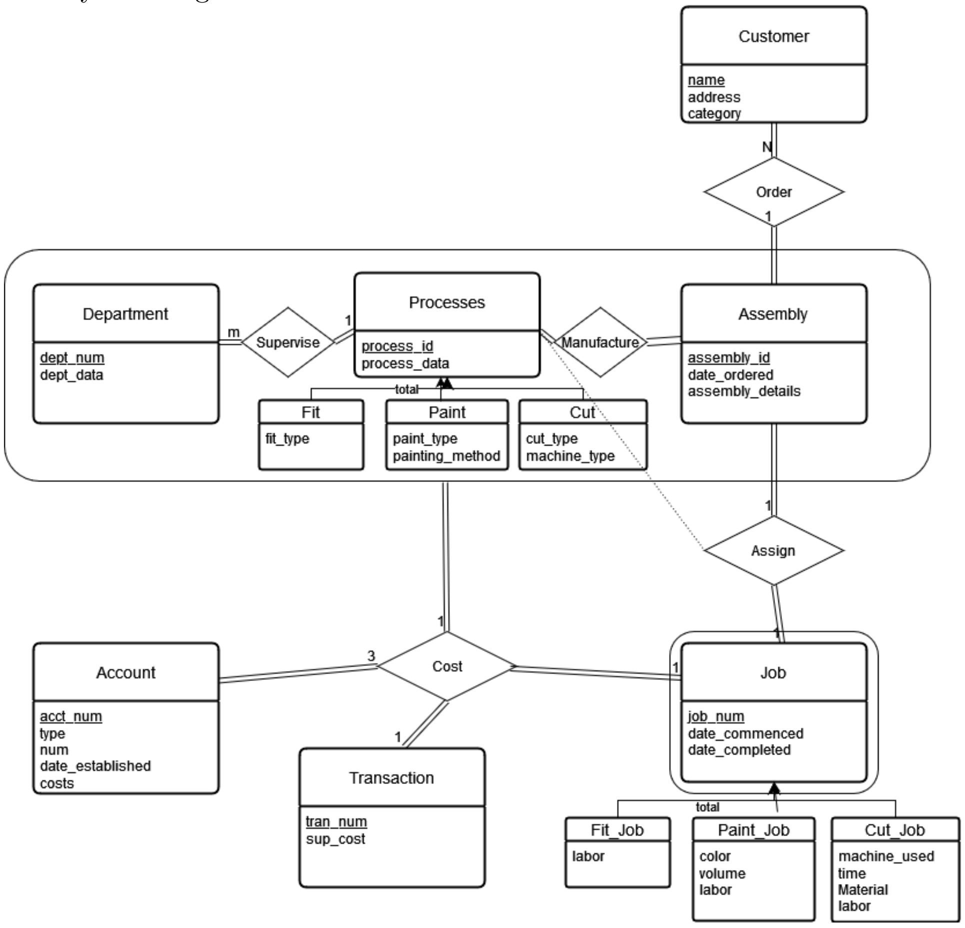
NAME: Nicholas Jacob
EMAIL: nicholas.c.jacob-1@ou.edu
STUDENT ID: # 113578513
Final Project
COURSE: CS/DSA 4513 DATABASE MANAGEMENT
SECTION: ONLINE
SEMESTER: FALL 2023
INSTRUCTOR: DR. LE GRUENWALD
SCORE:

Contents

1 ER Diagram

Here is my ER diagram



2 Relational Database Schema

Here are my schema:

Process(process_id,process_data)
Assemblies(assembly_id,date_ordered, assembly_details)
Manufacture(process_id,assembly_id)
Customer(name,address, category)
Order(name,assembly_id)
Department(dept_num,dept_data)
Supervise(dept_num,process_id)
Fit(process_id, fit_type)
Paint(process_id, paint_type, painting_method)
Cut(process_id,cutting_type, machine_type)
Account(acct_id, type, type_id, date_established, costs)
Job(job_num, job_date_commenced, job_completed)
Assign(job_num, assembly_id,process_id)
Transaction(tran_num, sup_cost)
Costs(job_num, acct_id,process_id, assembly_id, tran_num,dept_num)
Fit_Job(job_num, labor)
Paint_Job(job_num,color,volume, labor)
Cut_Job(job_num, machine_type, time, material, labor)

3 Storage

3.1 Storage Structures

Table Name	Query Number and Type	Search Key	Query Frequency	Selected File Organization	Justification
Customer	1 Insertion	name	30/Day	heap on name	At the moment adding lots of data and not accessing it directly often
Department	2 Insertion	dept_num	infrequent	Sequential on dept_num	Since this data is added infrequently but referenced by other tables often, sequential insertion seems appropriate.
Process (and sub categories)	3 Insertion	process_id, (sub category info)	infrequent	Sequential on process_id (and sub category id)	Infrequent insertion but often called
Supervises	3 Insertion	process_id and dept_num	infrequent	Sequential on process_id	Infrequent insertion but called often on process_id
Orders	4 Insertion	name, assembly_id	40/Day	dynamic hash on name and ass_id	This is a lot of orders to create each day. These will need to be joined with other tables frequently as is happening in our insertion so it is important to be easily accessible
Manufacture	4 Insertion	assembly_id	40/Day (but each assembly may have many processes)	dynamic hash on assembly_id	Frequent insertion with joins on other tables
Account	5 Insertion	type_acct and num	10/Day	Multitable clustering with type_acct for clustering and num sequential	This structure will make for fast access later and there is a fair amount of additions here.

Table Name	Query Number and Type	Search Key	Query Frequency	Selected File Organization	Justification
Job	6 Insertion	job_num	50/day	B tree on job_num	B tree is appropriate for often inserted and often called index.
Job	7 Random Search (Insertion of job_date_end)	job_num	50/Day	*B* tree on job_num	To enter completion data, you'll need a random search on job_num. *B* tree will be an efficient storage for all these records
Transaction and Costs	8 Random Search	tran_no for Transaction and tran_num, process_id for Costs	50/day	B tree on the tran_no and process_id	We'll need to update a lot of accounts here so it will be important to get to them quickly
Account	9 Random Search	type = Assembly and num	200/day	B tree on num	We have previously done clustering on these attributes so this will require nothing additional to the file
Job	10 Range Search	job_date_commenced and job_date_completed	20/day	Sequential index on both dates	Frequent call. If put in order can retrieve data faster
Manufacture	11 Random Search	assembly_id	100/day	Sequential index on assembly_id	This index was already created for Query 4.
Customer	12 Range Search	name (in order) by category	100/Day	Multitable Clustering with category for clustering and name stored in a B^+ tree	Since this data is accessed often this table should be pre-built. New customers are added often so B^+ tree storage on name will be most efficient within this multitable
Cut_Job	13 Range Search	job_num	1/Month	Sequential Index on job_num	Since we are doing a range search, we would like these to be in order.
Paint_Job	14 Random Search	job_num	1/Week	Dynamic Hash function on job_num	since we are accessing occasionally but adding lots of jobs, it would be nice to have quick access via a hash.

3.2 Storage Structures on Azure

Info on Azure indexing can be found here. Implementing these was a challenge. Azure uses B trees by default on the primary keys. This is great for random search but not so great for range searches. Since we knew this there were a few indexes that were unnecessary to create. Most of the rest were created especially if there were two attributes that were being indexed together. Sequential indexes (for range sort) were done by adding the ASC or DES tag to the attribute in question. Each index was created and added to the SQL code creating the tables necessary for indexing.

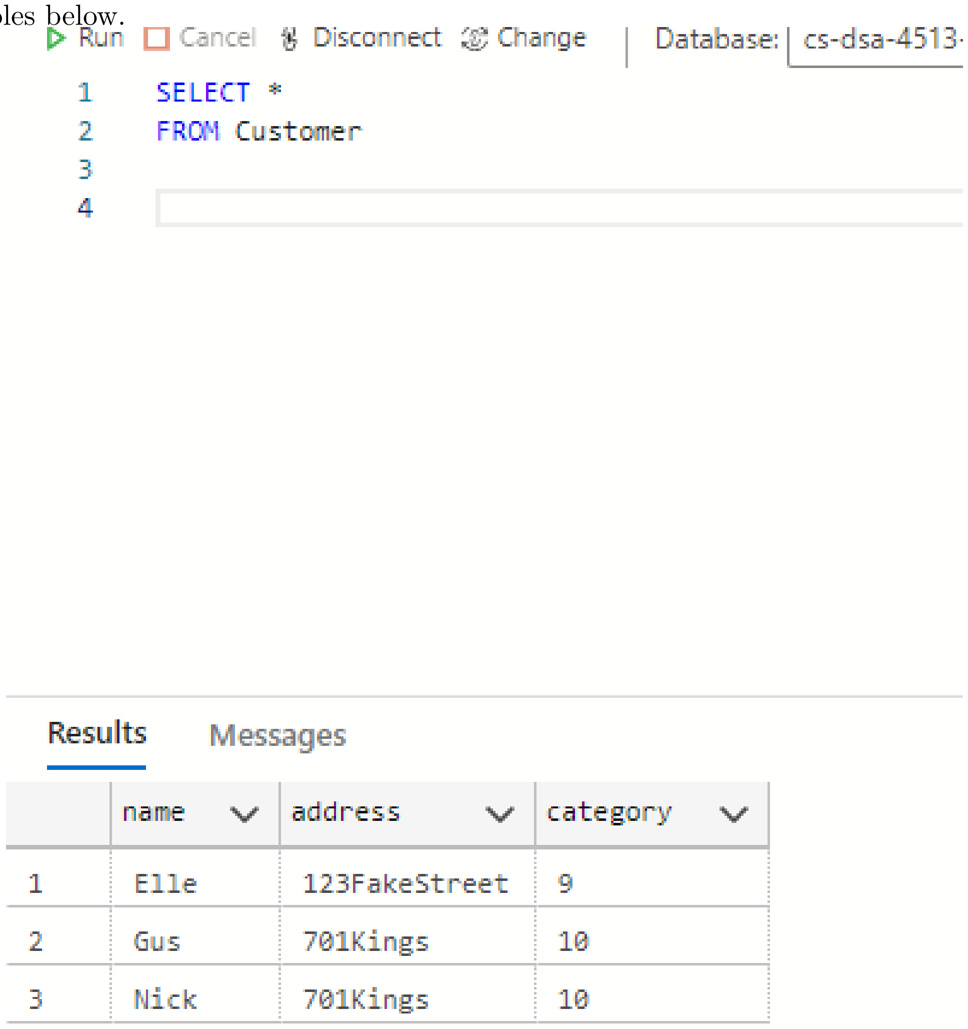
4 SQL and Azure

I have included my SQL file that creates the tables and indexes.

```
-- While working on the database design, it’s useful to start from scratch every time
-- Hence, we drop tables in reverse order they are created (so the foreign keyconstra
DROP TABLE IF EXISTS Enrollment
DROP TABLE IF EXISTS Student
DROP TABLE IF EXISTS Class
DROP TABLE IF EXISTS Cut_Job;
DROP TABLE IF EXISTS Paint_Job;
DROP TABLE IF EXISTS Fit_Job;
DROP TABLE IF EXISTS Costs;
DROP TABLE IF EXISTS Transact;
DROP TABLE IF EXISTS Assign;
DROP TABLE IF EXISTS Jobs;
DROP TABLE IF EXISTS Maintains;
DROP TABLE IF EXISTS Account;
DROP TABLE IF EXISTS Cut;
DROP TABLE IF EXISTS Paint;
DROP TABLE IF EXISTS Fit;
DROP TABLE IF EXISTS Supervise;
DROP TABLE IF EXISTS Department;
DROP TABLE IF EXISTS Orders;
DROP TABLE IF EXISTS Customer;
DROP TABLE IF EXISTS Manufacture;
DROP TABLE IF EXISTS Assemblies;
DROP TABLE IF EXISTS Processes;
-- Create tables
```

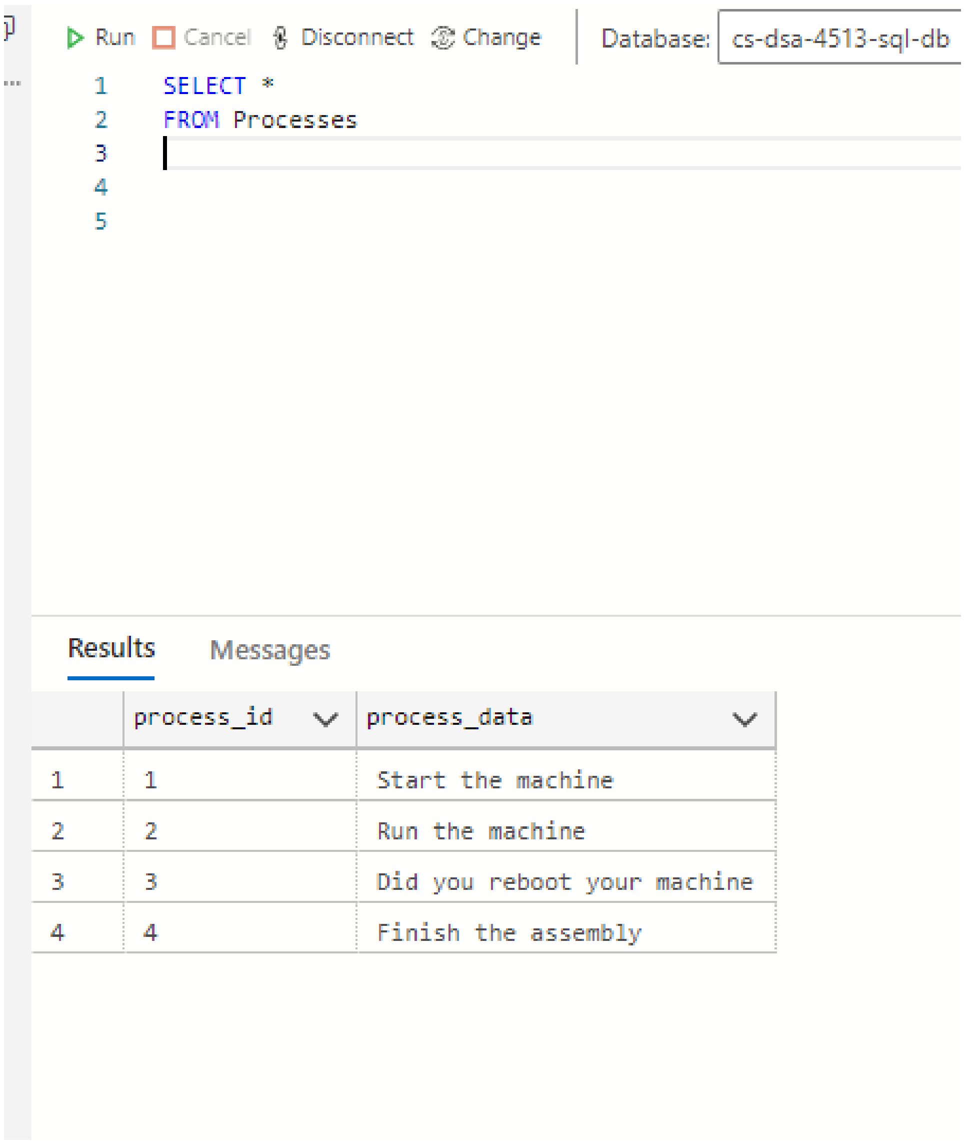
```
CREATE TABLE Processes(
process_id INT PRIMARY KEY,
process_data VARCHAR(64)
);
CREATE TABLE Assemblies(
assembly_id INT PRIMARY KEY,
date_ordered DATE,
assembly_details VARCHAR(64)
);
CREATE TABLE Manufacture (
process_id INT,
assembly_id INT,
CONSTRAINT FK_processid FOREIGN KEY(process_id) REFERENCES Processes,
CONSTRAINT FK_aid FOREIGN KEY(assembly_id) REFERENCES Assemblies
);
CREATE TABLE Customer(
name VARCHAR(64) PRIMARY KEY,
address VARCHAR(64),
category NUMERIC(2,0) NOT NULL,
CHECK(category>0 and category<11)
);
CREATE TABLE Orders (
name VARCHAR(64),
assembly_id INT,
CONSTRAINT PK_orders PRIMARY KEY (name, assembly_id),
CONSTRAINT FK_cname FOREIGN KEY(name) REFERENCES Customer,
CONSTRAINT FK_aidOrders FOREIGN KEY(assembly_id) REFERENCES Assemblies
);
CREATE TABLE Department (
dept_num INT PRIMARY KEY,
dept_data VARCHAR(128)
);
CREATE TABLE Supervise (
dept_num INT,
process_id INT,
CONSTRAINT PK_Supervises PRIMARY KEY(dept_num, process_id),
CONSTRAINT FK_deptnum FOREIGN KEY (dept_num) REFERENCES Department,
CONSTRAINT FK_proccessid FOREIGN KEY (process_id) REFERENCES Processes
);
CREATE TABLE Fit(
process_id INT PRIMARY KEY,
fit_type VARCHAR(64),
CONSTRAINT FK_fit_process FOREIGN KEY(process_id) REFERENCES Processes
);
CREATE TABLE Paint(
process_id INT PRIMARY KEY,
paint_type VARCHAR(64),
paint_method VARCHAR(64),
CONSTRAINT FK_paint_process FOREIGN KEY(process_id) REFERENCES Processes
);
CREATE TABLE Cut(
process_id INT PRIMARY KEY,
cutting_type VARCHAR(64),
machine_type VARCHAR(64),
CONSTRAINT FK_cut_process FOREIGN KEY(process_id) REFERENCES Processes
);
CREATE TABLE Account(
acct_id INT PRIMARY KEY,
type_acct VARCHAR(10) check (type_acct in ('Process','Assembly','Department')),
date_established DATE,
type_acct_id INT, --I should be a FK to Process, Assembly or department but could not
costs INT
);
/*
CREATE TABLE Maintains(
acct_id INT,
type_acct VARCHAR(10) check (type_acct in ('Process','Assembly','Department')),
--num INT,
CONSTRAINT PK_maintain PRIMARY KEY(acct_id,type_acct),
CONSTRAINT FK_maintain_acct FOREIGN KEY(acct_id) REFERENCES Account--should have FKey
);
*/
CREATE TABLE Jobs(
job_num INT PRIMARY KEY,
job_date_commenced DATE,
job_date_completed DATE
);
CREATE TABLE Assign(
job_num INT,
assembly_id INT,
process_id INT,--this gets the job started but not all of them?
CONSTRAINT PK_assign PRIMARY KEY(job_num,process_id,assembly_id),
CONSTRAINT FK_assign_process FOREIGN KEY(process_id) REFERENCES Processes,
CONSTRAINT FK_assign_job FOREIGN KEY(job_num) REFERENCES Jobs,
CONSTRAINT FK_assign_assembly FOREIGN KEY(assembly_id) REFERENCES Assemblies
);
CREATE TABLE Transact(
tran_num INT PRIMARY KEY,
sup_cost INT
);
CREATE TABLE Costs(--either transact or cost will need a process_id otherwise we won'
job_num INT,
tran_num INT,
process_id INT,
--CONSTRAINT PK_Costs PRIMARY KEY(job_num, tran_num),
CONSTRAINT FK_cost_process FOREIGN KEY(process_id) REFERENCES Processes,
--CONSTRAINT FK_cost_acct FOREIGN KEY(acct_id) REFERENCES Account,
--CONSTRAINT FK_cost_department FOREIGN KEY(dept_num) REFERENCES Department,
--CONSTRAINT FK_cost_assembly FOREIGN KEY(assembly_id) REFERENCES Assemblies,
CONSTRAINT FK_cost_transact FOREIGN KEY(tran_num) REFERENCES Transact,
CONSTRAINT FK_cost_job FOREIGN KEY(job_num) REFERENCES Jobs
);
CREATE TABLE Fit_Job(
job_num INT PRIMARY KEY,
labor NUMERIC(3,0),
CONSTRAINT FK_fit_job FOREIGN KEY(job_num) REFERENCES Jobs
);
CREATE TABLE Paint_Job(
job_num INT PRIMARY KEY,
color VARCHAR(10),
volume NUMERIC(3,2),
labor NUMERIC(3,0),
CONSTRAINT FK_paint_job FOREIGN KEY(job_num) REFERENCES Jobs
);
CREATE TABLE Cut_Job(
job_num INT PRIMARY KEY,
machine_type VARCHAR(10),
time NUMERIC(2,2),
material NUMERIC(2,2),
labor NUMERIC(3,0),
CONSTRAINT FK_cut_job FOREIGN KEY(job_num) REFERENCES Jobs
);
go
CREATE INDEX customer_name ON Customer(name)--query 1 insertion of customers
GO
CREATE INDEX dept_num ON Department(dept_num ASC) --query 2 insert of departments
GO
CREATE INDEX process ON Processes(process_id ASC) --query 3 making sequential indexes
CREATE INDEX process_cut ON Cut(process_id ASC)
CREATE INDEX process_paint ON Paint(process_id ASC)
CREATE INDEX process_fit ON Fit(process_id ASC)
GO
CREATE INDEX supervies ON Supervise(process_id, dept_num) --query 3 getting the super
GO
CREATE INDEX orders_index ON Orders(name, assembly_id) --query 4 keeping the name and

CREATE INDEX Manufacture_index ON Manufacture(assembly_id)--query4
GO
CREATE INDEX account_index ON Account(type_acct ASC, type_acct_id) --query5 this will
--No need to create 6 and 7 as B tree is created on Primary Key automatically
GO
CREATE INDEX transaction_index ON Transact(tran_num)
CREATE INDEX cost_index ON Costs(tran_num, process_id)--query8
GO
CREATE INDEX account_assembly ON Account(type_acct, type_acct_id)--query9
GO
CREATE INDEX job_date_index ON Jobs(job_date_commenced ASC, job_date_completed ASC)--
GO
--CREATE INDEX manufacture_index ON Manufacture(assembly_id ASC)--query11
CREATE INDEX customer_index ON Customer(name ASC, category)--query 12.  Joining the n
GO
CREATE INDEX cutjob_index ON Cut_Job(job_num)--query 13
GO
CREATE INDEX paintjob_index ON Paint_Job(job_num)--query 14
GO
```

I went ahead and added some data to each table so that I would be able to examine if the later queries were working correctly. We see a few of these tables below.



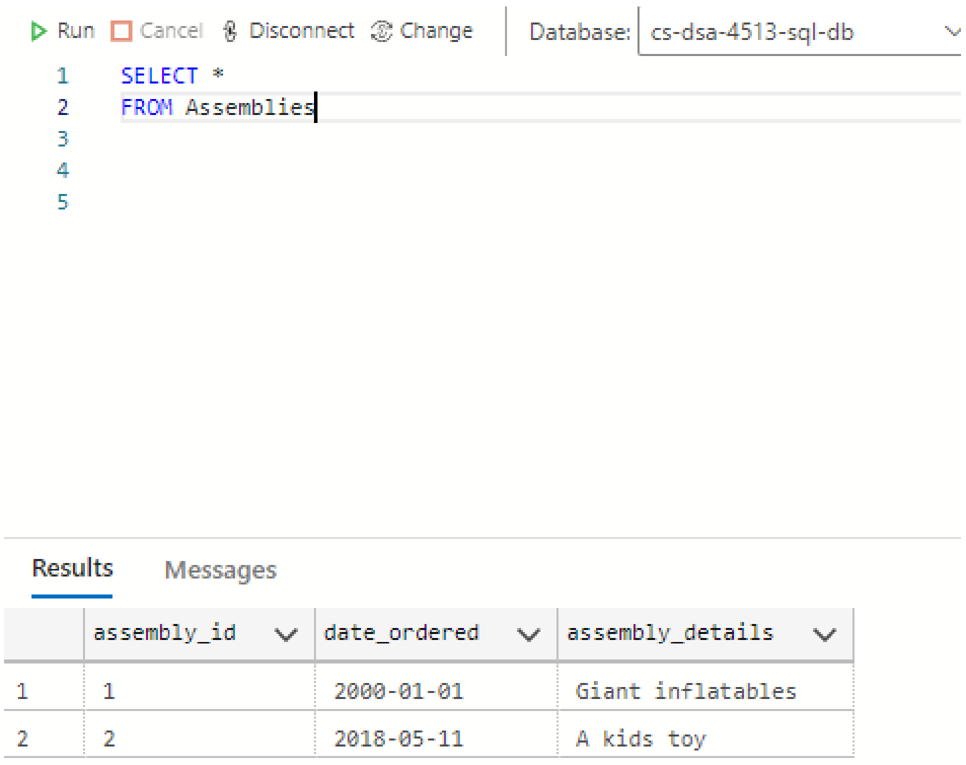
```
SELECT *
FROM Customer
```

	name	address	category
1	Elle	123FakeStreet	9
2	Gus	701Kings	10
3	Nick	701Kings	10



```
SELECT *
FROM Processes
```

	process_id	process_data
1	1	Start the machine
2	2	Run the machine
3	3	Did you reboot your machine
4	4	Finish the assembly

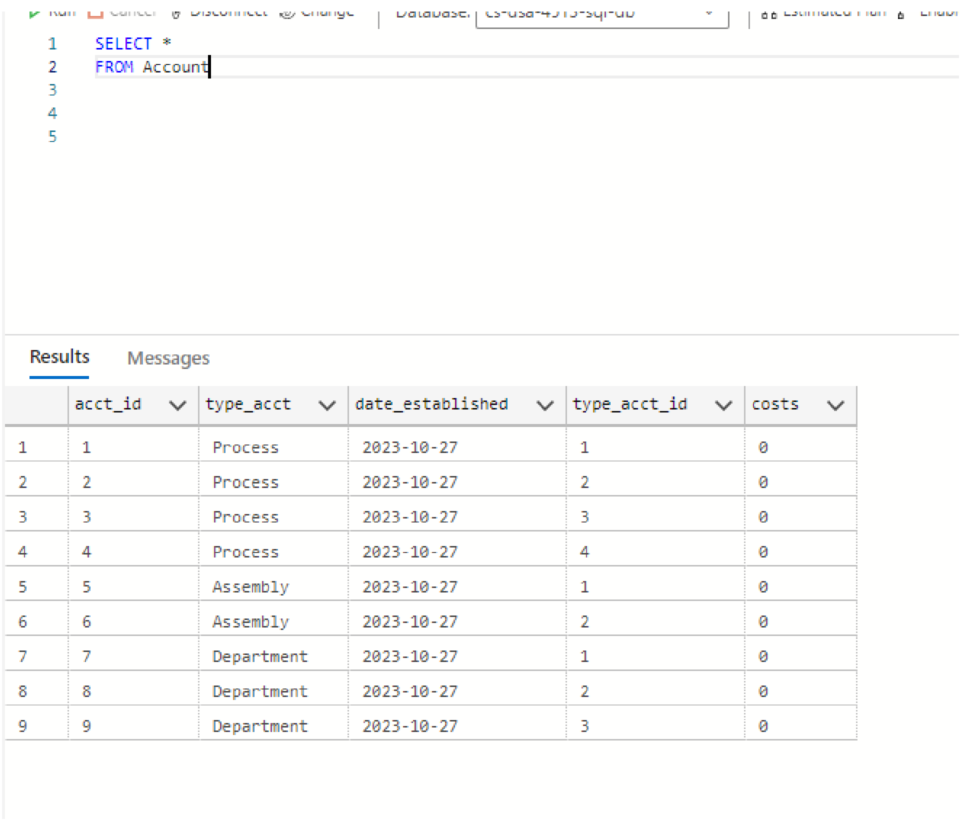


Run Cancel Disconnect Change Database: cs-dsa-4513-sql-db

```
SELECT *
FROM Assemblies
```

Results Messages

	assembly_id	date_ordered	assembly_details
1	1	2000-01-01	Giant inflatables
2	2	2018-05-11	A kids toy



```
SELECT *
FROM Account
```

	acct_id	type_acct	date_established	type_acct_id	costs
1	1	Process	2023-10-27	1	0
2	2	Process	2023-10-27	2	0
3	3	Process	2023-10-27	3	0
4	4	Process	2023-10-27	4	0
5	5	Assembly	2023-10-27	1	0
6	6	Assembly	2023-10-27	2	0
7	7	Department	2023-10-27	1	0
8	8	Department	2023-10-27	2	0
9	9	Department	2023-10-27	3	0

5 SQL and Java

5.1 SQL Transact

I put the transact calls in a new SQL file included below.

```
GO
DROP PROCEDURE IF EXISTS query1 --get rid of the procedure if you built it before

GO
CREATE PROCEDURE query1 --this is the first.  Need three inputs
        @name VARCHAR(64),
    @address VARCHAR(64),
    @category NUMERIC(2,0)
AS
BEGIN
        INSERT INTO Customer VALUES (@name, @address, @category) --insert me now
```

```
END
GO
--EXEC query1 @name = 'Nick', @address = NULL, @category = 10
GO

GO
DROP PROCEDURE IF EXISTS query2 --get rid of the procedure if you built it before

GO
CREATE PROCEDURE query2
        @dept_num INT,
    @dept_data VARCHAR(128)
AS
BEGIN
        INSERT INTO Department VALUES (@dept_num, @dept_data) --insert me now
END
GO
--EXEC query2 @dept_num = 1, @dept_data = NULL
GO
DROP PROCEDURE IF EXISTS query3 --get rid of the procedure if you built it before

GO
CREATE PROCEDURE query3 --this is the first.  Need three inputs
        @process_id INT,
    @process_data VARCHAR(64),
    @type VARCHAR(5),
    @type_type VARCHAR(64),
    @type_method VARCHAR(64)
AS
BEGIN
        INSERT INTO Processes VALUES (@process_id, @process_data) --insert into proce
        IF @type = 'Fit' INSERT INTO Fit VALUES (@process_id, @type_type)
        IF @type = 'Paint' INSERT INTO Paint VALUES (@process_id, @type_type, @type_m
        IF @type = 'Cut' INSERT INTO Cut VALUES(@process_id, @type_type, @type_method
END
GO
--EXEC query3 1,'','Fit',NULL,NULL
GO
DROP PROCEDURE IF EXISTS query4 --get rid of the procedure if you built it before
```

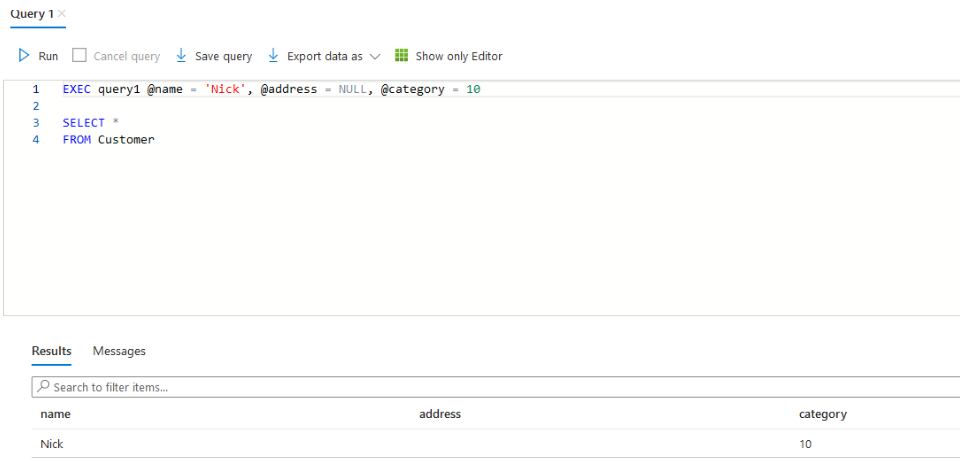
```
GO
CREATE PROCEDURE query4 --create assembly with all associated processes for customer
        @assembly_id INT,
    @date_ordered DATE,
    @assembly_details VARCHAR(64),
    @name VARCHAR(64),
    @process_ids VARCHAR(64)--take this as a string seperated by commas and we'll sli
AS
BEGIN
        INSERT INTO Assemblies VALUES (@assembly_id, @date_ordered, @assembly_details
        INSERT INTO Orders VALUES (@name,@assembly_id) --record what customer made th
        INSERT INTO Manufacture SELECT *,@assembly_id FROM STRING_SPLIT(@process_ids,
END
GO
--EXEC query4 1,'10/01/23',NULL,'Nick','1,1,1'
GO
DROP PROCEDURE IF EXISTS query5 --get rid of the procedure if you built it before

GO
CREATE PROCEDURE query5
    @acct_id INT,
    @type VARCHAR(10),
    @date_established DATE,
        @num INT
AS
BEGIN
        INSERT INTO Account VALUES (@acct_id,@type,@date_established,@num,0) --insert
        --INSERT INTO Maintains VALUES (@acct_id,@type)--,@num) --pass this info into
END
GO
--EXEC query5 1,'Process','10/10/20',1

GO



DROP PROCEDURE IF EXISTS query6 --get rid of the procedure if you built it before

GO
CREATE PROCEDURE query6
```

```
    @job_num INT,
    @job_date_commenced DATE,
    @assembly_id INT,
        @process_id INT
AS
BEGIN
        INSERT INTO Jobs (job_num,job_date_commenced) VALUES (@job_num,@job_date_comm
        INSERT INTO Assign VALUES (@job_num,@assembly_id,@process_id) --pass this inf
END

GO
--EXEC query6 50,NULL,1,1
GO

DROP PROCEDURE IF EXISTS query7 --get rid of the procedure if you built it before

GO
CREATE PROCEDURE query7
    @job_num INT,
    @job_date_completed DATE,
    @job_type VARCHAR(10),
        @labor NUMERIC(3,0),
        @machine_type VARCHAR(10),
        @time NUMERIC(2,2),
        @material NUMERIC(2,2),
        @color VARCHAR(10),
        @volume NUMERIC(3,2)

AS
BEGIN
        Update Jobs set job_date_completed = @job_date_completed where job_num = @job
        IF @job_type = 'Fit' INSERT INTO Fit_Job VALUES (@job_num, @labor)
        IF @job_type = 'Paint' INSERT INTO Paint_Job VALUES (@job_num, @color, @volum
        IF @job_type = 'Cut' INSERT INTO Cut_Job VALUES(@job_num, @machine_type, @tim
END
GO

--EXEC query7 @job_num = 50, @job_date_completed = '10/01/23', @job_type = 'Fit', @la
GO
```

```
GO

DROP PROCEDURE IF EXISTS query8 --get rid of the procedure if you built it before
GO
CREATE PROCEDURE query8
    @tran_num INT,
        @sup_cost INT,
        @job_num INT,
        @process_id INT
AS
BEGIN
        INSERT INTO Transact VALUES (@tran_num,@sup_cost)
        INSERT INTO Costs VALUES (@job_num, @tran_num, @process_id)
        UPDATE Account SET costs = costs + @sup_cost Where type_acct = 'Process' and
        UPDATE Account SET costs = costs + @sup_cost Where type_acct = 'Assembly' and
        UPDATE Account SET costs = costs + @sup_cost Where type_acct = 'Department' a
END
GO

--EXEC query8 @tran_num = 50, @sup_cost = 100, @job_num = 50, @process_id =1;

GO

DROP PROCEDURE IF EXISTS query9 --get rid of the procedure if you built it before
GO
CREATE PROCEDURE query9
    @assembly_id INT

AS
BEGIN
        Select * FROM Account WHERE type_acct_id = @assembly_id and type_acct = 'Asse
END
GO



GO

DROP PROCEDURE IF EXISTS query12 --get rid of the procedure if you built it before
```

```
GO
CREATE PROCEDURE query12
    @category NUMERIC(2,0)

AS
BEGIN
        Select name FROM Customer WHERE category = @category ORDER BY name ASC
END
GO



GO

DROP PROCEDURE IF EXISTS query13 --get rid of the procedure if you built it before
GO
CREATE PROCEDURE query13
    @job_num_start INT,
        @job_num_end INT

AS
BEGIN
        Delete FROM Jobs Where job_num in (SELECT Jobs.job_num FROM Jobs, Cut_Job Whe
        DELETE FROM Cut_Job where (job_num >= @job_num_start) and (job_num<=@job_num_
END
GO


EXEC query13 @job_num_start = 50, @job_num_end = 60;

GO

DROP PROCEDURE IF EXISTS query14 --get rid of the procedure if you built it before
GO
CREATE PROCEDURE query14
    @job_num INT,
        @color VARCHAR(10)

AS
BEGIN
```

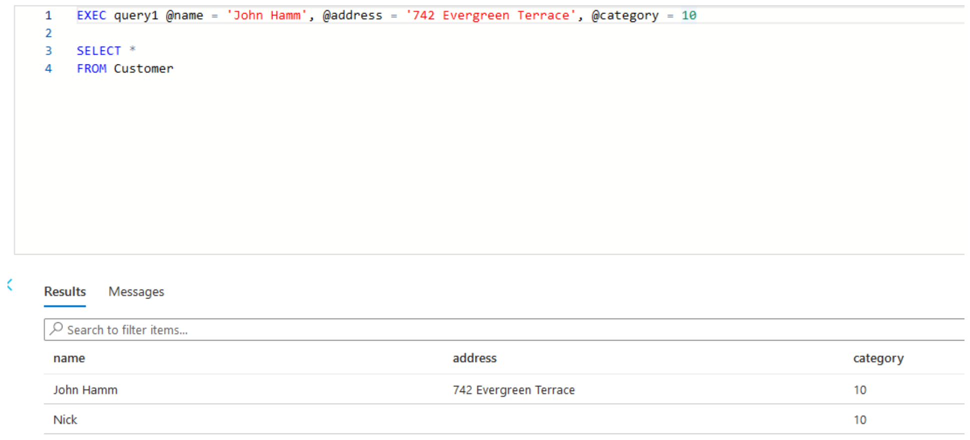
```
        Update Paint_Job set color = @color where job_num = @job_num
END
GO
```



```
EXEC query1 @name = 'Nick', @address = NULL, @category = 10

SELECT *
FROM Customer
```

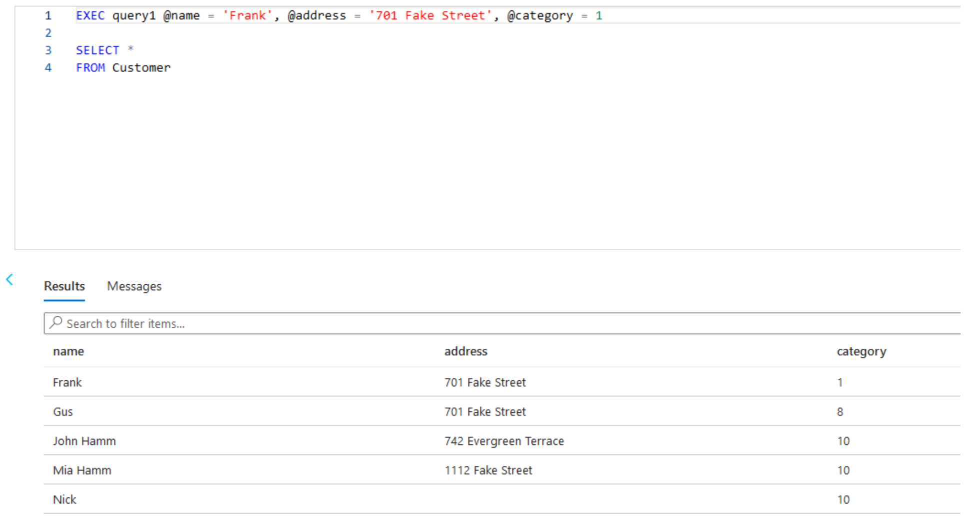
name	address	category
Nick		10

1.



```
EXEC query1 @name = 'John Hamm', @address = '742 Evergreen Terrace', @category = 10

SELECT *
FROM Customer
```

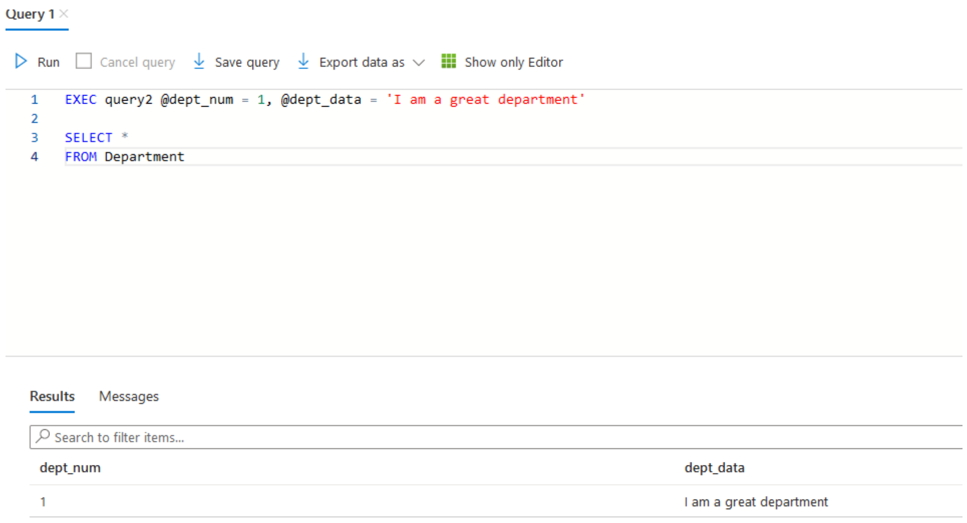
name	address	category
John Hamm	742 Evergreen Terrace	10
Nick		10



```sql
EXEC query1 @name = 'Frank', @address = '701 Fake Street', @category = 1

SELECT *
FROM Customer
```

Results Messages

name	address	category
Frank	701 Fake Street	1
Gus	701 Fake Street	8
John Hamm	742 Evergreen Terrace	10
Mia Hamm	1112 Fake Street	10
Nick		10

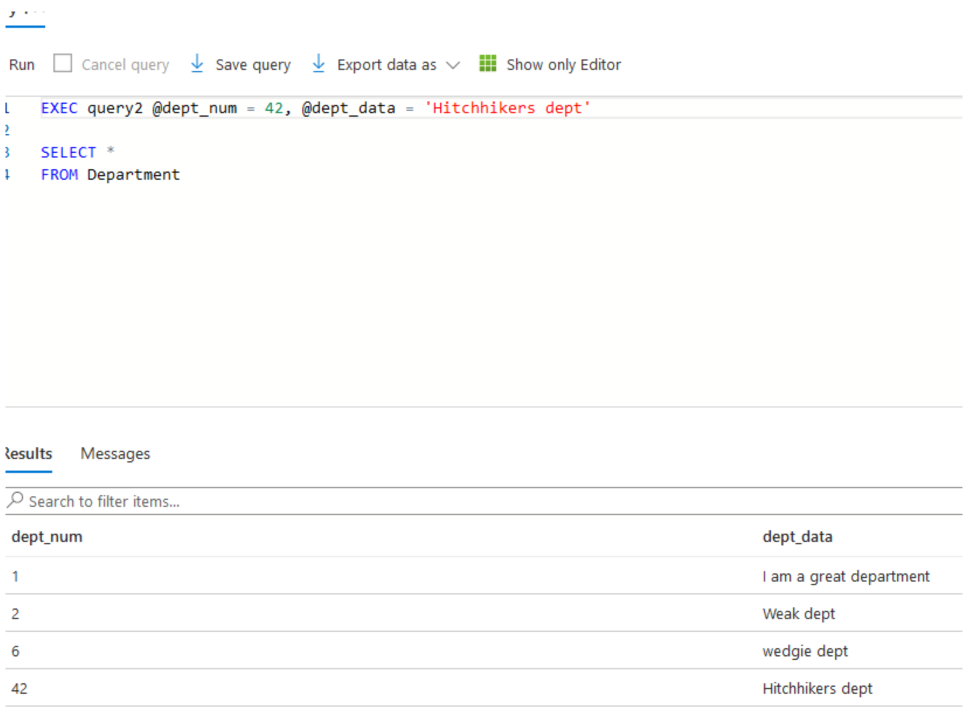


```sql
EXEC query2 @dept_num = 1, @dept_data = 'I am a great department'

SELECT *
FROM Department
```

Results Messages

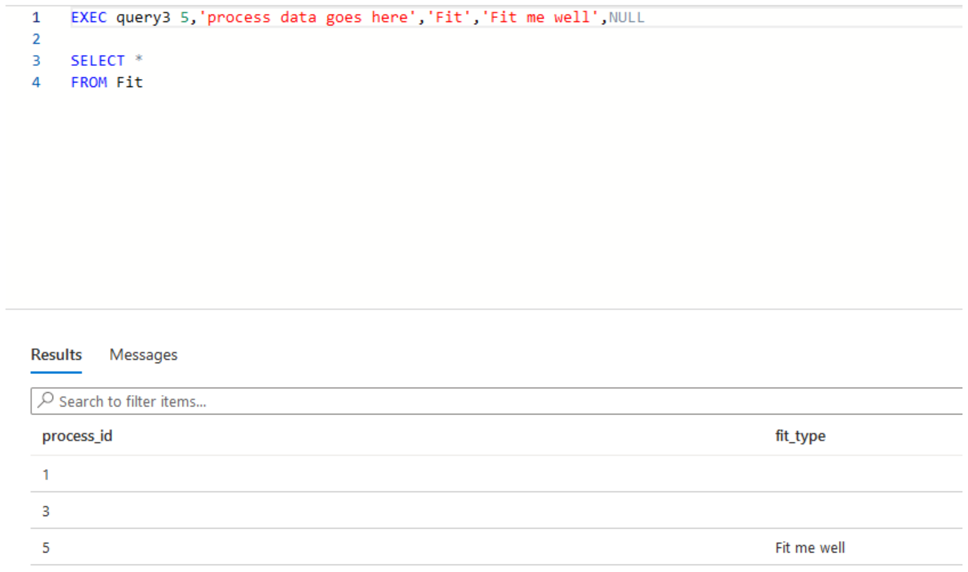
dept_num	dept_data
1	I am a great department

2.



```sql
EXEC query2 @dept_num = 42, @dept_data = 'Hitchhikers dept'

SELECT *
FROM Department
```

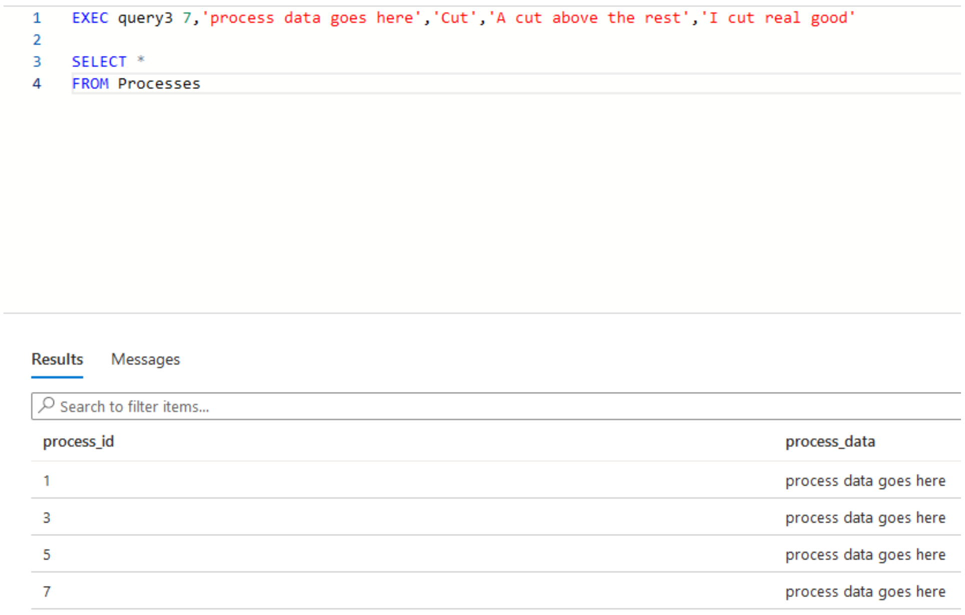
dept_num	dept_data
1	I am a great department
2	Weak dept
6	wedgie dept
42	Hitchhikers dept



```sql
EXEC query3 5,'process data goes here','Fit','Fit me well',NULL

SELECT *
FROM Fit
```

process_id	fit_type
1	
3	
5	Fit me well

3.

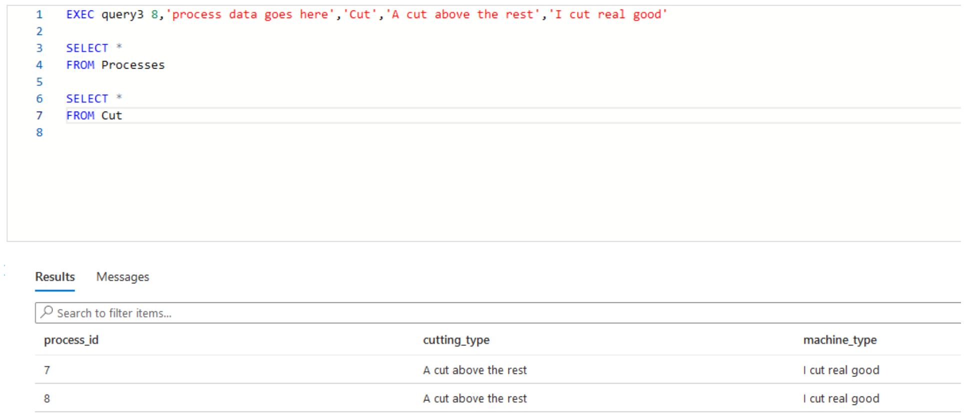


```
EXEC query3 7,'process data goes here','Cut','A cut above the rest','I cut real good'

SELECT *
FROM Processes
```

Results   Messages

Search to filter items...

process_id	process_data
1	process data goes here
3	process data goes here
5	process data goes here
7	process data goes here

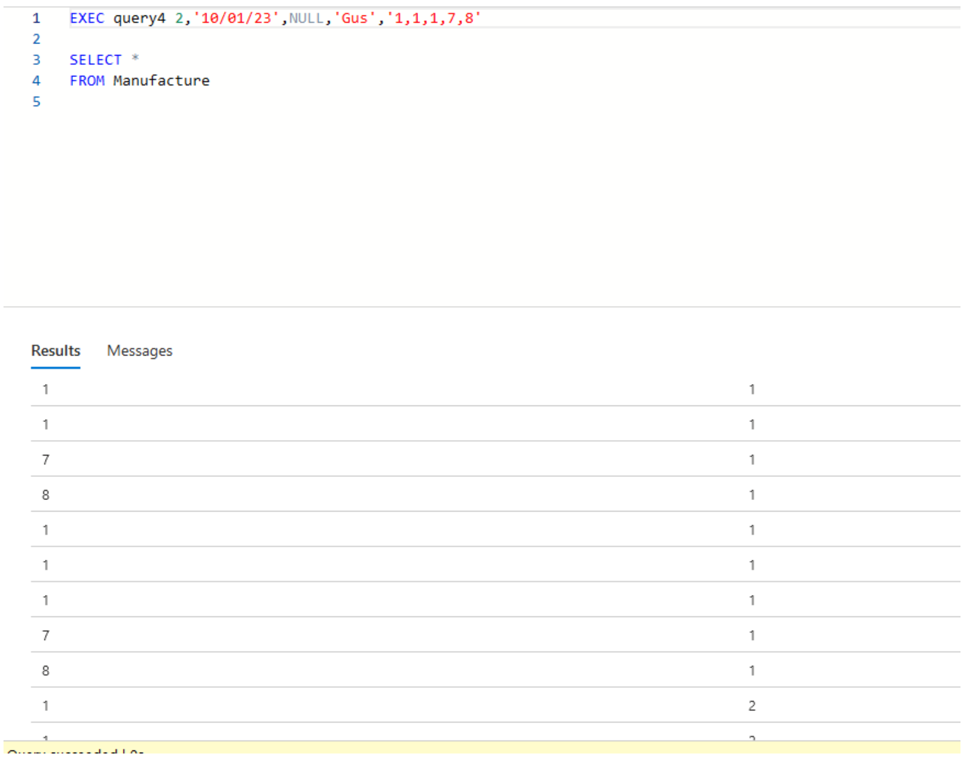


```
EXEC query3 8,'process data goes here','Cut','A cut above the rest','I cut real good'

SELECT *
FROM Processes

SELECT *
FROM Cut

```

Results   Messages

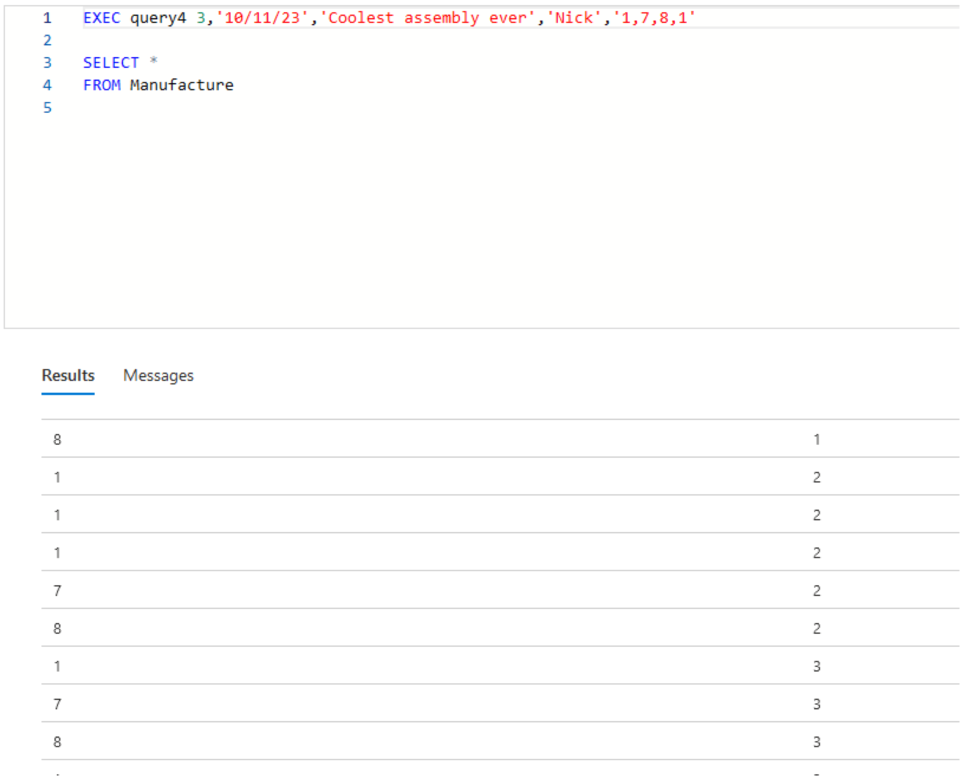
Search to filter items...

process_id	cutting_type	machine_type
7	A cut above the rest	I cut real good
8	A cut above the rest	I cut real good

4. 

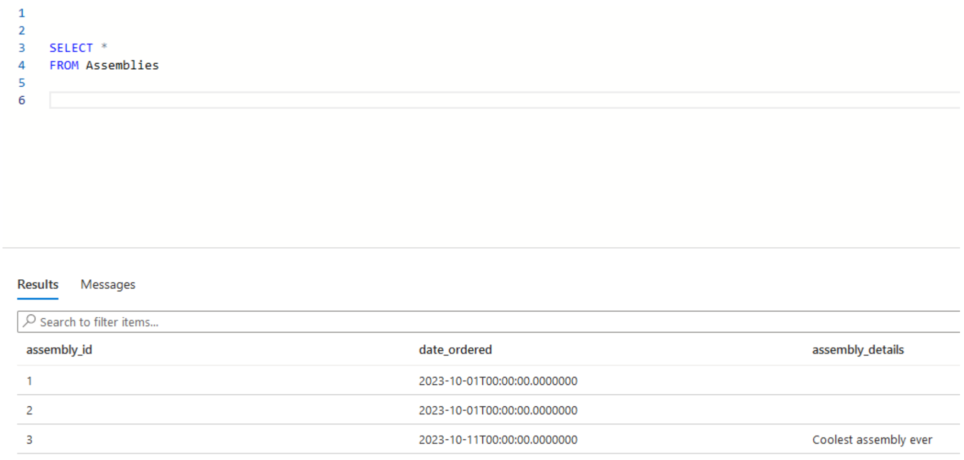
```sql
EXEC query4 2,'10/01/23',NULL,'Gus','1,1,1,7,8'

SELECT *
FROM Manufacture
```

```
1                                                               1
1                                                               1
7                                                               1
8                                                               1
1                                                               1
1                                                               1
1                                                               1
7                                                               1
8                                                               1
1                                                               2
```



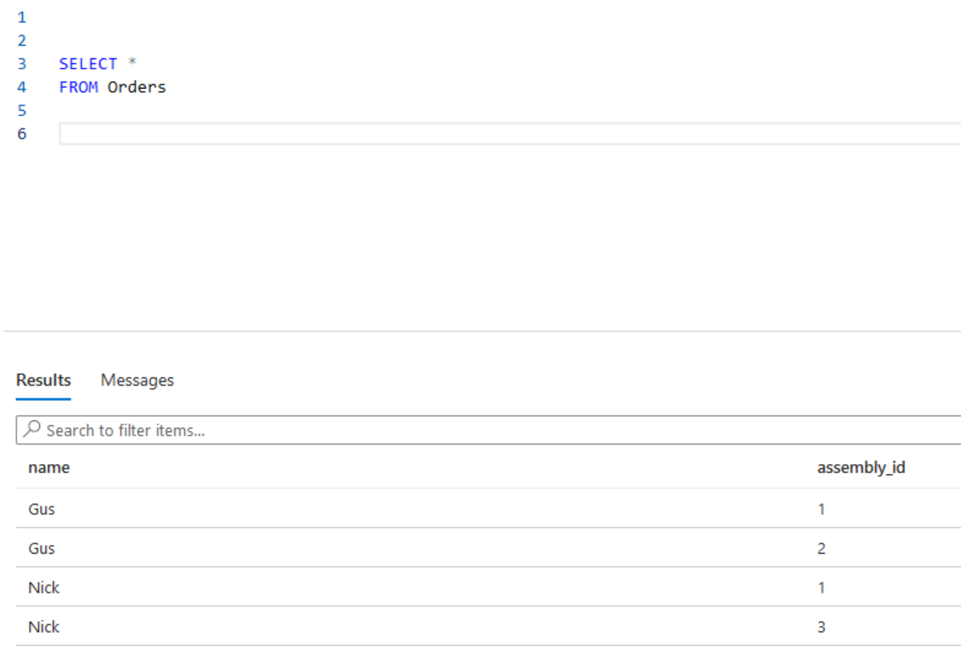
```sql
EXEC query4 3,'10/11/23','Coolest assembly ever','Nick','1,7,8,1'

SELECT *
FROM Manufacture
```

8	1
1	2
1	2
1	2
7	2
8	2
1	3
7	3
8	3



```sql
SELECT *
FROM Assemblies
```

assembly_id	date_ordered	assembly_details
1	2023-10-01T00:00:00.0000000	
2	2023-10-01T00:00:00.0000000	
3	2023-10-11T00:00:00.0000000	Coolest assembly ever

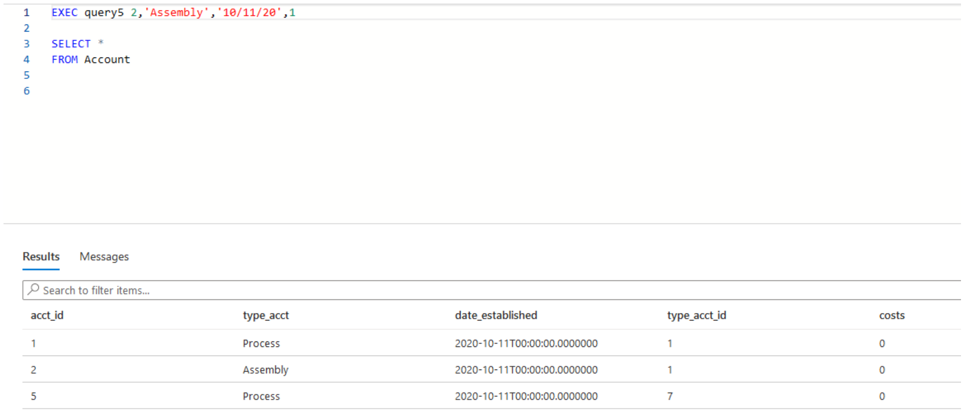


```sql
SELECT *
FROM Orders
```

Results Messages

Search to filter items...

name	assembly_id
Gus	1
Gus	2
Nick	1
Nick	3

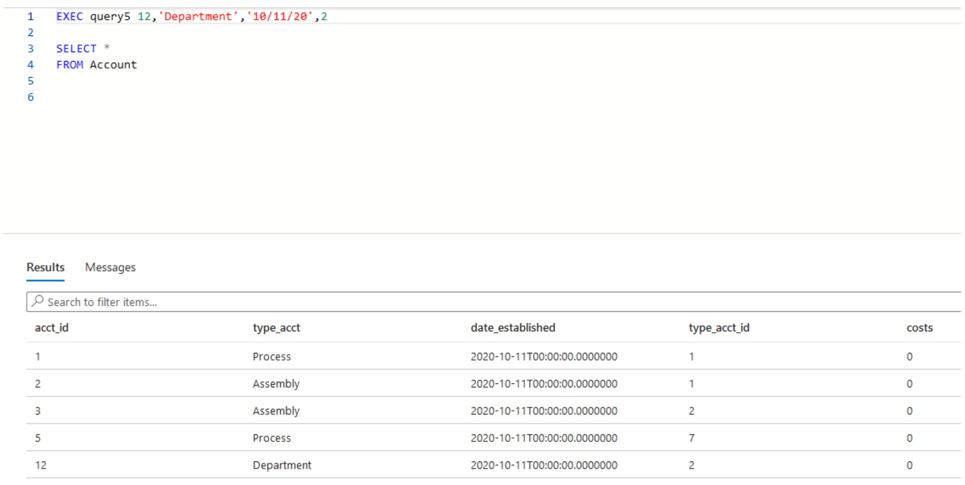
5. 

```sql
EXEC query5 2,'Assembly','10/11/20',1

SELECT *
FROM Account
```

Results Messages

Search to filter items...

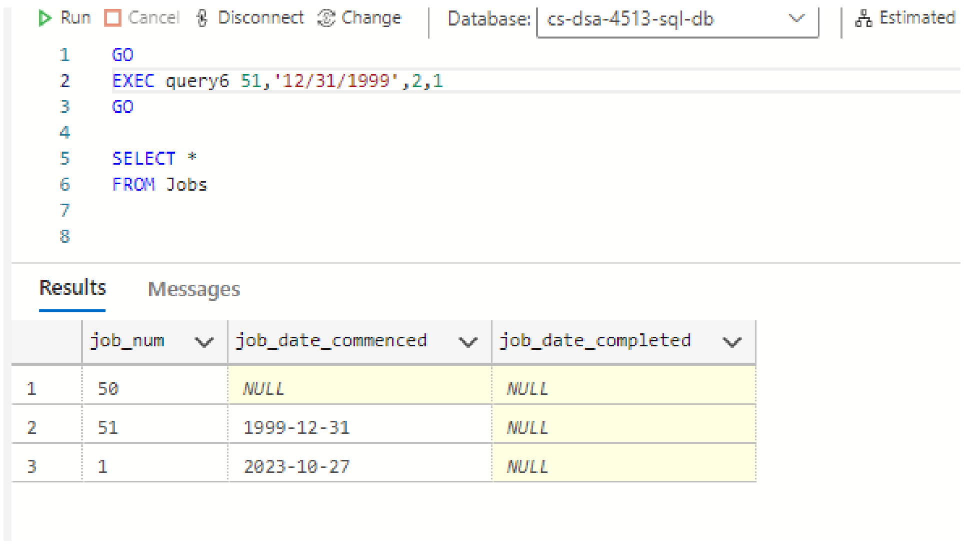
acct_id	type_acct	date_established	type_acct_id	costs
1	Process	2020-10-11T00:00:00.0000000	1	0
2	Assembly	2020-10-11T00:00:00.0000000	1	0
5	Process	2020-10-11T00:00:00.0000000	7	0



```sql
EXEC query5 12,'Department','10/11/20',2

SELECT *
FROM Account
```

Results Messages

acct_id	type_acct	date_established	type_acct_id	costs
1	Process	2020-10-11T00:00:00.0000000	1	0
2	Assembly	2020-10-11T00:00:00.0000000	1	0
3	Assembly	2020-10-11T00:00:00.0000000	2	0
5	Process	2020-10-11T00:00:00.0000000	7	0
12	Department	2020-10-11T00:00:00.0000000	2	0

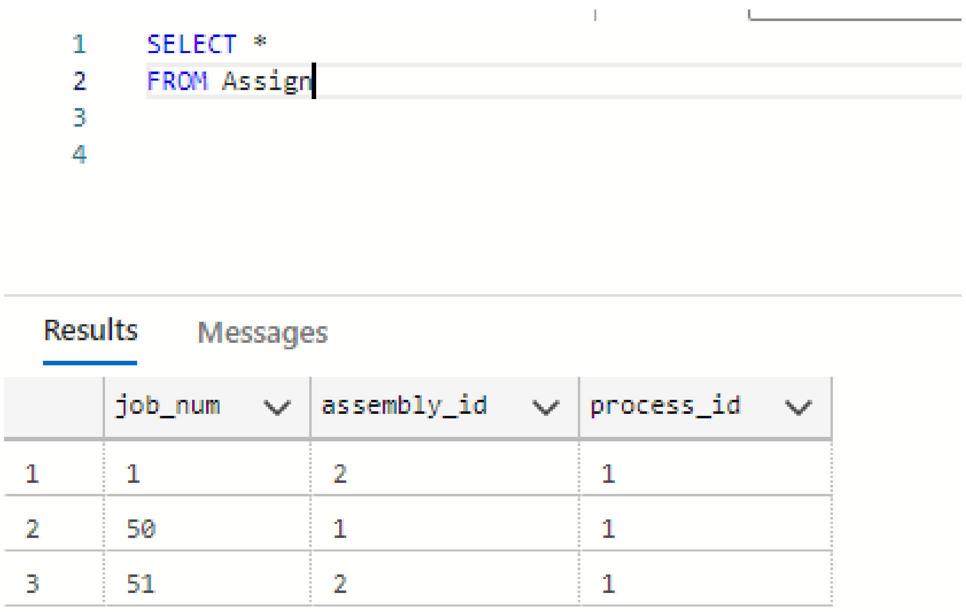
6. 

Run Cancel Disconnect Change Database: cs-dsa-4513-sql-db Estimated

```sql
GO
EXEC query6 51,'12/31/1999',2,1
GO

SELECT *
FROM Jobs
```

Results Messages

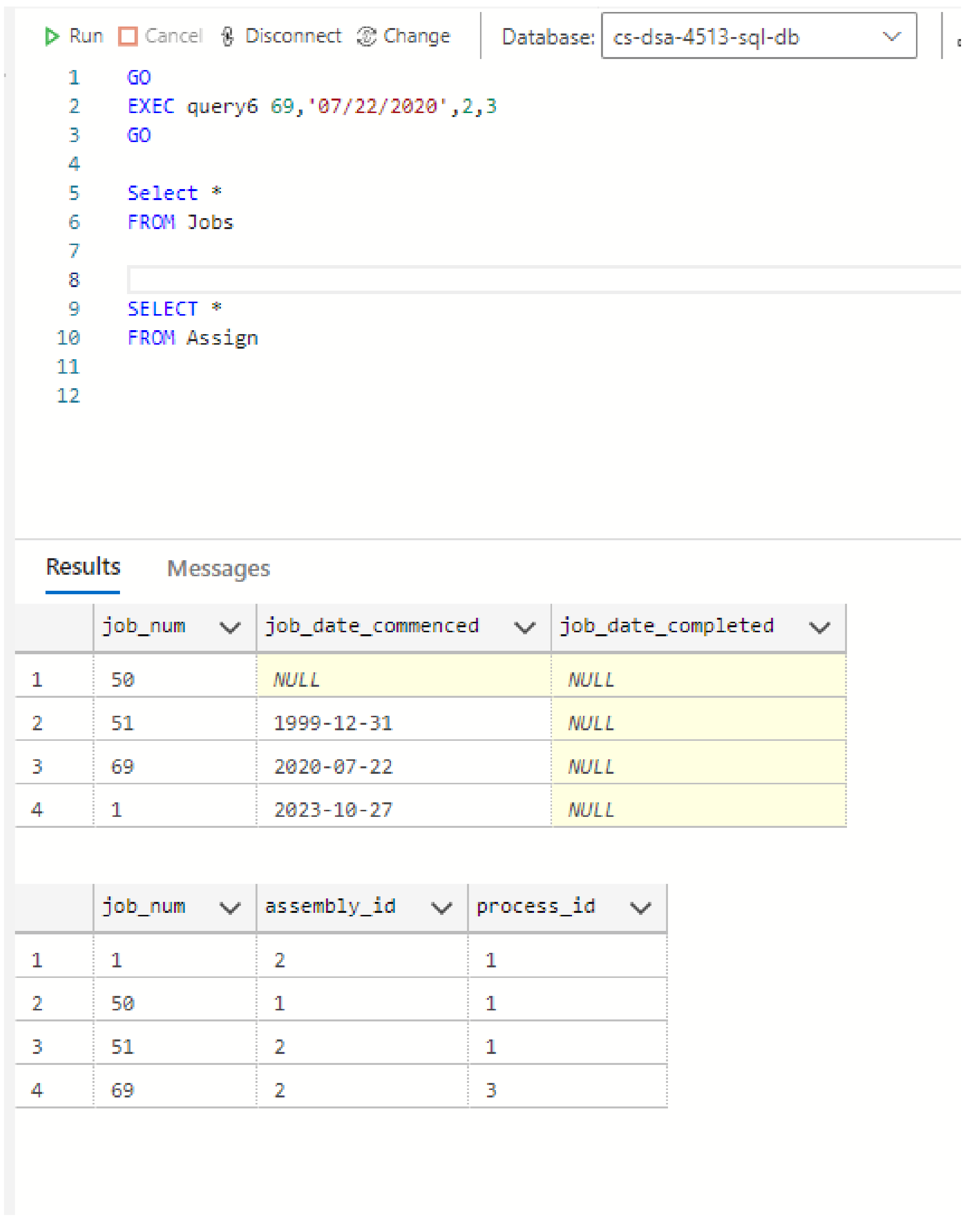
	job_num	job_date_commenced	job_date_completed
1	50	NULL	NULL
2	51	1999-12-31	NULL
3	1	2023-10-27	NULL



```sql
SELECT *
FROM Assign
```

Results Messages

	job_num	assembly_id	process_id
1	1	2	1
2	50	1	1
3	51	2	1

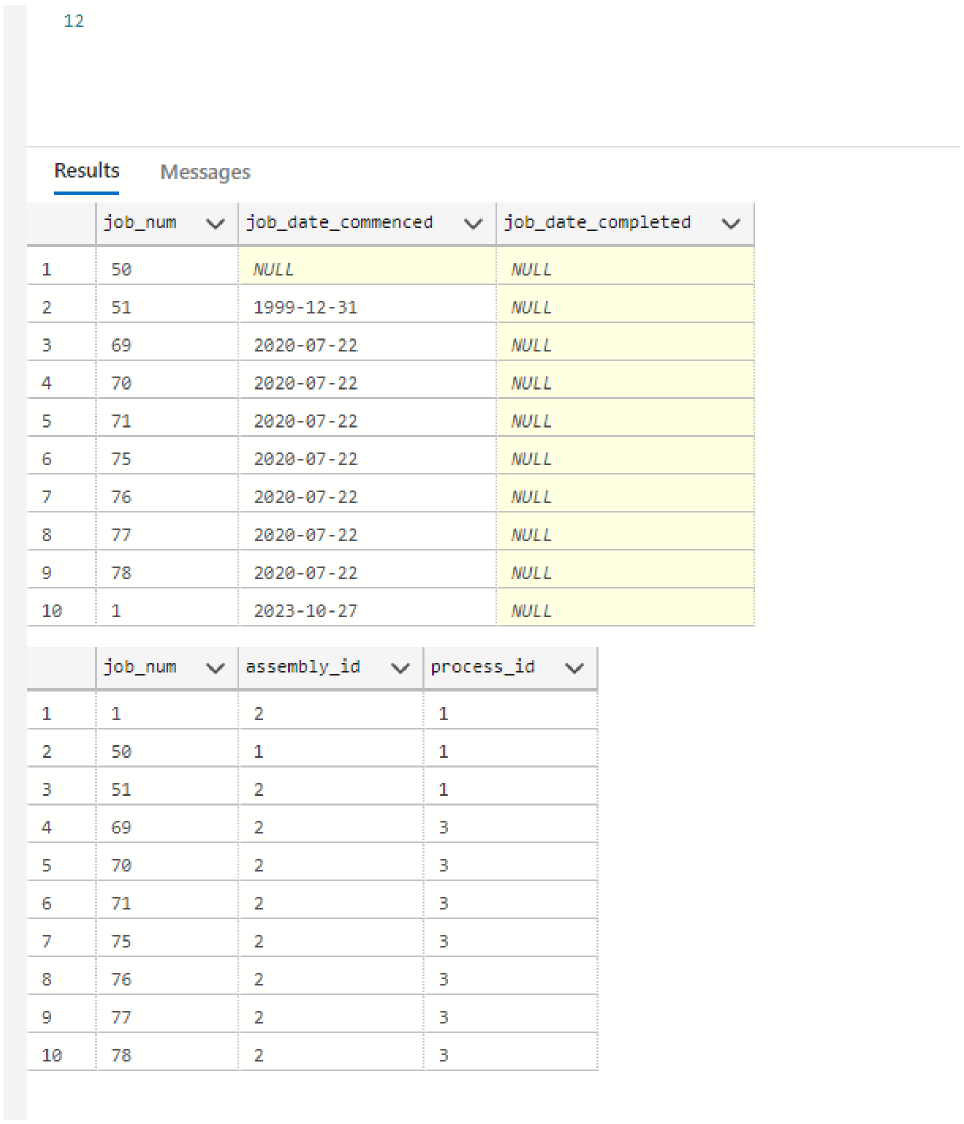

Run Cancel Disconnect Change Database: cs-dsa-4513-sql-db

```
GO
EXEC query6 69,'07/22/2020',2,3
GO

Select *
FROM Jobs


SELECT *
FROM Assign


```

Results Messages

	job_num	job_date_commenced	job_date_completed
1	50	NULL	NULL
2	51	1999-12-31	NULL
3	69	2020-07-22	NULL
4	1	2023-10-27	NULL

	job_num	assembly_id	process_id
1	1	2	1
2	50	1	1
3	51	2	1
4	69	2	3



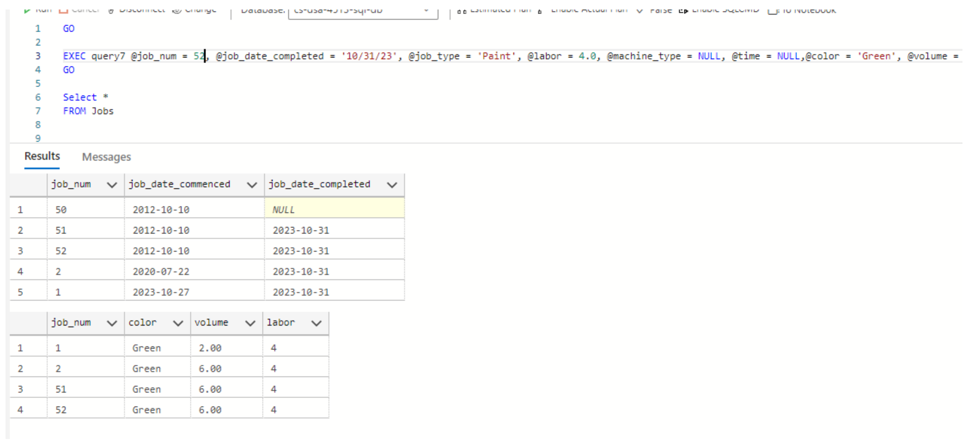
```
12
```

Results Messages

	job_num	job_date_commenced	job_date_completed
1	50	NULL	NULL
2	51	1999-12-31	NULL
3	69	2020-07-22	NULL
4	70	2020-07-22	NULL
5	71	2020-07-22	NULL
6	75	2020-07-22	NULL
7	76	2020-07-22	NULL
8	77	2020-07-22	NULL
9	78	2020-07-22	NULL
10	1	2023-10-27	NULL

	job_num	assembly_id	process_id
1	1	2	1
2	50	1	1
3	51	2	1
4	69	2	3
5	70	2	3
6	71	2	3
7	75	2	3
8	76	2	3
9	77	2	3
10	78	2	3

7.

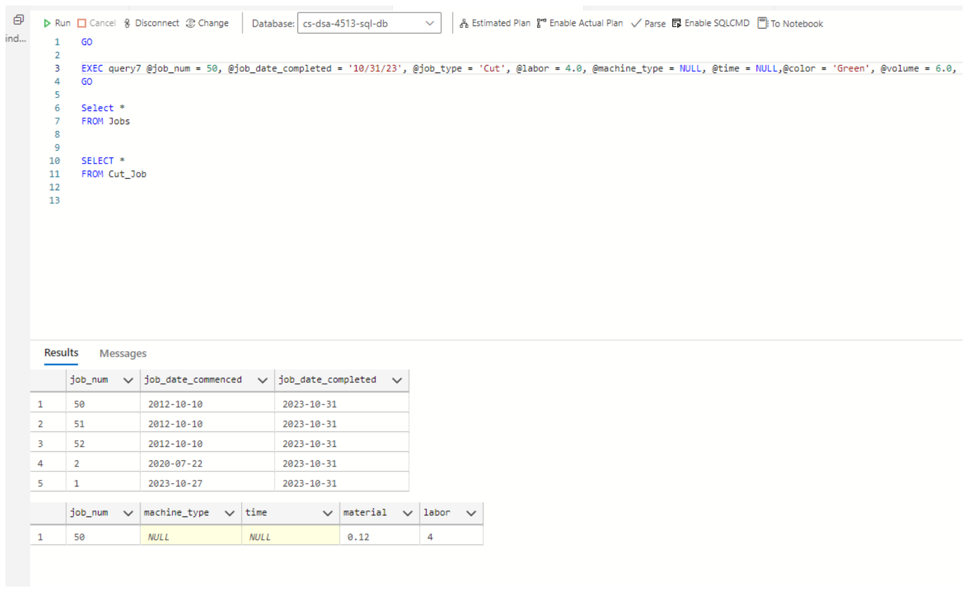


```
GO

EXEC query7 @job_num = 52, @job_date_completed = '10/31/23', @job_type = 'Paint', @labor = 4.0, @machine_type = NULL, @time = NULL,@color = 'Green', @volume =
GO

Select *
FROM Jobs

```

Results Messages

	job_num	job_date_commenced	job_date_completed
1	50	2012-10-10	NULL
2	51	2012-10-10	2023-10-31
3	52	2012-10-10	2023-10-31
4	2	2020-07-22	2023-10-31
5	1	2023-10-27	2023-10-31

	job_num	color	volume	labor
1	1	Green	2.00	4
2	2	Green	6.00	4
3	51	Green	6.00	4
4	52	Green	6.00	4



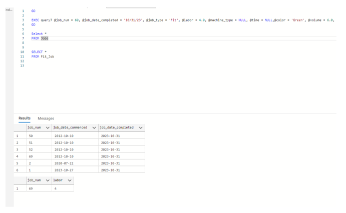
Database: cs-dsa-4513-sql-db

```
GO

EXEC query7 @job_num = 50, @job_date_completed = '10/31/23', @job_type = 'Cut', @labor = 4.0, @machine_type = NULL, @time = NULL,@color = 'Green', @volume = 6.0,
GO

Select *
FROM Jobs


SELECT *
FROM Cut_Job

```

Results Messages

	job_num	job_date_commenced	job_date_completed
1	50	2012-10-10	2023-10-31
2	51	2012-10-10	2023-10-31
3	52	2012-10-10	2023-10-31
4	2	2020-07-22	2023-10-31
5	1	2023-10-27	2023-10-31

	job_num	machine_type	time	material	labor
1	50	NULL	NULL	0.12	4



```
GO

EXEC query7 @job_num = 69, @job_date_completed = '10/31/23', @job_type = 'Fit', @labor = 4.0, @machine_type = NULL, @time = NULL,@color = 'Green', @volume = 6.0,
GO

Select *
FROM Jobs


SELECT *
FROM Fit_Job

```

Results Messages

	job_num	job_date_commenced	job_date_completed
1	50	2012-10-10	2023-10-31
2	51	2012-10-10	2023-10-31
3	52	2012-10-10	2023-10-31
4	69	2012-10-10	2023-10-31
5	2	2020-07-22	2023-10-31
6	1	2023-10-27	2023-10-31

	job_num	labor
1	69	4

8.

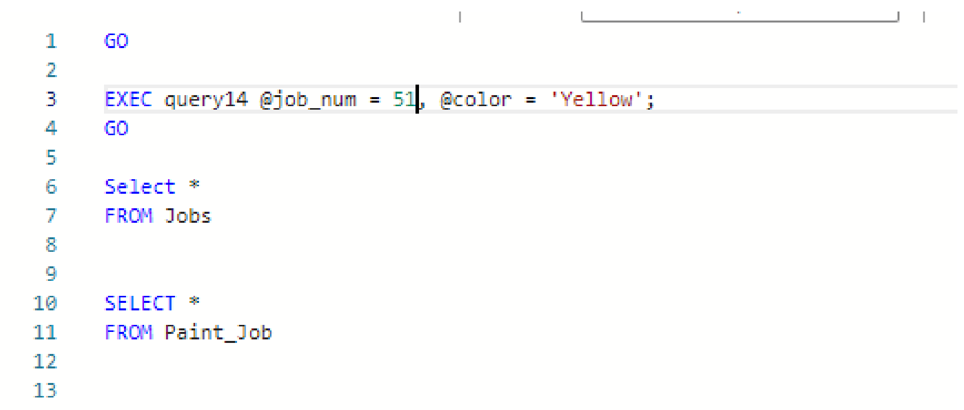
9.

10.

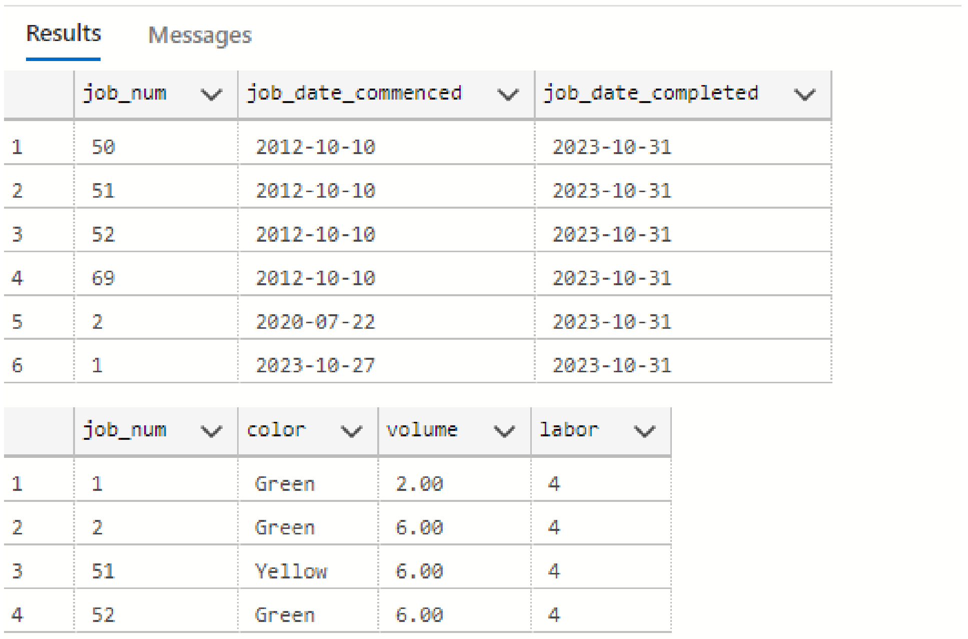
11.

12.

13.



```
GO

EXEC query14 @job_num = 51, @color = 'Yellow';
GO

Select *
FROM Jobs


SELECT *
FROM Paint_Job


```



Results   Messages

	job_num	job_date_commenced	job_date_completed
1	50	2012-10-10	2023-10-31
2	51	2012-10-10	2023-10-31
3	52	2012-10-10	2023-10-31
4	69	2012-10-10	2023-10-31
5	2	2020-07-22	2023-10-31
6	1	2023-10-27	2023-10-31

	job_num	color	volume	labor
1	1	Green	2.00	4
2	2	Green	6.00	4
3	51	Yellow	6.00	4
4	52	Green	6.00	4

14.

5.2 Java Implementation

```
import java.sql.Connection;
import java.sql.Statement;
import java.util.Scanner;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileWriter;
import java.io.IOException;

public class project {

    // Database credentials
    final static String HOSTNAME = "jaco0121-sql-server.database.windows.net";
    final static String DBNAME = "cs-dsa-4513-sql-db";
    final static String USERNAME = "njacob";
    final static String PASSWORD = "";

    // Database connection string
    final static String URL = String.format("jdbc:sqlserver://%s:1433;database=%s;use
            HOSTNAME, DBNAME, USERNAME, PASSWORD);

    // Query templates
    final static String QUERY_TEMPLATE_1 = "EXEC query1 @name = ?, @address = ?, @cat

    final static String QUERY_TEMPLATE_2 = "EXEC query2 @dept_num=?, @dept_data = ?;"

    final static String QUERY_TEMPLATE_3 = "EXEC query3 @process_id=?, @process_data

    final static String QUERY_TEMPLATE_4 = "EXEC query4 @assembly_id=?, @date_ordered

    final static String QUERY_TEMPLATE_5 = "EXEC query5 @acct_id = ?, @type = ?, @dat

    final static String QUERY_TEMPLATE_6 = "EXEC query6 @job_num = ?, @job_date_comme

    final static String QUERY_TEMPLATE_7 = "EXEC query7 @job_num = ?, @job_date_compl
final static String QUERY_TEMPLATE_8 = "EXEC query8 @tran_num =?, @sup_cost = ?,

final static String QUERY_TEMPLATE_9 = "EXEC query9 @assembly_id =?;";//call the

final static String QUERY_TEMPLATE_12 = "EXEC query12 @category =?;";//call the t

final static String QUERY_TEMPLATE_13 = "EXEC query13 @job_num_start =?, @job_num

final static String QUERY_TEMPLATE_14 = "EXEC query14 @job_num =?, @color =?;";//

// User input prompt//
final static String PROMPT =
        "\nPlease select one of the options below: \n" +
        "1) Enter a new customer; \n" +
        "2) Enter a new department; \n" +
        "3) Enter a new process; \n" +
        "4) Enter a new assembly; \n" +
        "5) Create a new account; \n" +
        "6) Enter a new job; \n" +
        "7) Complete a job; \n" +
        "8) Update costs; \n" +
        "9) Print cost on assembly id; \n" +
        "10) Print labor time by department; \n" +
        "11) Print assembly details; \n" +
        "12) Print customers by category; \n" +
        "13) Delete cut jobs; \n" +
        "14) Change color; \n" +
        "15) Import new customers; \n" +
        "16) Export customers by category; \n" +
        "17) Exit!";

public static void main(String[] args) throws SQLException {

    System.out.println("Welcome to my application!");

    final Scanner sc = new Scanner(System.in); // Scanner is used to collect the
    String option = ""; // Initialize user option selection as nothing
    while (!option.equals("17")) { // As user for options until option 17 is sele
        System.out.println(PROMPT); // Print the available options
```

```
option = sc.next(); // Read in the user option selection

switch (option) { // Switch between different options
    case "1": // Insert a new customer
        // Collect the new customer data from the user
        System.out.println("Please enter name for new customer:");
        sc.nextLine();
        final String name = sc.nextLine(); // Read in the user input of p

        System.out.println("Please enter customer address:");
        // Preceding nextInt, nextFloar, etc. do not consume new line cha
        // We call nextLine to consume that newline character, so that su
        //sc.nextLine();
        final String address = sc.nextLine(); // Read in user input of pe

        System.out.println("Please enter integer category for customer:")
        final int category = sc.nextInt(); // Read in the user input of a

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


                System.out.println("Dispatching the query...");
                // Actually execute the populated query
                final int rows_inserted = statement.executeUpdate();
                System.out.println(String.format("Done. %d rows inserted.
            }
            catch (SQLException sqle) {
                    System.out.println("Could not insert customer. " + sq
        }

        break;
    case "2": // Insert a new department
```

```
        // Collect the new department data from the user
        System.out.println("Please enter the department number:");
        sc.nextLine();
        final int dept_num = sc.nextInt(); // Read in the user input of p

        System.out.println("Please enter any department data:");
        // Preceding nextInt, nextFloar, etc. do not consume new line cha
        // We call nextLine to consume that newline character, so that su
        sc.nextLine();
        final String dept_data = sc.nextLine(); // Read in user input of

        System.out.println("Connecting to the database...");
        // Get a database connection and prepare a query statement
        try (final Connection connection = DriverManager.getConnection(UR
            try (
                final PreparedStatement statement = connection.prepareSta
                // Populate the query template with the data collected fr
                statement.setInt(1, dept_num);
                statement.setString(2, dept_data);



                System.out.println("Dispatching the query...");
                // Actually execute the populated query
                final int rows_inserted = statement.executeUpdate();
                System.out.println(String.format("Done. %d rows inserted.
            }
            catch (SQLException sqle) {
                    System.out.println("Could not insert department. " +
        }

        break;
    case "3": // Insert a new process
        // Collect the new process data from the user
        System.out.println("Please enter new process id:");
        sc.nextLine();
        final int process_id = sc.nextInt(); // Read in the user input of

        System.out.println("Please enter process data:");
        // Preceding nextInt, nextFloar, etc. do not consume new line cha
```

```
// We call nextLine to consume that newline character, so that su
sc.nextLine();
final String process_data = sc.nextLine(); // Read in user input

System.out.println("Please enter the type for the process (Fit, P
final String type = sc.nextLine(); // Read in the type
String type_type = null;
String type_method = null;

if (type.equalsIgnoreCase("Fit")) {
        System.out.println("Please enter the fit type:");
        type_type = sc.nextLine();
        type_method = null;
}
else if (type.equalsIgnoreCase("Paint")) {
        System.out.println("Please enter the paint type:");
        type_type = sc.nextLine();
        System.out.println("Please enter the paint method:");
        type_method = sc.nextLine();

}
else if (type.equalsIgnoreCase("Cut")) {
        System.out.println("Please enter the cut type:");
        type_type = sc.nextLine();
        System.out.println("Please enter the cut method:");
        type_method = sc.nextLine();
}
else {
        System.out.println("Why did you not input the type correc
}

System.out.println("Connecting to the database...");
// Get a database connection and prepare a query statement
try (final Connection connection = DriverManager.getConnection(UR
    try (
        final PreparedStatement statement = connection.prepareSta
        // Populate the query template with the data collected fr
        statement.setInt(1, process_id);
        statement.setString(2, process_data);
        statement.setString(3, type);
```

```
                statement.setString(4, type_type);
                statement.setString(5, type_method);


                System.out.println("Dispatching the query...");
                // Actually execute the populated query
                final int rows_inserted = statement.executeUpdate();
                System.out.println(String.format("Done. %d rows inserted.
            }
            catch (SQLException sqle) {
                    System.out.println("Could not insert process. " + sql
        }

        break;
    case "4": // Insert a new assembly
        // Collect the new assembly data from the user
        System.out.println("Please enter an assembly id:");
        sc.nextLine();
        final int assembly_id = sc.nextInt(); // Read in the user input o

        System.out.println("Please enter assembly date ordered in mm/dd/y
        // Preceding nextInt, nextFloar, etc. do not consume new line cha
        // We call nextLine to consume that newline character, so that su
        sc.nextLine();
        final String date_ordered = sc.nextLine(); // Read in user input

        System.out.println("Please enter assembly details:");
        final String assembly_details = sc.nextLine(); // Read in the use

        System.out.println("Please enter customer name:");
        final String name1 = sc.nextLine(); // Read in the user input of

        System.out.println("Please enter the process ids in a comma seper
        final String process_ids = sc.nextLine(); // Read in the user inp

        System.out.println("Connecting to the database...");
        // Get a database connection and prepare a query statement
        try (final Connection connection = DriverManager.getConnection(UR
            try (
                final PreparedStatement statement = connection.prepareSta
                // Populate the query template with the data collected fr
                statement.setInt(1, assembly_id);
                statement.setString(2, date_ordered);
                statement.setString(3, assembly_details);
                statement.setString(4, name1);
                statement.setString(5, process_ids);


                System.out.println("Dispatching the query...");
                // Actually execute the populated query
                final int rows_inserted = statement.executeUpdate();
                System.out.println(String.format("Done. %d rows inserted.
            }
            catch (SQLException sqle) {
                    System.out.println("Could not insert assembly. " + sq
        }

        break;
    case "5": // Insert a new account
        // Collect the new customer data from the user
        System.out.println("Please enter id for new account:");
        sc.nextLine();
        final int acct_id = sc.nextInt(); // Read in the user input of pe

        System.out.println("Please enter account type (Department, Proces
        // Preceding nextInt, nextFloar, etc. do not consume new line cha
        // We call nextLine to consume that newline character, so that su
        //sc.nextLine();
        final String type1 = sc.nextLine(); // Read in user input of perf

        System.out.println("Please enter the id this account references:"
        final int num = sc.nextInt(); // Read in the user input of age

        System.out.println("Please enter date established for this accoun
        final String date_established = sc.nextLine(); // Read in the use

        System.out.println("Connecting to the database...");
        // Get a database connection and prepare a query statement
        try (final Connection connection = DriverManager.getConnection(UR
            try (
                final PreparedStatement statement = connection.prepareSta
                // Populate the query template with the data collected fr
                statement.setInt(1, acct_id);
                statement.setString(2, type1);
                statement.setString(3, date_established);
                statement.setInt(4, num);


                System.out.println("Dispatching the query...");
                // Actually execute the populated query
                final int rows_inserted = statement.executeUpdate();
                System.out.println(String.format("Done. %d rows inserted.
            }
            catch (SQLException sqle) {
                    System.out.println("Could not insert account. " + sql
        }

        break;
    case "6": // Insert a new job
        // Collect the new job data from the user
        System.out.println("Please enter job number:");
        sc.nextLine();
        final int job_num = sc.nextInt(); // Read in the user input of pe

        System.out.println("Please enter date the job commenced:");
        // Preceding nextInt, nextFloar, etc. do not consume new line cha
        // We call nextLine to consume that newline character, so that su
        sc.nextLine();
        final String date_job_commenced = sc.nextLine(); // Read in date.

        System.out.println("Please enter the assembly id:");
        final int assembly_id2 = sc.nextInt(); // Read in the user input

        System.out.println("Please enter process id that starts this asse
        final int process_id2 = sc.nextInt(); // Read in the user input o

        System.out.println("Connecting to the database...");
        // Get a database connection and prepare a query statement
        try (final Connection connection = DriverManager.getConnection(UR
            try (
                final PreparedStatement statement = connection.prepareSta
                // Populate the query template with the data collected fr
                statement.setInt(1, job_num);
                statement.setString(2, date_job_commenced);
                statement.setInt(3, assembly_id2);
                statement.setInt(4, process_id2);


                System.out.println("Dispatching the query...");
                // Actually execute the populated query
                final int rows_inserted = statement.executeUpdate();
                System.out.println(String.format("Done. %d rows inserted.
            }
            catch (SQLException sqle) {
                    System.out.println("Could not insert job. " + sqle);
        }

        break;
    case "7": // End a job
        // Collect the new process data from the user
        System.out.println("Please enter job to end:");
        sc.nextLine();
        final int job_num1 = sc.nextInt(); // Read in the user input of p

        System.out.println("Please enter completion date of job:");
        // Preceding nextInt, nextFloar, etc. do not consume new line cha
        // We call nextLine to consume that newline character, so that su
        sc.nextLine();
        final String job_date_completed = sc.nextLine(); // Read in user

        System.out.println("Please enter the type for the process (Fit, P
        final String job_type = sc.nextLine(); // Read in the type
        double labor = 0.0;
        String machine_type = null;
        double time = 0.0;
        double material = 0.0;
        String color = null;
        double volume = 0.0;

        if (job_type.equalsIgnoreCase("Fit")) {
```

```
        System.out.println("Please enter the labor hours:");
        labor = sc.nextDouble();
}
else if (job_type.equalsIgnoreCase("Paint")) {
        System.out.println("Please enter the labor hours:");
        labor = sc.nextDouble();
        System.out.println("Please enter the paint color:");
        color = sc.nextLine();
        System.out.println("Please enter the paint volume:");
        volume = sc.nextDouble();

}
else if (job_type.equalsIgnoreCase("Cut")) {
        System.out.println("Please enter the labor hours:");
        labor = sc.nextDouble();
        System.out.println("Please enter the machine type:");
        machine_type = sc.nextLine();
        System.out.println("Please enter the time:");
        time = sc.nextDouble();
        System.out.println("Please enter the material:");
        material = sc.nextDouble();
}
else {
        System.out.println("Why did you not input the type correc
}

System.out.println("Connecting to the database...");
// Get a database connection and prepare a query statement
try (final Connection connection = DriverManager.getConnection(UR
    try (
        final PreparedStatement statement = connection.prepareSta
        // Populate the query template with the data collected fr
        statement.setInt(1, job_num1);
        statement.setString(2, job_date_completed);
        statement.setString(3, job_type);
        statement.setDouble(4, labor);
        statement.setString(5, machine_type);
        statement.setDouble(6, time);
        statement.setDouble(7, material);
        statement.setString(8, color);
```

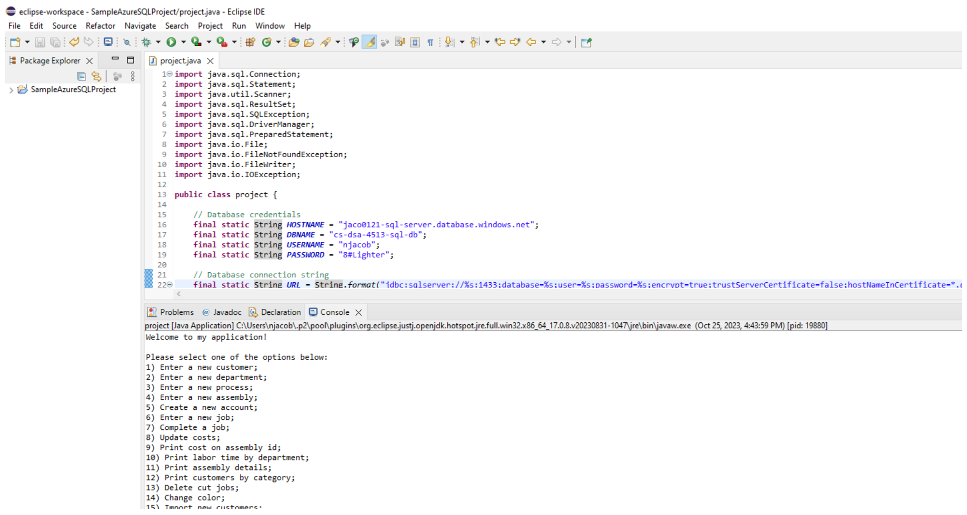
```
                statement.setDouble(9, volume);


                System.out.println("Dispatching the query...");
                // Actually execute the populated query
                final int rows_inserted = statement.executeUpdate();
                System.out.println(String.format("Done. %d rows inserted.
            }
            catch (SQLException sqle) {
                    System.out.println("Could not insert process. " + sql
        }

        break;

    case "8": // Insert a new cost
        // Collect the cost data from the user
        System.out.println("Please enter transaction number:");
        sc.nextLine();
        final int tran_num = sc.nextInt(); // Read in the user input of p

        System.out.println("Please enter the cost for this transaction:")
        // Preceding nextInt, nextFloar, etc. do not consume new line cha
        // We call nextLine to consume that newline character, so that su
        sc.nextLine();
        final double sup_cost = sc.nextDouble(); // Read in date.

        System.out.println("Please enter the job number:");
        final int job_num3 = sc.nextInt(); // Read in the user input asse

        System.out.println("Please enter process for this transaction:");
        final int process_id3 = sc.nextInt(); // Read in the user input o

        System.out.println("Connecting to the database...");
        // Get a database connection and prepare a query statement
        try (final Connection connection = DriverManager.getConnection(UR
            try (
                final PreparedStatement statement = connection.prepareSta
                // Populate the query template with the data collected fr
                statement.setInt(1, tran_num);
                statement.setDouble(2, sup_cost);
                statement.setInt(3, job_num3);
                statement.setInt(4, process_id3);


                System.out.println("Dispatching the query...");
                // Actually execute the populated query
                final int rows_inserted = statement.executeUpdate();
                System.out.println(String.format("Done. %d transaction co
            }
            catch (SQLException sqle) {
                    System.out.println("Could not complete transaction. "
        }

        break;

    case "9":
            System.out.println("Please enter an assembly id:");
        sc.nextLine();
        final int assembly_id3 = sc.nextInt(); // Read in the user input
        // Get the database connection, create statement and execute it r
        try (final Connection connection = DriverManager.getConnection(UR
            System.out.println("Dispatching the query...");
            try (
                    final PreparedStatement statement = connection.prepar
                    // Populate the query template with the data collecte
                    statement.setInt(1, assembly_id3);
                    final ResultSet resultSet = statement.executeQuery();
                    System.out.println(String.format("Costs of Assembly %


                    // Unpack the tuples returned by the database and pri
                    while (resultSet.next()) {
                        System.out.println(String.format("%s",
                            resultSet.getDouble(1)));
                    }
            }
        }

        break;
```

```
case "12":
        System.out.println("Please enter category number:");
    sc.nextLine();
    final int catnum = sc.nextInt(); // Read in the user input of cat
    System.out.println("Connecting to the database...");
    // Get the database connection, create statement and execute it r
    try (final Connection connection = DriverManager.getConnection(UR
        System.out.println("Dispatching the query...");
        final PreparedStatement statement = connection.prepareStateme
            // Populate the query template with the data collected fr
            statement.setInt(1, catnum);


            //System.out.println("Dispatching the query...");
            // Actually execute the populated query
            final ResultSet resultSet = statement.executeQuery();



                System.out.println("Contents of the Customer table:")
                System.out.println("name");

                // Unpack the tuples returned by the database and pri
                while (resultSet.next()) {
                    System.out.println(String.format("%s",
                        resultSet.getString(1)));
                }
        }
        }


    break;


case "13": // delete cut jobs
    // Collect the new customer data from the user
    System.out.println("Please enter start number for range of cut jo
    sc.nextLine();
    final int job_num_start = sc.nextInt(); // Read in the user input
```

```
        System.out.println("Please enter end number for range of cut jobs
        final int job_num_end = sc.nextInt(); // Read in the user input o

        System.out.println("Connecting to the database...");
        // Get a database connection and prepare a query statement
        try (final Connection connection = DriverManager.getConnection(UR
            try (
                final PreparedStatement statement = connection.prepareSta
                // Populate the query template with the data collected fr
                statement.setInt(1, job_num_start);
                statement.setInt(2, job_num_end);


                System.out.println("Dispatching the query...");
                // Actually execute the populated query
                final int rows_inserted = statement.executeUpdate();
                System.out.println(String.format("Done. %d rows deleted."
            }
            catch (SQLException sqle) {
                    System.out.println("Could not delete rows. " + sqle);
        }
        break;
    case "14": // update paint job
        // Collect the new customer data from the user
        System.out.println("Please enter job number for paint job:");
        sc.nextLine();
        final int job_num2 = sc.nextInt(); // Read in the user input of p

        System.out.println("Please enter the new color:");
        sc.nextLine();
        final String color1 = sc.nextLine(); // Read in the user input of

        System.out.println("Connecting to the database...");
        // Get a database connection and prepare a query statement
        try (final Connection connection = DriverManager.getConnection(UR
            try (
                final PreparedStatement statement = connection.prepareSta
                // Populate the query template with the data collected fr
                statement.setInt(1, job_num2);
                statement.setString(2, color1);
                System.out.println("Dispatching the query...");
                // Actually execute the populated query
                final int rows_inserted = statement.executeUpdate();
                System.out.println(String.format("Done. %d rows modified.
            }
            catch (SQLException sqle) {
                    System.out.println("Could not modify rows. " + sqle);
        }
        break;


    case "15":
            System.out.println("Enter path for file to input:");
            sc.nextLine();
            final String pathtofile = sc.nextLine();
            File file = new File(pathtofile);
            try (Scanner scanfile = new Scanner(file)){
            while (scanfile.hasNextLine())
            try (final Connection connection = DriverManager.getConnectio
                try (
                    final PreparedStatement statement = connection.prepar
                    // Populate the query template with the data collecte
                    statement.setString(1, scanfile.next());
                    statement.setString(2, scanfile.next());
                    statement.setString(3, scanfile.next());


                    System.out.println("Dispatching the query...");
                    // Actually execute the populated query
                    final int rows_inserted = statement.executeUpdate();
                    System.out.println(String.format("Done. %d row insert
                    //System.out.println(scanfile.nextLine());
            }}} catch (FileNotFoundException e) {
                    System.out.println("File not found");
            }
            break;

    case "16":
```

```
            System.out.println("Enter path for export file:");
            sc.nextLine();
            final String pathtofile2 = sc.nextLine();
        try {
            FileWriter myWriter = new FileWriter(pathtofile2);
            try (final Connection connection = DriverManager.getConnectio
            try (
                    final Statement statement = connection.createStatemen
                    final ResultSet resultSet = statement.executeQuery("S

                    while (resultSet.next()) {
                    myWriter.write(String.format("%s  %s  %s %n",
                        resultSet.getString(1),
                        resultSet.getString(2),
                        resultSet.getString(3)));
                }

            }
            //myWriter.write("Files in Java might be tricky, but it is fu
            myWriter.close();
            System.out.println("Successfully wrote to the file.");}
          } catch (IOException e) {
            System.out.println("An error occurred.");
            e.printStackTrace();
          }
            break;
    case "33":
        System.out.println("Connecting to the database...");
        // Get the database connection, create statement and execute it r
        try (final Connection connection = DriverManager.getConnection(UR
            System.out.println("Dispatching the query...");
            try (
                final Statement statement = connection.createStatement();
                final ResultSet resultSet = statement.executeQuery("SELEC

                    System.out.println("Contents of the Performer table:"
                    System.out.println("ID | name | years of experience |

                    // Unpack the tuples returned by the database and pri
                    while (resultSet.next()) {
```

```
                                    System.out.println(String.format("%s | %s | %s |
                                        resultSet.getString(1),
                                        resultSet.getString(2),
                                        resultSet.getString(3),
                                        resultSet.getString(4)));
                                }
                        }
                    }

                    break;
                case "17": // Do nothing, the while loop will terminate upon the next
                    System.out.println("Exiting! Good-bye!");
                    break;
                default: // Unrecognized option, re-prompt the user for the correct o
                    System.out.println(String.format(
                        "Unrecognized option: %s\n" +
                        "Please try again!",
                        option));
                    break;
            }
        }

        sc.close(); // Close the scanner before exiting the application
    }
}
```



```
import java.sql.Connection;
import java.sql.Statement;
import java.util.Scanner;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileWriter;
import java.io.IOException;

public class project {

    // Database credentials
    final static String HOSTNAME = "jaco0121-sql-server.database.windows.net";
    final static String DBNAME = "cs-dsa-4513-sql-db";
    final static String USERNAME = "njacob";
    final static String PASSWORD = "8#Lighter";

    // Database connection string
    final static String URL = String.format("jdbc:sqlserver://%s:1433;database=%s;user=%s;password=%s;encrypt=true;trustServerCertificate=false;hostNameInCertificate=*.
```

```
project [Java Application] C:\Users\njacob\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.8.v20230831-1047\jre\bin\javaw.exe (Oct 25, 2023, 4:43:59 PM) [pid: 19880]
Welcome to my application!

Please select one of the options below:
1) Enter a new customer;
2) Enter a new department;
3) Enter a new process;
4) Enter a new assembly;
5) Create a new account;
6) Enter a new job;
7) Complete a job;
8) Update costs;
9) Print cost on assembly id;
10) Print labor time by department;
11) Print assembly details;
12) Print customers by category;
13) Delete cut jobs;
14) Change color;
15) Import new customers;
```

6 Java Execution

6.1 Query 1

6.2 Query 2

6.3 Query 3

6.4 Query 4

6.5 Query 5

6.6 Query 6

6.7 Query 7

6.8 Query 8

6.9 Query 9

6.10 Query 10

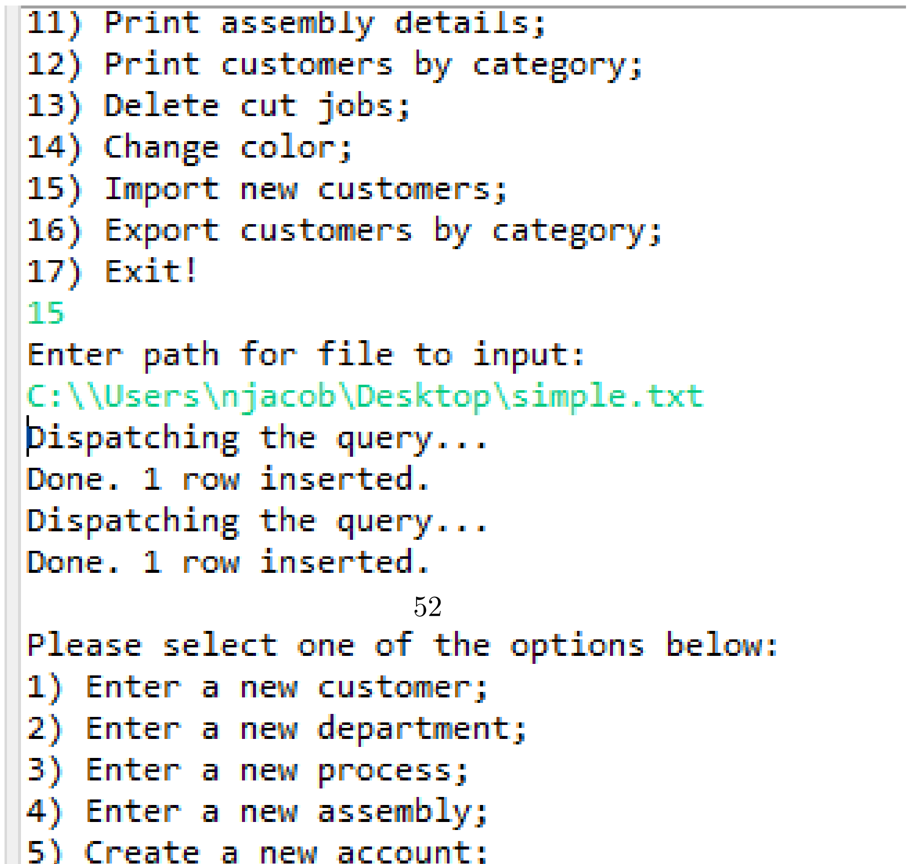
6.11 Query 11

6.12 Query 12

6.13 Query 13

6.14 Query 14

6.15 Query Import/Export

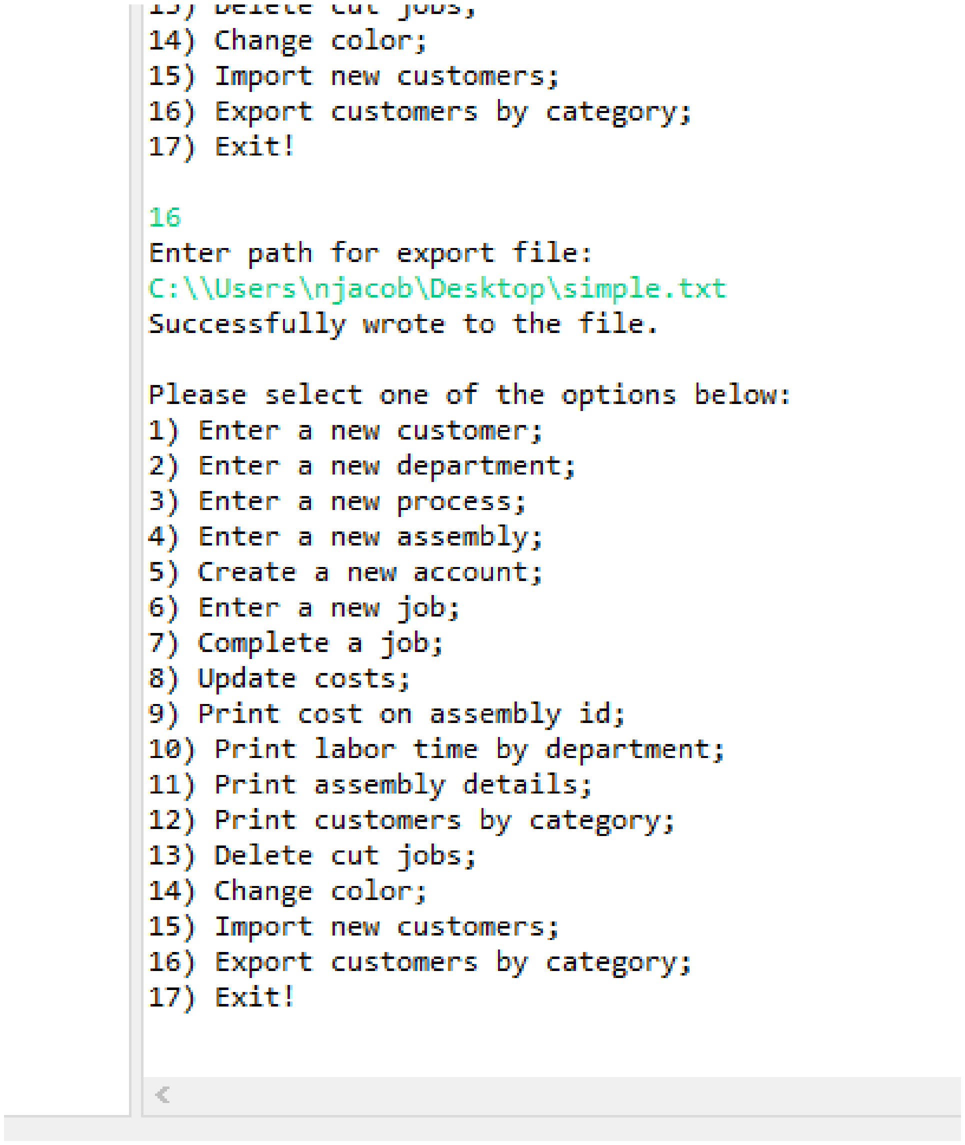


```
11) Print assembly details;
12) Print customers by category;
13) Delete cut jobs;
14) Change color;
15) Import new customers;
16) Export customers by category;
17) Exit!
15
Enter path for file to input:
C:\\Users\njacob\Desktop\simple.txt
Dispatching the query...
Done. 1 row inserted.
Dispatching the query...
Done. 1 row inserted.
```

```
Please select one of the options below:
1) Enter a new customer;
2) Enter a new department;
3) Enter a new process;
4) Enter a new assembly;
5) Create a new account;
```

Here is what is started with

```
Jimmy  701King  8
John  123Fake  9
```



```
14) Change color;
15) Import new customers;
16) Export customers by category;
17) Exit!

16
Enter path for export file:
C:\\Users\njacob\Desktop\simple.txt
Successfully wrote to the file.

Please select one of the options below:
1) Enter a new customer;
2) Enter a new department;
3) Enter a new process;
4) Enter a new assembly;
5) Create a new account;
6) Enter a new job;
7) Complete a job;
8) Update costs;
9) Print cost on assembly id;
10) Print labor time by department;
11) Print assembly details;
12) Print customers by category;
13) Delete cut jobs;
14) Change color;
15) Import new customers;
16) Export customers by category;
17) Exit!
```

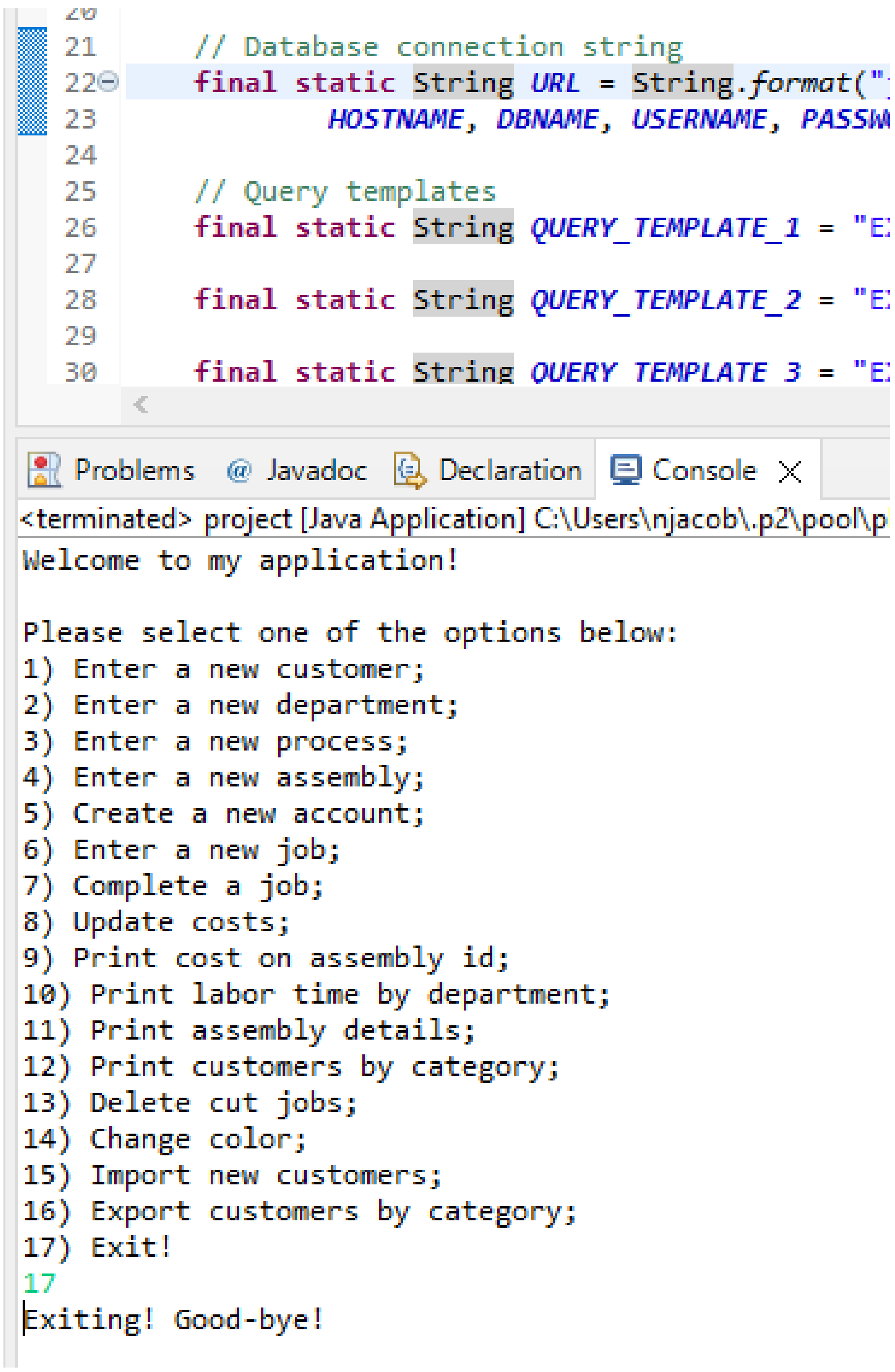
Here is the file it printed

```
Elle  123FakeStreet  9
Gus  701Kings  10
```

```
Jimmy  701King  8
John  123Fake  9
Nick  701Kings  10
```

6.16 Errors

6.17 Quitting



```
    // Database connection string
    final static String URL = String.format("
            HOSTNAME, DBNAME, USERNAME, PASSW

    // Query templates
    final static String QUERY_TEMPLATE_1 = "E

    final static String QUERY_TEMPLATE_2 = "E

    final static String QUERY_TEMPLATE_3 = "E
```

```
<terminated> project [Java Application] C:\Users\njacob\.p2\pool\p
Welcome to my application!

Please select one of the options below:
1) Enter a new customer;
2) Enter a new department;
3) Enter a new process;
4) Enter a new assembly;
5) Create a new account;
6) Enter a new job;
7) Complete a job;
8) Update costs;
9) Print cost on assembly id;
10) Print labor time by department;
11) Print assembly details;
12) Print customers by category;
13) Delete cut jobs;
14) Change color;
15) Import new customers;
16) Export customers by category;
17) Exit!
17
Exiting! Good-bye!
```

7 Web Database

7.1 Source Code

Data Handler

```
package jsp_azure_test;

import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.DriverManager;
import java.sql.PreparedStatement;

public class DataHandler {

    private Connection conn;

    // Azure SQL connection credentials
    final static String server = "jaco0121-sql-server.database.windows.net";
    final static String database = "cs-dsa-4513-sql-db";
    final static String username = "njacob";
    final static String password = "";

    // Resulting connection string
    final private String url =
            String.format("jdbc:sqlserver://%s:1433;database=%s;user=%s;password=%s;e
                    server, database, username, password);

    // Initialize and save the database connection

    private void getDBConnection() throws SQLException {
        if (conn != null) {
            return;
        }

        this.conn = DriverManager.getConnection(url);
    }


    // Add a customer to the table
    public boolean addCustomer(
            String cname, String address, int category) throws SQLException {

        getDBConnection(); // Prepare the database connection

        // Prepare the SQL statement
        final String sqlQuery =
                "INSERT INTO Customer " +
                    "(name, address, category) " +
                "VALUES " +
                "(?, ?, ?)";
        final PreparedStatement stmt = conn.prepareStatement(sqlQuery);

        // Replace the '?' in the above statement with the given attribute values
        stmt.setString(1, cname);
        stmt.setString(2, address);
        stmt.setInt(3, category);


        // Execute the query, if only one record is updated, then we indicate success
        return stmt.executeUpdate() == 1;
    }

// Return the result of selecting all customers based on category
public ResultSet getAllCustomers(int category) throws SQLException {
    getDBConnection();

    final String sqlQuery = "SELECT * FROM Customer WHERE category = ?;";
    final PreparedStatement stmt = conn.prepareStatement(sqlQuery);
    stmt.setInt(1, category);

    return stmt.executeQuery();
        }
}
```

Add customer

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
```

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Query Result</title>
</head>
    <body>
    <%@page import="jsp_azure_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 startTime = request.getParameter("cname");
    String movieName = request.getParameter("address");
    String durationString = request.getParameter("category");


    /*
     * If the user hasn't filled out all the time, movie name and duration. This is v
     */
    if (startTime.equals("") || movieName.equals("") || durationString.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.addCustomer(startTime, movieName, duration);
        if (!success) { // Something went wrong
            %>
                <h2>There was a problem inserting the customer</h2>
            <%
        } else { // Confirm success to the user
            %>
            <h2>The Customer:</h2>

            <ul>
                <li>Customer Name: <%=startTime%></li>
                <li>Address: <%=movieName%></li>
                <li>Category: <%=durationString%></li>

            </ul>

            <h2>Was successfully inserted.</h2>

            <a href="get_all_customers_form.jsp">See all customers.</a>
            <%
        }
    }
    %>
    </body>
</html>
```

Add customer form

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

Get all customers

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
    <meta charset="UTF-8">
        <title>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
            final DataHandler handler = new DataHandler();
```

```
                String categoryString = request.getParameter("category");
                int category = Integer.parseInt(categoryString);
            final ResultSet movies = handler.getAllCustomers(category);
        %>
        <!-- The table for displaying all the movie records -->
        <table cellspacing="2" cellpadding="2" border="1">
            <tr> <!-- The table headers row -->
              <td align="center">
                <h4>Customer 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 time = movies.getString("name");
                   final String name = movies.getString("address");
                   final String duration = movies.getString("category");


                   out.println("<tr>"); // Start printing out the new table row
                   out.println( // Print each attribute value
                        "<td align=\"center\">" + time +
                        "</td><td align=\"center\"> " + name +
                        "</td><td align=\"center\"> " + duration + "</td>");
                   out.println("</tr>");
               }
               %>
          </table>
                     <a href="add_customer_form.jsp">Add another customers.</a>
    </body>
</html>
```

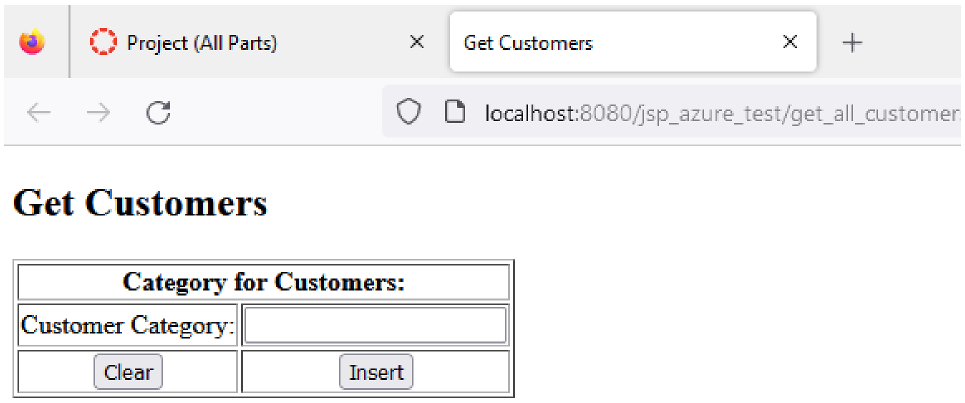
Get all customers form

```
<!DOCTYPE html>
```

```
<html>
    <head>
        <meta charset="UTF-8">
        <title>Get Customers</title>
    </head>
    <body>
        <h2>Get Customers</h2>
        <!--
            Form for collecting user input for the new movie_night record.
            Upon form submission, add_movie.jsp file will be invoked.
        -->
        <form action="get_all_customers.jsp">
            <!-- The form organized in an HTML table for better clarity. -->
            <table border=1>
                <tr>
                    <th colspan="2">Category for Customers:</th>
                </tr>
                <tr>
                    <td>Customer 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>
```

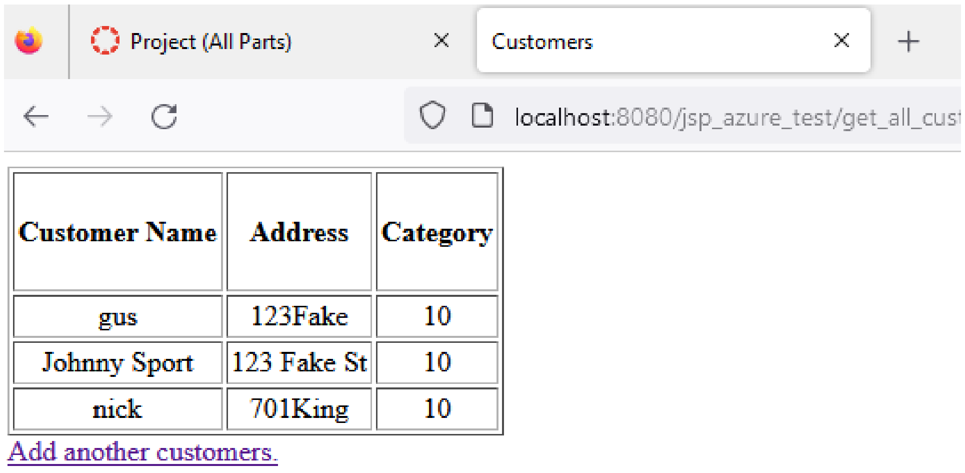
7.2 Screenshots

Holy cow I didn't think I would get this to work!



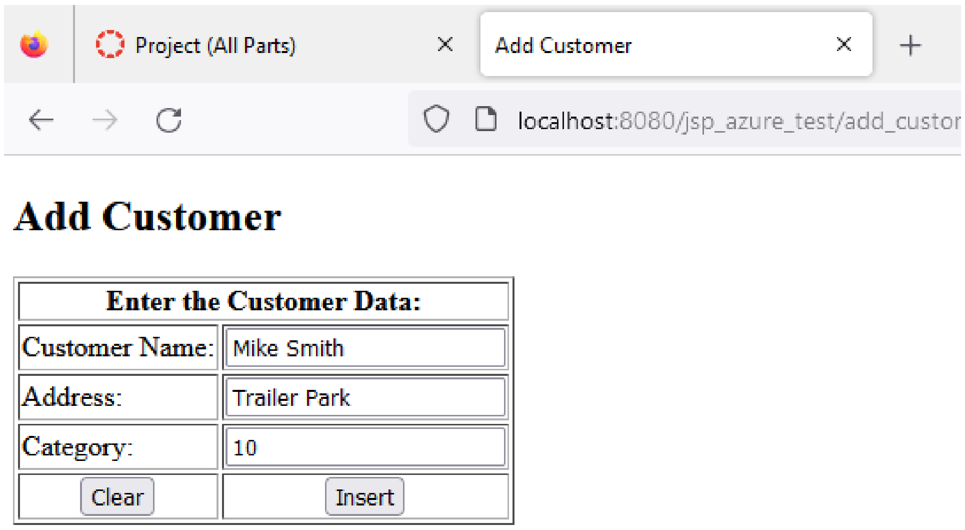
Get Customers

Category for Customers:	
Customer Category:	
Clear	Insert



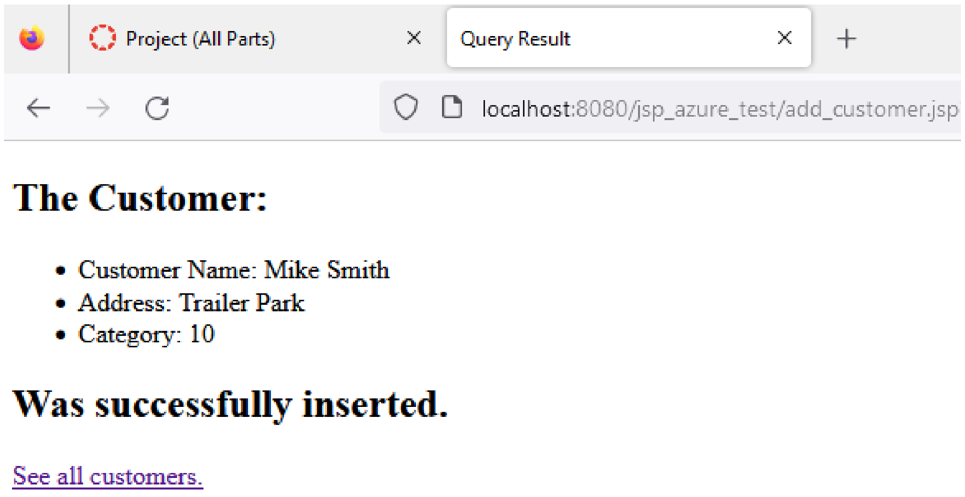
Customer Name	Address	Category
gus	123Fake	10
Johnny Sport	123 Fake St	10
nick	701King	10

Add another customers.



Add Customer

Enter the Customer Data:	
Customer Name:	Mike Smith
Address:	Trailer Park
Category:	10
Clear	Insert

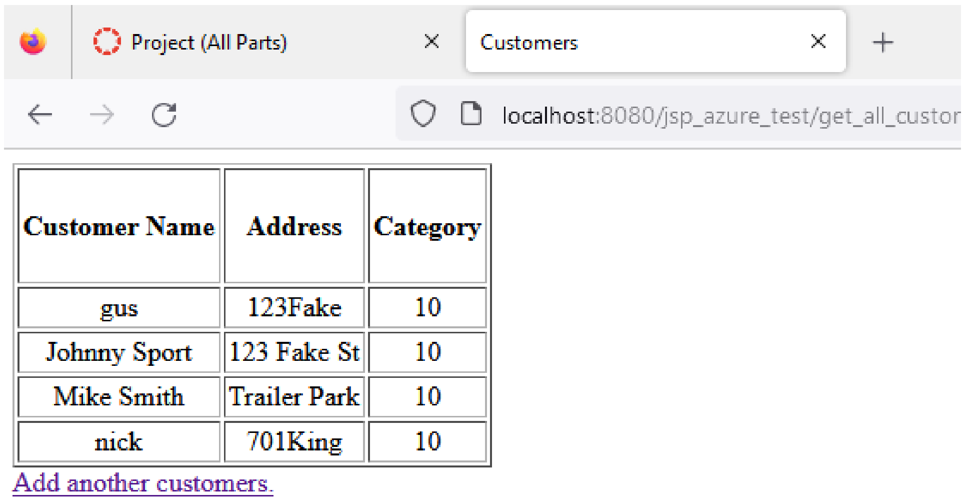


The Customer:

- Customer Name: Mike Smith
- Address: Trailer Park
- Category: 10

Was successfully inserted.

See all customers.



Customer Name	Address	Category
gus	123Fake	10
Johnny Sport	123 Fake St	10
Mike Smith	Trailer Park	10
nick	701King	10

Add another customers.