COURSE: CS/DSA-4513-DATABASE MANAGEMENT SYSTEMS

SECTION: 001

SEMESTER: FALL 2023

INSTRUCTOR'S NAME: LE GRUENWALD

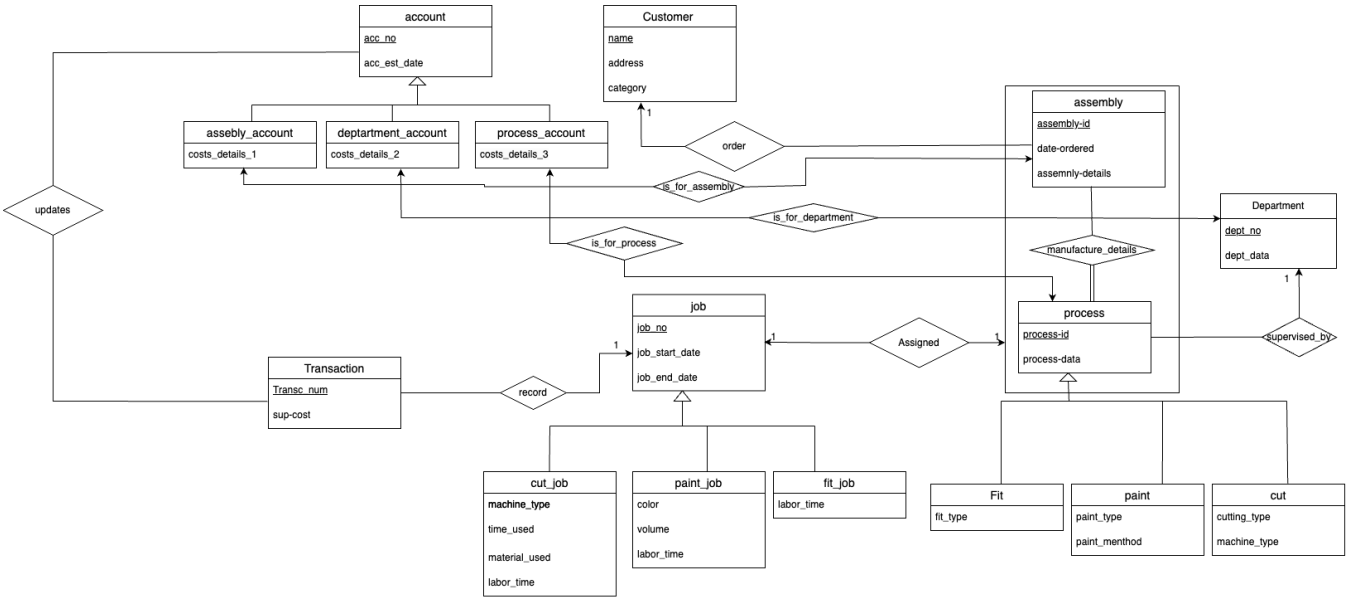
AUTHOR'S NAME: Rama Dinesh Kumar

AUTHOR'S ID: 113582682

EMAIL ADDRESS: Dinesh.Kumar.Rama-1@ou.edu

PROJECT TITLE: JOB-SHOP ACCOUNTING DATABASE SYSTEM

Task 1: ER - Diagram for Job-shop Accounting System:



Task 2: Relational Database Schemas:

customer (name , address, category)

assembly (name, assembly_id, date-ordered, assembly-details)

manufacture_details (assembly_id, process_id, job_no)

process (process_id, process-data, dept_no)

fit (process_id, fit_type)

paint (process_id, paint_type, paint_method)

cut (process_id, cutting_type, machine_type)

department (dept_no, dept_data)

job (job_no, job_start_date, job_end_date)

cut_job (job_no , machine_type, time_used, material_used, labor_time)

paint_job (job_no, colour, volume, labor_time)

fit_job (job_no, labor_time)

transaction_data (Transc_num, sup-cost, job_no)

updates(acc_no, Transc_num)

account (aac_no, acc_est_date)

assembly_account (acc_no, cost_details_1, assembly_id)

department_account (acc_no, cost_details_2, dept_no)

process_account (acc_no, cost_details_3, process_id)

Task 3:

3.1 Discussion of storage structures for tables

Table Name	Query # And Type	Search Key	Query Frequency	Selected File Organization	Justifications
customer	QUERY 1) Insert QUERY 12) Range	 category	30/d 100/d	B+ tree (Secondary Index) on category	B+ tree supports all types of queries efficiently. Range searches are highly efficient due to the linked leaves in B+ trees. The secondary index on 'category' enables faster lookups for range queries, which are the most frequent.
Account	QUERY 5) Insert QUERY 8) Random	 acc_no	10/d 50/d	Hash file on acc_no	with a high frequency of random searches hash file is a suitable choice
department	QUERY 2) Insert QUERY 3) Random QUERY 5) Random QUERY 11) Random	 dept_no dept_no dept_no	Infrequent Infrequent 10/d 100/d	Hash file on dept_no	With the high frequency of random searches on dept_no, a hash file is effective because hashing provides fast access for random search queries
Process	QUERY 3) Insert QUERY 5) Random QUERY 6) Random	 dept_no process_id	Infrequent 50/d 10/d 20/d	B+ tree Primary index on process_id And Secondary index on dept_no	B+ tree is a good choice here because there is a mix of query types. The primary index on process_id allows efficient insertions and random searches, while the secondary index on dept_no acco any queries that involve department-related searches.

	QUERY 10) Random QUERY 11) Random		100/d		
fit	QUERY 3) Insert		Infrequent	Heap file	I chose a heap file here because there is only insertion operation making heap file suitable for it.
cut	QUERY 3) Insert		Infrequent	Heap file	I chose a heap file here because there is only insertion operation making heap file suitable for it.
paint	QUERY 3) Insert		Infrequent	Heap file	I chose a heap file here because there is only insertion operation making heap file suitable for it.
assembly	QUERY 4)Insert QUERY 5)Random QUERY6)Random	 assembly_id assembly_id	40/d 10/d 50/d	Hash file on assembly_id	Hashing provides very fast access for random searches and is best for equality searches on assembly_id.
Job	QUERY 6) Insert QUERY 7) Random QUERY 10) random QUERY 11) order by	 job_no date_completed date_commenced	50/d 50/d 20/d 100/d	B+ tree on job_no	B+ tree is suitable because it efficiently handles and provides fast access for the high frequency of insertions and random searches on job_no, as well as 'order by' queries on dates.
cut_job	QUERY 7) Insert QUERY 13) Range QUERY 10) Random	 job_no job_no	50/d 1/m 20/d	B+ tree (on job_no)	I have chosen B+ tree because it is efficient for the range and random access queries involving the job_no search key, while also handling frequent insert effectively, making it suitable for both the query types and their frequencies.

paint_job	QUERY 7) Insert QUERY 14) random QUERY 10) random	 job_no job_no	50/d 1/w 20/d	Hash file on job_no	Since there are no range queries and a high frequency of inserts, a hash file is suitable.
fit_job	QUERY 7)Insert QUERY 10)Random	 job_no	50/d 20/d	B+ Tree	B+ tree is well-suited for the 'fit_job' table as it balances both the frequent inserts and efficient random access queries.
manufacture_details	QUERY 4) Insert QUERY 6) Random QUERY 10) Random QUERY 11) Random	 job_no assembly_id	40/d 50/d 20/d 100/d	B+ Tree on job_no(secondary index) And primary index on assembly_id	Here I chose B+ tree because it is best for handling both insert and random queries.
dept_account	QUERY 5) Insert QUERY 8) Random	 acc_no		Hash file on acc_no	with a high frequency of random searches hash file is a suitable choice
process_account	QUERY 5) Insert QUERY 8) Random	 acc_no		Hash file on acc_no	with a high frequency of random searches hash file is a suitable choice
assembly_account	QUERY 5) Insert QUERY 8) Random QUERY 9) Random	 acc_no assembly_id	10/d 50/d 200/d	B+ tree on assembly_id	Here I chose B+ tree because B+ tree handles multiple search keys efficiently.
Transaction	QUERY 8) Insert		50/d	Heap file	Heap files are efficient for tables with a high frequency of inserts

Updates	QUERY 8) Insert		50/d	Heap file	Here I chose heap file because it is efficient for insertions.

3.2 Discuss choices of storage structures for each relational table when implementing it in Azure SQL Database (if different from the previous choices specified in 3.1).

Table Name	Selected File Organizationin 3.1	Selected File Organization in Azure SQL Database
customer	B+ tree (Secondary Index) on category	B+ tree Primary indexing on primary key is the default indexing in azure sql database. But here I am creating a secondary indexing on category No Changes here.
assembly	Hash file on assembly_id	In Azure sql database Hash file is not supported. Since i am creating hash file on assembly_id the azure will create B+ tree index on assembly_id.
department	Hash file on dept_no	In Azure sql database Hash file is not supported. Since i am creating hash file on dept_no the azure will create B+ tree index on dept_no.
Job	B+ tree on job_no	B+ tree Primary indexing on primary key is the default indexing in azure sql database. Here it is job_no. So, no change here.
cut_job	B+ tree on job_no	I B+ tree Primary indexing on primary key is the default indexing in azure sql database. Here the indexing is on job_no. So, No cahnges here
paint_job	Hash file on job_no	In Azure sql database Hash file is not supported. Since i am creating hash file on job_no the azure will create B+ tree index on job_no.

fit_job	B+ Tree	B+ tree Primary indexing on primary key is the default indexing in azure sql database. So, No changes here
Process	B+ tree Primary on process_id Secondary index on dept_no	B+ tree Primary indexing on primary key is the default indexing in azure sql database. But I need to create secondary indexing on dept_no.
fit	Heap file	Heap file is accepted by the azure sql database. So, no changes here.
cut	Heap file	Heap file is accepted by the azure sql database. So, no changes here.
paint	Heap file	Heap file is accepted by the azure sql database. So, no changes here.
Transaction	Heap file	Heap file is accepted by the azure sql database. So, no changes here.
Account	Hash file on acc_no	In Azure sql database Hash file is not supported. Since i am creating hash file on acc_no the azure will create B+ tree index on acc_no.
dept_account	Hash file on acc_no	In Azure sql database Hash file is not supported. Since i am creating hash file on acc_no the azure will create B+ tree index on acc_no.
process_account	Hash file on acc_no	In Azure sql database Hash file is not supported. Since i am creating hash file on acc_no the azure will create B+ tree index on acc_no.
assembly_account	B+ tree on assembly_id	B+ tree Primary indexing on primary key is the default indexing in azure sql database. Here, the indexing is on assembly_id
manufacture_details	B+ Tree on job_no(secondary index) And primary index on assembly_id	B+ tree Primary indexing on primary key is the default indexing in azure sql database. Here, the primary indexing is on assembly_id. And secondary indexing is on job_no.

Updates	Heap file	Heap file is accepted by the azure sql database. So, no changes here.

Task 4:

SQL statements and screenshots showing the creation of tables in Azure SQL Database

```
--DROP STATEMENTS IF TABLE EXISTS


DROP TABLE if exists updates;
DROP TABLE if exists transaction_data;
DROP TABLE if exists assembly_account;
DROP TABLE if exists department_account;
DROP TABLE if exists process_account;
DROP TABLE if exists manufacture_details;
DROP TABLE if exists fit_job;
DROP TABLE if exists paint_job;
DROP TABLE if exists cut_job;
DROP TABLE if exists job;
DROP TABLE if exists assembly;
DROP TABLE if exists fit;
DROP TABLE if exists paint;
DROP TABLE if exists cut;
DROP TABLE if exists process;
DROP TABLE if exists department;
DROP TABLE if exists account;
DROP TABLE if exists customer;



-- CREATE TABLES


-- CREATE customer TABLE
```

```
CREATE TABLE customer (
name VARCHAR(255) PRIMARY KEY,
address VARCHAR(255),
category INT
);
-- creating non clustered B+ index on category in customer table.
create nonclustered index idx_category on customer(category);


-- CREATE account TABLE
CREATE TABLE account (
acc_no INT PRIMARY KEY,
acc_est_date DATE
);
-- Since clustered index is created by default i am commenting it
--create clustered index idx_acc_no on account(account_no);


-- CREATE department TABLE
CREATE TABLE department (
dept_no INT PRIMARY KEY,
dept_data VARCHAR(255)
);


-- Since clustered index is created by default i am commenting it
-- create clustered index idx_dept_no on department(dept_no);


-- CREATE process TABLE
CREATE TABLE process (
process_id INT PRIMARY KEY,
process_data VARCHAR(255),
dept_no INT,
FOREIGN KEY (dept_no) REFERENCES department(dept_no)
);
create nonclustered index idx_process_dept_no on process(dept_no);


-- CREATE cut TABLE
CREATE TABLE cut (
process_id INT PRIMARY KEY nonclustered,
cutting_type VARCHAR(255),
machine_type VARCHAR(255),
FOREIGN KEY (process_id) REFERENCES process(process_id)
```

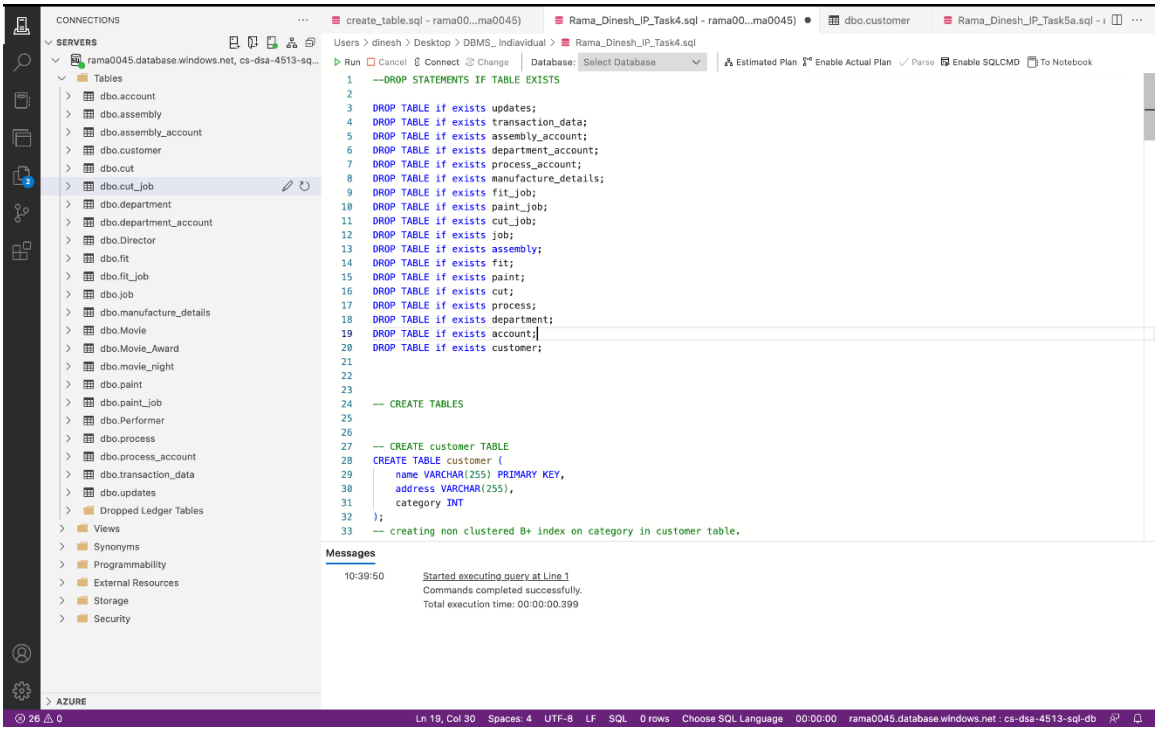
```
);


-- CREATE paint TABLE
CREATE TABLE paint (
process_id INT PRIMARY KEY nonclustered,
paint_type VARCHAR(255),
paint_method VARCHAR(255),
FOREIGN KEY (process_id) REFERENCES process(process_id)
);


-- CREATE fit TABLE
CREATE TABLE fit (
process_id INT PRIMARY KEY nonclustered,
fit_type VARCHAR(255),
FOREIGN KEY (process_id) REFERENCES process(process_id)
);


-- CREATE assembly TABLE
CREATE TABLE assembly (
assembly_id INT PRIMARY KEY,
name VARCHAR(255),
date_ordered DATE,
assembly_details VARCHAR(255),
FOREIGN KEY (name) REFERENCES customer(name)
);


-- Since clustered index is created i am commenting it out
--create clustered index idx_assembly_id on assembly(assembly_id);


-- CREATE job TABLE
CREATE TABLE job (
job_no INT PRIMARY KEY,
job_start_date DATE,
job_end_date DATE
);
create nonclustered index idx_date_ended on job(job_end_date);


-- CREATE cut_job TABLE
CREATE TABLE cut_job (
```

```
job_no INT PRIMARY KEY,
machine_type VARCHAR(255),
time_used INT,
material_used VARCHAR(255),
labor_time INT,
FOREIGN KEY (job_no) REFERENCES job(job_no)
);


-- Since clustered index is created default i am commenting it
--create clustered index idx_cut_job_no on cut_job(job_no);


-- CREATE paint_job TABLE
CREATE TABLE paint_job (
job_no INT PRIMARY KEY,
colour VARCHAR(255),
volume VARCHAR(255),
labor_time INT,
FOREIGN KEY (job_no) REFERENCES job(job_no)
);


-- Since clustered index is created by default i am commenting it
--create clustered index idx_paint_job_no on paint_job(job_no);


-- CREATE fit_job TABLE
CREATE TABLE fit_job (
job_no INT PRIMARY KEY,
labor_time INT,
FOREIGN KEY (job_no) REFERENCES job(job_no)
);


-- Since clustered index is created by default i am commenting it
--create clustered index idx_fit_job_no on fit_job(job_no);


-- CREATE manufacture_details TABLE
CREATE TABLE manufacture_details (
assembly_id INT,
process_id INT,
job_no INT,
PRIMARY KEY (assembly_id, process_id),
```

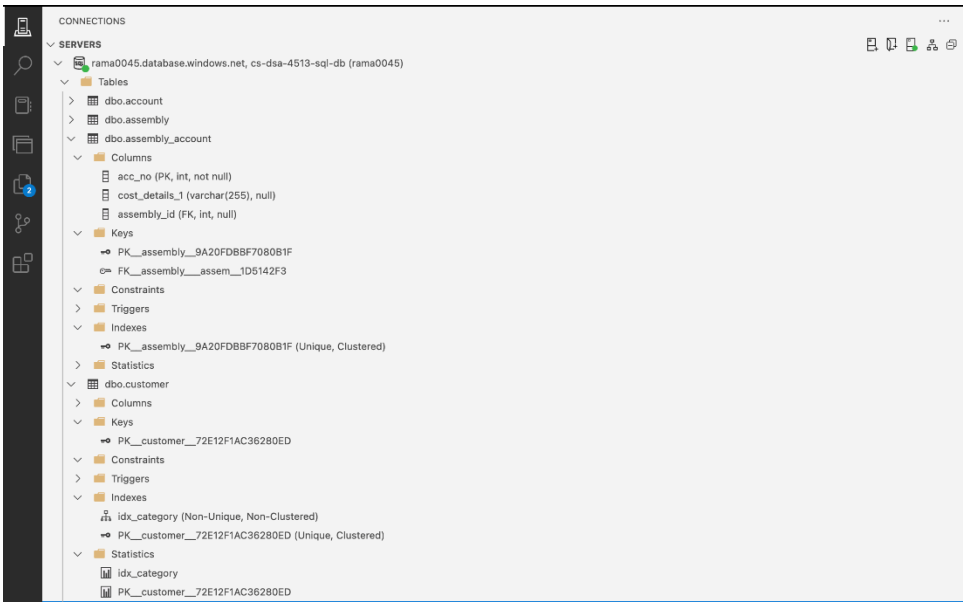
```
FOREIGN KEY (assembly_id) REFERENCES assembly(assembly_id),
FOREIGN KEY (process_id) REFERENCES process(process_id),
FOREIGN KEY (job_no) REFERENCES job(job_no)
);
create nonclustered index idx_manufacture_details_assembly_id on manufacture_details(assembly_id);


-- CREATE process_account TABLE
CREATE TABLE process_account (
acc_no INT PRIMARY KEY,
cost_details_3 VARCHAR(255),
process_id INT,
FOREIGN KEY (acc_no) REFERENCES account(acc_no),
FOREIGN KEY (process_id) REFERENCES process(process_id)
);
-- Since clustered index is created by default i am commenting it
-- create clustered index idx_process_account_number on process_account(account_no);


-- CREATE department_account TABLE
CREATE TABLE department_account (
acc_no INT PRIMARY KEY,
cost_details_2 VARCHAR(255),
dept_no INT,
FOREIGN KEY (acc_no) REFERENCES account(acc_no),
FOREIGN KEY (dept_no) REFERENCES department(dept_no)
);
-- Since clustered index is created by default i am commenting it
--create clustered index idx_department_account_number on department_account(account_no);


-- CREATE assembly_account TABLE
CREATE TABLE assembly_account (
acc_no INT PRIMARY KEY,
cost_details_1 VARCHAR(255),
assembly_id INT,
FOREIGN KEY (assembly_id) REFERENCES assembly(assembly_id)
);
create nonclustered index idx_assembly_acc_no on assembly_account(assembly_id);


-- CREATE transaction_data TABLE
CREATE TABLE transaction_data (
transc_num INT PRIMARY KEY nonclustered,
sup_cost INT,
```

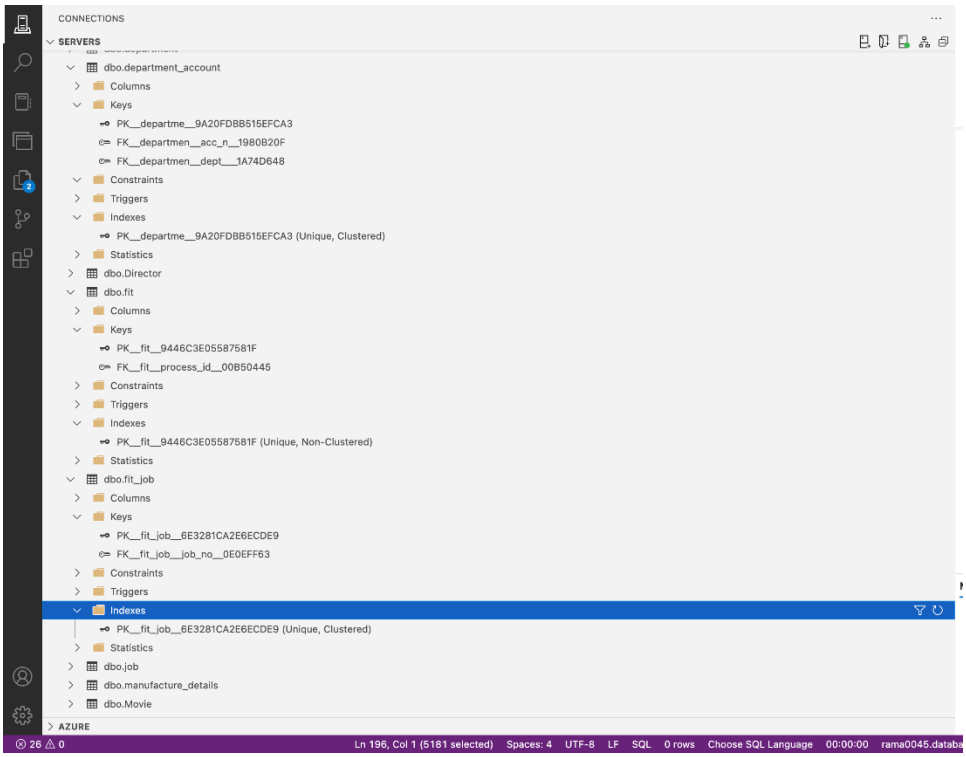
```
job_id INT,
FOREIGN KEY (job_id) REFERENCES job(job_no)
);


-- CREATE updates TABLE
CREATE TABLE updates (
acc_no INT ,
transc_num INT,
PRIMARY KEY nonclustered(acc_no, transc_num),
FOREIGN KEY (transc_num) REFERENCES transaction_data(transc_num),
FOREIGN KEY (acc_no) REFERENCES account(acc_no)
);
```

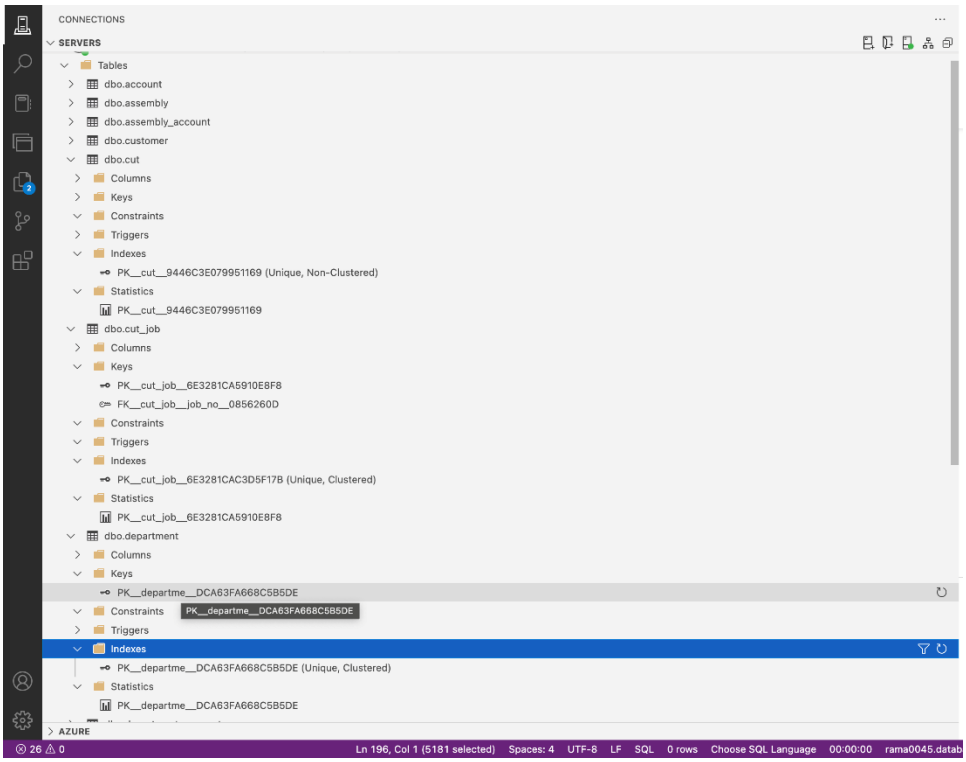
Screenshot showing the tables are created as expected.

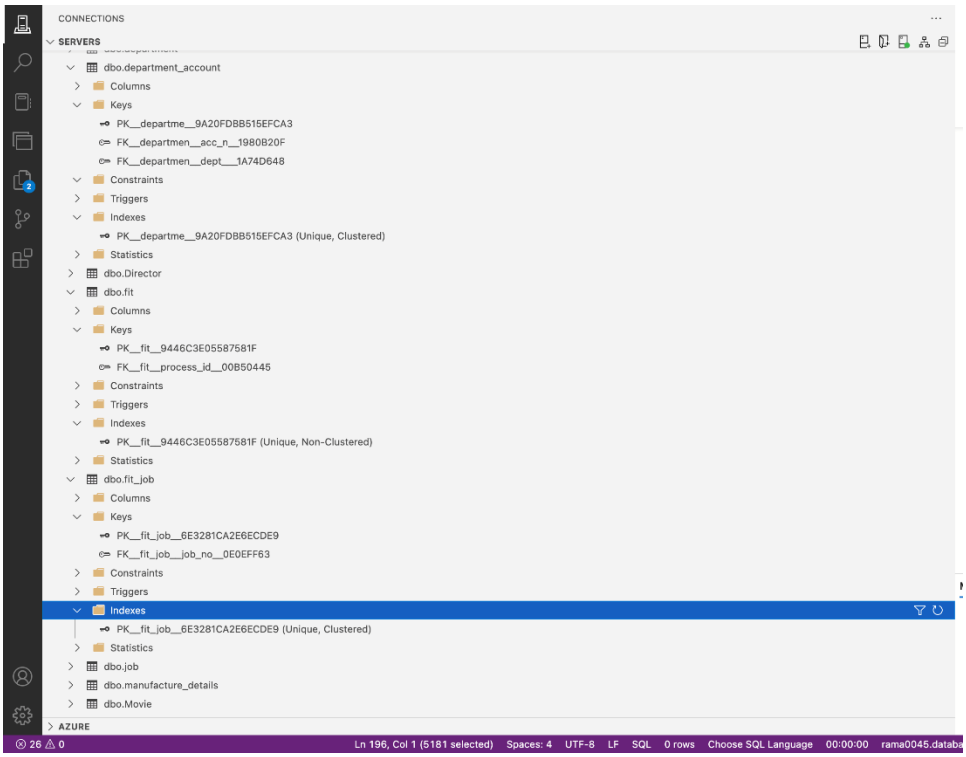


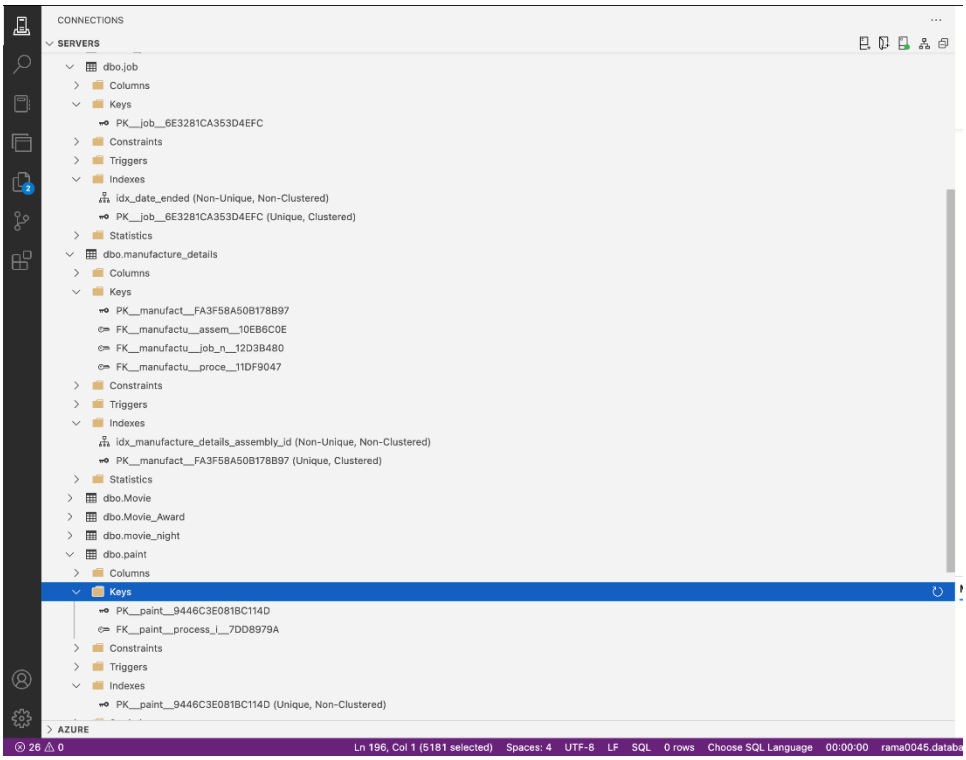

CONNECTIONS
SERVERS
rama0045.database.windows.net, cs-dsa-4513-sql-db (rama0045)
Tables
dbo.account
Columns
Keys
PK__account__9A20FDBB274DD9EE
Constraints
Triggers
Indexes
PK__account__9A20FDBBE2007672 (Unique, Clustered)
Statistics
dbo.assembly
Columns
assembly_id (PK, int, not null)
name (FK, varchar(255), null)
date_ordered (date, null)
assembly_details (varchar(255), null)
Keys
PK__assembly__F37B349B491EB9D3
FK__assembly__name__039170F0
Constraints
Triggers
Indexes
PK__assembly__F37B349B491EB9D3 (Unique, Clustered)
Statistics

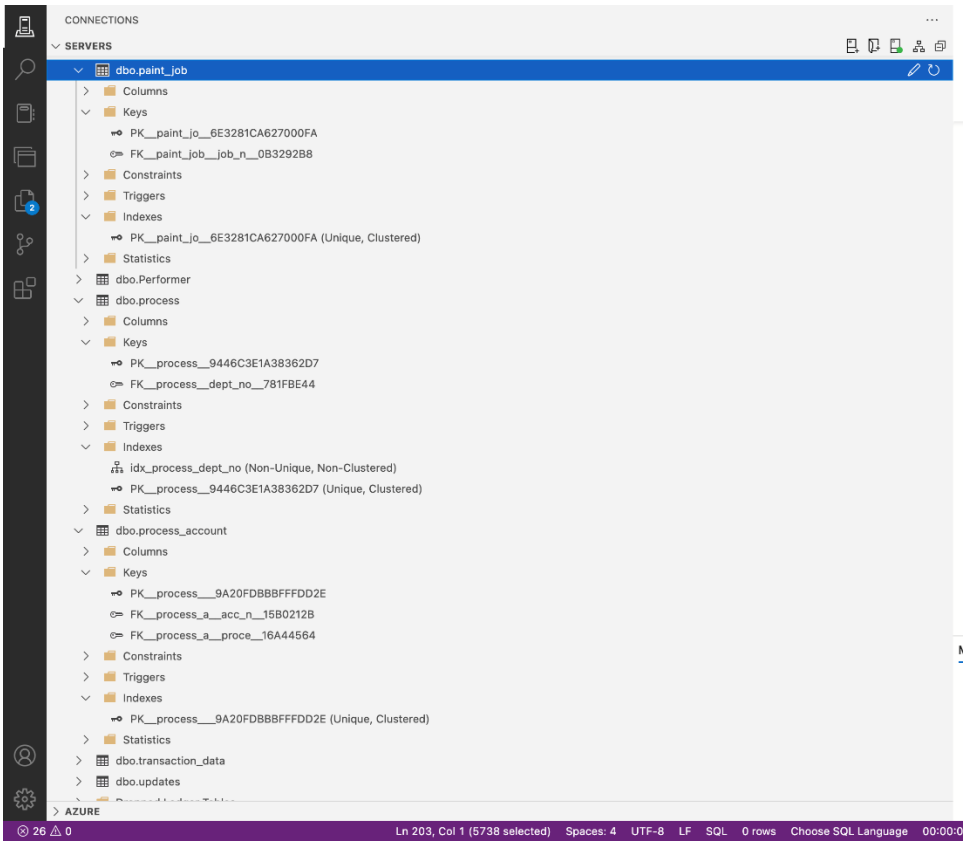

CONNECTIONS
SERVERS
rama0045.database.windows.net, cs-dsa-4513-sql-db (rama0045)
Tables
dbo.account
dbo.assembly
dbo.assembly_account
Columns
acc_no (PK, int, not null)
cost_details_1 (varchar(255), null)
assembly_id (FK, int, null)
Keys
PK__assembly__9A20FDBBF7080B1F
FK__assembly___assem__1D5142F3
Constraints
Triggers
Indexes
PK__assembly__9A20FDBBF7080B1F (Unique, Clustered)
Statistics
dbo.customer
Columns
Keys
PK__customer__72E12F1AC36280ED
Constraints
Triggers
Indexes
idx_category (Non-Unique, Non-Clustered)
PK__customer__72E12F1AC36280ED (Unique, Clustered)
Statistics
idx_category
PK__customer__72E12F1AC36280ED

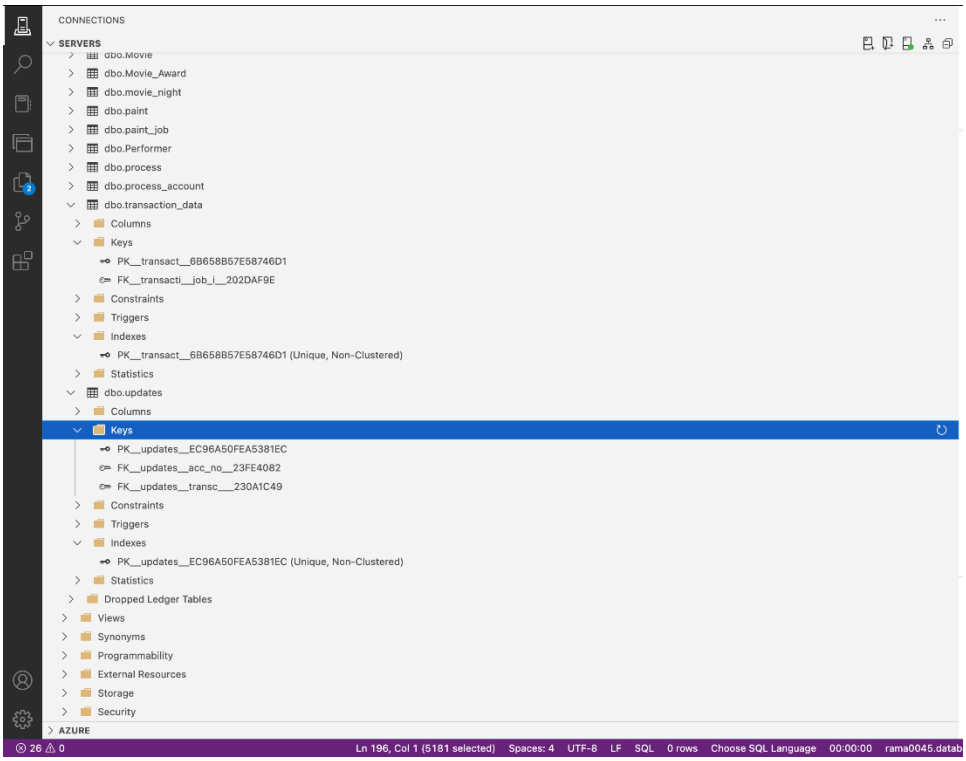

CONNECTIONS
SERVERS
dbo.department_account
Columns
Keys
PK__departme__9A20FDBB515EFCA3
FK__departmen__acc_n__1980B20F
FK__departmen__dept___1A74D648
Constraints
Triggers
Indexes
PK__departme__9A20FDBB515EFCA3 (Unique, Clustered)
Statistics
dbo.Director
dbo.fit
Columns
Keys
PK__fit__9446C3E05587581F
FK__fit__process_id__00B50445
Constraints
Triggers
Indexes
PK__fit__9446C3E05587581F (Unique, Non-Clustered)
Statistics
dbo.fit_job
Columns
Keys
PK__fit_job__6E3281CA2E6ECDE9
FK__fit_job__job_no__0E0EFF63
Constraints
Triggers
Indexes
PK__fit_job__6E3281CA2E6ECDE9 (Unique, Clustered)
Statistics
dbo.job
dbo.manufacture_details
dbo.Movie
AZURE
26 0
Ln 196, Col 1 (5181 selected)
Spaces: 4
UTF-8
LF
SQL
0 rows
Choose SQL Language
00:00:00


CONNECTIONS
SERVERS
Tables
dbo.account
dbo.assembly
dbo.assembly_account
dbo.customer
dbo.cut
Columns
Keys
Constraints
Triggers
Indexes
PK__cut__9446C3E079951169 (Unique, Non-Clustered)
Statistics
PK__cut__9446C3E079951169
dbo.cut_job
Columns
Keys
PK__cut_job__6E3281CA5910E8F8
FK__cut_job__job_no__0856260D
Constraints
Triggers
Indexes
PK__cut_job__6E3281CAC3D5F17B (Unique, Clustered)
Statistics
PK__cut_job__6E3281CA5910E8F8
dbo.department
Columns
Keys
PK__departme__DCA63FA668C5B5DE
Constraints
PK__departme__DCA63FA668C5B5DE
Triggers
Indexes
PK__departme__DCA63FA668C5B5DE (Unique, Clustered)
Statistics
PK__departme__DCA63FA668C5B5DE
AZURE
26 0
Ln 196, Col 1 (5181 selected)
Spaces: 4
UTF-8
LF
SQL
0 rows
Choose SQL Language
00:00:00


CONNECTIONS
SERVERS
dbo.department_account
Columns
Keys
PK__departme__9A20FDBB515EFCA3
FK__departmen__acc_n__1980B20F
FK__departmen__dept___1A74D648
Constraints
Triggers
Indexes
PK__departme__9A20FDBB515EFCA3 (Unique, Clustered)
Statistics
dbo.Director
dbo.fit
Columns
Keys
PK__fit__9446C3E05587581F
FK__fit__process_id__00B50445
Constraints
Triggers
Indexes
PK__fit__9446C3E05587581F (Unique, Non-Clustered)
Statistics
dbo.fit_job
Columns
Keys
PK__fit_job__6E3281CA2E6ECDE9
FK__fit_job__job_no__0E0EFF63
Constraints
Triggers
Indexes
PK__fit_job__6E3281CA2E6ECDE9 (Unique, Clustered)
Statistics
dbo.job
dbo.manufacture_details
dbo.Movie
AZURE
26 0
Ln 196, Col 1 (5181 selected)
Spaces: 4
UTF-8
LF
SQL
0 rows
Choose SQL Language
00:00:00
rama0045.databa


CONNECTIONS
SERVERS
dbo.job
Columns
Keys
PK__job__6E3281CA353D4EFC
Constraints
Triggers
Indexes
idx_date_ended (Non-Unique, Non-Clustered)
PK__job__6E3281CA353D4EFC (Unique, Clustered)
Statistics
dbo.manufacture_details
Columns
Keys
PK__manufact__FA3F58A50B178B97
FK__manufactu__assem__10EB6C0E
FK__manufactu__job_n__12D3B480
FK__manufactu__proce__11DF9047
Constraints
Triggers
Indexes
idx_manufacture_details_assembly_id (Non-Unique, Non-Clustered)
PK__manufact__FA3F58A50B178B97 (Unique, Clustered)
Statistics
dbo.Movie
dbo.Movie_Award
dbo.movie_night
dbo.paint
Columns
Keys
PK__paint__9446C3E081BC114D
FK__paint__process_i__7DD8979A
Constraints
Triggers
Indexes
PK__paint__9446C3E081BC114D (Unique, Non-Clustered)
AZURE
26 0
Ln 196, Col 1 (5181 selected)
Spaces: 4
UTF-8
LF
SQL
0 rows
Choose SQL Language
00:00:00
rama0045.databa


CONNECTIONS
SERVERS
dbo.paint_job
Columns
Keys
PK__paint_jo__6E3281CA627000FA
FK__paint_job__job_n__0B3292B8
Constraints
Triggers
Indexes
PK__paint_jo__6E3281CA627000FA (Unique, Clustered)
Statistics
dbo.Performer
dbo.process
Columns
Keys
PK__process__9446C3E1A38362D7
FK__process__dept_no__781FBE44
Constraints
Triggers
Indexes
idx_process_dept_no (Non-Unique, Non-Clustered)
PK__process__9446C3E1A38362D7 (Unique, Clustered)
Statistics
dbo.process_account
Columns
Keys
PK__process___9A20FDBBBFFFDD2E
FK__process_a__acc_n__15B0212B
FK__process_a__proce__16A44564
Constraints
Triggers
Indexes
PK__process___9A20FDBBBFFFDD2E (Unique, Clustered)
Statistics
dbo.transaction_data
dbo.updates
AZURE
26 0
Ln 203, Col 1 (5738 selected)
Spaces: 4
UTF-8
LF
SQL
0 rows
Choose SQL Language


CONNECTIONS
SERVERS
dbo.Movie_Award
dbo.movie_night
dbo.paint
dbo.paint_job
dbo.Performer
dbo.process
dbo.process_account
dbo.transaction_data
Columns
Keys
PK__transact__6B658B57E58746D1
FK__transacti__job_i__202DAF9E
Constraints
Triggers
Indexes
PK__transact__6B658B57E58746D1 (Unique, Non-Clustered)
Statistics
dbo.updates
Columns
Keys
PK__updates__EC96A50FEA5381EC
FK__updates__acc_no__23FE4082
FK__updates__transc___230A1C49
Constraints
Triggers
Indexes
PK__updates__EC96A50FEA5381EC (Unique, Non-Clustered)
Statistics
Dropped Ledger Tables
Views
Synonyms
Programmability
External Resources
Storage
Security
AZURE
26 0
Ln 196, Col 1 (5181 selected)
Spaces: 4
UTF-8
LF
SQL
0 rows
Choose SQL Language
00:00:00
rama0045.datab

Task 5:

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

```
DROP PROCEDURE IF EXISTS ChangePaintJobColor;

DROP PROCEDURE IF EXISTS DeleteCutJobsInRange;

DROP PROCEDURE IF EXISTS GetCustomersByCategoryRange;

DROP PROCEDURE IF EXISTS GetProcessesForAssembly;

DROP PROCEDURE IF EXISTS GetTotalLaborTimeInDepartment;

DROP PROCEDURE IF EXISTS GetTotalCostForAssembly;

DROP PROCEDURE IF EXISTS UpdateAccountCosts;

DROP PROCEDURE IF EXISTS CompleteJobWithDetails;

DROP PROCEDURE IF EXISTS EnterNewJob;

DROP PROCEDURE IF EXISTS CreateNewAccountAndAssociate;

DROP PROCEDURE IF EXISTS InsertNewAssembly;

DROP PROCEDURE IF EXISTS InsertProcess;

DROP PROCEDURE IF EXISTS InsertDepartment;

DROP PROCEDURE IF EXISTS InsertCustomer;



--1

GO

CREATE PROCEDURE InsertCustomer

@customer_name VARCHAR(255),

@address VARCHAR(255),

@category INT

AS

BEGIN
```

```
INSERT INTO customer (name, address, category)
VALUES (@customer_name, @address, @category);
END;


--2
go
CREATE PROCEDURE InsertDepartment
@dept_no INT,
@dept_data VARCHAR(255)
AS
BEGIN
INSERT INTO Department (dept_no, dept_data)
VALUES (@dept_no, @dept_data);
END;


-- 3


GO
CREATE PROCEDURE InsertProcess
@p_process_id INT,
@p_process_data VARCHAR(255),
@p_dept_no INT,
@p_dept_data VARCHAR(255),
@p_type VARCHAR(50),
@p_type_info1 VARCHAR(255),
@p_type_info2 VARCHAR(255)
```

```
AS
BEGIN
-- Insert into process table
INSERT INTO process(process_id, process_data, dept_no) VALUES (@p_process_id, @p_process_data, @p_dept_no);


-- Insert into appropriate type table based on the given type
IF @p_type = 'fit'
BEGIN
INSERT INTO fit(process_id, fit_type) VALUES (@p_process_id, @p_type_info1);
END
ELSE IF @p_type = 'cut'
BEGIN
INSERT INTO cut(process_id, cutting_type, machine_type) VALUES (@p_process_id, @p_type_info1, @p_type_info2);
END
ELSE IF @p_type = 'paint'
BEGIN
INSERT INTO paint(process_id, paint_type, paint_method) VALUES (@p_process_id, @p_type_info1, @p_type_info2);
END


END;



--4


GO
CREATE PROCEDURE InsertNewAssembly
```

```
@customerName NVARCHAR(255),
@assemblyDetails NVARCHAR(255),
@assemblyID INT,
@dateOrdered DATE,
@processIDs NVARCHAR(MAX)
AS
BEGIN
DECLARE @i INT = 0;

-- Insert into the assembly table
INSERT INTO assembly (assembly_id, name, assembly_details, date_ordered)
VALUES (@assemblyID, @customerName, @assemblyDetails, @dateOrdered);

-- Associate the assembly with specified processes
DECLARE @processID INT;
DECLARE @pos INT;

WHILE LEN(@processIDs) > 0
BEGIN
SET @pos = CHARINDEX(',', @processIDs);

IF @pos > 0
SET @processID = CAST(LEFT(@processIDs, @pos - 1) AS INT);
ELSE
SET @processID = CAST(@processIDs AS INT);
```

```
INSERT INTO manufacture_details (assembly_id, process_id)
VALUES (@assemblyID, @processID);

IF @pos > 0
SET @processIDs = SUBSTRING(@processIDs, @pos + 1, LEN(@processIDs) - @pos);
ELSE
SET @processIDs = '';
END;
END;

--5
GO
CREATE PROCEDURE CreateNewAccountAndAssociate
@acc_no INT,
@acc_est_date DATE,
@process_id INT = NULL,
@assembly_id INT = NULL,
@dept_no INT = NULL
AS
BEGIN
-- Insert into the account table
INSERT INTO account (acc_no, acc_est_date)
VALUES (@acc_no, @acc_est_date);

-- Associate the account with process, assembly, or department accounts
```

```
IF @process_id <> 0
BEGIN
INSERT INTO process_account (acc_no, process_id)
VALUES (@acc_no, @process_id);
END

IF @assembly_id <>0
BEGIN
INSERT INTO assembly_account (acc_no, assembly_id)
VALUES (@acc_no, @assembly_id);
END

IF @dept_no <>0
BEGIN
INSERT INTO department_account (acc_no, dept_no)
VALUES (@acc_no, @dept_no);
END
END


--6
GO
CREATE PROCEDURE EnterNewJob
@job_no INT,
@assembly_id INT,
@process_id INT,
```

```
@job_start_date DATE
AS
BEGIN
-- Insert the new job
INSERT INTO job (job_no, job_start_date)
VALUES (@job_no, @job_start_date);


-- Associate the job with assembly and process
UPDATE manufacture_details
SET job_no = @job_no
WHERE assembly_id = @assembly_id
AND process_id = @process_id;
END


--7
GO
CREATE PROCEDURE CompleteJobWithDetails
@job_no INT,
@completion_date DATE,
@job_type VARCHAR(50),
@machine_type VARCHAR(255) = NULL,
@time_used INT = NULL,
@material_used VARCHAR(255) = NULL,
@labor_time INT = NULL,
@color VARCHAR(255) = NULL,
@volume VARCHAR(255) = NULL,
@job_labor INT = NULL
```

```
AS
BEGIN
-- Update the job with the completion date
UPDATE job
SET job_end_date = @completion_date
WHERE job_no = @job_no;


-- Determine the job type and insert information
IF @job_type = 'cut-job'
BEGIN
-- Insert Cut-job specific information into cut_job table
INSERT INTO cut_job (job_no, machine_type, time_used, material_used, labor_time)
VALUES (@job_no, @machine_type, @time_used, @material_used, @labor_time);
END
ELSE IF @job_type = 'paint-job'
BEGIN
-- Insert Paint-job specific information into paint_job table
INSERT INTO paint_job (job_no, colour, volume, labor_time)
VALUES (@job_no, @color, @volume, @job_labor);
END
ELSE IF @job_type = 'fit-job'
BEGIN
-- Insert Fit-job specific information into fit_job table
INSERT INTO fit_job (job_no, labor_time)
VALUES (@job_no, @job_labor);
END
END
```

```
--8
GO
CREATE PROCEDURE UpdateAccountCosts
@transc_num INT,
@sup_cost INT,
@acc_no INT
AS
BEGIN



-- Update the process account cost details
UPDATE process_account
SET cost_details_3 = ISNULL(cost_details_3, 0) + @sup_cost
WHERE acc_no = @acc_no


-- Update the department account cost details
UPDATE department_account
SET cost_details_2 = ISNULL(cost_details_2, 0) + @sup_cost
WHERE acc_no = @acc_no


-- Update the assembly account cost details
UPDATE assembly_account
SET cost_details_1 = ISNULL(cost_details_1, 0) + @sup_cost
WHERE acc_no = @acc_no
```

```
-- Insert the transaction into the updates table
INSERT INTO transaction_data (transc_num, sup_cost)
VALUES (@transc_num, @sup_cost)

INSERT INTO updates (acc_no, transc_num)
VALUES (@acc_no, @transc_num);
END;



--9

GO
CREATE PROCEDURE GetTotalCostForAssembly
@assembly_id INT
AS
BEGIN
DECLARE @total_cost DECIMAL(10, 2);

-- Calculate the total cost for the given assembly ID
SELECT @total_cost = ISNULL(SUM(CAST(ISNULL(aa.cost_details_1, 0) AS DECIMAL(10, 2))), 0)
FROM assembly_account aa
WHERE aa.assembly_id = @assembly_id;
```

```
-- Return the total cost
SELECT @total_cost AS TotalCost;
END



--10
GO
CREATE PROCEDURE GetTotalLaborTimeInDepartment
@targetDate DATE,
@deptNo INT
AS
BEGIN
SELECT
d.dept_no,
d.dept_data AS department_name,
SUM(COALESCE(cut_job.labor_time, 0) + COALESCE(paint_job.labor_time, 0) + COALESCE(fit_job.labor_time, 0)) AS
total_labor_time
FROM
department d
LEFT JOIN process p ON d.dept_no = p.dept_no
LEFT JOIN cut c ON p.process_id = c.process_id
LEFT JOIN paint pa ON p.process_id = pa.process_id
LEFT JOIN fit f ON p.process_id = f.process_id
LEFT JOIN manufacture_details md ON p.process_id = md.process_id
LEFT JOIN cut_job ON md.job_no = cut_job.job_no
LEFT JOIN paint_job ON md.job_no = paint_job.job_no
LEFT JOIN fit_job ON md.job_no = fit_job.job_no
LEFT JOIN job j ON md.job_no = j.job_no
WHERE
```

```
j.job_end_date = @targetDate
AND d.dept_no = @deptNo
GROUP BY
d.dept_no, d.dept_data;
END;



--11
GO
CREATE PROCEDURE GetProcessesForAssembly
@assemblyId INT
AS
BEGIN
-- Retrieve processes and department information for the given assembly ID
SELECT p.process_id, p.process_data, d.dept_data, j.job_start_date
FROM manufacture_details md
JOIN process p ON md.process_id = p.process_id
JOIN department d ON p.dept_no = d.dept_no
JOIN job j ON md.job_no = j.job_no
WHERE md.assembly_id = @assemblyId
ORDER BY j.job_start_date;
END;


--12


GO
```

```
CREATE PROCEDURE GetCustomersByCategoryRange
@minCategory INT,
@maxCategory INT
AS
BEGIN
-- Retrieve customers within the specified category range in name order
SELECT name, address, category
FROM customer
WHERE category BETWEEN @minCategory AND @maxCategory
ORDER BY name;
END


--13
GO
CREATE PROCEDURE DeleteCutJobsInRange
@minJobNo INT,
@maxJobNo INT
AS
BEGIN
-- Delete cut-jobs whose job numbers are in the specified range
DELETE FROM cut_job
WHERE job_no BETWEEN @minJobNo AND @maxJobNo;
END


--14
GO
CREATE PROCEDURE ChangePaintJobColor
```

```
@job_no INT,
@newColor VARCHAR(255)
AS
BEGIN
-- Update the color of the specified paint job
UPDATE paint_job
SET colour = @newColor
WHERE job_no = @job_no;
END


--ERROR Checking
--1
Insert into customer values ('Akhil', 'Texas', 10)
--2
INSERT INTO account(acc_no, acc_est_date)
values(123,'2023-11-14',100.00)
--3
INSERT into departmen_account VALUES (20,30,40)
```

5.2 The Java source program and screenshots showing its successful compilation

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.sql.CallableStatement;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Types;
import java.util.Date;
import java.util.Scanner;

public class Rama_Dinesh_IP_Task5b {
   //  Database URL COnnection
private static final String DATABASE_URL = "jdbc:sqlserver://rama0045.database.windows.net:1433;" +
      "database=cs-dsa-4513-sql-db;user=rama0045;password=Dkronaldo7@;" +
"encrypt=true;trustServerCertificate=false;hostNameInCertificate=*.database.windows.net;loginTimeout=30;";

   public static void main(String[] args) {
           //Creating Database connection
      try (Connection connection = DriverManager.getConnection(DATABASE_URL)) {
         Scanner scanner = new Scanner(System.in);
```

```
int choice;

do {
   // Display menu
   System.out.println("Welcome to Job-Shop Accounting Database System:");
   System.out.println("1. Insert a new customer");
   System.out.println("2. Insert a new department");
   System.out.println("3. Insert a new process");
   System.out.println("4. Insert a new assembly");
   System.out.println("5. Insert a new account");
   System.out.println("6. Enter the new job");
   System.out.println("7. Enter complete job details");
   System.out.println("8. Update account cost details");
   System.out.println("9. Retrieve total cost on assembly-id ");
   System.out.println("10. Retrieve total labor time in a given date");
   System.out.println("11. Retrieve process");
   System.out.println("12. Retrieve customer details between given category range ");
   System.out.println("13. Delete cut-jobs between given job range");
   System.out.println("14. Change color of paint-job");
   System.out.println("15. Import customer details from a file");
   System.out.println("16. Export customer details to a file");
   System.out.println("17. QUIT!");



   System.out.println("Enter your choice: ");
   choice = scanner.nextInt();
   scanner.nextLine(); // Consume newline
```

```
switch (choice) {
    case 1:
        // Insert a new customer
        System.out.println("Enter customer name: ");
        String customerName = scanner.nextLine();
        System.out.println("Enter address: ");
        String address = scanner.nextLine();
        System.out.println("Enter category: ");
        int category = scanner.nextInt();

        insertCustomer(connection, customerName, address, category);
        break;



    case 2:
        // Insert a new department
        System.out.println("Enter department number: ");
        int deptNo = scanner.nextInt();
        scanner.nextLine(); // Consume newline
        System.out.println("Enter department data: ");
        String deptData = scanner.nextLine();

        insertDepartment(connection, deptNo, deptData);
        break;

    case 3:
```

```
    // Insert a new process
System.out.println("Enter process ID: ");
int processId = scanner.nextInt();
scanner.nextLine(); // Consume newline
System.out.println("Enter process data: ");
String process_data = scanner.nextLine();
System.out.println("Enter department number: ");
int deptNoForProcess = scanner.nextInt();
scanner.nextLine(); // Consume newline
System.out.println("Enter department data: ");
String deptDataForProcess = scanner.nextLine();
System.out.println("Enter process type (fit/cut/paint): ");
String processType = scanner.nextLine();

// Additional information based on process type
String typeInfo1 = "";
String typeInfo2 = "";
if (processType.equals("fit") || processType.equals("cut") || processType.equals("paint")) {
   System.out.println("Enter type info 1: ");
   typeInfo1 = scanner.nextLine();
}
if (processType.equals("cut")) {
   System.out.println("Enter type info 2: ");
   typeInfo2 = scanner.nextLine();
}

insertProcess(connection, processId, process_data, deptNoForProcess, deptDataForProcess,
processType, typeInfo1, typeInfo2);
break;
```

```
            case 4:
                // Insert a new assembly
                System.out.println("Enter customer name: ");
                customerName = scanner.nextLine();
                System.out.println("Enter assembly details: ");
                String assemblyDetails = scanner.nextLine();
                System.out.println("Enter assembly ID: ");
                int assemblyID = scanner.nextInt();
                scanner.nextLine(); // Consume newline
                System.out.println("Enter date ordered (YYYY-MM-DD): ");
                String dateOrderedStr = scanner.nextLine();
                System.out.println("Enter comma-separated process IDs: ");
                String processIDs = scanner.nextLine();

                // Convert date string to java.sql.Date
                java.sql.Date dateOrdered = java.sql.Date.valueOf(dateOrderedStr);

                // Call the stored procedure
                try {
                    insertNewAssembly(connection, customerName, assemblyDetails, assemblyID, dateOrdered,
processIDs);
                    System.out.println("Assembly inserted successfully!");
                } catch (SQLException e) {
                    e.printStackTrace();
                    System.out.println("Error inserting assembly. Please try again.");
                }
                break;
```

```
            case 5:
                // Create a new account and associate with process, assembly, or department
                System.out.println("Enter account number: ");
                int accountNumber = scanner.nextInt();
                scanner.nextLine(); // Consume newline
                System.out.println("Enter account establishment date (YYYY-MM-DD): ");
                String accountEstablishmentDateStr = scanner.nextLine();

                // Convert date string to java.sql.Date
                java.sql.Date accountEstablishmentDate =
java.sql.Date.valueOf(accountEstablishmentDateStr);

                System.out.println("Enter process ID (or enter 0 if not applicable): ");
                int processID = scanner.nextInt();
                scanner.nextLine(); // Consume newline

                System.out.println("Enter assembly ID (or enter 0 if not applicable): ");
                assemblyID = scanner.nextInt();
                scanner.nextLine(); // Consume newline

                System.out.println("Enter department number (or enter 0 if not applicable): ");
                int departmentNumber = scanner.nextInt();
                scanner.nextLine(); // Consume newline

                // Call the stored procedure
                try {
                    createNewAccountAndAssociate(connection, accountNumber, accountEstablishmentDate,
processID, assemblyID, departmentNumber);
                    System.out.println("Account created and associated successfully!");
```

```
    } catch (SQLException e) {
        e.printStackTrace();
        System.out.println("Error creating account. Please try again.");
    }
    break;
case 6:
    // Enter a new job
    System.out.println("Enter job number: ");
    int jobNumber = scanner.nextInt();
    scanner.nextLine(); // Consume newline
    System.out.println("Enter assembly ID: ");
    int assemblyIDForJob = scanner.nextInt();
    scanner.nextLine(); // Consume newline
    System.out.println("Enter process ID: ");
    int processIDForJob = scanner.nextInt();
    scanner.nextLine(); // Consume newline
    System.out.println("Enter job start date (YYYY-MM-DD): ");
    String jobStartDateStr = scanner.nextLine();

    // Convert date string to java.sql.Date
    java.sql.Date jobStartDate = java.sql.Date.valueOf(jobStartDateStr);

    // Call the stored procedure
    try {
        enterNewJob(connection, jobNumber, assemblyIDForJob, processIDForJob, jobStartDate);
        System.out.println("Job entered successfully!");
    } catch (SQLException e) {
        e.printStackTrace();
```

```
            System.out.println("Error entering job. Please try again.");
        }
        break;
    case 7:
            // Complete a job with details
        System.out.println("Enter job number: ");
        jobNumber = scanner.nextInt();
        scanner.nextLine(); // Consume newline
        System.out.println("Enter completion date (YYYY-MM-DD): ");
        String completionDateStr = scanner.nextLine();
        System.out.println("Enter job type (cut-job/paint-job/fit-job): ");
        String jobType = scanner.nextLine();

        // Optional details based on job type
        String machineType = null;
        int timeUsed = 0;
        String materialUsed = null;
        int laborTime = 0;
        String color = null;
        String volume = null;
        int jobLabor = 0;

        if (jobType.equals("cut-job")) {
            System.out.println("Enter machine type: ");
            machineType = scanner.nextLine();
            System.out.println("Enter time used: ");
            timeUsed = scanner.nextInt();
            scanner.nextLine(); // Consume newline
```

```
            System.out.println("Enter material used: ");
            materialUsed = scanner.nextLine();
            System.out.println("Enter labor time: ");
            laborTime = scanner.nextInt();
            scanner.nextLine(); // Consume newline
        } else if (jobType.equals("paint-job")) {
            System.out.println("Enter color: ");
            color = scanner.nextLine();
            System.out.println("Enter volume: ");
            volume = scanner.nextLine();
            System.out.println("Enter job labor: ");
            jobLabor = scanner.nextInt();
            scanner.nextLine(); // Consume newline
        } else if (jobType.equals("fit-job")) {
            System.out.println("Enter job labor: ");
            jobLabor = scanner.nextInt();
            scanner.nextLine(); // Consume newline
        }

        // Convert date string to java.sql.Date
        java.sql.Date completionDate = java.sql.Date.valueOf(completionDateStr);

        // Call the stored procedure
        try {
            completeJobWithDetails(connection, jobNumber, completionDate, jobType, machineType,
timeUsed, materialUsed, laborTime, color, volume, jobLabor);
            System.out.println("Job completed successfully!");
        } catch (SQLException e) {
            e.printStackTrace();
```

```
            System.out.println("Error completing job. Please try again.");
        }
        break;
    case 8:
            // Update account costs
        System.out.println("Enter transaction number: ");
        int transcNumber = scanner.nextInt();
        scanner.nextLine(); // Consume newline
        System.out.println("Enter supplier cost: ");
        int supplierCost = scanner.nextInt();
        System.out.println("Enter account number: ");
        int acc_no = scanner.nextInt();
        scanner.nextLine(); // Consume newline

        // Call the stored procedure
        try {
            updateAccountCosts(connection, transcNumber, supplierCost, acc_no);
            System.out.println("Account costs updated successfully!");
        } catch (SQLException e) {
            e.printStackTrace();
            System.out.println("Error updating account costs. Please try again.");
        }
        break;
    case 9:
            // Get total cost for assembly
        System.out.println("Enter assembly ID: ");
        assemblyID = scanner.nextInt();
        scanner.nextLine(); // Consume newline
```

```
                // Call the stored procedure
                try {
                    double totalCost = getTotalCostForAssembly(connection, assemblyID);
                    System.out.println("Total cost for assembly " + assemblyID + ": $" + totalCost);
                } catch (SQLException e) {
                    e.printStackTrace();
                    System.out.println("Error getting total cost for assembly. Please try again.");
                }
                break;

            case 10:
                 // Get total labor time in department
                System.out.println("Enter department number: ");
                departmentNumber = scanner.nextInt();
                scanner.nextLine(); // Consume newline
                System.out.println("Enter completion date (YYYY-MM-DD): ");
                completionDateStr = scanner.nextLine();

                // Call the stored procedure
                try {

                  double result= getTotalLaborTimeInDepartment(connection, departmentNumber,
completionDateStr);
                    System.out.println("Total labor time retrieved for department " + result);
                } catch (SQLException e) {
                    e.printStackTrace();
                    System.out.println("Error getting total labor time in department. Please try again.");
                }
```

```
    break;
case 11:
        // Get processes for assembly
    System.out.println("Enter assembly ID: ");
    assemblyID = scanner.nextInt();
    scanner.nextLine(); // Consume newline

    // Call the stored procedure
    try {
        getProcessesForAssembly(connection, assemblyID);
    } catch (SQLException e) {
        e.printStackTrace();
        System.out.println("Error getting processes for assembly. Please try again.");
    }
    break;
case 12:
        // Get customers by category range
    System.out.println("Enter minimum category: ");
    int minCategory = scanner.nextInt();
    System.out.println("Enter maximum category: ");
    int maxCategory = scanner.nextInt();

    // Call the stored procedure
    try {
        getCustomersByCategoryRange(connection, minCategory, maxCategory);
    } catch (SQLException e) {
        e.printStackTrace();
        System.out.println("Error getting customers by category range. Please try again.");
```

```
    }
    break;
case 13:
        // Delete cut jobs based on given range
        System.out.println("Enter minimum job number: ");
    int minJobNo = scanner.nextInt();
    System.out.println("Enter maximum job number: ");
    int maxJobNo = scanner.nextInt();

    // Call the stored procedure
    try {
        deleteCutJobsInRange(connection, minJobNo, maxJobNo);
        System.out.println("Cut jobs deleted successfully.");
    } catch (SQLException e) {
        e.printStackTrace();
        System.out.println("Error deleting cut jobs. Please try again.");
    }
    break;

case 14:
        // Change paint job color
    System.out.println("Enter job number: ");
    jobNumber = scanner.nextInt();
    scanner.nextLine(); // Consume newline
    System.out.println("Enter new color: ");
    String newColor = scanner.nextLine();

    // Call the stored procedure
```

```
            try {
                changePaintJobColor(connection, jobNumber, newColor);
                System.out.println("Paint job color changed successfully.");
            } catch (SQLException e) {
                e.printStackTrace();
                System.out.println("Error changing paint job color. Please try again.");
            }
            break;
        case 15:
                // Import customers from a data file
            System.out.println("Enter the input file name: ");
            String inputFile = scanner.nextLine();

            // Call the importCustomersFromFile method
            try {
                importCustomersFromFile(connection, inputFile);
                System.out.println("Customers imported successfully.");
            } catch (IOException | SQLException e) {
                e.printStackTrace();
                System.out.println("Error importing customers. Please try again.");
            }
            break;
        case 16:
                // Export customers to a data file
            System.out.println("Enter minimum category: ");
            int minCategory1 = scanner.nextInt();
            scanner.nextLine(); // Consume newline
            System.out.println("Enter maximum category: ");
```

```
            int maxCategory2 = scanner.nextInt();
            scanner.nextLine(); // Consume newline
            System.out.println("Enter the output file name: ");
            String outputFile = scanner.nextLine();

            // Call the exportCustomersToFile method
            try {
                exportCustomersToFile(connection, minCategory1, maxCategory2, outputFile);
                System.out.println("Customers exported successfully.");
            } catch (SQLException | IOException e) {
                e.printStackTrace();
                System.out.println("Error exporting customers. Please try again.");
            }
            break;




        case 17:
            // Exit the program
            System.out.println("Exiting program!");
            return;
        default:
            System.out.println("Invalid choice. Please try again.");
      }
   }while (choice != 17);
} catch (SQLException e) {
```

```
            e.printStackTrace();
        }
    }

    // insert customer method
    private static void insertCustomer(Connection connection, String customerName, String address, int
category) throws SQLException {
        String sql = "EXEC InsertCustomer ?, ?, ?";
        try (PreparedStatement statement = connection.prepareStatement(sql)) {
            statement.setString(1, customerName);
            statement.setString(2, address);
            statement.setInt(3, category);
         // Execute the stored procedure
            statement.executeUpdate();
            System.out.println("Customer inserted successfully!");

        }
    }

    // inssert department method
    private static void insertDepartment(Connection connection, int deptNo, String deptData) throws
SQLException {
        String sql = "EXEC InsertDepartment ?, ?";
        try (PreparedStatement statement = connection.prepareStatement(sql)) {
            statement.setInt(1, deptNo);
            statement.setString(2, deptData);
         // Execute the stored procedure
            statement.executeUpdate();
            System.out.println("Department inserted successfully!");
```

```
        }
    }


    // insert process method
    private static void insertProcess(Connection connection, int processId, String process_data, int deptNo,
String deptData, String type, String typeInfo1, String typeInfo2) throws SQLException {
        String sql = "EXEC InsertProcess ?, ?, ?, ?, ?, ?, ?";
        try (PreparedStatement statement = connection.prepareStatement(sql)) {
            statement.setInt(1, processId);
            statement.setString(2, process_data);
            statement.setInt(3, deptNo);
            statement.setString(4, deptData);
            statement.setString(5, type);
            statement.setString(6, typeInfo1);
            statement.setString(7, typeInfo2);
         // Execute the stored procedure
            statement.executeUpdate();
            System.out.println("Process inserted successfully!");
        }
    }


    private static void insertNewAssembly(Connection connection, String customerName, String
assemblyDetails, int assemblyID, java.sql.Date dateOrdered, String processIDs) throws SQLException {
        String sql = "{CALL InsertNewAssembly(?, ?, ?, ?, ?)}";


        try (CallableStatement statement = connection.prepareCall(sql)) {
            // Set the input parameters
            statement.setString(1, customerName);
            statement.setString(2, assemblyDetails);
```

```
            statement.setInt(3, assemblyID);
            statement.setDate(4, dateOrdered);
            statement.setString(5, processIDs);

            // Execute the stored procedure
            statement.execute();
        }
    }

    private static void createNewAccountAndAssociate(Connection connection, int accountNumber,
java.sql.Date accountEstablishmentDate, int processID, int assemblyID, int departmentNumber) throws
SQLException {
        String sql = "{CALL CreateNewAccountAndAssociate(?, ?, ?, ?, ?)}";

        try (CallableStatement statement = connection.prepareCall(sql)) {
            // Set the input parameters
            statement.setInt(1, accountNumber);
            statement.setDate(2, accountEstablishmentDate);
            statement.setInt(3, processID);
            statement.setInt(4, assemblyID);
            statement.setInt(5, departmentNumber);

            // Execute the stored procedure
            statement.execute();
        }
    }

    private static void enterNewJob(Connection connection, int jobNumber, int assemblyID, int processID,
java.sql.Date jobStartDate) throws SQLException {
        String sql = "{CALL EnterNewJob(?, ?, ?, ?)}";
```

```
    try (CallableStatement statement = connection.prepareCall(sql)) {
        // Set the input parameters
        statement.setInt(1, jobNumber);
        statement.setInt(2, assemblyID);
        statement.setInt(3, processID);
        statement.setDate(4, jobStartDate);

        // Execute the stored procedure
        statement.execute();
    }
}

  private static void completeJobWithDetails(Connection connection, int jobNumber, java.sql.Date
completionDate, String jobType, String machineType, int timeUsed, String materialUsed, int laborTime,
String color, String volume, int jobLabor) throws SQLException {
    String sql = "{CALL CompleteJobWithDetails(?, ?, ?, ?, ?, ?, ?, ?, ?, ?)}";

    try (CallableStatement statement = connection.prepareCall(sql)) {
        // Set the input parameters
        statement.setInt(1, jobNumber);
        statement.setDate(2, completionDate);
        statement.setString(3, jobType);
        statement.setString(4, machineType);
        statement.setInt(5, timeUsed);
        statement.setString(6, materialUsed);
        statement.setInt(7, laborTime);
        statement.setString(8, color);
        statement.setString(9, volume);
        statement.setInt(10, jobLabor);
```

```
            // Execute the stored procedure
            statement.execute();
        }
    }


    private static void updateAccountCosts(Connection connection, int transcNumber, int supplierCost, int
acc_no) throws SQLException {
        String sql = "{CALL UpdateAccountCosts(?, ?, ?)}";

        try (CallableStatement statement = connection.prepareCall(sql)) {
            // Set the input parameters
            statement.setInt(1, transcNumber);
            statement.setInt(2, supplierCost);
            statement.setInt(3, acc_no);

            // Execute the stored procedure
            statement.execute();
        }
    }


    private static double getTotalCostForAssembly(Connection connection, int assemblyID) throws
SQLException {
        String sql = "{CALL GetTotalCostForAssembly(?)}";

        try (CallableStatement statement = connection.prepareCall(sql)) {
            // Set the input parameter
```

```
            statement.setInt(1, assemblyID);

            // Execute the stored procedure
            ResultSet resultSet = statement.executeQuery();

            // Retrieve the total cost from the result set
            if (resultSet.next()) {
                return resultSet.getDouble("TotalCost");
            } else {
                throw new SQLException("No result returned from the stored procedure");
            }
        }
    }

    private static double getTotalLaborTimeInDepartment(Connection connection, int departmentNumber,
String completionDateStr) throws SQLException {
        String sql = "{CALL GetTotalLaborTimeInDepartment(?, ?)}";
        double totalLaborTime=0;
        try (CallableStatement statement = connection.prepareCall(sql)) {
            // Set the input parameters
            statement.setInt(2, departmentNumber);
            statement.setDate(1, java.sql.Date.valueOf(completionDateStr));
            // Execute the stored procedure

            ResultSet resultSet= statement.executeQuery();
            if (resultSet.next()) {
                totalLaborTime = resultSet.getDouble("total_labor_time");
            }
        }
```

```
        return totalLaborTime;
    }



    private static void getProcessesForAssembly(Connection connection, int assemblyID) throws
SQLException {
        String sql = "{CALL GetProcessesForAssembly(?)}";

        try (CallableStatement statement = connection.prepareCall(sql)) {
            // Set the input parameter
            statement.setInt(1, assemblyID);

            // Execute the stored procedure
            ResultSet resultSet = statement.executeQuery();

            // Display the result
            while (resultSet.next()) {
                String processName = resultSet.getString("process_data");
                String departmentName = resultSet.getString("dept_data");
                int processId = resultSet.getInt("process_id");
                Date job_date = resultSet.getDate("job_start_date");
                System.out.println("ProcessId:"+processId+", Process: " + processName + ", Department: " +
departmentName+", job start date:"+job_date);
            }
        }
    }



    private static void getCustomersByCategoryRange(Connection connection, int minCategory, int
maxCategory) throws SQLException {
```

```
    String sql = "{CALL GetCustomersByCategoryRange(?, ?)}";

    try (CallableStatement statement = connection.prepareCall(sql)) {
        // Set the input parameters
        statement.setInt(1, minCategory);
        statement.setInt(2, maxCategory);

        // Execute the stored procedure
        ResultSet resultSet = statement.executeQuery();

        // Display the result
        while (resultSet.next()) {
            String name = resultSet.getString("name");
            String address = resultSet.getString("address");
            int category = resultSet.getInt("category");
            System.out.println("Name: " + name + ", Address: " + address + ", Category: " + category);
        }
    }
}


  private static void deleteCutJobsInRange(Connection connection, int minJobNo, int maxJobNo) throws
SQLException {
    String sql = "{CALL DeleteCutJobsInRange(?, ?)}";

    try (CallableStatement statement = connection.prepareCall(sql)) {
        // Set the input parameters
        statement.setInt(1, minJobNo);
        statement.setInt(2, maxJobNo);
```

```
            // Execute the stored procedure
            statement.execute();
        }
    }


    private static void changePaintJobColor(Connection connection, int jobNumber, String newColor) throws
SQLException {
        String sql = "{CALL ChangePaintJobColor(?, ?)}";

        try (CallableStatement statement = connection.prepareCall(sql)) {
            // Set the input parameters
            statement.setInt(1, jobNumber);
            statement.setString(2, newColor);

            // Execute the stored procedure
            statement.execute();
        }
    }

    private static void importCustomersFromFile(Connection connection, String fileName) throws
IOException, SQLException {
        try (BufferedReader reader = new BufferedReader(new FileReader(fileName))) {
            String line;
            while ((line = reader.readLine()) != null) {
                // Assuming each line in the file contains customer data in the format: name,address,category
                String[] parts = line.split(",");
                if (parts.length == 3) {
```

```
                String name = parts[0];
                String address = parts[1];
                int category = Integer.parseInt(parts[2]);

                // Insert customer into the database
                insertCustomer(connection, name, address, category);
            }
        }
    }
}


    private static void exportCustomersToFile(Connection connection, int minCategory, int maxCategory,
String fileName) throws SQLException, IOException {
        String sql = "{CALL GetCustomersByCategoryRange(?, ?)}";
        try (CallableStatement statement = connection.prepareCall(sql)) {
            // Set the input parameters
            statement.setInt(1, minCategory);
            statement.setInt(2, maxCategory);

            // Execute the stored procedure
            ResultSet resultSet = statement.executeQuery();

            // Write the results to the output file
            try (BufferedWriter writer = new BufferedWriter(new FileWriter(fileName))) {
                while (resultSet.next()) {
                    String name = resultSet.getString("name");
                    String address = resultSet.getString("address");
                    int category = resultSet.getInt("category");
```

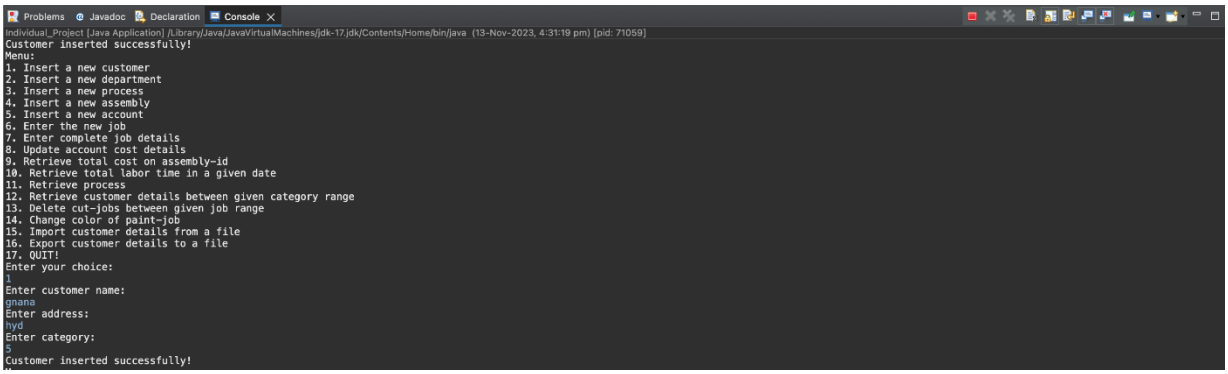
```
                // Write customer data to the file
                writer.write(String.format("%s,%s,%d%n", name, address, category));
            }
        }
    }
}
```

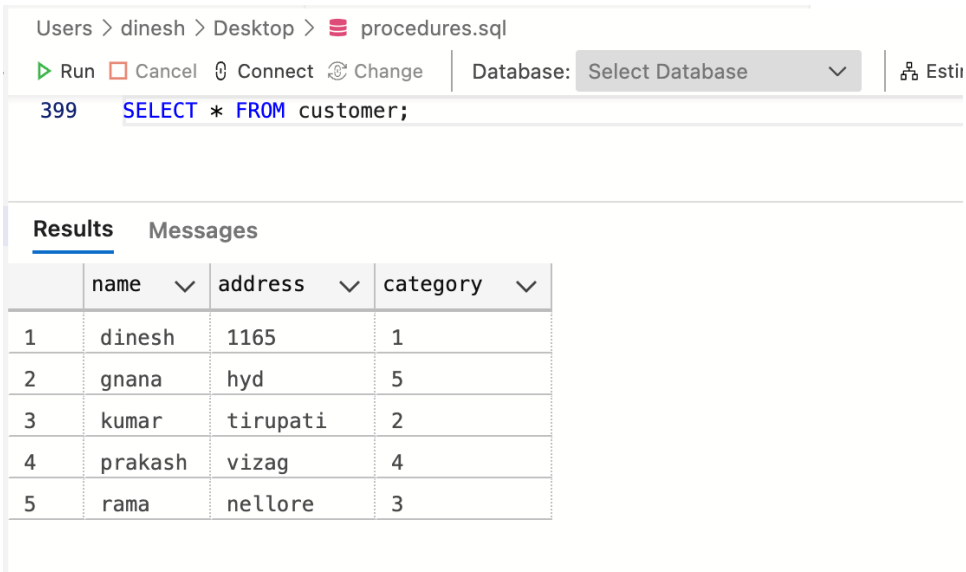
Task 6. Java program Execution

6.1 Screenshots showing the testing of query 1

Query1:

Java Code Screenshot:

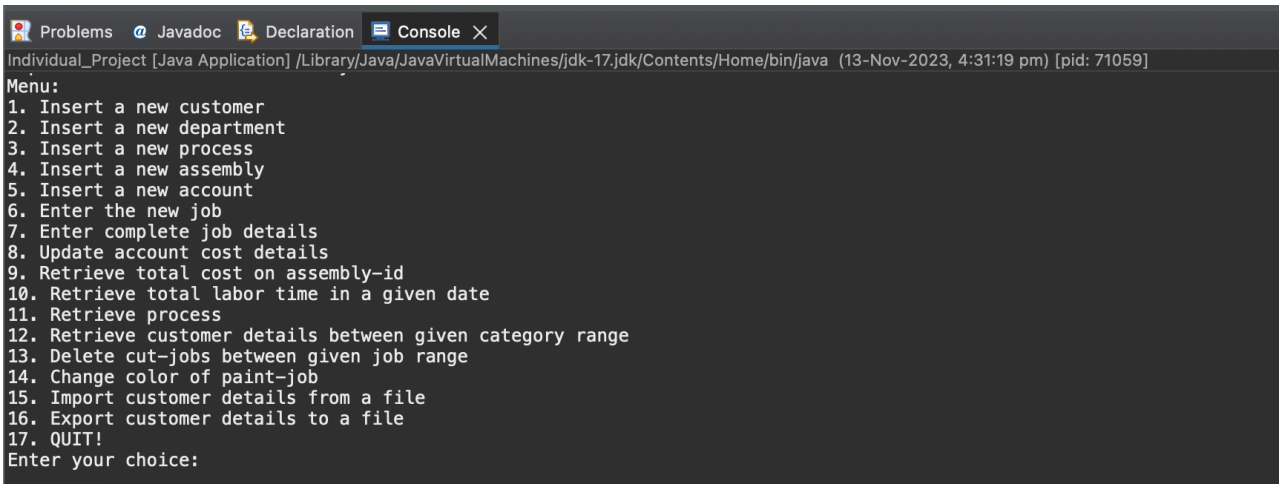


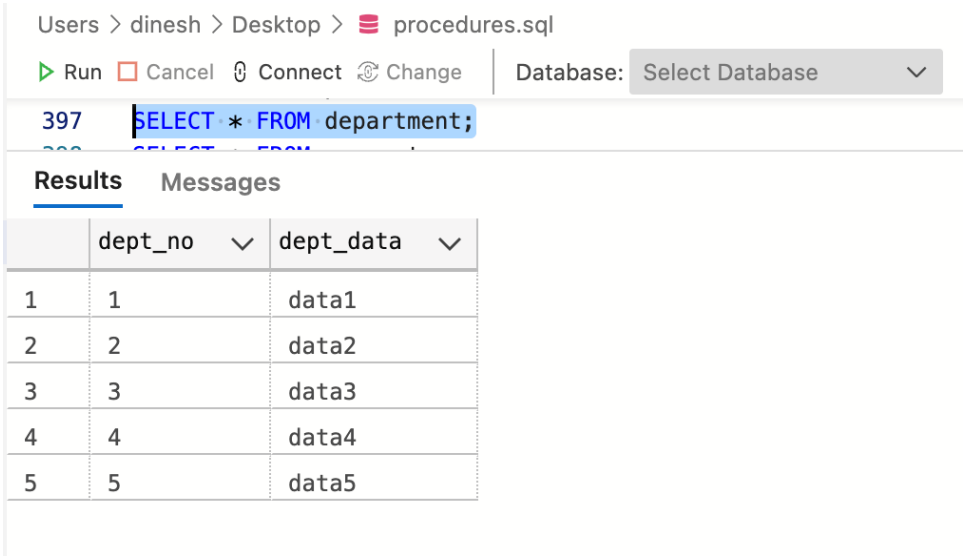


	name	address	category
1	dinesh	1165	1
2	gnana	hyd	5
3	kumar	tirupati	2
4	prakash	vizag	4
5	rama	nellore	3

6.2 Screenshots showing the testing of query 2

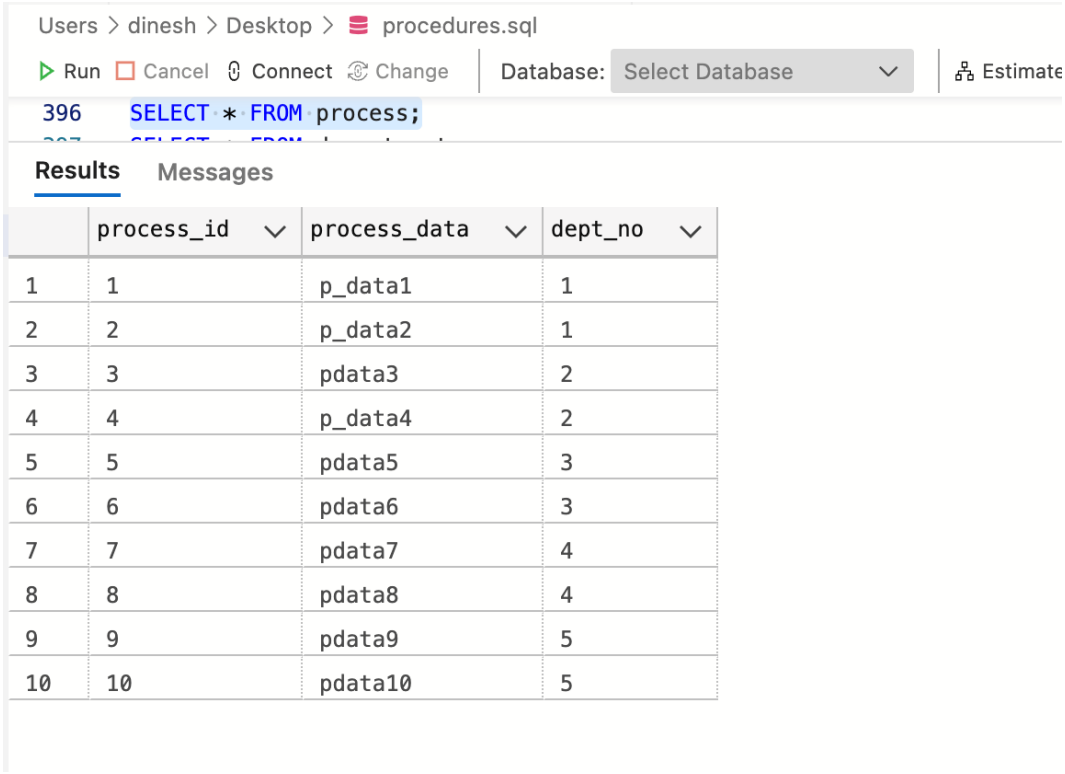
Query 2:





6.3 Screenshots showing the testing of query 3

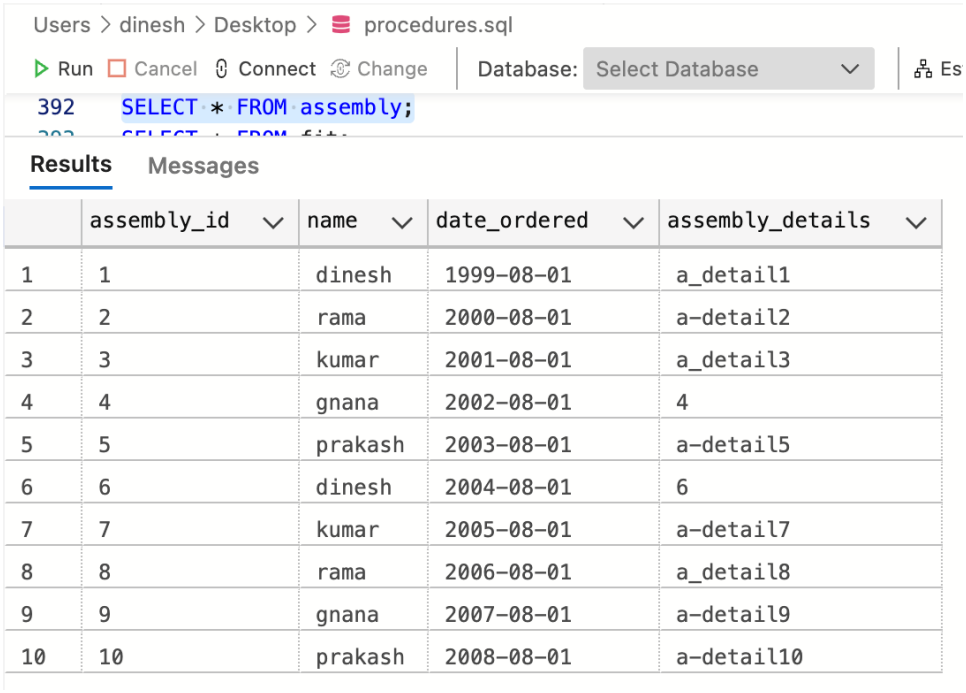
Query 3:



	process_id	process_data	dept_no
1	1	p_data1	1
2	2	p_data2	1
3	3	pdata3	2
4	4	p_data4	2
5	5	pdata5	3
6	6	pdata6	3
7	7	pdata7	4
8	8	pdata8	4
9	9	pdata9	5
10	10	pdata10	5

6.4 Screenshots showing the testing of query 4

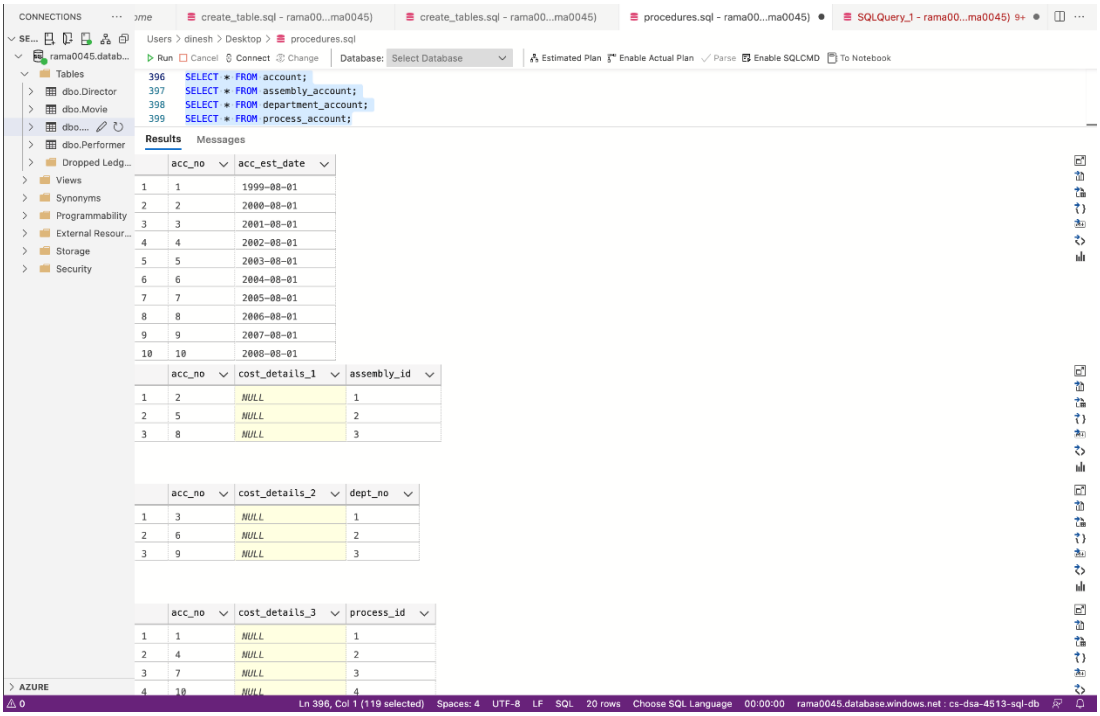
Query 4:



	assembly_id	name	date_ordered	assembly_details
1	1	dinesh	1999-08-01	a_detail1
2	2	rama	2000-08-01	a-detail2
3	3	kumar	2001-08-01	a_detail3
4	4	gnana	2002-08-01	4
5	5	prakash	2003-08-01	a-detail5
6	6	dinesh	2004-08-01	6
7	7	kumar	2005-08-01	a-detail7
8	8	rama	2006-08-01	a_detail8
9	9	gnana	2007-08-01	a-detail9
10	10	prakash	2008-08-01	a-detail10

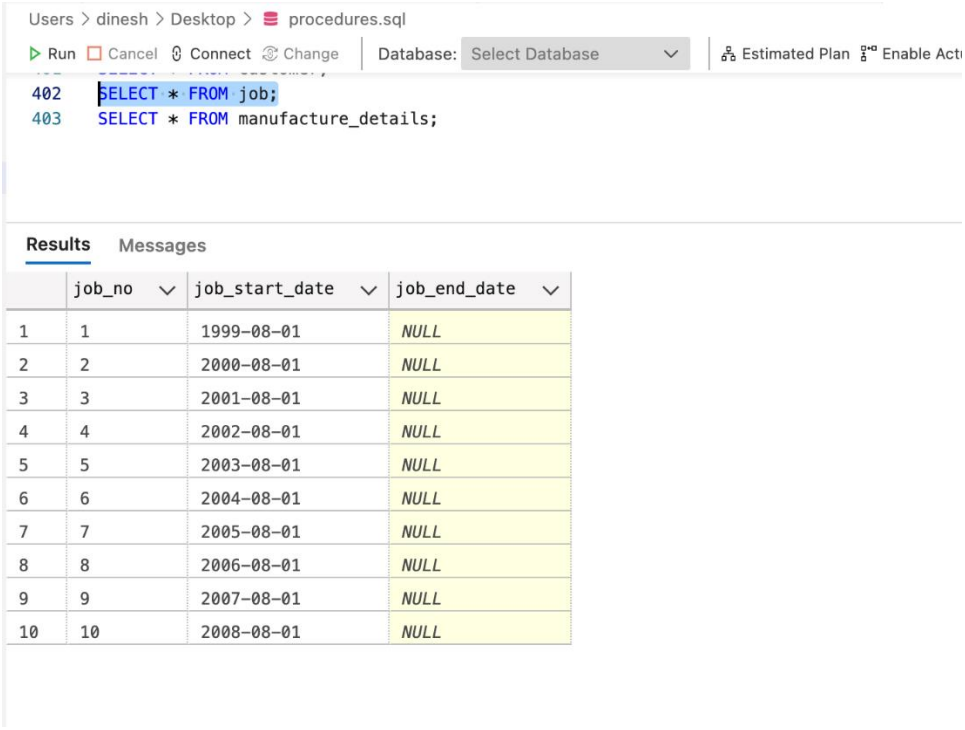
6.5 Screenshots showing the testing of query 5

Query 5:

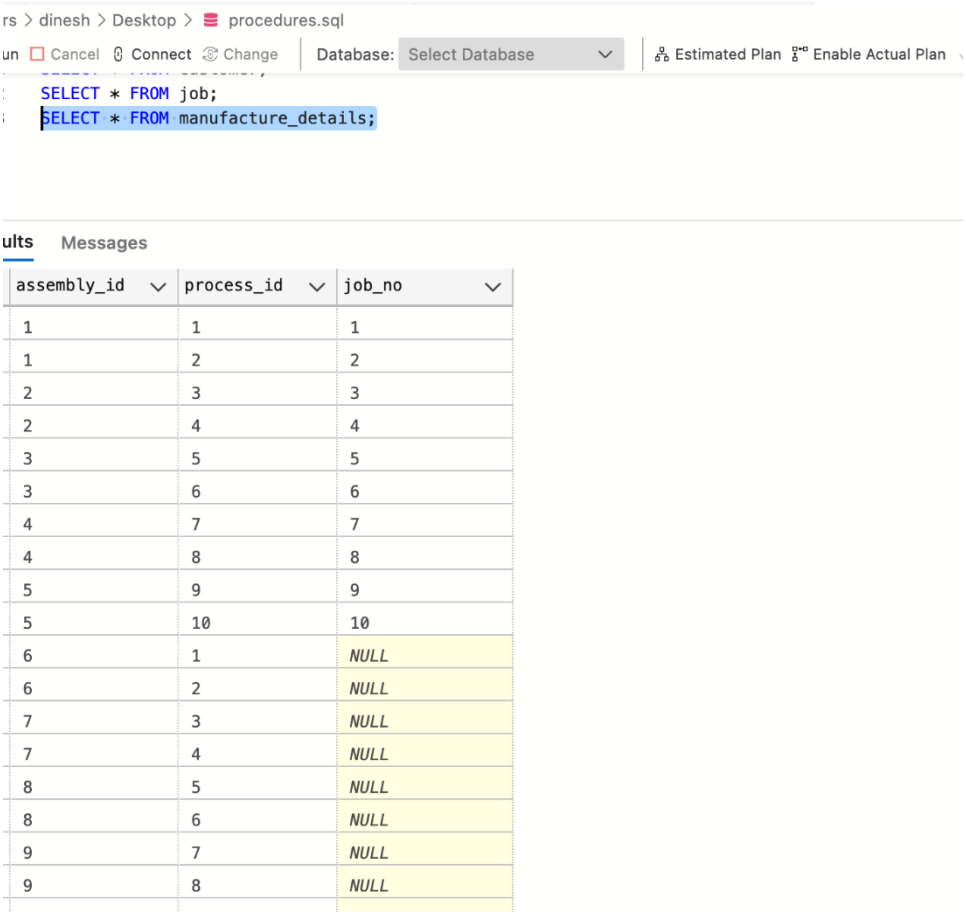


6.6 Screenshots showing the testing of query 6

Query 6:

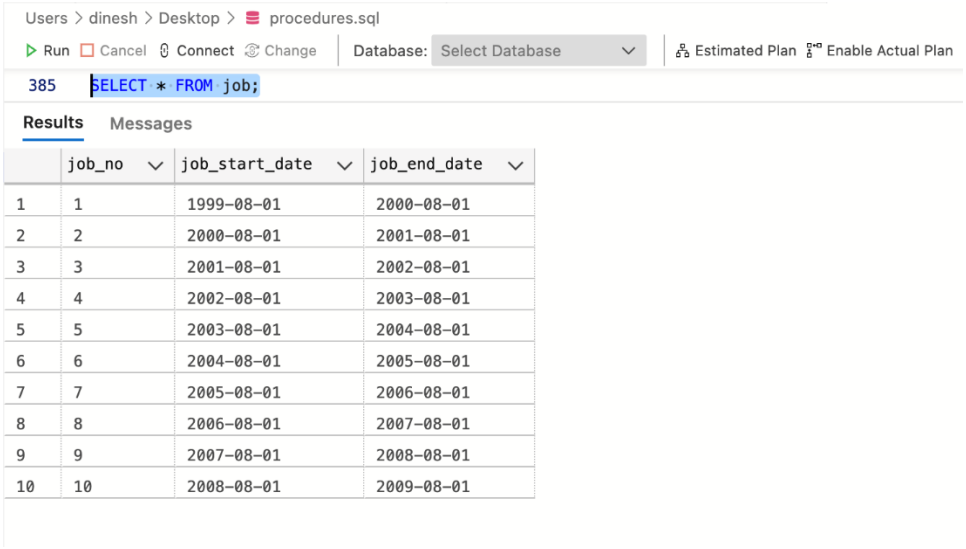


	job_no	job_start_date	job_end_date
1	1	1999-08-01	NULL
2	2	2000-08-01	NULL
3	3	2001-08-01	NULL
4	4	2002-08-01	NULL
5	5	2003-08-01	NULL
6	6	2004-08-01	NULL
7	7	2005-08-01	NULL
8	8	2006-08-01	NULL
9	9	2007-08-01	NULL
10	10	2008-08-01	NULL

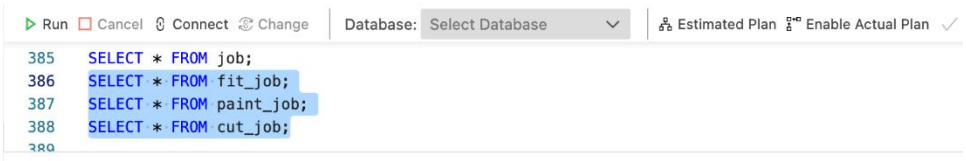

rs > dinesh > Desktop > procedures.sql
un Cancel Connect Change
Database: Select Database
Estimated Plan
Enable Actual Plan
SELECT * FROM job;
SELECT * FROM manufacture_details;
ults
Messages
assembly_id
process_id
job_no
1 1 1
1 2 2
2 3 3
2 4 4
3 5 5
3 6 6
4 7 7
4 8 8
5 9 9
5 10 10
6 1 NULL
6 2 NULL
7 3 NULL
7 4 NULL
8 5 NULL
8 6 NULL
9 7 NULL
9 8 NULL

6.7 Screenshots showing the testing of query 7

Query 7:



	job_no	job_start_date	job_end_date
1	1	1999-08-01	2000-08-01
2	2	2000-08-01	2001-08-01
3	3	2001-08-01	2002-08-01
4	4	2002-08-01	2003-08-01
5	5	2003-08-01	2004-08-01
6	6	2004-08-01	2005-08-01
7	7	2005-08-01	2006-08-01
8	8	2006-08-01	2007-08-01
9	9	2007-08-01	2008-08-01
10	10	2008-08-01	2009-08-01



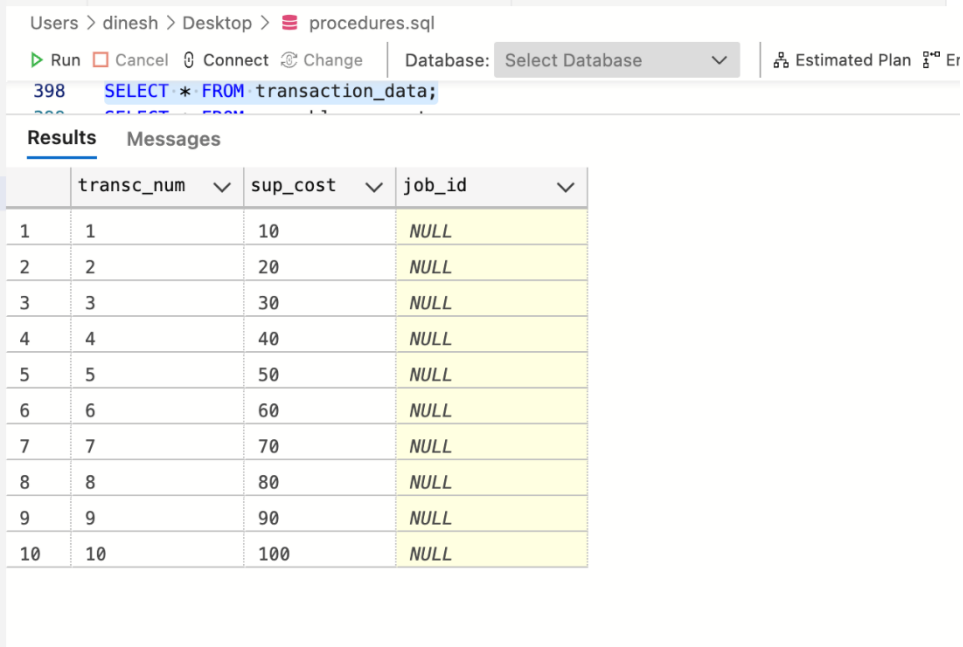
	job_no	labor_time
1	7	8
2	8	23

	job_no	colour	volume	labor_time
1	4	red	5	7
2	5	green	7	8
3	6	blue	9	10
4	10	yellow	8	9

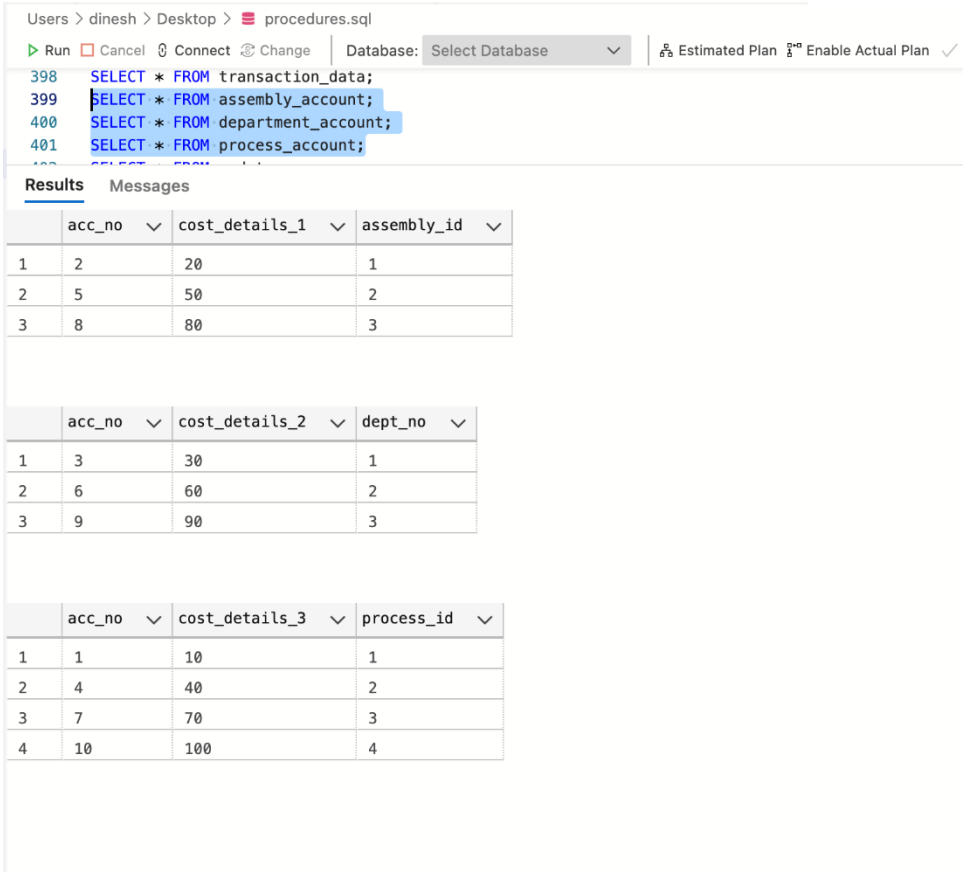
	job_no	machine_type	time_used	material_used	labor_time
1	1	mtype1	5	saw	5
2	2	mtype2	5	knife	5
3	3	mtype3	7	saw	8
4	9	mtype9	89	saw	87

6.8 Screenshots showing the testing of query 8

Query 8:



	transc_num	sup_cost	job_id
1	1	10	NULL
2	2	20	NULL
3	3	30	NULL
4	4	40	NULL
5	5	50	NULL
6	6	60	NULL
7	7	70	NULL
8	8	80	NULL
9	9	90	NULL
10	10	100	NULL



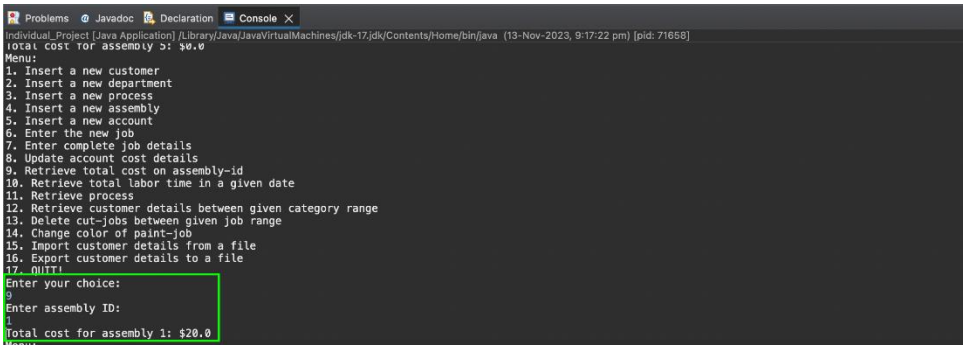
	acc_no	cost_details_1	assembly_id
1	2	20	1
2	5	50	2
3	8	80	3

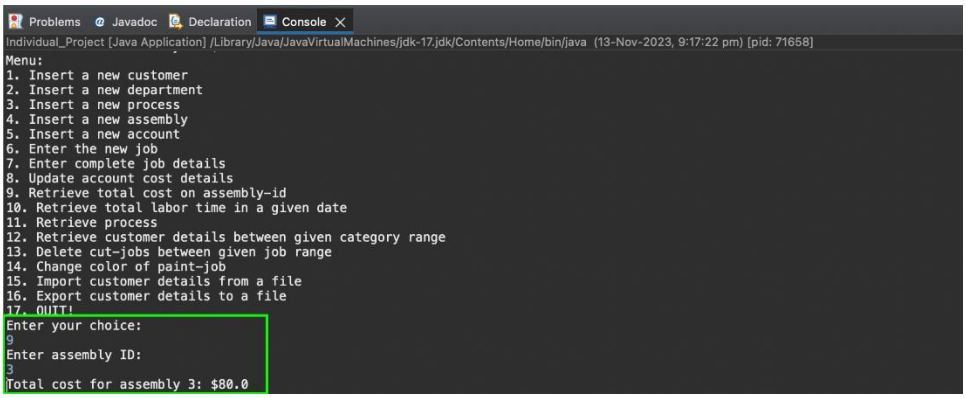
	acc_no	cost_details_2	dept_no
1	3	30	1
2	6	60	2
3	9	90	3

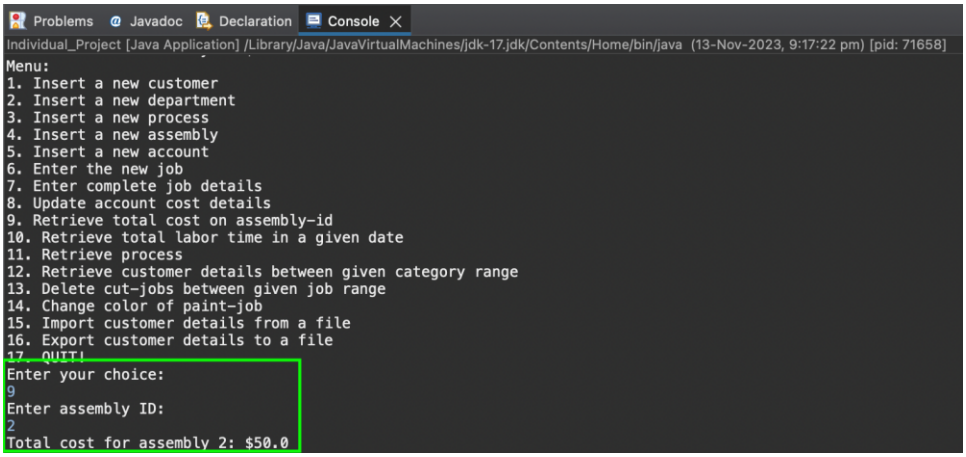
	acc_no	cost_details_3	process_id
1	1	10	1
2	4	40	2
3	7	70	3
4	10	100	4

6.9 Screenshots showing the testing of query 9

Query 9:




Problems
Javadoc
Declaration
Console
Individual_Project [Java Application] /Library/Java/JavaVirtualMachines/jdk-17.jdk/Contents/Home/bin/java (13-Nov-2023, 9:17:22 pm) [pid: 71658]
Menu:
1. Insert a new customer
2. Insert a new department
3. Insert a new process
4. Insert a new assembly
5. Insert a new account
6. Enter the new job
7. Enter complete job details
8. Update account cost details
9. Retrieve total cost on assembly-id
10. Retrieve total labor time in a given date
11. Retrieve process
12. Retrieve customer details between given category range
13. Delete cut-jobs between given job range
14. Change color of paint-job
15. Import customer details from a file
16. Export customer details to a file
17. QUIT!
Enter your choice:
9
Enter assembly ID:
3
Total cost for assembly 3: $80.0


Problems
Javadoc
Declaration
Console
Individual_Project [Java Application] /Library/Java/JavaVirtualMachines/jdk-17.jdk/Contents/Home/bin/java (13-Nov-2023, 9:17:22 pm) [pid: 71658]
Menu:
1. Insert a new customer
2. Insert a new department
3. Insert a new process
4. Insert a new assembly
5. Insert a new account
6. Enter the new job
7. Enter complete job details
8. Update account cost details
9. Retrieve total cost on assembly-id
10. Retrieve total labor time in a given date
11. Retrieve process
12. Retrieve customer details between given category range
13. Delete cut-jobs between given job range
14. Change color of paint-job
15. Import customer details from a file
16. Export customer details to a file
17. QUIT!
Enter your choice:
9
Enter assembly ID:
2
Total cost for assembly 2: $50.0

Users > dinesh > Desktop > procedures.sql

Run | Cancel | Connect | Change | Database: Select Database

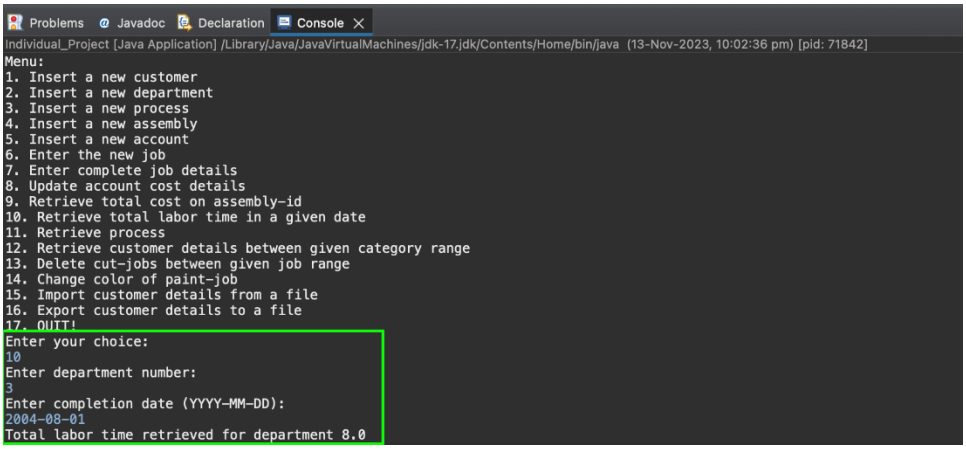
```
399  SELECT * FROM assembly_account;
```

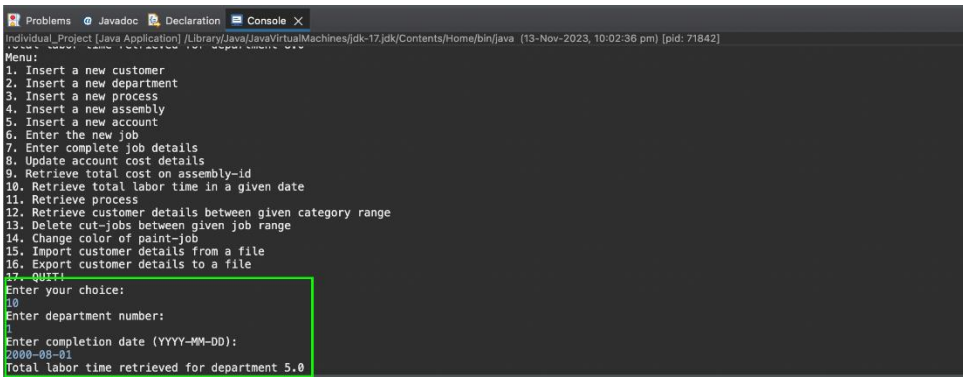
Results | Messages

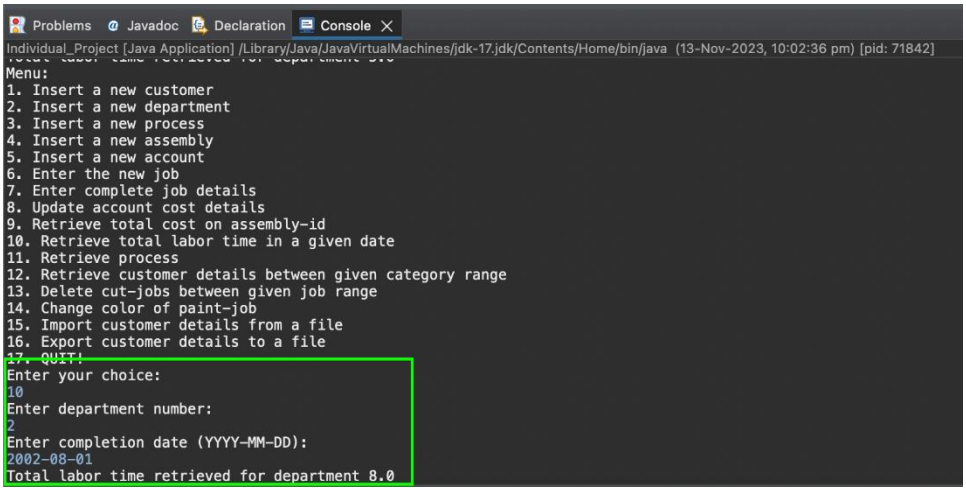
	acc_no	cost_details_1	assembly_id
1	2	20	1
2	5	50	2
3	8	80	3

6.10 Screenshots showing the testing of query 10

Query 10:

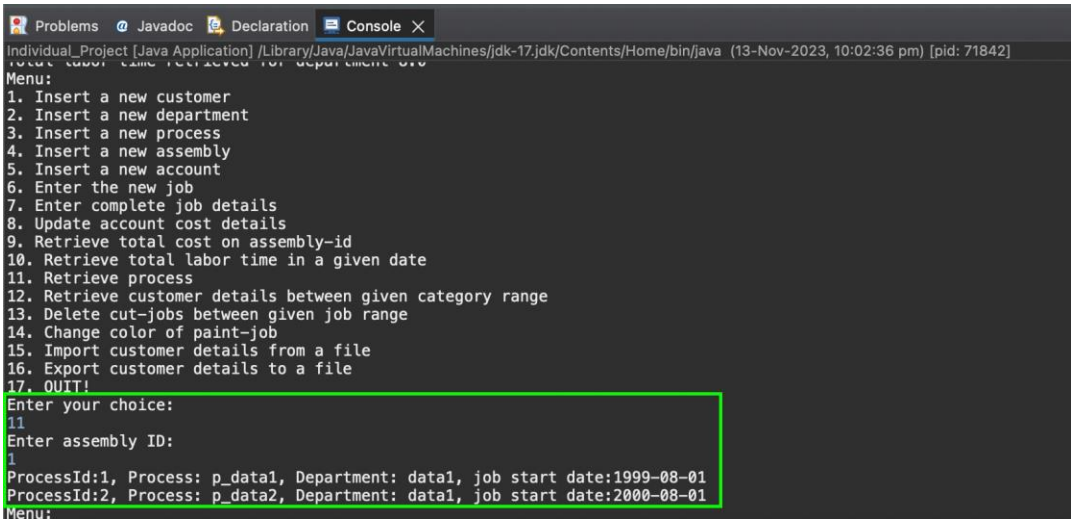


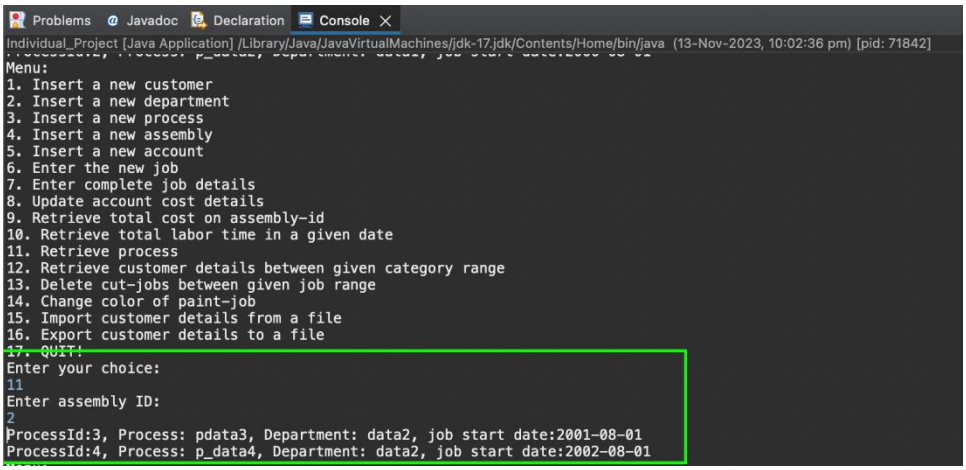

Problems
Javadoc
Declaration
Console
Individual_Project [Java Application] /Library/Java/JavaVirtualMachines/jdk-17.jdk/Contents/Home/bin/java (13-Nov-2023, 10:02:36 pm) [pid: 71842]
Menu:
1. Insert a new customer
2. Insert a new department
3. Insert a new process
4. Insert a new assembly
5. Insert a new account
6. Enter the new job
7. Enter complete job details
8. Update account cost details
9. Retrieve total cost on assembly-id
10. Retrieve total labor time in a given date
11. Retrieve process
12. Retrieve customer details between given category range
13. Delete cut-jobs between given job range
14. Change color of paint-job
15. Import customer details from a file
16. Export customer details to a file
17. QUIT!
Enter your choice:
10
Enter department number:
1
Enter completion date (YYYY-MM-DD):
2000-08-01
Total labor time retrieved for department 5.0

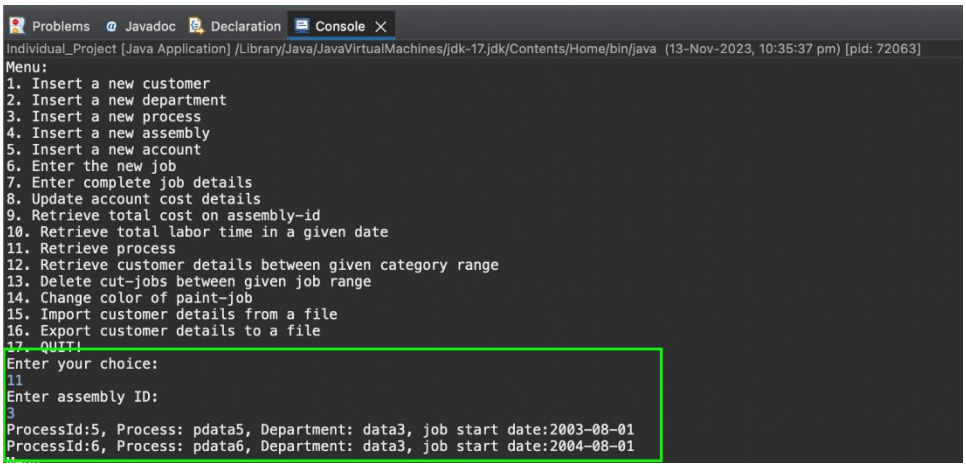

Problems
Javadoc
Declaration
Console
Individual_Project [Java Application] /Library/Java/JavaVirtualMachines/jdk-17.jdk/Contents/Home/bin/java (13-Nov-2023, 10:02:36 pm) [pid: 71842]
Menu:
1. Insert a new customer
2. Insert a new department
3. Insert a new process
4. Insert a new assembly
5. Insert a new account
6. Enter the new job
7. Enter complete job details
8. Update account cost details
9. Retrieve total cost on assembly-id
10. Retrieve total labor time in a given date
11. Retrieve process
12. Retrieve customer details between given category range
13. Delete cut-jobs between given job range
14. Change color of paint-job
15. Import customer details from a file
16. Export customer details to a file
17. QUIT!
Enter your choice:
10
Enter department number:
2
Enter completion date (YYYY-MM-DD):
2002-08-01
Total labor time retrieved for department 8.0

6.11 Screenshots showing the testing of query 11

Query 11:

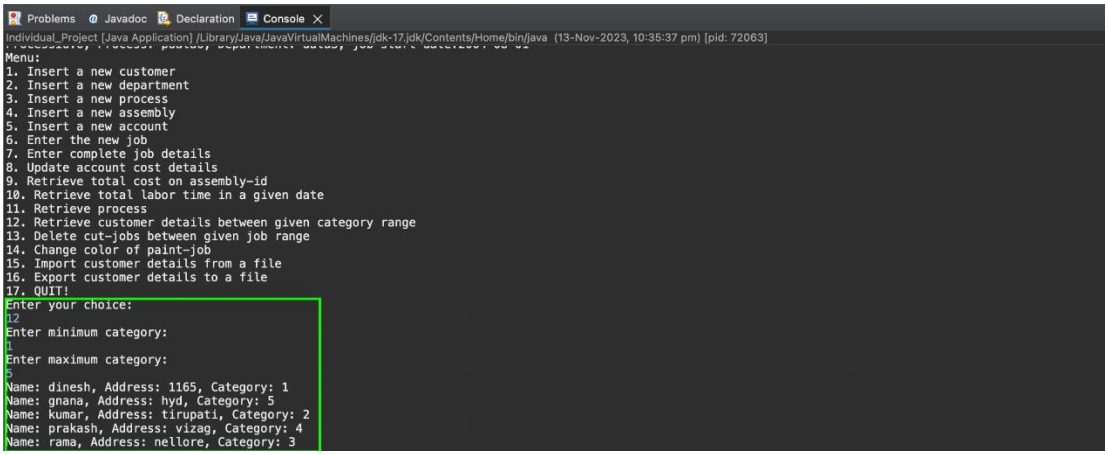


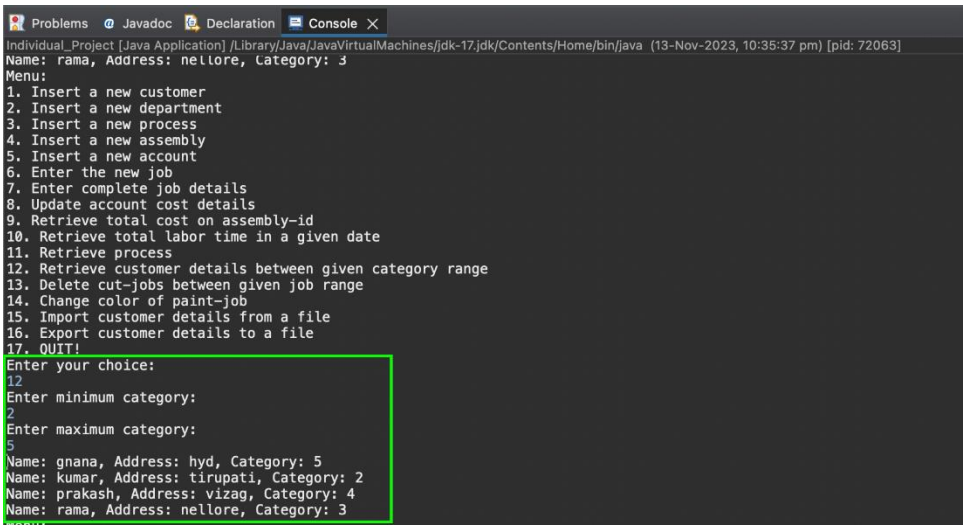


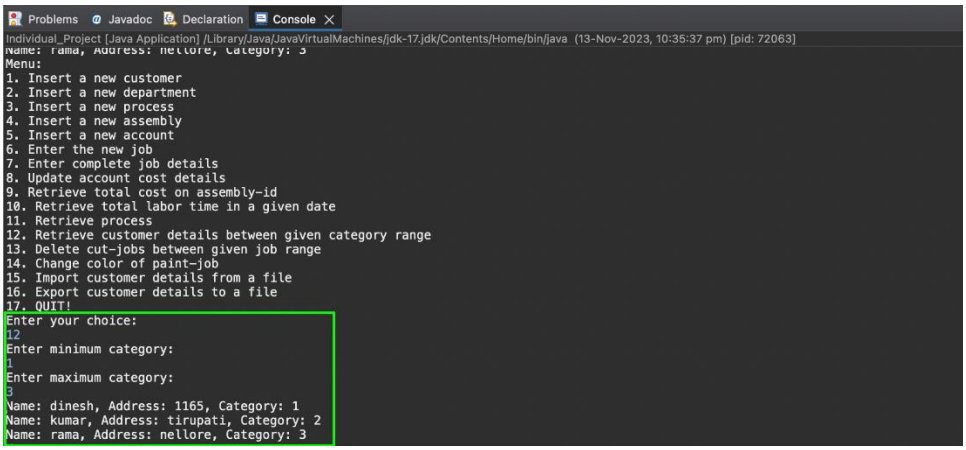


6.12 Screenshots showing the testing of query 12

Query 12:

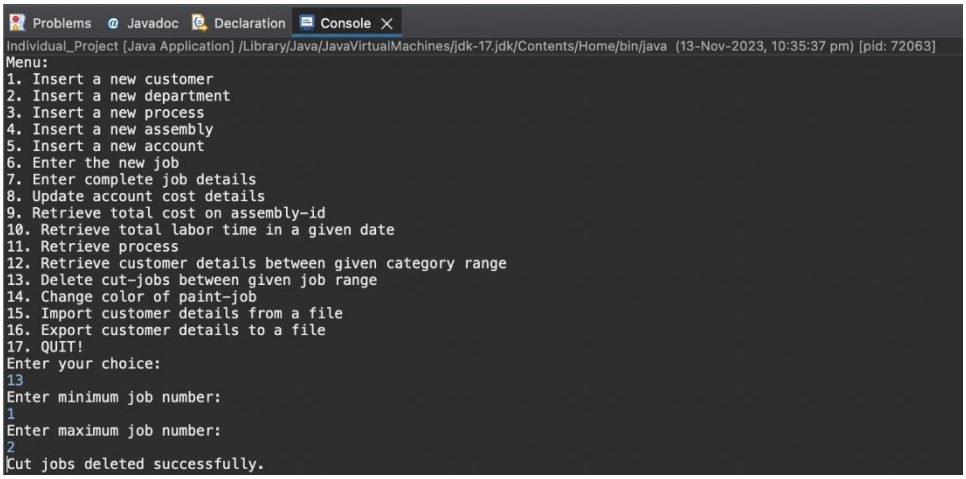


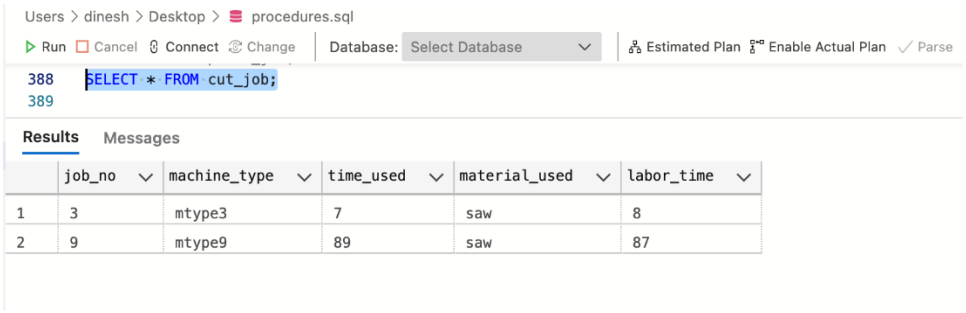

Problems
Javadoc
Declaration
Console
Individual_Project [Java Application] /Library/Java/JavaVirtualMachines/jdk-17.jdk/Contents/Home/bin/java (13-Nov-2023, 10:35:37 pm) [pid: 72063]
Name: rama, Address: nellore, Category: 3
Menu:
1. Insert a new customer
2. Insert a new department
3. Insert a new process
4. Insert a new assembly
5. Insert a new account
6. Enter the new job
7. Enter complete job details
8. Update account cost details
9. Retrieve total cost on assembly-id
10. Retrieve total labor time in a given date
11. Retrieve process
12. Retrieve customer details between given category range
13. Delete cut-jobs between given job range
14. Change color of paint-job
15. Import customer details from a file
16. Export customer details to a file
17. QUIT!
Enter your choice:
12
Enter minimum category:
2
Enter maximum category:
5
Name: gnana, Address: hyd, Category: 5
Name: kumar, Address: tirupati, Category: 2
Name: prakash, Address: vizag, Category: 4
Name: rama, Address: nellore, Category: 3


Problems
Javadoc
Declaration
Console
Individual_Project [Java Application] /Library/Java/JavaVirtualMachines/jdk-17.jdk/Contents/Home/bin/java (13-Nov-2023, 10:35:37 pm) [pid: 72063]
Name: rama, Address: nellore, Category: 3
Menu:
1. Insert a new customer
2. Insert a new department
3. Insert a new process
4. Insert a new assembly
5. Insert a new account
6. Enter the new job
7. Enter complete job details
8. Update account cost details
9. Retrieve total cost on assembly-id
10. Retrieve total labor time in a given date
11. Retrieve process
12. Retrieve customer details between given category range
13. Delete cut-jobs between given job range
14. Change color of paint-job
15. Import customer details from a file
16. Export customer details to a file
17. QUIT!
Enter your choice:
12
Enter minimum category:
1
Enter maximum category:
3
Name: dinesh, Address: 1165, Category: 1
Name: kumar, Address: tirupati, Category: 2
Name: rama, Address: nellore, Category: 3

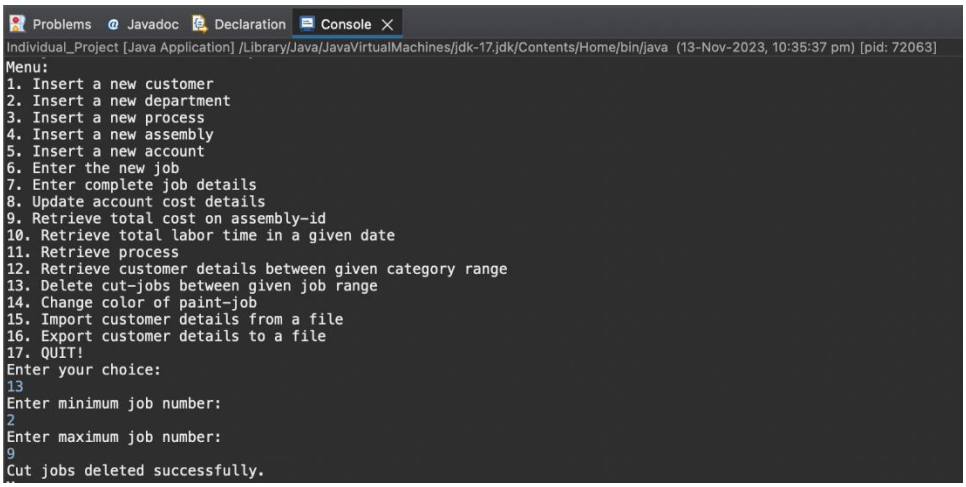
6.13 Screenshots showing the testing of query 13

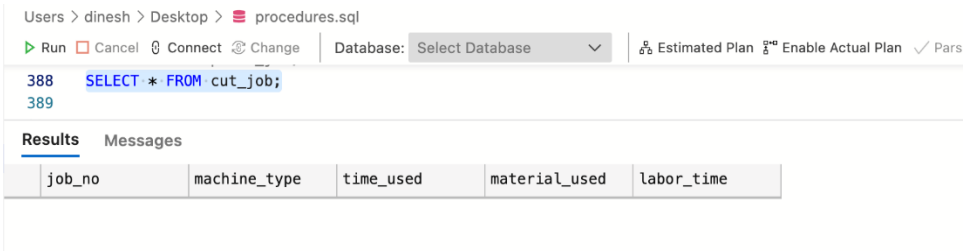
Query 13:

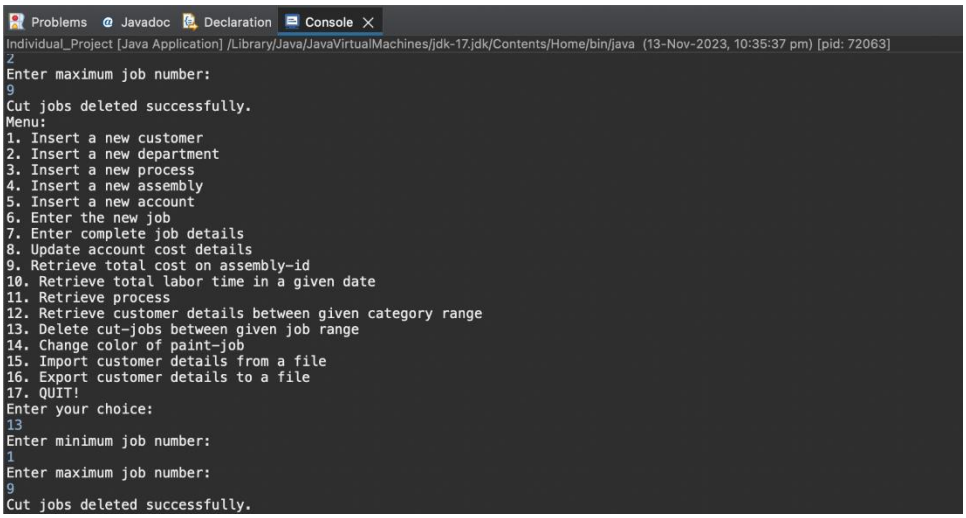


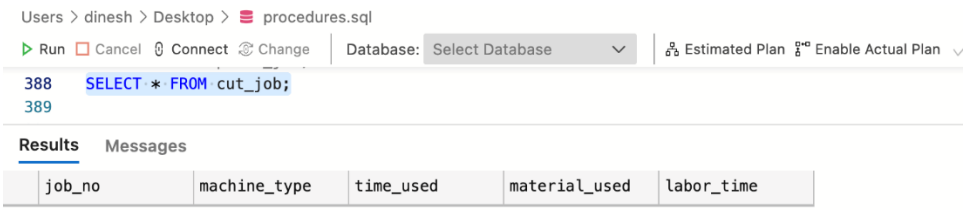


	job_no	machine_type	time_used	material_used	labor_time
1	3	mtype3	7	saw	8
2	9	mtype9	89	saw	87


Problems
Javadoc
Declaration
Console
Individual_Project [Java Application] /Library/Java/JavaVirtualMachines/jdk-17.jdk/Contents/Home/bin/java (13-Nov-2023, 10:35:37 pm) [pid: 72063]
Menu:
1. Insert a new customer
2. Insert a new department
3. Insert a new process
4. Insert a new assembly
5. Insert a new account
6. Enter the new job
7. Enter complete job details
8. Update account cost details
9. Retrieve total cost on assembly-id
10. Retrieve total labor time in a given date
11. Retrieve process
12. Retrieve customer details between given category range
13. Delete cut-jobs between given job range
14. Change color of paint-job
15. Import customer details from a file
16. Export customer details to a file
17. QUIT!
Enter your choice:
13
Enter minimum job number:
2
Enter maximum job number:
9
Cut jobs deleted successfully.

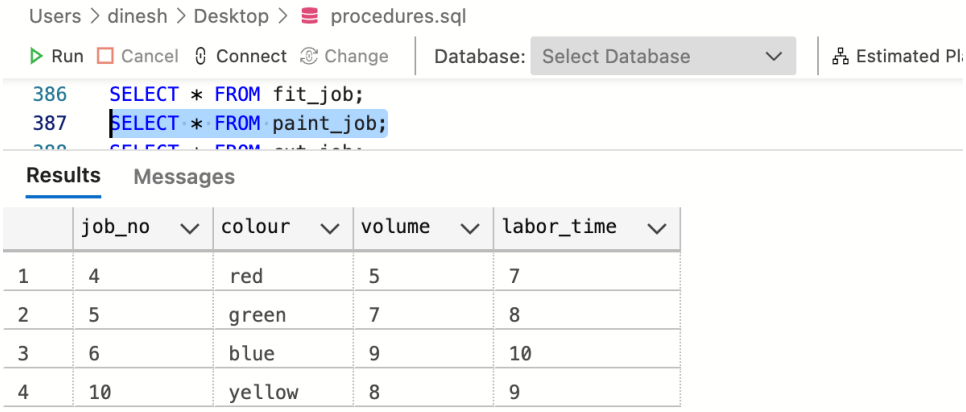

Users > dinesh > Desktop > procedures.sql
Run
Cancel
Connect
Change
Database:
Select Database
Estimated Plan
Enable Actual Plan
388 SELECT * FROM cut_job;
389
Results
Messages
job_no
machine_type
time_used
material_used
labor_time


Problems
Javadoc
Declaration
Console
Individual_Project [Java Application] /Library/Java/JavaVirtualMachines/jdk-17.jdk/Contents/Home/bin/java (13-Nov-2023, 10:35:37 pm) [pid: 72063]
2
Enter maximum job number:
9
Cut jobs deleted successfully.
Menu:
1. Insert a new customer
2. Insert a new department
3. Insert a new process
4. Insert a new assembly
5. Insert a new account
6. Enter the new job
7. Enter complete job details
8. Update account cost details
9. Retrieve total cost on assembly-id
10. Retrieve total labor time in a given date
11. Retrieve process
12. Retrieve customer details between given category range
13. Delete cut-jobs between given job range
14. Change color of paint-job
15. Import customer details from a file
16. Export customer details to a file
17. QUIT!
Enter your choice:
13
Enter minimum job number:
1
Enter maximum job number:
9
Cut jobs deleted successfully.

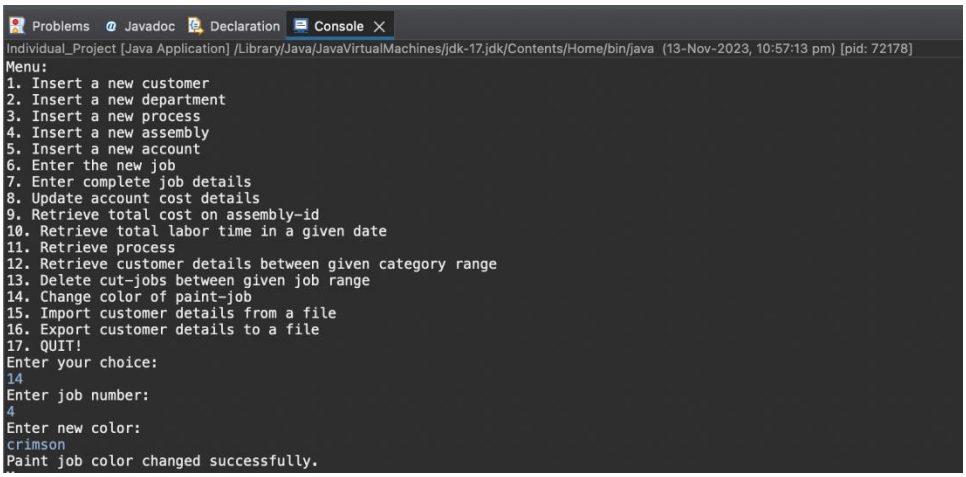


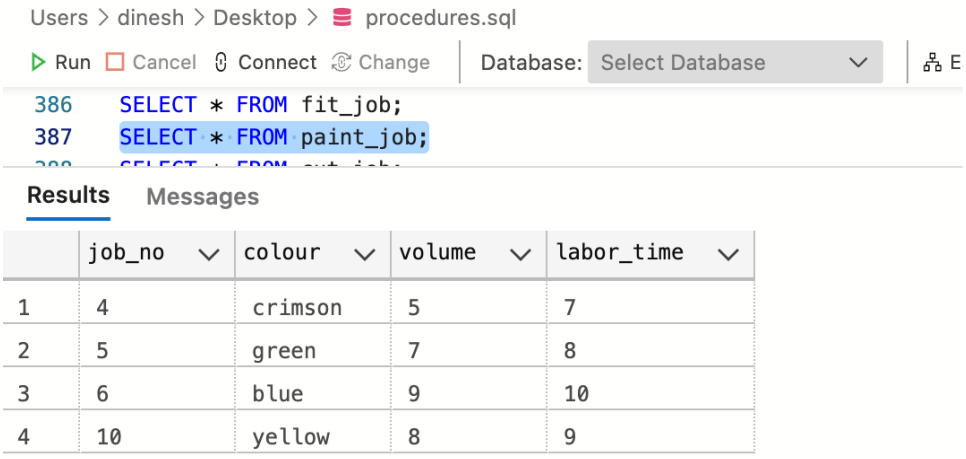
6.14 Screenshots showing the testing of query 14

Query 14:

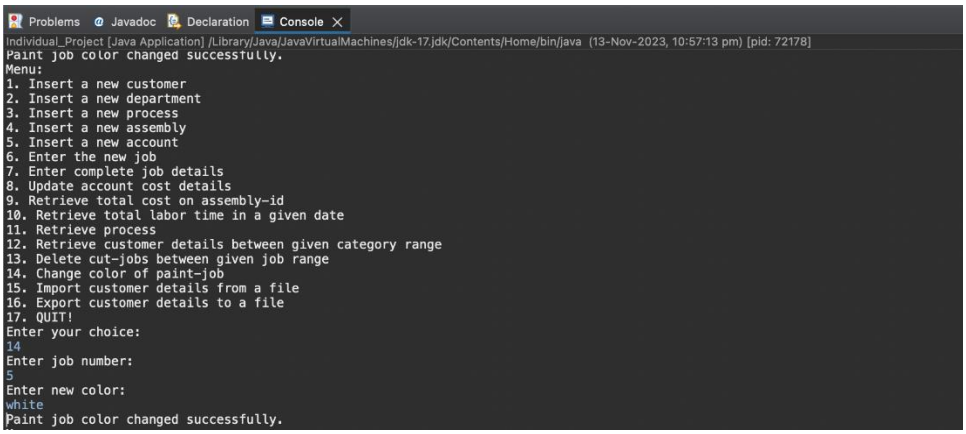


	job_no	colour	volume	labor_time
1	4	red	5	7
2	5	green	7	8
3	6	blue	9	10
4	10	yellow	8	9





	job_no	colour	volume	labor_time
1	4	crimson	5	7
2	5	green	7	8
3	6	blue	9	10
4	10	yellow	8	9



Users > dinesh > Desktop > procedures.sql

Run Cancel Connect Change | Database: Select Database | Estimated Plan Enable Actual Plan

```
387  SELECT * FROM paint_job;
388  SELECT * FROM cut_job;
```

Results Messages

	job_no	colour	volume	labor_time
1	4	crimson	5	7
2	5	white	7	8
3	6	blue	9	10
4	10	yellow	8	9

Problems Javadoc Declaration Console

```
Individual_Project [Java Application] /Library/Java/JavaVirtualMachines/jdk-17.jdk/Contents/Home/bin/java (13-Nov-2023, 10:57:13 pm) [pid: 72178]
Paint job color changed successfully.
Menu:
1. Insert a new customer
2. Insert a new department
3. Insert a new process
4. Insert a new assembly
5. Insert a new account
6. Enter the new job
7. Enter complete job details
8. Update account cost details
9. Retrieve total cost on assembly-id
10. Retrieve total labor time in a given date
11. Retrieve process
12. Retrieve customer details between given category range
13. Delete cut-jobs between given job range
14. Change color of paint-job
15. Import customer details from a file
16. Export customer details to a file
17. QUIT!
Enter your choice:
14
Enter job number:
10
Enter new color:
maroon
Paint job color changed successfully.
```

Users > dinesh > Desktop > procedures.sql

Run Cancel Connect Change | Database: Select Database | Estimated Plan Enable Actual Plan Parse Enable SQLCMD To Notebook

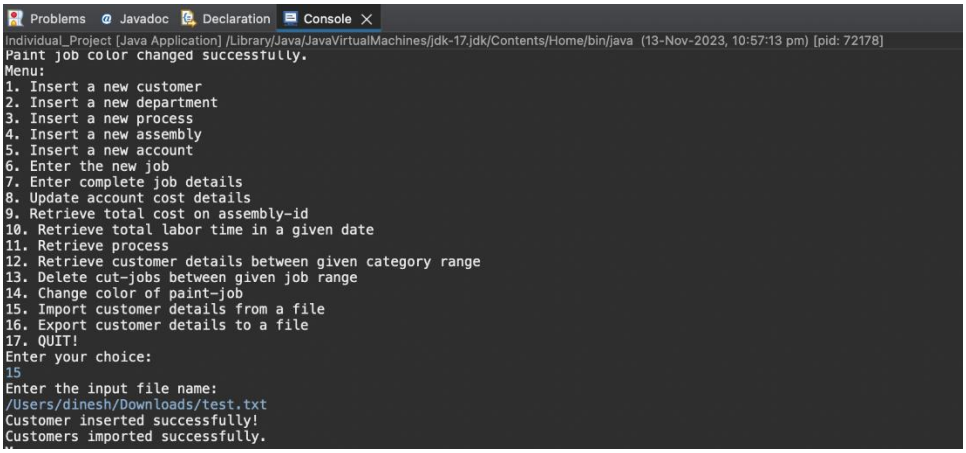
```
387  SELECT * FROM paint_job;
388  SELECT * FROM cut_job;
```

Results Messages

	job_no	colour	volume	labor_time
1	4	crimson	5	7
2	5	white	7	8
3	6	blue	9	10
4	10	maroon	8	9

6.15 Screenshots showing the testing of query 15

Query 15:



Users > dinesh > Desktop > procedures.sql

Run Cancel Connect Change Database: Select Database Estimated Plan Enable Actual Plan Parse Er

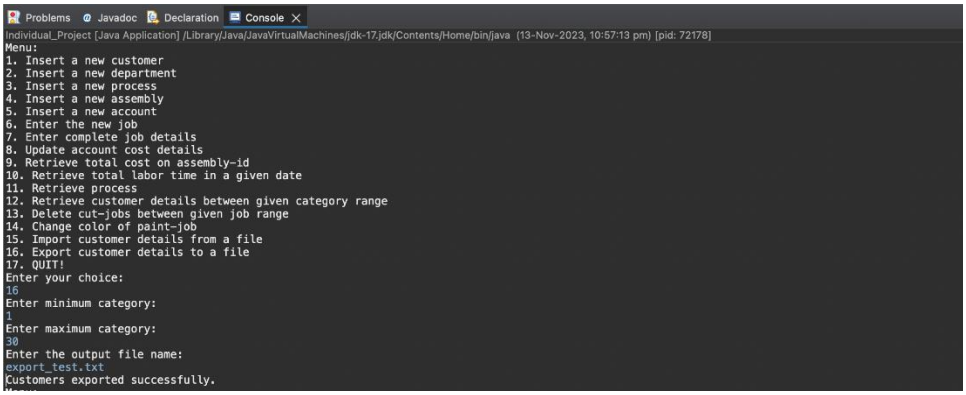
```
402  SELECT * FROM updates;
403  SELECT * FROM customer;
```

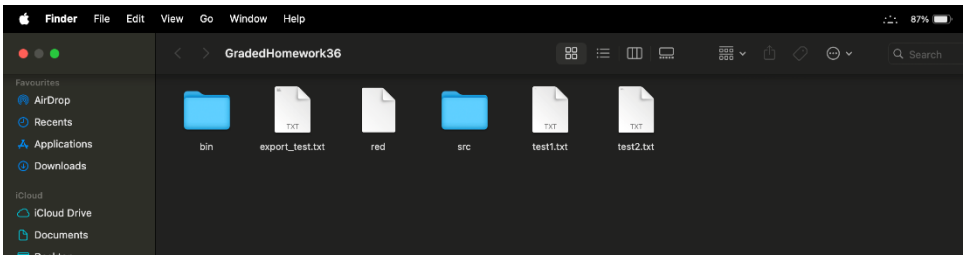
Results Messages

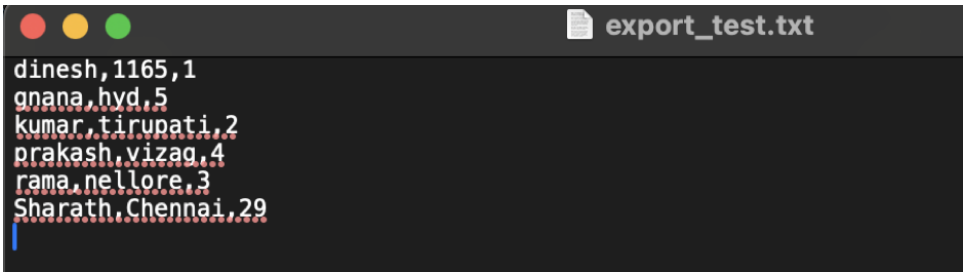
	name	address	category
1	dinesh	1165	1
2	gnana	hyd	5
3	kumar	tirupati	2
4	prakash	vizag	4
5	rama	nellore	3
6	Sharath	Chennai	29

6.16 Screenshots showing the testing of query 16

Query 16:



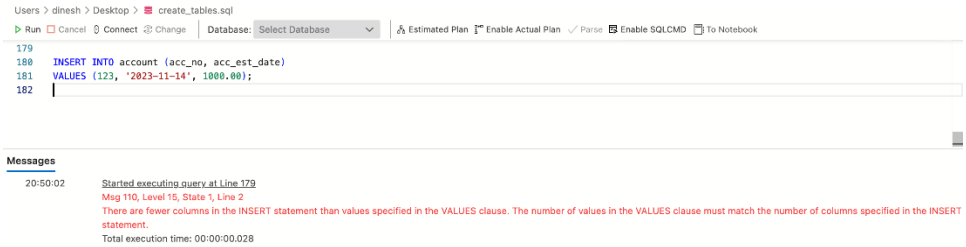

Finder File Edit View Go Window Help
87%
GradedHomework36
Search
Favourites
AirDrop
Recents
Applications
Downloads
iCloud
iCloud Drive
Documents
bin
export_test.txt
red
src
test1.txt
test2.txt


export_test.txt
dinesh,1165,1
gnana,hyd,5
kumar,tirupati,2
prakash,vizag,4
rama,nellore,3
Sharath,Chennai,29

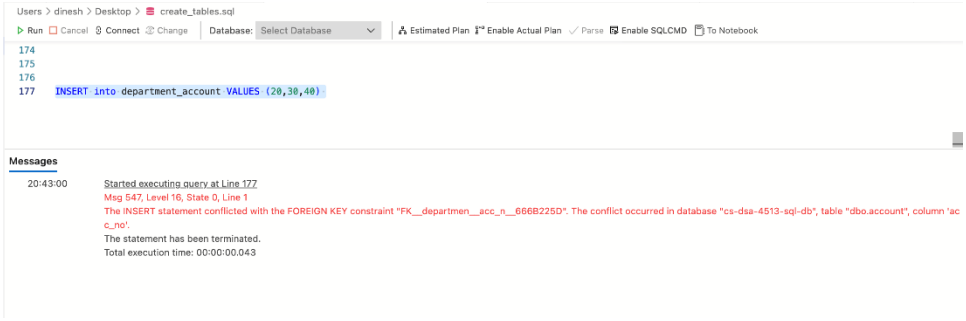
Error Checking:

1) Entering customer that already exists which shows the primary key error.

2) Insert statement error.

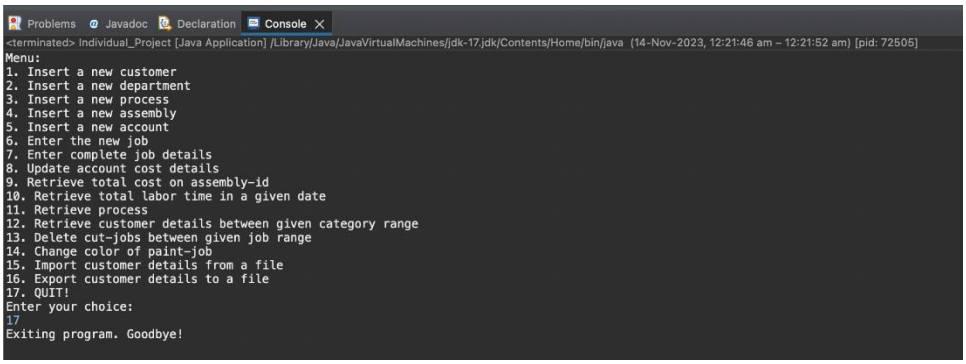


3.) Showing Foreign key erorr



6.17 Screenshots showing the testing of query 17

Query 17: EXIT.



Task 7: Web Database application and its execution

7.1 Web database application source program and screenshots showing and its successful compilation.

Rama_Dinesh_IP_Task7_DataHandlerQuery.java

```
package jsp_azure_test;

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

public class Rama_Dinesh_IP_Task7_DataHandler {
```

```
    private Connection conn;

    // Azure SQL connection credentials
    private String server = "rama0045.database.windows.net";
    private String database = "cs-dsa-4513-sql-db";
    private String username = "rama0045";
    private String password = "Dkronaldo7@";

    // Resulting connection string
    final private String url =
String.format("jdbc:sqlserver://%s:1433;database=%s;user=%s;password=%s;encrypt=true;trustServerC
ertificate=false;hostNameInCertificate=*.database.windows.net;loginTimeout=30;", server, database,
username, password);

    // Initialize and save the database connection
    private void getDBConnection() throws SQLException {
        if (conn != null) {
            return;
        }
        this.conn = DriverManager.getConnection(url);
    }

    // Returning the result from the customer table within a category range
    public ResultSet getCustomersByCategoryRange(int minCategory, int maxCategory) throws
SQLException {
        getDBConnection();
        final String sqlQuery = "SELECT * FROM customer WHERE category BETWEEN ? AND ? ORDER BY
name;";
        final PreparedStatement stmt = conn.prepareStatement(sqlQuery);
        stmt.setInt(1, minCategory);
        stmt.setInt(2, maxCategory);
        return stmt.executeQuery();
    }

    // Inserting a record into the customer table using a stored procedure with the given attribute values
    public boolean addCustomer(String customerName, String customerAddress, int category) throws
SQLException {
        getDBConnection(); // Prepare the database connection
        // Prepare the SQL statement
        final String sqlQuery = "{CALL InsertCustomer(?, ?, ?)}";
        final CallableStatement stmt = conn.prepareCall(sqlQuery);
```

```
        stmt.setString(1, customerName);
        stmt.setString(2, customerAddress);
        stmt.setInt(3, category);

        // Execute the stored procedure
        stmt.execute();

        // Indicate success by returning true
        return true;
    }
}
```

Rama_Dinesh_IP_Task7_InsertCustomer.jsp

```
<%@ page import="java.sql.*" %>
<%@ page import="jsp_azure_test.Rama_Dinesh_IP_Task7_DataHandler" %>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
<title>Customer Insertion</title>
<style>
body {
text-align: center;
}


form {
display: inline-block;
text-align: left;
}
</style>
</head>
```

```
<body>
<h1>Insert Customer</h1>
<form action=" Rama_Dinesh_IP_Task7_ InsertCustomerConfirmation.jsp" method="post">
<label for="customerName">Customer Name:</label>
<input type="text" name="customerName" required><br>


<label for="customerAddress">Customer Address:</label>
<input type="text" name="customerAddress" required><br>


<label for="category">Category:</label>
<input type="number" name="category" required><br>


<input type="submit" value="Submit">
</form>
</body>
</html>
```

Rama_Dinesh_IP_Task7_InsertCustomerConfirmation.jsp

```
<%@ page import="java.sql.*" %>
<%@ page import="jsp_azure_test.Rama_Dinesh_IP_Task7_DataHandler" %>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ page import="java.io.*,java.util.*"%>
<html>
<head>
<title>Customer Insertion Confirmation</title>
<style>
body {
text-align: center;
}
```

```
.output {
display: inline-block;
text-align: left;
}
</style>
</head>
<body>
<div class="output">
<%
String customerName = request.getParameter("customerName");
String customerAddress = request.getParameter("customerAddress");
int category = Integer.parseInt(request.getParameter("category"));


Rama_Dinesh_IP_Task7_DataHandler dataHandler = new Rama_Dinesh_IP_Task7_DataHandler();
boolean isSuccess = dataHandler.addCustomer(customerName, customerAddress, category);


if (isSuccess) {
out.println("<p>Customer inserted successfully!</p>");
} else {
out.println("<p>Failed to insert customer.</p>");
}
%>
</div>
</body>
</html>
```

Rama_Dinesh_IP_Task7_RetrieveCustomers.jsp

```
<%@ page import="java.sql.*" %>
<%@ page import="jsp_azure_test.Rama_Dinesh_IP_Task7_DataHandler" %>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
<title>Retrieve Customers</title>
```

```
<style>
body {
text-align: center;
}


form {
display: inline-block;
text-align: left;
}
</style>
</head>
<body>
<h1>Retrieve Customers</h1>
<form action=" Rama_Dinesh_IP_Task7_ DisplayCustomers.jsp" method="post">
<label for="minCategory">Minimum Category:</label>
<input type="number" name="minCategory" required><br>


<label for="maxCategory">Maximum Category:</label>
<input type="number" name="maxCategory" required><br>


<input type="submit" value="Retrieve Customers">
</form>
</body>
</html>
```

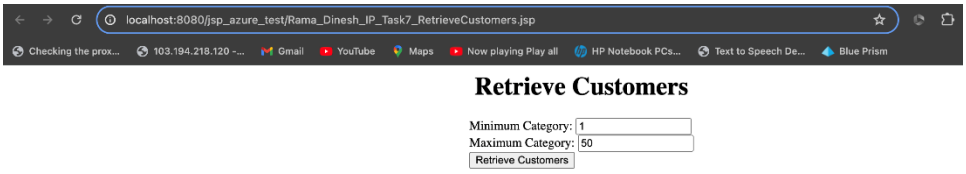
Rama_Dinesh_IP_Task7_ DisplayCustomers.jsp

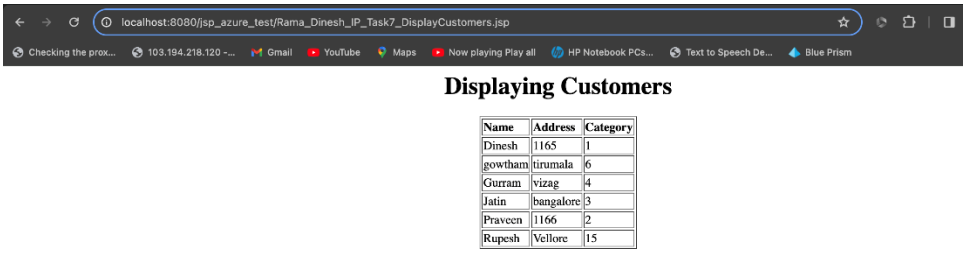
```
<%@ page import="java.sql.*" %>
<%@ page import="jsp_azure_test.Rama_Dinesh_IP_Task7_DataHandler" %>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ page import="java.io.*,java.util.*"%>
<html>
<head>
<title>Display Customers</title>
<style>
body {
text-align: center;
}


table {
margin: 0 auto;
text-align: left;
}
</style>
</head>
<body>
<h1>Displaying Customers</h1>
<table border='1'>
<tr><th>Name</th><th>Address</th><th>Category</th></tr>


<%
int minCategory = Integer.parseInt(request.getParameter("minCategory"));
int maxCategory = Integer.parseInt(request.getParameter("maxCategory"));


Rama_Dinesh_IP_Task7_DataHandler dataHandler = new Rama_Dinesh_IP_Task7_DataHandler();
ResultSet resultSet = dataHandler.getCustomersByCategoryRange(minCategory, maxCategory);


while (resultSet.next()) {
out.println("<tr>");
out.println("<td>" + resultSet.getString("name") + "</td>");
out.println("<td>" + resultSet.getString("address") + "</td>");
out.println("<td>" + resultSet.getInt("category") + "</td>");
out.println("</tr>");
}
```

```
%>


</table>
</body>
</html>
```

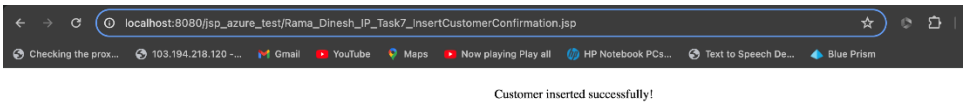
7.2. Screenshots showing the testing of the Web database application

Query 12: Displaying the customer table

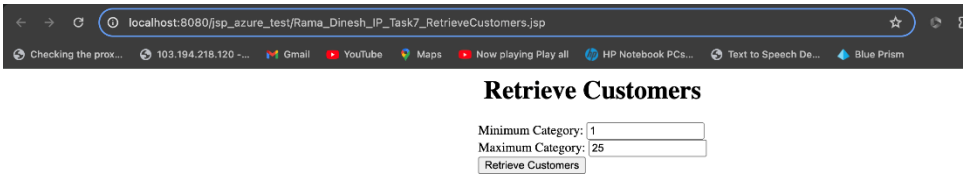




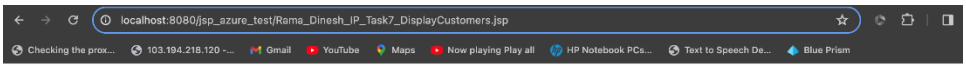
Query 1: Inserting new customer details



Query 12: Again Running this query to display the inserted details.



Here we can see that Akhil, Texas, 10 is added to the customer details table.



Displaying Customers

Name	Address	Category
Akhil	Texas	10
Dinesh	1165	1
gowtham	tirumala	6
Gurram	vizag	4
Jatin	bangalore	3
Praveen	1166	2
Rupesh	Vellore	15