DSA 4513 - A JOB-SHOP ACCOUNTING SYSTEM

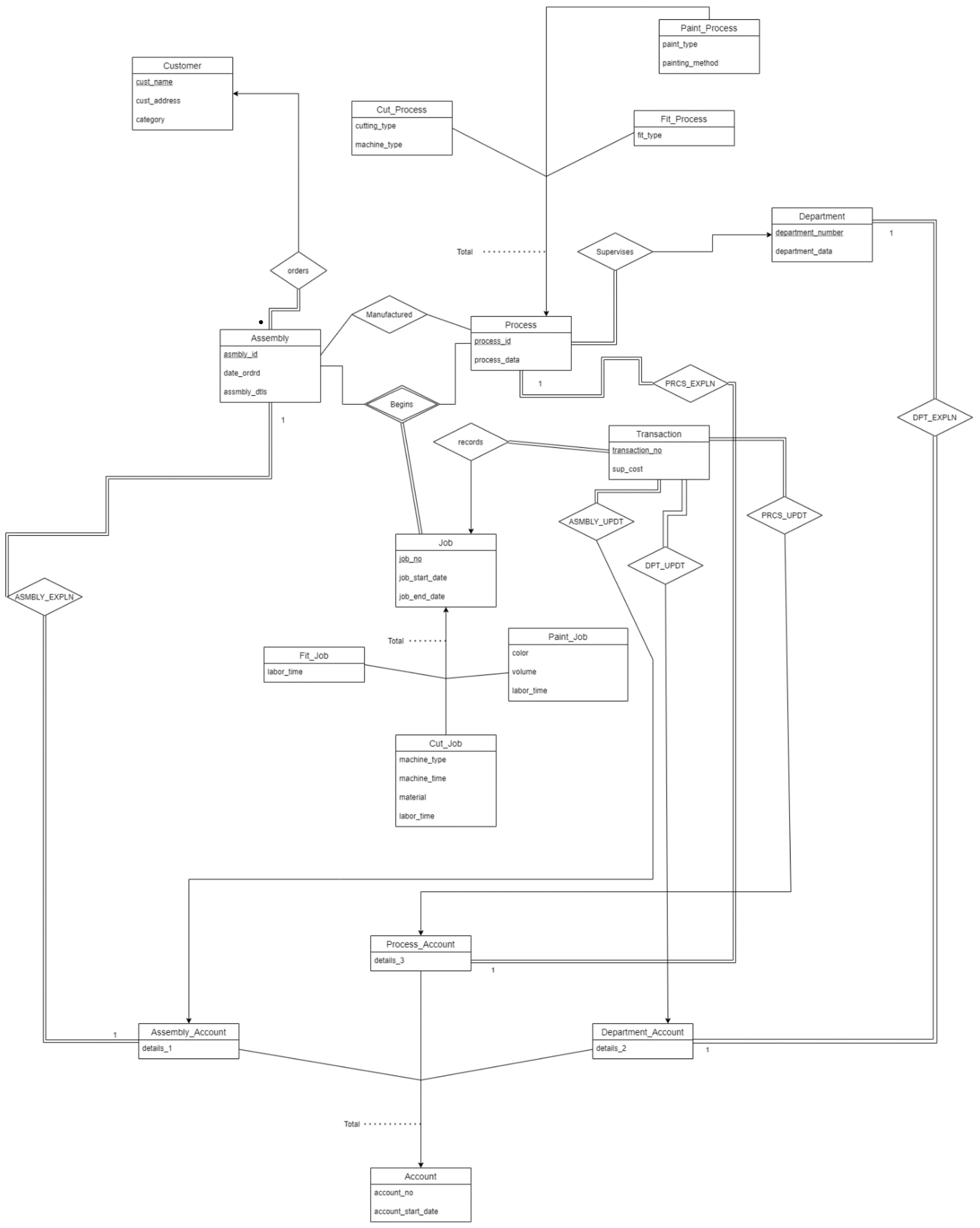
Course Name	: Database Management Systems
Course Number	: DSA 4513
Section Number	: 995
Semester	: Fall
Year	: 2021
Instructor	: Dr. Gia-Loi Gruenwald
Author	: Harinadh Appidi
Student ID	: 113496970
Email ID	: harinadhappidi@ou.edu
Title of Project	: A JOB SHOP ACCOUNTING SYSTEM

DSA 4513 - A JOB-SHOP ACCOUNTING SYSTEM

DSA 4513 - A JOB-SHOP ACCOUNTING SYSTEM

Task 1

1.1. ER Diagram



1.2. Relational Database Schema

i) Customer(<u>cust_name</u>, cust_address, category)

ii) Asmbly(<u>asmbly_id</u>, date_ordrd, asmbly_dtls, cust_name)

iii) Department(<u>department_no</u>, department_data)

iv) Process(<u>process_id</u>, process_data, department_no)

v) Fit_Process(<u>process_id</u>, fit_type)

vi) Cut_Process(<u>process_id</u>, cutting_type, machine_type)

vii) Paint_Process(<u>process_id</u>, paint_type, painting_method)

viii) Job(<u>job_no</u>, job_start_date, job_end_date, asmbly_id, process_id)

ix) Paint_Job(<u>job_no</u>, color, volume,labor_time)

x) Cut_Job(<u>job_no</u>, machine_type, machine_time, material, labor_time)

xi) Fit_Job(<u>job_no</u>, labor_time)

xii) Account(<u>account_no</u>, account_start_date)

xiii) Department_Account(<u>account_no</u>, details_2, department_no)

xiv) Asmbly_Account(<u>account_no</u>, details_1, asmbly_id)

xv) Process_Account(<u>account_no</u>, details_3, process_id)

xvi) Trnsctn(<u>transaction_no</u>, sup_cost, job_no, dep_acno, asmb_acno, prcs_acno)

xvii) Maufactured(<u>asmbly_id</u>, <u>process_id</u>)

TASK 2. DATA DICTIONARY

Note: Varchar max size is +2 bytes more than the defined value in dictionary.

i.e., VARCHAR(20) can have maximum size is 22 bytes and byte range is 0-22 bytes as per MS SQL server documentation.

Reference : https://docs.microsoft.com/en-us/sql/t-sql/data-types/char-and-varchar-transact-sql?view=sql-server-ver15.

Please refer to the remarks section on the above page

Customer

Attribute Name	Attribute Type	Attribute Size	Constraints	References
cust_name	VARCHAR(20)	22	PRIMARY KEY	
cust_address	VARCHAR(120)	122	NOT NULL	
category	INT	4	CHECK BETWEEN 1 and 10	

Asmbly

Attribute Name	Attribute Type	Attribute Size	Constraints	References
asmbly_id	VARCHAR(10)	12	PRIMARY KEY	
date_ordrd	DATE	3	NOT NULL	
asmbly_dtls	VARCHAR(100)	102		
cust_name	VARCHAR(20)	22	FOREIGN KEY	**Customer**

Department

Attribute Name	Attribute Type	Attribute Size	Constraints	References
department_no	VARCHAR(10)	12	PRIMARY KEY	
department_data	VARCHAR(100)	102		

Process

Attribute Name	Attribute Type	Attribute Size	Constraints	References
process_id	VARCHAR(10)	12	PRIMARY KEY	
process_data	VARCHAR(100)	102		
department_no	VARCHAR(10)	12	FOREIGN KEY	Department

Fit_Process

Attribute Name	Attribute Type	Attribute Size	Constraints	References
process_id	VARCHAR(10)	12	PRIMARY KEY, FOREIGN KEY	Process
fit_type	VARCHAR(10)	12	NOT NULL	

Cut_Process

Attribute Name	Attribute Type	Attribute Size	Constraints	References
process_id	VARCHAR(10)	12	PRIMARY KEY, FOREIGN KEY	Process
cutting_type	VARCHAR(10)	12		
machine_type	VARCHAR(10)	12		

Paint_Process

Attribute Name	Attribute Type	Attribute Size	Constraints	References
process_id	VARCHAR(10)	12	PRIMARY KEY, FOREIGN KEY	Process
paint_type	VARCHAR(10)	12	NOT NULL	
painting_method	VARCHAR(10)	12	NOT NULL	

Job

Attribute Name	Attribute Type	Attribute Size	Constraints	References
job_no	INT	4	PRIMARY KEY	
job_start_date	DATE	3	NOT NULL	
job_end_date	DATE	3		
asmbly_id	VARCHAR(10)	12	FOREIGN KEY, NOT NULL	Asmbly
process_id	VARCHAR(10)	12	FOREIGN KEY, NOT NULL	Process

Paint_Job

Attribute Name	Attribute Type	Attribute Size	Constraints	References
job_no	INT	4	PRIMARY KEY, FOREIGN KEY	Job
color	VARCHAR(10)	12	NOT NULL	
volume	REAL	4	DEFAULT 0	
labor_time	INT	3	DEFAULT 0	

Cut_Job

Attribute Name	Attribute Type	Attribute Size	Constraints	References
job_no	INT	4	PRIMARY KEY, FOREIGN KEY	Job
machine_type	VARCHAR(10)	12	NOT NULL	
machine_time	INT	3	DEFAULT 0	
labor_time	INT	3	DEFAULT 0	
material	VARCHAR(20)	22	NOT NULL	

Fit_Job

Attribute Name	Attribute Type	Attribute Size	Constraints	References
job_no	INT	4	PRIMARY KEY, FOREIGN KEY	Job

labor_time	INT	3	DEFAULT 0	

Account				
Attribute Name	Attribute Type	Attribute Size	Constraints	References
account_no	VARCHAR(10)	12	PRIMARY KEY	
account_start_date	DATE	3	NOT NULL	

Department_Account				
Attribute Name	Attribute Type	Attribute Size	Constraints	References
account_no	VARCHAR(10)	12	PRIMARY KEY, FOREIGN KEY	Account
details_2	REAL	4	DEFAULT 0	
department_no	VARCHAR(10)	12	FOREIGN KEY, UNIQUE, NOT NULL	Department

Asmbly_Account				
Attribute Name	Attribute Type	Attribute Size	Constraints	References
account_no	VARCHAR(10)	12	PRIMARY KEY, FOREIGN KEY	Account
details_1	REAL	4	DEFAULT 0	
asmbly_id	VARCHAR(10)	12	FOREIGN KEY, UNIQUE, NOT NULL	Asmbly

Process_Account				
Attribute Name	Attribute Type	Attribute Size	Constraints	References
account_no	VARCHAR(10)	12	PRIMARY KEY, FOREIGN KEY	Account
details_3	REAL	4	DEFAULT 0	
process_id	VARCHAR(10)	12	FOREIGN KEY, UNIQUE, NOT NULL	Process

Transctn				
Attribute Name	Attribute Type	Attribute Size	Constraints	References
transaction_no	VARCHAR(20)	22	PRIMARY KEY	
sup_cost	REAL	4	NOT NULL DEFAULT 0	
job_no	INT	4	FOREIGN KEY, NOT NULL	Job
dep_acno	VARCHAR(10)	12	FOREIGN KEY	Department_Account(account_no)
asmb_acno	VARCHAR(10)	12	FOREIGN KEY	Asmbly_Account(account_no)
prcs_acno	VARCHAR(10)	12	FOREIGN KEY	Process_Account(account_no)

Manufactured				
Attribute Name	Attribute Type	Attribute Size	Constraints	References
asmbly_id	VARCHAR(10)	12	PRIMARY KEY, FOREIGN KEY	Asmbly
process_id	VARCHAR(10)	12	PRIMARY KEY ,FOREIGN KEY	Process

TASK 3.

3.1. Discussion of storage structures for tables

Table Name	Query# and Type	Search Key	Query Frequency	Selected File Organization	Justifications
Customer	1 & insertion 13 & range search	NA category	30/day 100/day	B+ - Tree index with indexing on category	Frequency of range search > insertion, B+ tree organization is selected for faster retrieval
Department	2 & insertion	NA	Infrequent	Heap	Only Insert records are affecting this table, so heap orgn is selected for quicker insertion.
Process	3 & insertion 8 & random search 10 & random search 11 & random search 12 & random search	NA process_id, department_no department_no process_id department_no	Infrequent 50/day 20/day 100/day 20/day	Dynamic Hashing with haskey on process_id	Dynamic hashing as random search is more frequent and hash key on process_id as random search frequency on it is more than other search keys.
Fit_Process	3 & insertion	NA	Infrequent	Heap	Heap as only insertions are affecting this table
Cut_Process	3 & insertion	NA	Infrequent	Heap	Heap as only insertions are affecting this table

Paint_Process	3 & insertion	NA	Infrequent	Heap	Heap as only insertions are affecting this table
Asmbly	4 & insertion	NA	40/day	Heap	Heap selected as only insertions are affecting this table
Process_Account	5 & insertion 8& random search 8 & update (Random search)	NA process_id udpate w.r.t random search – account_no	10/day 50/day 50/day	Dynamic hashing with hash key account_no	As random search is frequent than insertion. account_no as it is primary key
Asmbly_Account	5 & insertion 8& random search 8 & update(random search) 9 & random search	NA asmbly_id random search – account_no random search – asmbly_id	10/day 50/day 50/day 200/day	Dynamic Hashing with hashkey on asmbly_id	queries on asmbly_id are frequent than queries on account_no
Department_Account	5 & insertion 8 & random search 8& update	NA department_no account_no	10/day 50/day 50/day	Dynamic Hashing with hashkey on account_no	account_no as it is primary key
Job	6 & insertion 7 & random search 7 & update 8 & random search 10 & random search	NA job_no update w.r.t - job_no job_no process_id, job_end_date	50/day 50/day 50/day 50/day 20/day	Dynamic hashing with hash_key on job_no	random search is frequent and queries on job_no are more frequent than other search keys

	11 & random search 12 & random search 14 & delete w.r.t range search	asmbly_id job_end_date job_no	100/day 20/day 1/month		
Fit_Job	7 & insertion 10 & random search 12 & random search	NA job_no job_no	50/day 20/day 20/day	Dynamic Hashing with hashkey job_no	Though random search frequency is slightly less than insertion frequency, query retrieve time would be more if heap is used.
Cut_Job	7 & insertion 10 & random search 12& random_search 14& delete w.r.t range search	NA job_no job_no job_no	50/day 20/day 20/day 1/month	B+ - Tree with index on job_no.	As range search and insertion may take more time if dynamic hashing is used
Paint_Job	7 & insertion 10 & random search 12 & random search, 15 & update w.r.t random search	NA job_no job_no job_no	50/day 20/day 20/day 1/week	Dynamic Hashing with hashkey on job_no	Though random search frequency is slightly less than insertion frequency, query retrieve time would be more if heap is used.
Transaction	8 & insertion		50/day	Heap	Heap as only insertions

					are affecting table
Manufactured	4 & insertion		40/day	Heap	Heap as only insertions ar affecting table

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

As most of tables are having random searches and Dynamic hashing woulld work better such cases, Azure SQL & SQL server documentation informs that hash indexing requires in SQL server requires memory optimized tables instead of disk based tables.

After trying to create a memory optimized table in Azure Database and noticing that memory optimized tables are not supported in the free tier service version, I have decided to use non clustered indices on following tables in addition to the clustered indices created on primary keys in SQL server by query optimizer.

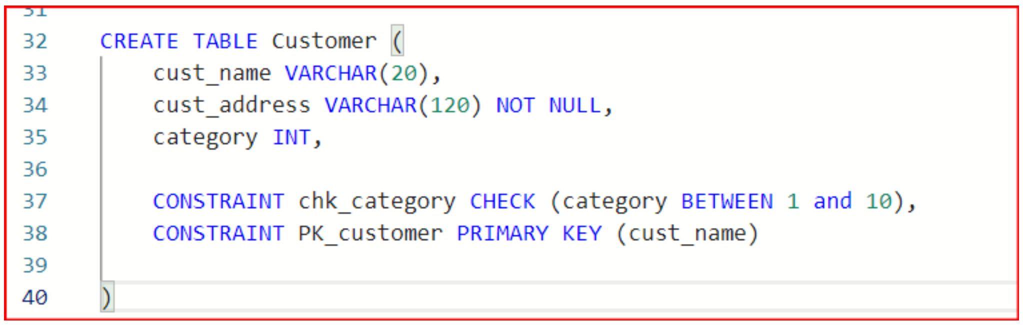
Below error snapshot is for reference.

```
Msg 40536, Level 16, State 2, Line 4
'MEMORY_OPTIMIZED tables' is not supported in this service tier of the database. See Books Onl
ine for more details on feature support in different service tiers of Windows Azure SQL Databa
se.
Total execution time: 00:00:00.102
```

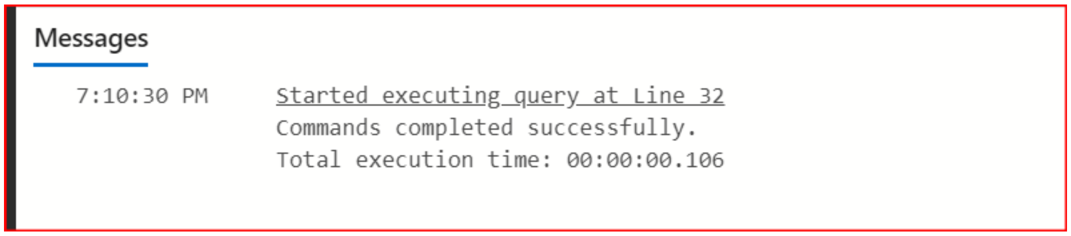
Table Name	Secondary Index
Customer	category
Process	department_no
Job	asmbly_id
Job	job_end_date , process_id
Department_Account	department_no
Asmbly_Account	asmbly_id
Process_Account	process_id

Task 4. SQL statements and screenshots showing the creation of tables in Azure SQL Database

Creating Table Customer



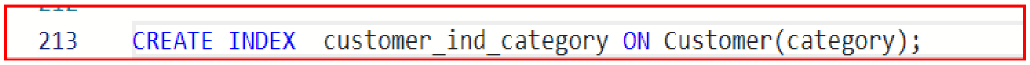
```sql
CREATE TABLE Customer (
    cust_name VARCHAR(20),
    cust_address VARCHAR(120) NOT NULL,
    category INT,

    CONSTRAINT chk_category CHECK (category BETWEEN 1 and 10),
    CONSTRAINT PK_customer PRIMARY KEY (cust_name)

)
```



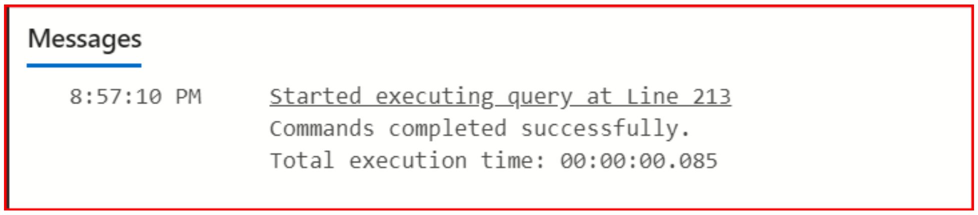
Messages

```
7:10:30 PM      Started executing query at Line 32
                Commands completed successfully.
                Total execution time: 00:00:00.106
```

Creating Index on category for Cusomter Table



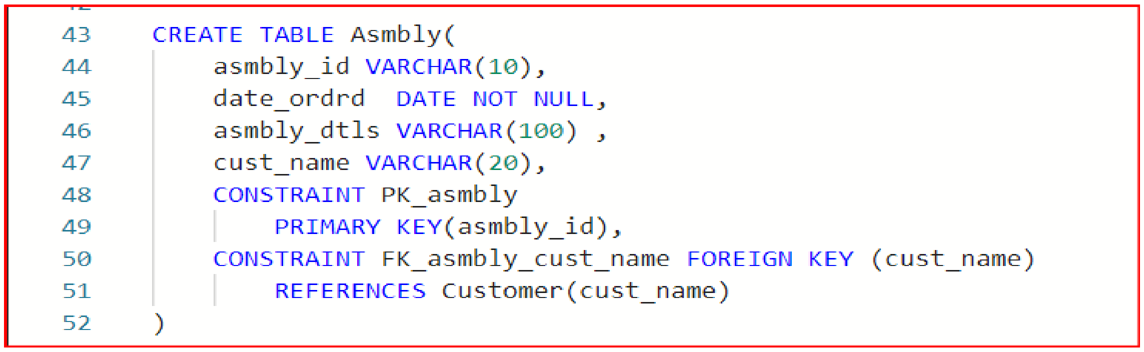
```sql
CREATE INDEX  customer_ind_category ON Customer(category);
```



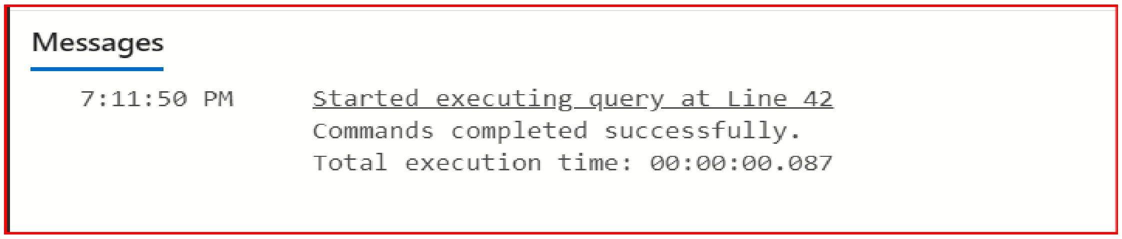
Messages

```
8:57:10 PM      Started executing query at Line 213
                Commands completed successfully.
                Total execution time: 00:00:00.085
```

Creating Table Asmbly

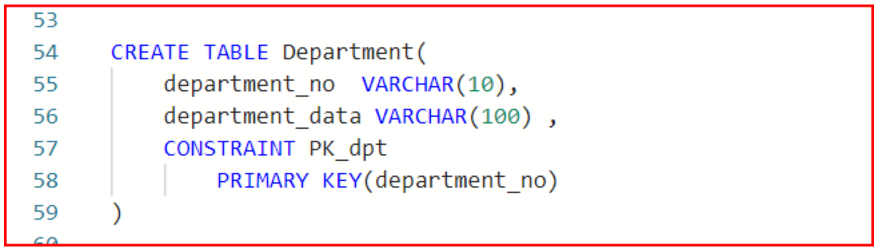


```
CREATE TABLE Asmbly(
    asmbly_id VARCHAR(10),
    date_ordrd  DATE NOT NULL,
    asmbly_dtls VARCHAR(100) ,
    cust_name VARCHAR(20),
    CONSTRAINT PK_asmbly
        PRIMARY KEY(asmbly_id),
    CONSTRAINT FK_asmbly_cust_name FOREIGN KEY (cust_name)
        REFERENCES Customer(cust_name)
)
```

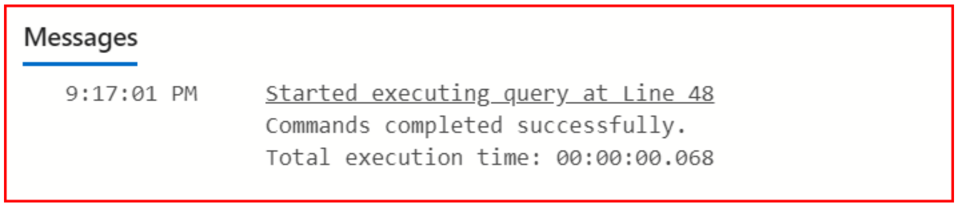


```
Messages
   7:11:50 PM     Started executing query at Line 42
                  Commands completed successfully.
                  Total execution time: 00:00:00.087
```

Creating Table Department

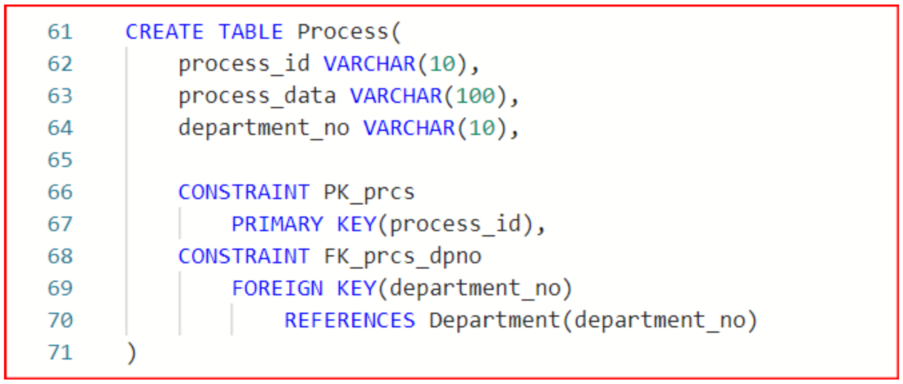


```
CREATE TABLE Department(
    department_no  VARCHAR(10),
    department_data VARCHAR(100) ,
    CONSTRAINT PK_dpt
        PRIMARY KEY(department_no)
)
```

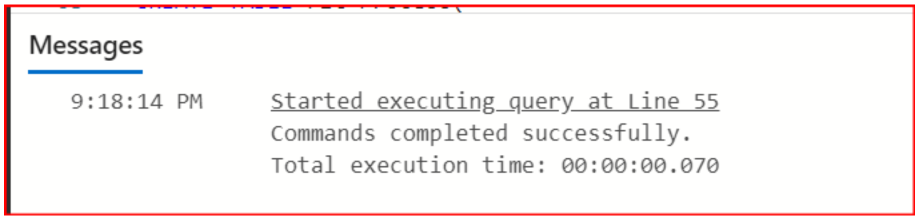


```
Messages
   9:17:01 PM     Started executing query at Line 48
                  Commands completed successfully.
                  Total execution time: 00:00:00.068
```

Creating Table Process

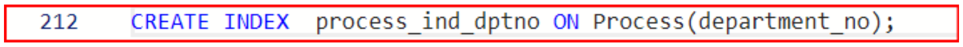


```
CREATE TABLE Process(
    process_id VARCHAR(10),
    process_data VARCHAR(100),
    department_no VARCHAR(10),

    CONSTRAINT PK_prcs
        PRIMARY KEY(process_id),
    CONSTRAINT FK_prcs_dpno
        FOREIGN KEY(department_no)
            REFERENCES Department(department_no)
)
```

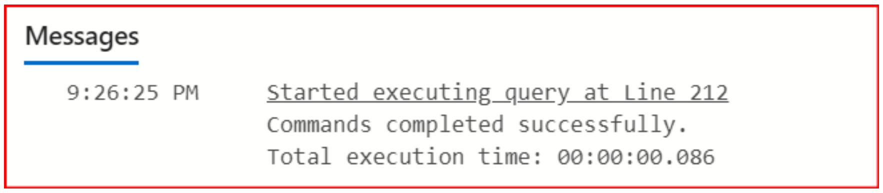


```
Messages
    9:18:14 PM      Started executing query at Line 55
                    Commands completed successfully.
                    Total execution time: 00:00:00.070
```

Creating Index on department_no for Process Table

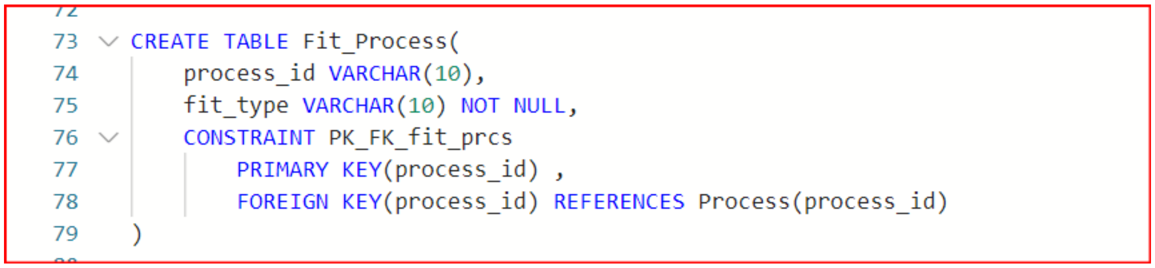


```
CREATE INDEX  process_ind_dptno ON Process(department_no);
```

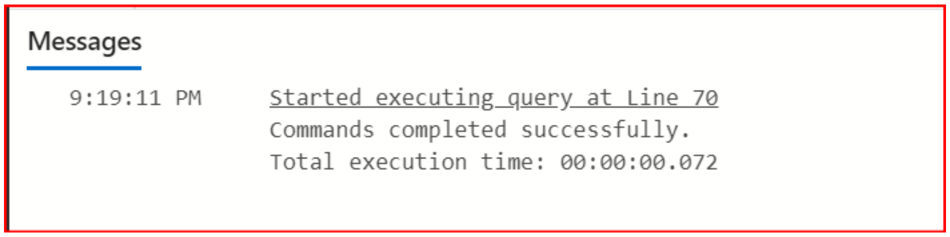


```
Messages
    9:26:25 PM      Started executing query at Line 212
                    Commands completed successfully.
                    Total execution time: 00:00:00.086
```

Creating Table Fit_Process

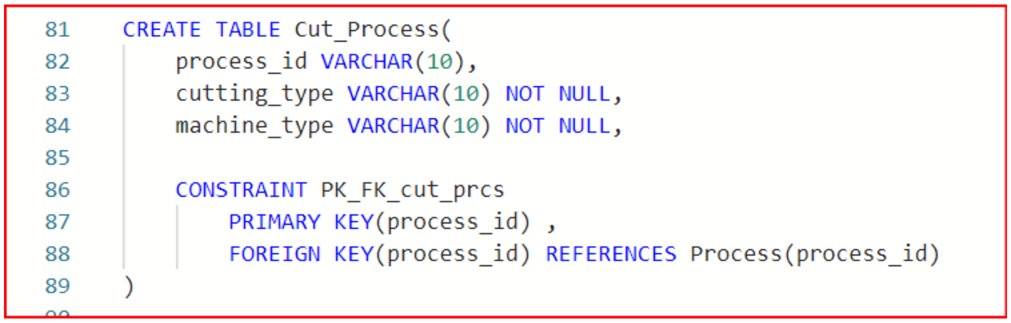


```
CREATE TABLE Fit_Process(
    process_id VARCHAR(10),
    fit_type VARCHAR(10) NOT NULL,
    CONSTRAINT PK_FK_fit_prcs
        PRIMARY KEY(process_id) ,
        FOREIGN KEY(process_id) REFERENCES Process(process_id)
)
```

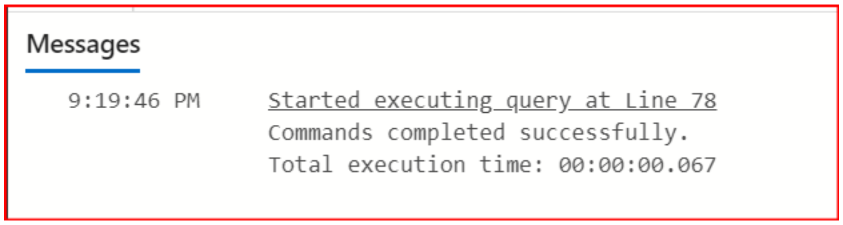


```
Messages
    9:19:11 PM      Started executing query at Line 70
                    Commands completed successfully.
                    Total execution time: 00:00:00.072
```

Creating Table Cut_Process



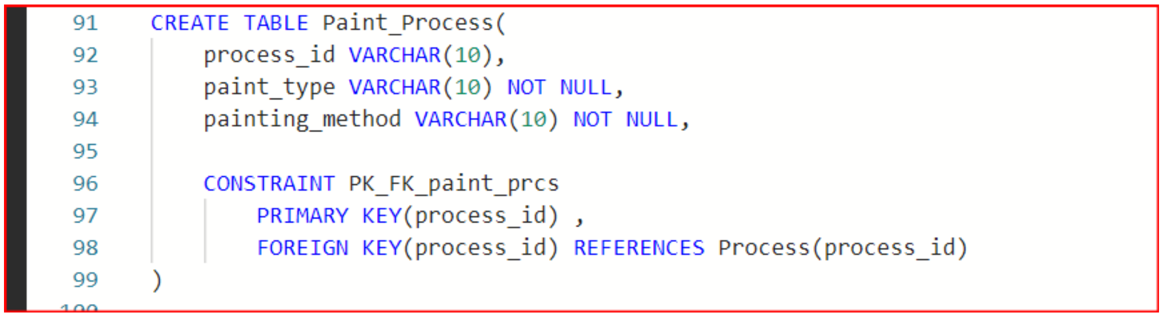
```
CREATE TABLE Cut_Process(
    process_id VARCHAR(10),
    cutting_type VARCHAR(10) NOT NULL,
    machine_type VARCHAR(10) NOT NULL,

    CONSTRAINT PK_FK_cut_prcs
        PRIMARY KEY(process_id) ,
        FOREIGN KEY(process_id) REFERENCES Process(process_id)
)
```



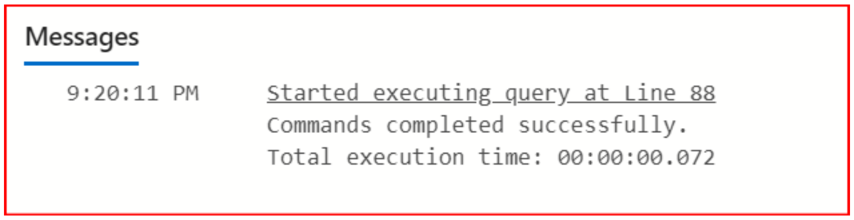
Messages

```
9:19:46 PM      Started executing query at Line 78
                Commands completed successfully.
                Total execution time: 00:00:00.067
```

Creating Table Paint_Process



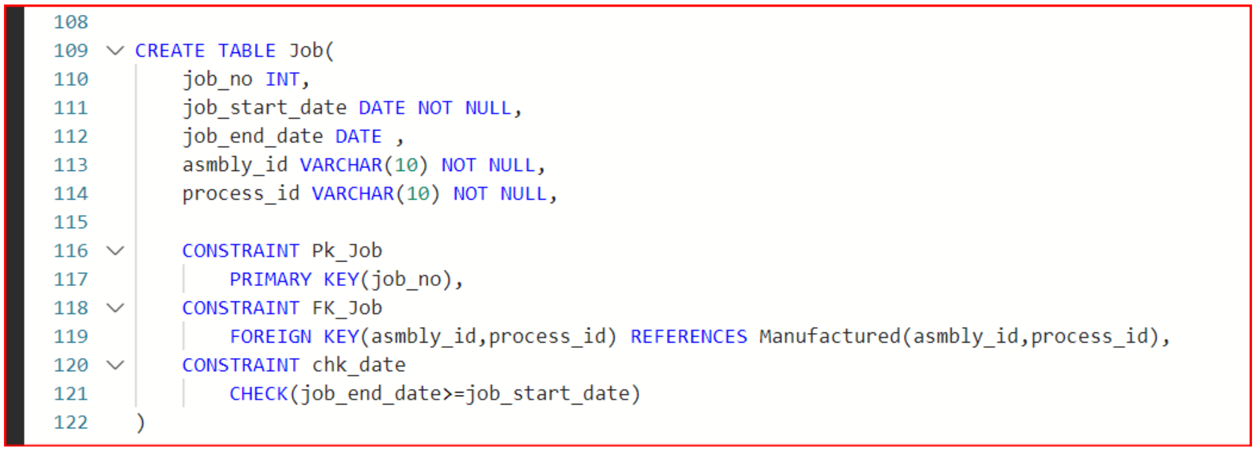
```
CREATE TABLE Paint_Process(
    process_id VARCHAR(10),
    paint_type VARCHAR(10) NOT NULL,
    painting_method VARCHAR(10) NOT NULL,

    CONSTRAINT PK_FK_paint_prcs
        PRIMARY KEY(process_id) ,
        FOREIGN KEY(process_id) REFERENCES Process(process_id)
)
```



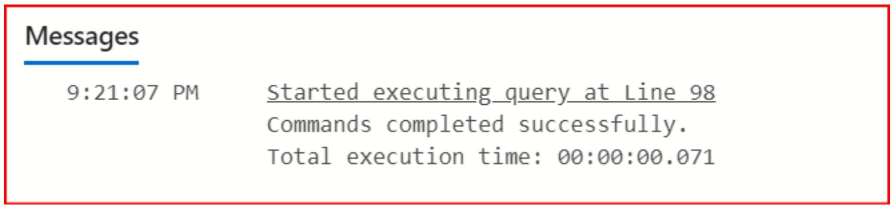
Messages

```
9:20:11 PM      Started executing query at Line 88
                Commands completed successfully.
                Total execution time: 00:00:00.072
```

Creating Table Job



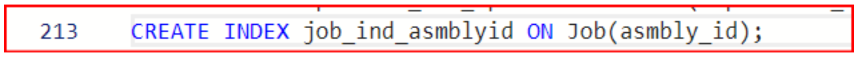
```

CREATE TABLE Job(
    job_no INT,
    job_start_date DATE NOT NULL,
    job_end_date DATE ,
    asmbly_id VARCHAR(10) NOT NULL,
    process_id VARCHAR(10) NOT NULL,

    CONSTRAINT Pk_Job
        PRIMARY KEY(job_no),
    CONSTRAINT FK_Job
        FOREIGN KEY(asmbly_id,process_id) REFERENCES Manufactured(asmbly_id,process_id),
    CONSTRAINT chk_date
        CHECK(job_end_date>=job_start_date)
)
```



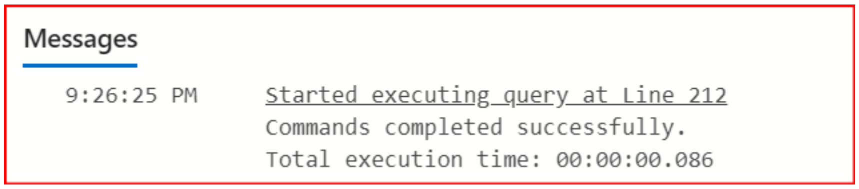
Messages

```
9:21:07 PM      Started executing query at Line 98
                Commands completed successfully.
                Total execution time: 00:00:00.071
```

Creating index on asmbly_id for Job table



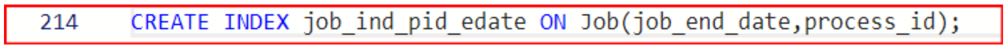
```
213     CREATE INDEX job_ind_asmblyid ON Job(asmbly_id);
```



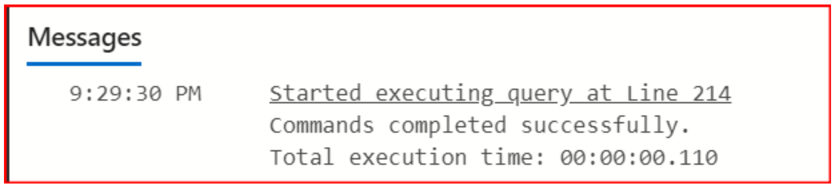
Messages

```
9:26:25 PM      Started executing query at Line 212
                Commands completed successfully.
                Total execution time: 00:00:00.086
```

Creating index on job_end_date and process_id



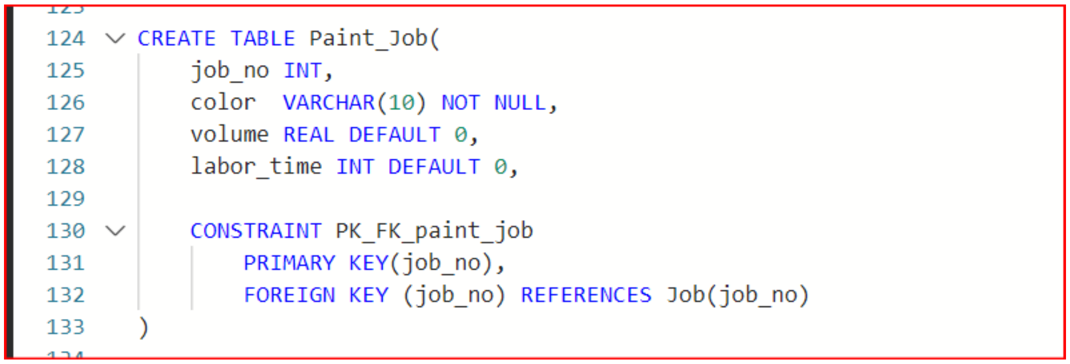
```
214     CREATE INDEX job_ind_pid_edate ON Job(job_end_date,process_id);
```



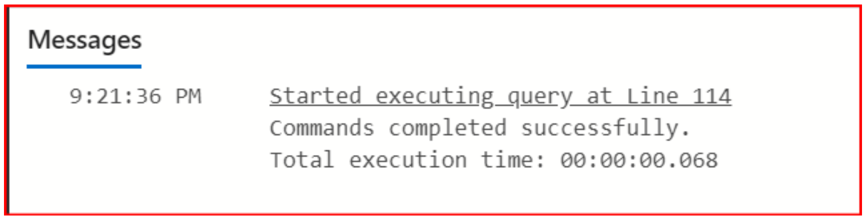
Messages

```
9:29:30 PM      Started executing query at Line 214
                Commands completed successfully.
                Total execution time: 00:00:00.110
```

Creating Table Paint_Job

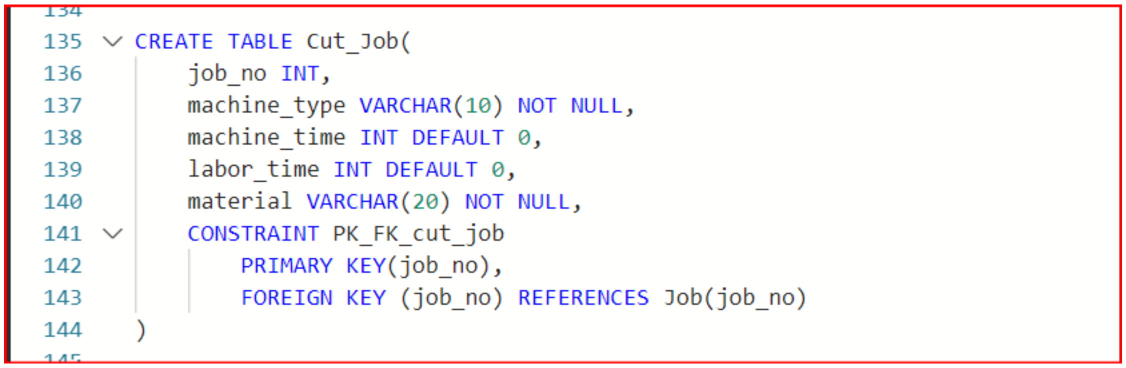


```sql
CREATE TABLE Paint_Job(
    job_no INT,
    color  VARCHAR(10) NOT NULL,
    volume REAL DEFAULT 0,
    labor_time INT DEFAULT 0,

    CONSTRAINT PK_FK_paint_job
        PRIMARY KEY(job_no),
        FOREIGN KEY (job_no) REFERENCES Job(job_no)
)
```

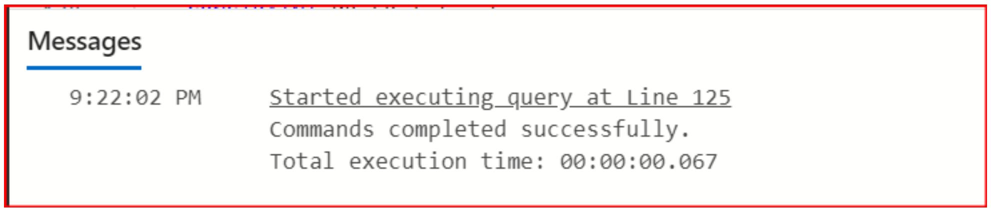


```
Messages
    9:21:36 PM      Started executing query at Line 114
                    Commands completed successfully.
                    Total execution time: 00:00:00.068
```

Creating Table Cut_Job

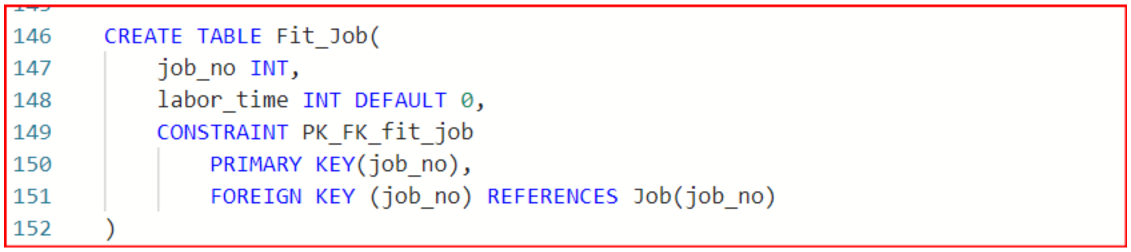


```sql
CREATE TABLE Cut_Job(
    job_no INT,
    machine_type VARCHAR(10) NOT NULL,
    machine_time INT DEFAULT 0,
    labor_time INT DEFAULT 0,
    material VARCHAR(20) NOT NULL,
    CONSTRAINT PK_FK_cut_job
        PRIMARY KEY(job_no),
        FOREIGN KEY (job_no) REFERENCES Job(job_no)
)
```

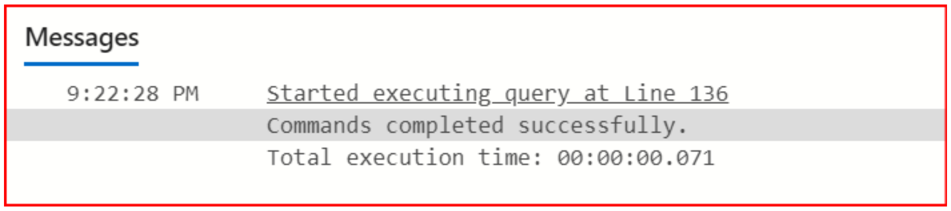


```
Messages
    9:22:02 PM      Started executing query at Line 125
                    Commands completed successfully.
                    Total execution time: 00:00:00.067
```

Creating Table Fit_Job

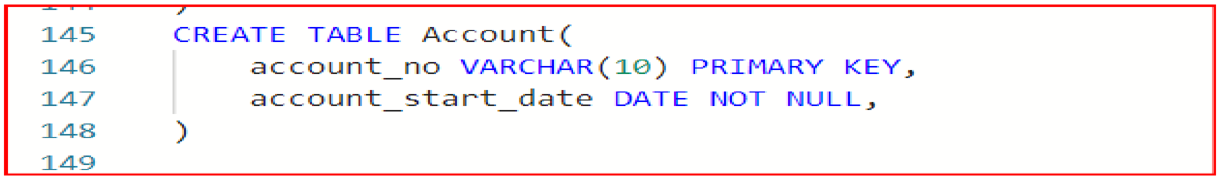


```sql
CREATE TABLE Fit_Job(
    job_no INT,
    labor_time INT DEFAULT 0,
    CONSTRAINT PK_FK_fit_job
        PRIMARY KEY(job_no),
        FOREIGN KEY (job_no) REFERENCES Job(job_no)
)
```

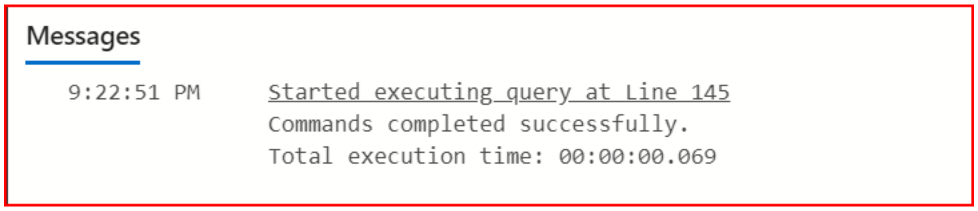


```
Messages
    9:22:28 PM    Started executing query at Line 136
                  Commands completed successfully.
                  Total execution time: 00:00:00.071
```

Creating Table Account

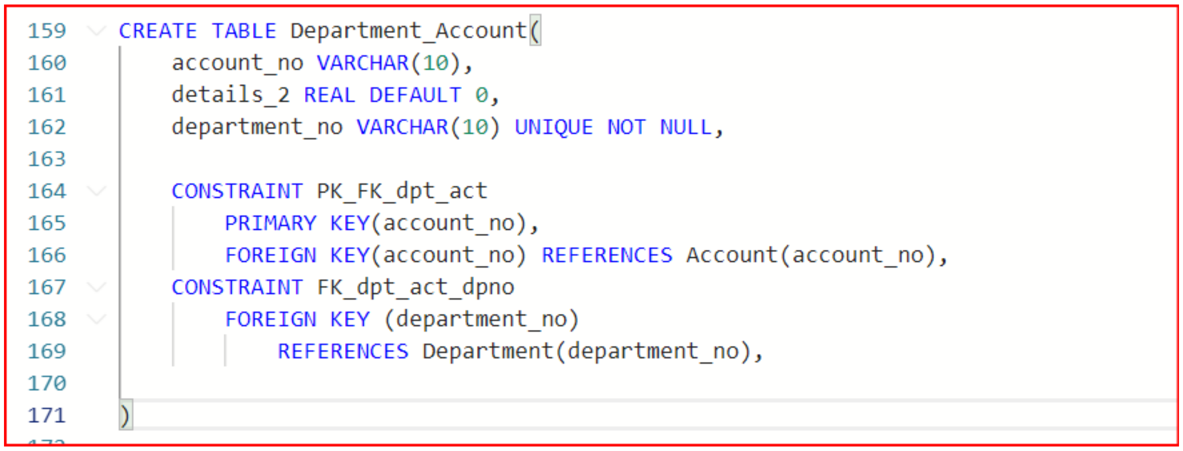


```sql
CREATE TABLE Account(
    account_no VARCHAR(10) PRIMARY KEY,
    account_start_date DATE NOT NULL,
)
```

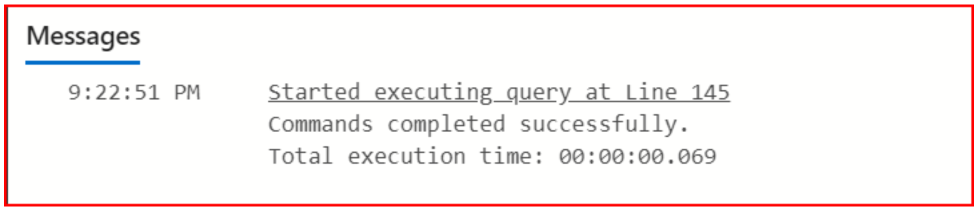


```
Messages
    9:22:51 PM    Started executing query at Line 145
                  Commands completed successfully.
                  Total execution time: 00:00:00.069
```

Creating Table Department_Account

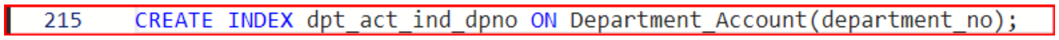


```sql
CREATE TABLE Department_Account(
    account_no VARCHAR(10),
    details_2 REAL DEFAULT 0,
    department_no VARCHAR(10) UNIQUE NOT NULL,

    CONSTRAINT PK_FK_dpt_act
        PRIMARY KEY(account_no),
        FOREIGN KEY(account_no) REFERENCES Account(account_no),
    CONSTRAINT FK_dpt_act_dpno
        FOREIGN KEY (department_no)
            REFERENCES Department(department_no),

)
```



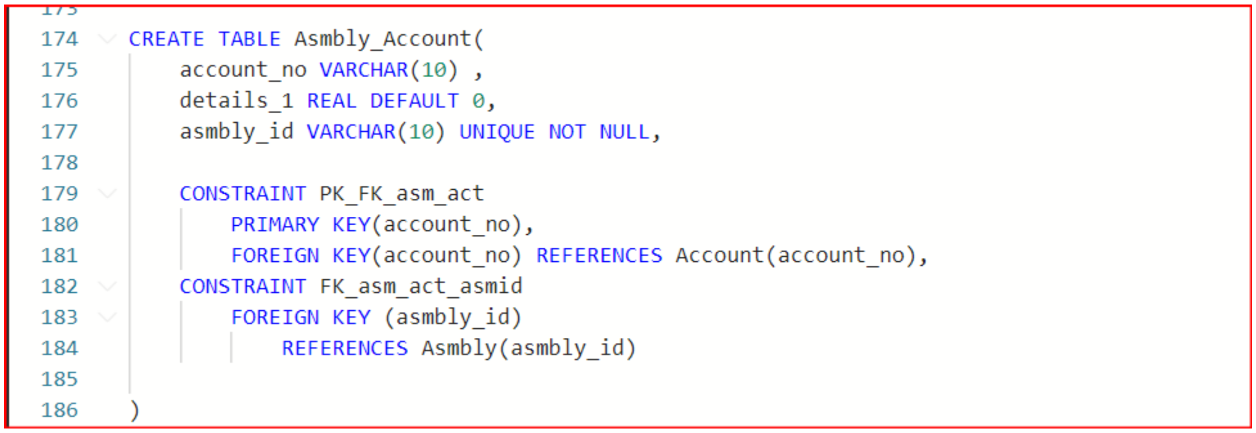
```
Messages
    9:22:51 PM    Started executing query at Line 145
                  Commands completed successfully.
                  Total execution time: 00:00:00.069
```

Creating index on department_no for table Department_Account table

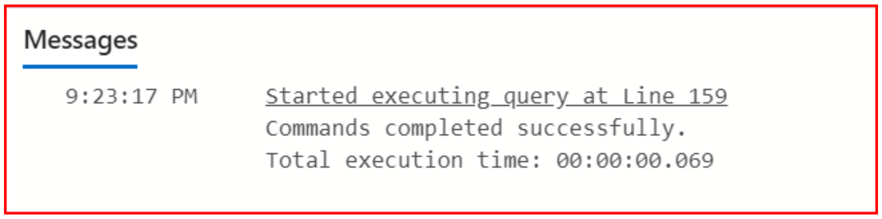


```sql
CREATE INDEX dpt_act_ind_dpno ON Department_Account(department_no);
```

Creating Table Asmbly_Account



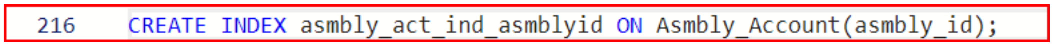
```sql
CREATE TABLE Asmbly_Account(
    account_no VARCHAR(10) ,
    details_1 REAL DEFAULT 0,
    asmbly_id VARCHAR(10) UNIQUE NOT NULL,

    CONSTRAINT PK_FK_asm_act
        PRIMARY KEY(account_no),
        FOREIGN KEY(account_no) REFERENCES Account(account_no),
    CONSTRAINT FK_asm_act_asmid
        FOREIGN KEY (asmbly_id)
            REFERENCES Asmbly(asmbly_id)

)
```



Messages

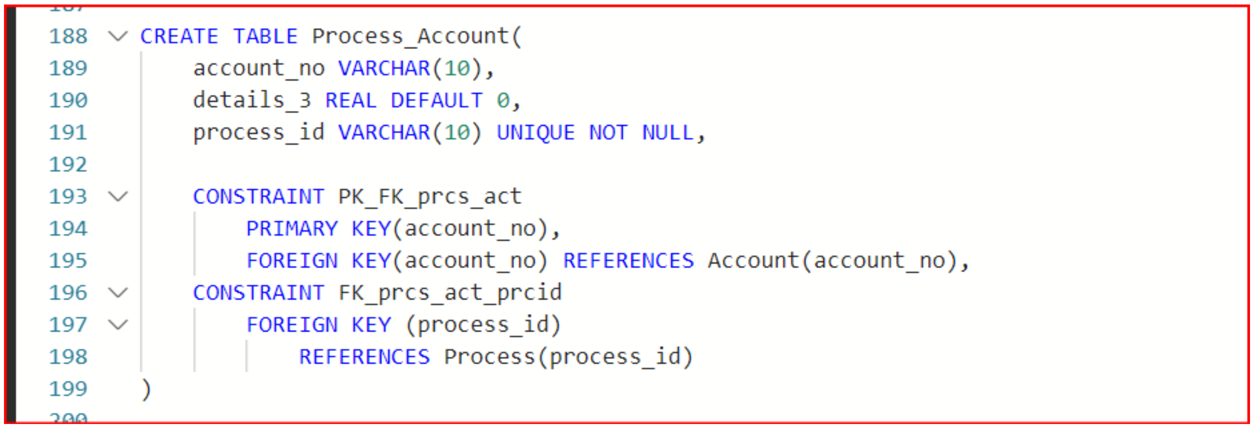
```
9:23:17 PM      Started executing query at Line 159
                Commands completed successfully.
                Total execution time: 00:00:00.069
```

Creating index on asmbly_id for Asmbly_Account table

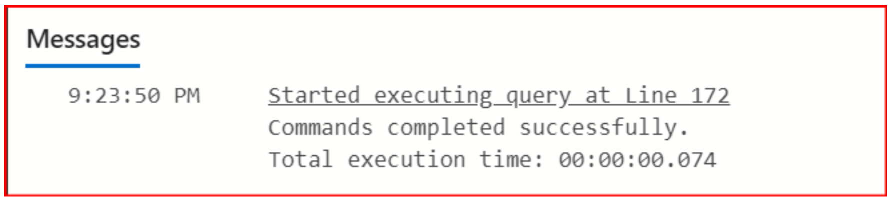


```sql
CREATE INDEX asmbly_act_ind_asmblyid ON Asmbly_Account(asmbly_id);
```

Creating Table Process_Account



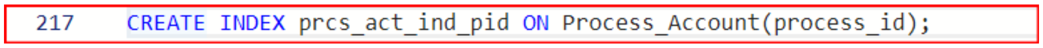
```sql
CREATE TABLE Process_Account(
    account_no VARCHAR(10),
    details_3 REAL DEFAULT 0,
    process_id VARCHAR(10) UNIQUE NOT NULL,

    CONSTRAINT PK_FK_prcs_act
        PRIMARY KEY(account_no),
        FOREIGN KEY(account_no) REFERENCES Account(account_no),
    CONSTRAINT FK_prcs_act_prcid
        FOREIGN KEY (process_id)
            REFERENCES Process(process_id)
)
```



Messages

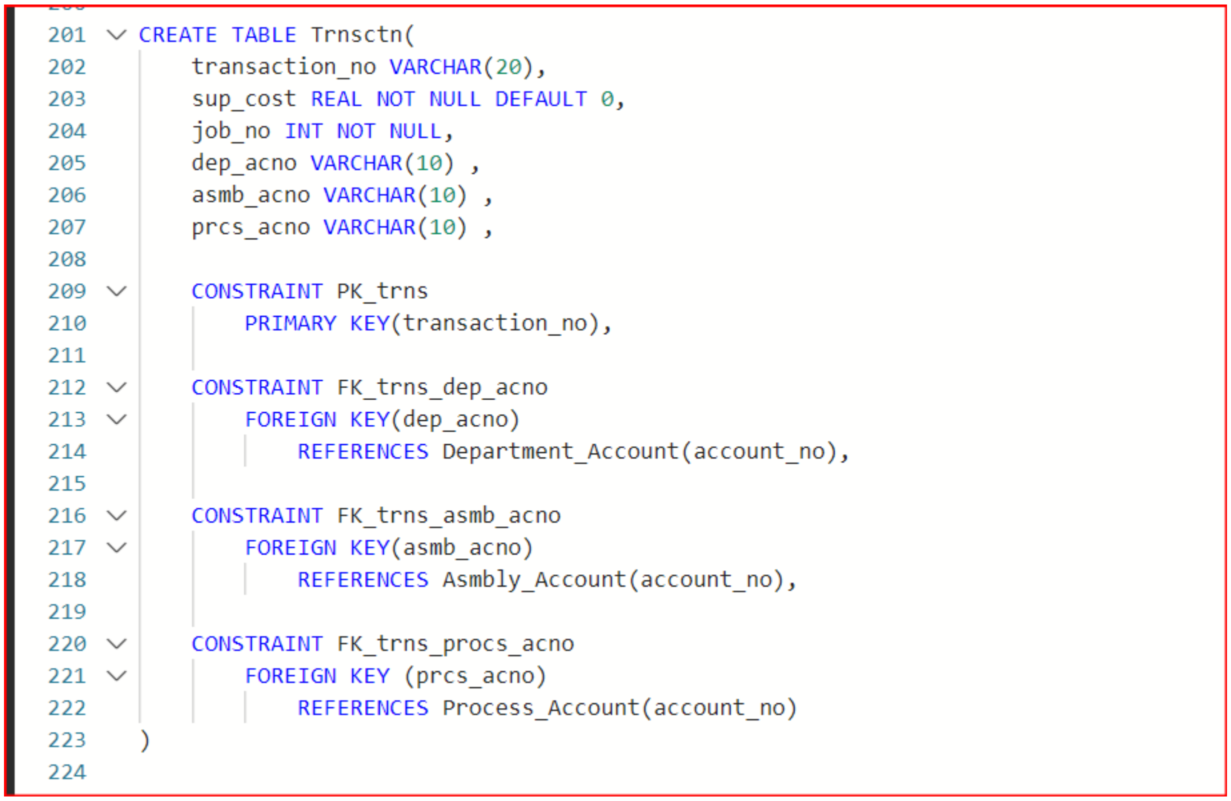
```
9:23:50 PM      Started executing query at Line 172
                Commands completed successfully.
                Total execution time: 00:00:00.074
```

Creating index on process_id for table Process_Account table

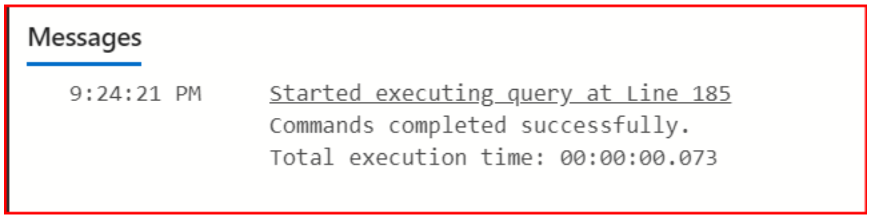


```sql
CREATE INDEX prcs_act_ind_pid ON Process_Account(process_id);
```

Creating Table Trnsctn

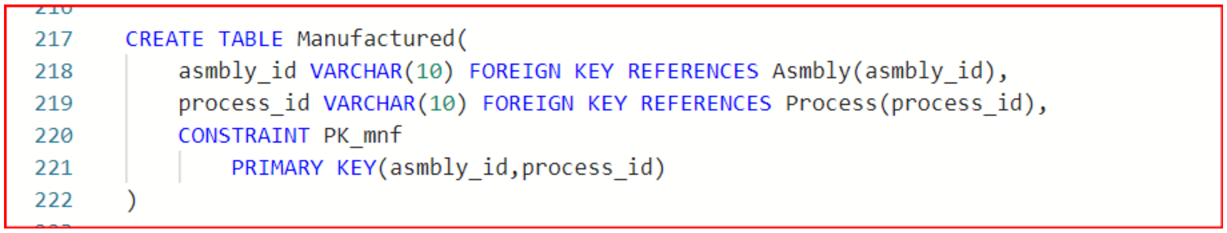


```sql
CREATE TABLE Trnsctn(
    transaction_no VARCHAR(20),
    sup_cost REAL NOT NULL DEFAULT 0,
    job_no INT NOT NULL,
    dep_acno VARCHAR(10) ,
    asmb_acno VARCHAR(10) ,
    prcs_acno VARCHAR(10) ,

    CONSTRAINT PK_trns
        PRIMARY KEY(transaction_no),

    CONSTRAINT FK_trns_dep_acno
        FOREIGN KEY(dep_acno)
            REFERENCES Department_Account(account_no),

    CONSTRAINT FK_trns_asmb_acno
        FOREIGN KEY(asmb_acno)
            REFERENCES Asmbly_Account(account_no),

    CONSTRAINT FK_trns_procs_acno
        FOREIGN KEY (prcs_acno)
            REFERENCES Process_Account(account_no)
)
```

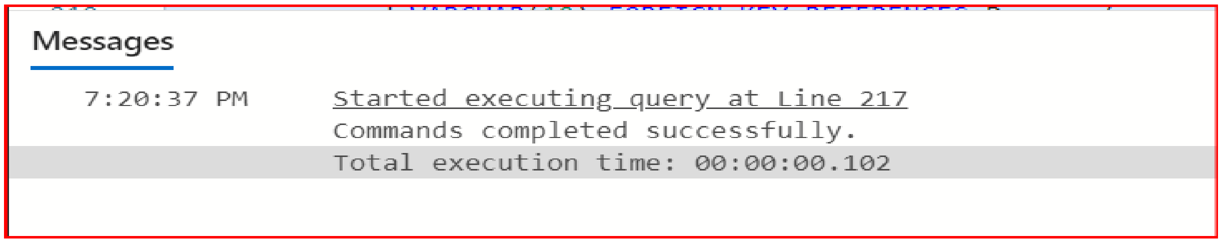


```
Messages

9:24:21 PM        Started executing query at Line 185
                  Commands completed successfully.
                  Total execution time: 00:00:00.073
```

Creating Table Manufactured



```sql
CREATE TABLE Manufactured(
    asmbly_id VARCHAR(10) FOREIGN KEY REFERENCES Asmbly(asmbly_id),
    process_id VARCHAR(10) FOREIGN KEY REFERENCES Process(process_id),
    CONSTRAINT PK_mnf
        PRIMARY KEY(asmbly_id,process_id)
)
```



```
Messages

7:20:37 PM        Started executing query at Line 217
                  Commands completed successfully.
                  Total execution time: 00:00:00.102
```

Task 5.

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

Please refer Appidi_Harinadh_Task5a.SQL File for the SQL statements of queries 1-15 which are Implemented as Stored Procedures.

```
DROP PROCEDURE IF EXISTS qproc_1;
DROP PROCEDURE IF EXISTS qproc_2;
DROP PROCEDURE IF EXISTS qproc_3;
DROP PROCEDURE IF EXISTS qproc_3_1;
DROP PROCEDURE IF EXISTS qproc_3_2;
DROP PROCEDURE IF EXISTS qproc_3_3;
DROP PROCEDURE IF EXISTS qproc_4;
DROP PROCEDURE IF EXISTS qproc_4_1;
DROP PROCEDURE IF EXISTS qproc_5;
DROP PROCEDURE IF EXISTS qproc_5_1;
DROP PROCEDURE IF EXISTS qproc_5_2;
DROP PROCEDURE IF EXISTS qproc_5_3;
DROP PROCEDURE IF EXISTS qproc_6;
DROP PROCEDURE IF EXISTS qproc_7;
DROP PROCEDURE IF EXISTS qproc_7_1;
DROP PROCEDURE IF EXISTS qproc_7_2;
DROP PROCEDURE IF EXISTS qproc_7_3;
DROP PROCEDURE IF EXISTS qproc_8;
DROP PROCEDURE IF EXISTS qproc_9;
DROP PROCEDURE IF EXISTS qproc_10;
DROP PROCEDURE IF EXISTS qproc_11;
DROP PROCEDURE IF EXISTS qproc_12_1;
DROP PROCEDURE IF EXISTS qproc_12_2;
DROP PROCEDURE IF EXISTS qproc_12_3;
DROP PROCEDURE IF EXISTS qproc_13;
DROP PROCEDURE IF EXISTS qproc_14;
DROP PROCEDURE IF EXISTS qproc_15;
```

1. Enter a new customer (30/day)

```
GO
    CREATE PROCEDURE qproc_1
        @cust_name VARCHAR(20),
        @cust_address VARCHAR(100),
        @category INT
        AS
            BEGIN
                INSERT INTO Customer (
                    [cust_name],[cust_address],[category]
                )
                VALUES(
                    @cust_name,@cust_address,@category
                )
            END
```

2. Enter a new department (infrequent)

```
GO
    CREATE PROCEDURE qproc_2
        @department_no VARCHAR(10),
        @department_data VARCHAR(100)
        AS
            BEGIN
                INSERT INTO Department (
                    [department_no],[department_data]
                )
                VALUES(
                    @department_no,@department_data
                )

            END
```

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

```
GO
    CREATE PROCEDURE qproc_3
        @process_id VARCHAR(10),
        @process_data VARCHAR(100),
        @department_no VARCHAR(10)
        AS
            BEGIN
                INSERT INTO Process(
                    [process_id],[process_data],[department_no]
                )
                VALUES(
                    @process_id, @process_data, @department_no
                )

            END
GO
    CREATE PROCEDURE qproc_3_1
        @process_id VARCHAR(10),
        @fit_type VARCHAR(10)
        AS
            BEGIN
                INSERT INTO Fit_Process (
                    [process_id],[fit_type]
                )
                VALUES(
                    @process_id, @fit_type
                )
            END
GO
    CREATE PROCEDURE qproc_3_2
        @process_id VARCHAR(10),
        @paint_type VARCHAR(10),
        @painting_method VARCHAR(10)
        AS
            BEGIN
                INSERT INTO Paint_Process (
                    [process_id],[paint_type],[painting_method]
                )
                VALUES(
                    @process_id, @paint_type, @painting_method
                )

            END

GO
    CREATE PROCEDURE qproc_3_3
        @process_id VARCHAR(10),
        @cutting_type VARCHAR(10),
        @machine_type VARCHAR(10)
        AS
            BEGIN
                INSERT INTO Cut_Process (
                    [process_id],[cutting_type],[machine_type]
                )
                VALUES(
                    @process_id, @cutting_type,@machine_type
                )

            END
```

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

```
    GO
        CREATE PROCEDURE qproc_4
            @asmbly_id VARCHAR(10),
            @date_ordrd VARCHAR(10),
            @asmbly_dtls VARCHAR(100),
            @cust_name VARCHAR(20)
            AS
                BEGIN
                    INSERT INTO Asmbly (
                        [asmbly_id],[date_ordrd],[asmbly_dtls],[cust_name]
                    )
                    VALUES(
                        @asmbly_id, CAST(@date_ordrd AS DATE),@asmbly_dtls,@cust_name
                    )
            END
    GO
        CREATE PROCEDURE qproc_4_1
            @asmbly_id VARCHAR(10),
            @process_id VARCHAR(10)
            AS
                BEGIN
                    INSERT INTO Manufactured (
                        [asmbly_id],[process_id]
                    )
                    VALUES(
                        @asmbly_id, @process_id
                    )

                END
```

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

```
    GO
        CREATE PROCEDURE qproc_5
            @account_no VARCHAR(10),
            @account_start_date VARCHAR(10)

            AS
                BEGIN
                    INSERT INTO Account(
                        [account_no],[account_start_date]
                    )
                    VALUES(
                        @account_no, cast(@account_start_date as DATE)
                    )

                END

    GO

        CREATE PROCEDURE qproc_5_1
            @account_no VARCHAR(10),
            -- @details_2 REAL,
            @department_no VARCHAR(10)

            AS
                BEGIN
                    INSERT INTO Department_Account (
                        [account_no],[department_no]
                    )
                    VALUES(
                        @account_no, @department_no
                    )

                END
```

```
GO
    CREATE PROCEDURE qproc_5_2
        @account_no VARCHAR(10),
        -- @account_start_date VARCHAR(10),
        -- @details_1 REAL,
        @asmbly_id VARCHAR(10)

        AS
            BEGIN
                INSERT INTO Asmbly_Account (
                    [account_no],[asmbly_id]
                )
                VALUES(
                    @account_no, @asmbly_id
                )

            END


GO
    CREATE PROCEDURE qproc_5_3
        @account_no VARCHAR(10),
        -- @account_start_date VARCHAR(10),
        -- @details_3 REAL,
        @process_id VARCHAR(10)

        AS
            BEGIN
                INSERT INTO Process_Account (
                    [account_no],[process_id]
                )
                VALUES(
                    @account_no, @process_id
                )

            END
```

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

```
GO
    CREATE PROCEDURE qproc_6
        @job_no VARCHAR(10),
        @job_start_date VARCHAR(10),
        @asmbly_id VARCHAR(10),
        @process_id VARCHAR(10)

        AS
            BEGIN
                INSERT INTO Job (
                    [job_no],[job_start_date],[asmbly_id],[process_id]
                )
                VALUES(
                    @job_no, CAST(@job_start_date AS DATE), @asmbly_id, @process_id
                )

            END
```

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

```
GO
    CREATE PROCEDURE qproc_7
        @job_no VARCHAR(10),
        @job_end_date VARCHAR(10)

        AS
            BEGIN
                UPDATE Job
                    SET job_end_date = CAST(@job_end_date AS DATE)
                        WHERE job_no = @job_no

            END


GO
    CREATE PROCEDURE qproc_7_1
        @job_no VARCHAR(10),
        @labor_time INT

        AS
            BEGIN
                INSERT INTO Fit_Job (
                    [job_no],[labor_time]
                )
                VALUES(
                    @job_no, @labor_time
                )

            END


GO
    CREATE PROCEDURE qproc_7_2
        @job_no VARCHAR(10),
        @labor_time INT,
        @color VARCHAR(10),
        @volume REAL

        AS
            BEGIN
                INSERT INTO Paint_Job (
                    [job_no],[labor_time],[color],[volume]
                )
                VALUES(
                    @job_no, @labor_time , @color, @volume
                )

            END


GO
    CREATE PROCEDURE qproc_7_3
        @job_no VARCHAR(10),
        @machine_type VARCHAR(10),
        @machine_time INT,
        @labor_time INT,
        @material VARCHAR(20)

        AS
            BEGIN
                INSERT INTO Cut_Job (
                    [job_no],[machine_type],[machine_time],[labor_time],[material]
                )
                VALUES(
                    @job_no, @machine_type, @machine_time, @labor_time, @material
                )

            END
```

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

```
    GO
        CREATE PROCEDURE qproc_8
            @transaction_no VARCHAR(20),
            @sup_cost REAL,
            @job_no VARCHAR(10)

            AS
                BEGIN
                    DECLARE @dep_acno VARCHAR(10),
                            @asmb_acno VARCHAR(10),
                            @prcs_acno VARCHAR(10)

                    SET @dep_acno = (SELECT account_no FROM Department_Account,Process, Job
                                        WHERE Job.job_no = @job_no and
                            Department_Account.department_no = Process.department_no and
                                                   Process.process_id = Job.process_id);

                    SET @asmb_acno = (SELECT account_no FROM Asmbly_Account, Job
                  WHERE Job.job_no = @job_no and Asmbly_Account.asmbly_id = Job.asmbly_id);

                   SET @prcs_acno = (SELECT account_no FROM Process_Account, Job
               WHERE Job.job_no = @job_no and Process_Account.process_id = Job.process_id);
```

```
                INSERT INTO Trnsctn (
                [transaction_no],[sup_cost],[job_no],[dep_acno],[asmb_acno],[prcs_acno]
                )
                VALUES(
                   @transaction_no, @sup_cost, @job_no, @dep_acno,@asmb_acno,@prcs_acno
                )

                UPDATE Department_Account
                    SET details_2 = details_2 + @sup_cost
                    WHERE account_no = @dep_acno

                UPDATE Asmbly_Account
                    SET details_1 = details_1 + @sup_cost
                    WHERE account_no = @asmb_acno

                UPDATE Process_Account
                    SET details_3 = details_3 + @sup_cost
                    WHERE account_no = @prcs_acno

            END
```

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

```
    GO
        CREATE PROCEDURE qproc_9
            @asmbly_id VARCHAR(10)
            AS
            BEGIN
                SELECT details_1 from Asmbly_Account
                    WHERE Asmbly_Account.asmbly_id = @asmbly_id

            END
```

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

```
    GO
        CREATE PROCEDURE qproc_10
            @department_no VARCHAR(10),
            @job_end_date VARCHAR(10)

            AS
            BEGIN
                DECLARE @fit_labr_time   INT,
                        @cut_labr_time   INT,
                        @paint_labr_time INT,
                        @total_labr_time INT;

                SET @fit_labr_time = (SELECT labor_time from Fit_Job
                WHERE Fit_Job.job_no in (SELECT distinct(job_no) from Job
                  WHERE Job.process_id in (SELECT distinct(process_id) FROM Process
                      WHERE Process.department_no = @department_no) and
                         Job.job_end_date <= cast(@job_end_date As date)));

               SET @cut_labr_time = ( SELECT labor_time from Cut_Job
                 WHERE Cut_Job.job_no in (
                      SELECT distinct(job_no) from Job
                       WHERE Job.process_id in (SELECT distinct(process_id) FROM Process
                             WHERE Process.department_no = @department_no) and
                                   Job.job_end_date <= cast(@job_end_date As date)));

               SET @paint_labr_time = ( SELECT labor_time from Paint_Job
               WHERE Paint_Job.job_no in (
                 SELECT distinct(job_no) from Job
                       WHERE Job.process_id in (SELECT distinct(process_id) FROM Process
                           WHERE Process.department_no = @department_no) and
                               Job.job_end_date <= cast(@job_end_date As date)));

               IF @fit_labr_time IS NULL SET @fit_labr_time = 0;
               IF @cut_labr_time IS NULL SET @cut_labr_time = 0;
               IF @paint_labr_time IS NULL SET @paint_labr_time = 0;
               SET @total_labr_time = @fit_labr_time + @cut_labr_time + @paint_labr_time;
               SELECT @total_labr_time;

           END
```

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

```
       GO
           CREATE PROCEDURE qproc_11
               @asmbly_id VARCHAR(10)
               AS
               BEGIN
                   SELECT Job.process_id, Process.department_no, Job.job_start_date
                       FROM Job, Process
                       WHERE Job.asmbly_id = @asmbly_id AND Process.process_id = Job.process_i
       d ORDER BY Job.job_start_date;

               END
```

12. Retrieve the jobs (together with their type information and assembly-id) completed during a given date in a given department (20/day).

```
       GO
           CREATE PROCEDURE qproc_12_1
               @job_end_date VARCHAR(10),
               @department_no VARCHAR(10)

               AS
               BEGIN
                   SELECT DISTINCT(Job.job_no), Job.asmbly_id, Fit_Job.labor_time
                       FROM Job, Fit_Job
                 WHERE Job.job_end_date <= cast(@job_end_date AS DATE) and Job.process_id in
                       (SELECT Process.process_id FROM Process, Department
                           WHERE Process.department_no = Department.department_no)
```

```
                AND Job.job_no = fit_Job.job_no

        END



GO
    CREATE PROCEDURE qproc_12_2
        @job_end_date VARCHAR(10),
        @department_no VARCHAR(10)

        AS
        BEGIN
         SELECT DISTINCT(Job.job_no), Job.asmbly_id, Paint_Job.color, Paint_Job.volume,
                      Paint_Job.labor_time
                FROM Job, Paint_Job
                WHERE Job.job_end_date <= cast(@job_end_date  AS DATE)
                AND Job.process_id
                IN  (SELECT Process.process_id FROM Process, Department
                        WHERE Process.department_no = Department.department_no)
                AND Job.job_no = Paint_Job.job_no

        END


GO
    CREATE PROCEDURE qproc_12_3
        @job_end_date VARCHAR(10),
        @department_no VARCHAR(10)

        AS
        BEGIN
               SELECT DISTINCT(Job.job_no), Job.asmbly_id, Cut_Job.machine_type,
                      Cut_Job.machine_time, Cut_Job.material, Cut_Job.labor_time
                FROM Job, Cut_Job
            WHERE Job.job_end_date <= cast(@job_end_date AS DATE) and Job.process_id in
                (SELECT Process.process_id FROM Process, Department
                    WHERE Process.department_no = Department.department_no)
                AND Job.job_no = Cut_Job.job_no

        END
```

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

```
GO
    CREATE PROCEDURE qproc_13
        @clower INT,
        @cupper INT

        AS
        BEGIN
          SELECT * FROM Customer
          WHERE category >=@clower AND category<=@cupper
          ORDER BY cust_name
```

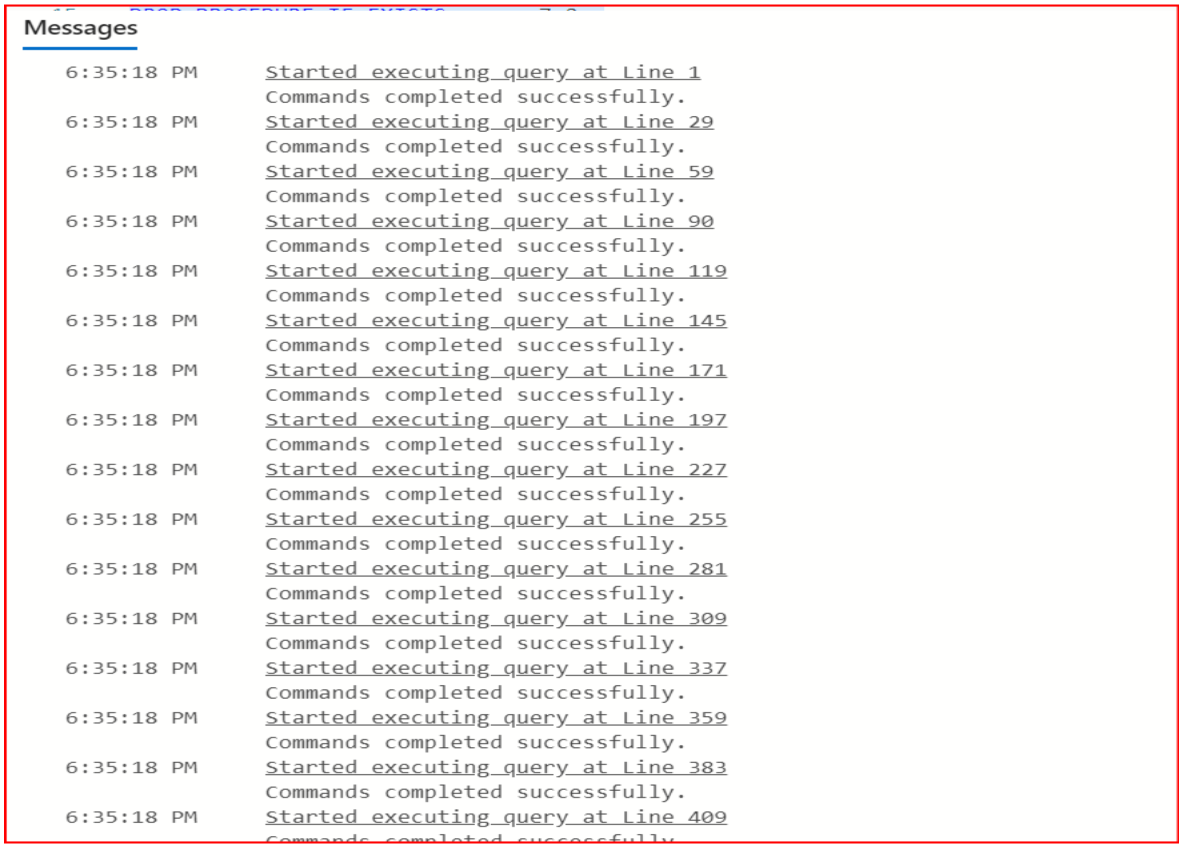
```
            END
```

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

```
    GO
        CREATE PROCEDURE qproc_14
            @jno_lower INT,
            @jno_upper INT

            AS
            BEGIN
               DELETE FROM Cut_Job WHERE job_no >=@jno_lower AND job_no <= @jno_upper

            END
```

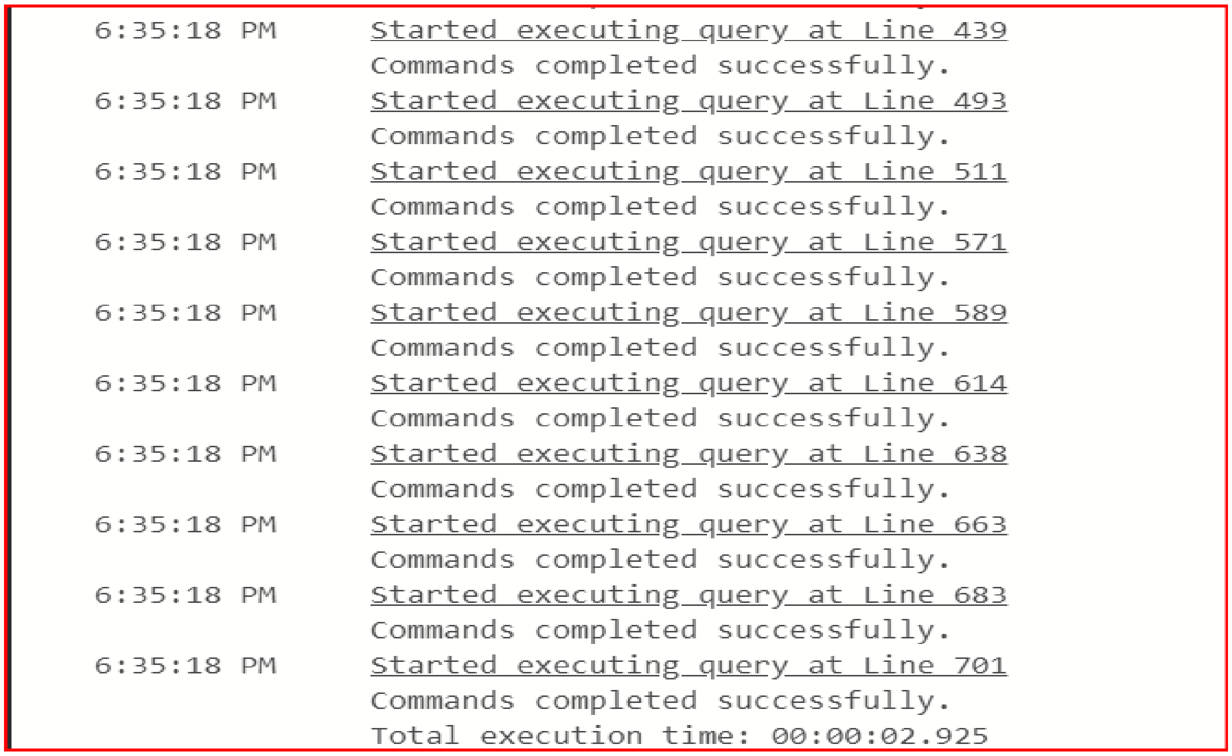
15. Change the color of a given paint job (1/week).

```
    GO
        CREATE PROCEDURE qproc_15
            @job_no INT,
            @color VARCHAR(10)
            AS
            BEGIN
               UPDATE Paint_Job SET color = @color WHERE job_no = @job_no

            END
```



Messages

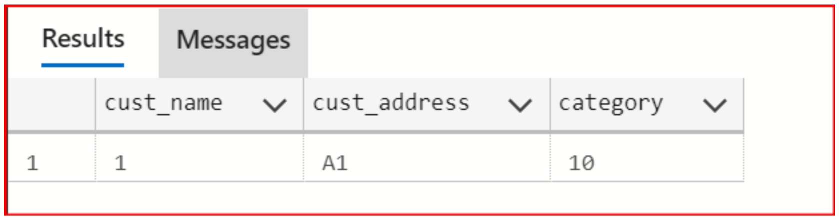
```
   6:35:18 PM     Started executing query at Line 1
                  Commands completed successfully.
   6:35:18 PM     Started executing query at Line 29
                  Commands completed successfully.
   6:35:18 PM     Started executing query at Line 59
                  Commands completed successfully.
   6:35:18 PM     Started executing query at Line 90
                  Commands completed successfully.
   6:35:18 PM     Started executing query at Line 119
                  Commands completed successfully.
   6:35:18 PM     Started executing query at Line 145
                  Commands completed successfully.
   6:35:18 PM     Started executing query at Line 171
                  Commands completed successfully.
   6:35:18 PM     Started executing query at Line 197
                  Commands completed successfully.
   6:35:18 PM     Started executing query at Line 227
                  Commands completed successfully.
   6:35:18 PM     Started executing query at Line 255
                  Commands completed successfully.
   6:35:18 PM     Started executing query at Line 281
                  Commands completed successfully.
   6:35:18 PM     Started executing query at Line 309
                  Commands completed successfully.
   6:35:18 PM     Started executing query at Line 337
                  Commands completed successfully.
   6:35:18 PM     Started executing query at Line 359
                  Commands completed successfully.
   6:35:18 PM     Started executing query at Line 383
                  Commands completed successfully.
   6:35:18 PM     Started executing query at Line 409
                  Commands completed successfully.
```



```
6:35:18 PM     Started executing query at Line 439
               Commands completed successfully.
6:35:18 PM     Started executing query at Line 493
               Commands completed successfully.
6:35:18 PM     Started executing query at Line 511
               Commands completed successfully.
6:35:18 PM     Started executing query at Line 571
               Commands completed successfully.
6:35:18 PM     Started executing query at Line 589
               Commands completed successfully.
6:35:18 PM     Started executing query at Line 614
               Commands completed successfully.
6:35:18 PM     Started executing query at Line 638
               Commands completed successfully.
6:35:18 PM     Started executing query at Line 663
               Commands completed successfully.
6:35:18 PM     Started executing query at Line 683
               Commands completed successfully.
6:35:18 PM     Started executing query at Line 701
               Commands completed successfully.
               Total execution time: 00:00:02.925
```

ERROR HANDLING

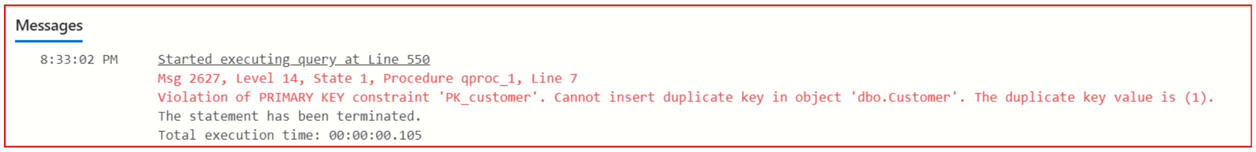
```
SELECT * FROM Customer;
```



	cust_name	cust_address	category
1	1	A1	10

Inserting another row with same primary key to check for primary key violation

```
        EXEC qproc_1 @cust_name=1,@cust_address ='A2',@category=9;
```



```
8:33:02 PM     Started executing query at Line 550
               Msg 2627, Level 14, State 1, Procedure qproc_1, Line 7
               Violation of PRIMARY KEY constraint 'PK_customer'. Cannot insert duplicate key in object 'dbo.Customer'. The duplicate key value is (1).
               The statement has been terminated.
               Total execution time: 00:00:00.105
```

Here Error handling is shown only for one table by Azure SQL , However, it is same for all the other tables.
Error handling is done completely by Azure SQL Database and no error handling is done by Java program.

5. 2 The Java source program and screenshots showing its successful compilation

Please refer to the Appidi_Harinadh_Task5b.java file for the source program.
The screenshot for successful compilation can be found below.

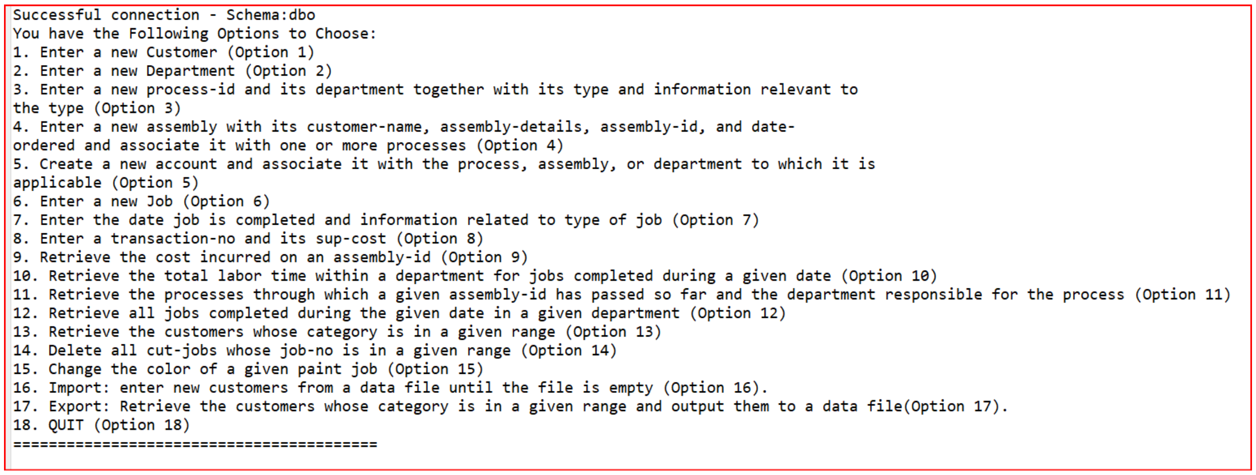
```
                         //Importing the necessary Packages

import java.util.Scanner;
import java.sql.Connection;
import java.sql.Statement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.DriverManager;
import java.io.File;
import java.io.FileWriter;

public class Main {

     // Database credentials
    final static String HOSTNAME = "dbms-fall2021.database.windows.net";
    final static String DBNAME = "cs-dsa-4513-sql-db";
    final static String USERNAME = "appi0005";
    final static String PASSWORD = "Schrodinger9618@";

    // Database connection string
    final static String URL = 
String.format("jdbc:sqlserver://%s:1433;database=%s;user=%s;password=%s;encrypt=t
rue;trustServerCertificate=false;hostNameInCertificate=*.database.windows.net;log
inTimeout=30;",
            HOSTNAME, DBNAME, USERNAME, PASSWORD);


    public static void main(String[] args) throws SQLException {
        // Connect to database

        try (final Connection connection = DriverManager.getConnection(URL)) {


            //Scanner myScan = new Scanner(System.in);
            int Choice = 0;
            while(Choice != 18)
            {

            // printing out the available choices for the User to choose.
            System.out.println("****************** START
*************************");
            System.out.println("You have the Following Options to Choose:");
            System.out.println("1. Enter a new Customer (Option 1)");
            System.out.println("2. Enter a new Department (Option 2)");
            System.out.println("3. Enter a new process-id and its department
together with its type and information relevant to \r\n"
                    + "the type (Option 3)");
            System.out.println("4. Enter a new assembly with its customer-name,
assembly-details, assembly-id, and date- \r\n"
                    + "ordered and associate it with one or more processes
(Option 4)");
            System.out.println("5. Create a new account and associate it with the
process, assembly, or department to which it is \r\n"
                    + "applicable (Option 5)");
            System.out.println("6. Enter a new Job (Option 6)");
            System.out.println("7. Enter the date job is completed and
information related to type of job (Option 7)");
            System.out.println("8. Enter a transaction-no and its sup-cost
(Option 8)");
            System.out.println("9. Retrieve the cost incurred on an assembly-id
(Option 9)");
            System.out.println("10. Retrieve the total labor time within a
department for jobs completed during a given date (Option 10)");
            System.out.println("11. Retrieve the processes through which a given
assembly-id has passed so far and the department responsible for the process
(Option 11)");
            System.out.println("12. Retrieve all jobs completed during the given
date in a given department (Option 12)");
            System.out.println("13. Retrieve the customers whose category is in a
given range (Option 13)");
            System.out.println("14. Delete all cut-jobs whose job-no is in a
given range (Option 14)");
            System.out.println("15. Change the color of a given paint job (Option
15)");
            System.out.println("16. Import: enter new customers from a data file
until the file is empty (Option 16).");
            System.out.println("17. Export: Retrieve the customers whose category
is in a given range and output them to a data file(Option 17).");
            System.out.println("18. QUIT (Option 18)");
            System.out.println("\n ****************** END
************************* \n");


            // initializing Scanner object
            Scanner myScan = new Scanner(System.in);
            //reading the input given by the user
            Choice = myScan.nextInt();

            //Depending on the choice made by the user, we are defining the
operations to be done

            switch (Choice) {

            case 1:
                // try and catch are used to not terminate loop in case of error.
                try {

                //Declaring the variables
                int category;
                String cname, caddress;

                //Taking customer name from user
                System.out.println("Enter the Customer Name");
                cname = myScan.next();

                // Taking Customer Address from the user
                System.out.println("Enter the Customer Address");
                caddress = myScan.next();

                // Taking customer category from the user
                System.out.println("Enter the Customer Category");
                category = myScan.nextInt();

                // Executing procedure for Query 1.
                final String Sql1 = "EXEC qproc_1 @cust_name = '"+cname+"',
@cust_address = '"+caddress+"', @category = '"+category+"';";

                final Statement statement1 = connection.createStatement();
                    statement1.executeUpdate(Sql1);

                    System.out.println("Customer Record Inserted Sucessfully.");

                } catch (Exception e) {
                    System.err.println("Error! Returning to main menu");
                }
                break;

            case 2:
                // try and catch are used to not terminate loop in case of error.
                try {

                // Declaring variables
                String department_no, department_data;

                // Taking Department Number from user.
                System.out.println("Enter the Department Number\n");
                department_no = myScan.next();

                // Taking Department data from user.
                System.out.println("Enter the Department Data\n");
                department_data = myScan.next();

                // Executing Procedure for Query 2.
                final String Sql2 = "EXEC qproc_2 @department_no =
'"+department_no+"', @department_data = '"+department_data+"';";

                final Statement statement2 = connection.createStatement();
                    statement2.executeUpdate(Sql2);

                    System.out.println("Department record inserted
successfully.\n");

                } catch (Exception e) {
                    System.err.println("Error! Returning to main menu");
                }
                break;
            case 3:
                // try and catch are used to not terminate loop in case of error.
                try {

                // Declaring Variables
                String process_id, process_data, department_no;

                // Taking process-id from the user
                System.out.println("Enter Process ID");
                process_id = myScan.next();

                // Taking process data from the user
                System.out.println("Enter Process Data");
                process_data = myScan.next();

                // Taking Department-no from the user
                System.out.println("Enter Department No");
                department_no = myScan.next();
                //Executing Procedure for Query 3
                final String Sql3 = "EXEC qproc_3 @process_id = 
'"+process_id+"'," + 
                        " @process_data = '"+process_data+"', @department_no = 
'"+department_no+"';";

                final Statement statement3 = connection.createStatement();
                    statement3.executeUpdate(Sql3);

                    System.out.println("Process record inserted 
successfully.\n");

                // Asking user to enter the type of procedure
                System.out.println("Choose one of the following type of 
process:\n 1.Fit\n 2. Paint\n 3.Cut\n");
                int prcs_choice;
                // Taking the type of process from the user
                prcs_choice = myScan.nextInt();

                if (prcs_choice ==1) {

                    // Declaring Variables
                    String fit_type;

                    // Taking fit type from user
                    System.out.println("Enter fit type\n");
                    fit_type = myScan.next();

                    // Executing Procedure to insert data in Fit_Process table
                    final String Sql3_1 = "EXEC qproc_3_1 @process_id = 
'"+process_id+"',  @fit_type = '"+fit_type+"';";

                    final Statement statement3_1 = connection.createStatement();
                        statement3_1.executeUpdate(Sql3_1);

                        System.out.println("Fit Process Record inserted 
successfully.\n");

                }

                if (prcs_choice ==2) {

                    // Declaring Variables
                    String paint_type, painting_method;
                    //Taking paint type from the user
                    System.out.println("Enter paint type");
                    paint_type = myScan.next();

                    // Taking paint method from the user
                    System.out.println("Enter painting method");
                    painting_method = myScan.next();

                    // Executing procedure to insert data into Paint_Process
table
                    final String Sql3_2 = "EXEC qproc_3_2 @process_id =
'"+process_id+"', " +
                            " @paint_type = '"+paint_type+"', @painting_method =
'"+painting_method+"';";

                    final Statement statement3_2 = connection.createStatement();
                        statement3_2.executeUpdate(Sql3_2);

                        System.out.println("Paint Process record inserted
successfully.\n");

                }

                if (prcs_choice ==3) {

                    // Declaring Variables
                    String cutting_type, machine_type;

                    //Taking cut type from the user
                    System.out.println("Enter cutting type");
                    cutting_type = myScan.next();

                    // Take machine type from user
                    System.out.println("Enter machine type");
                    machine_type = myScan.next();

                    //Executing Procedure to insert data into Process_cut table.
                    final String Sql3_3 = "EXEC qproc_3_3 @process_id =
'"+process_id+"', " +
                            " @cutting_type = '"+cutting_type+"', @machine_type =
'"+machine_type+"';";

                    final Statement statement3_3 = connection.createStatement();
                        statement3_3.executeUpdate(Sql3_3);
                        System.out.println("Cut Process record inserted
successfully.\n");

                }
                } catch (Exception e) {
                    System.out.println("You got an error!. Returning to main
menu");
                }
                break;

            case 4:
                // try and catch are used to not terminate loop in case of error.
                try {
                    // Declaring Variables
                String asmbly_id, asmbly_dtls, cust_name;
                String date_ordrd;

                // Taking Assembly-id from user
                System.out.println("Enter Assembly ID");
                asmbly_id = myScan.next();

                // Taking date ordered from user
                System.out.println("Enter Date Ordered in YYYY-MM-DD format");
                date_ordrd = myScan.next();

                // Taking Assembly details from user
                System.out.println("Enter Assembly Details");
                asmbly_dtls = myScan.next();

                // Taking customer name from user
                System.out.println("Enter the Customer Name");
                cust_name = myScan.next();

                // Taking Processes associated with this assembly
                System.out.println("Enter number of processes associated with
this Assembly:\n");
                int n = myScan.nextInt();
                int count=0;




                // Executing procedure for Query 4
                final String Sql4 = "EXEC qproc_4 @asmbly_id = '"+asmbly_id+"',
@date_ordrd = '"+date_ordrd+"'," +
                            "@asmbly_dtls = '"+asmbly_dtls+"', @cust_name =
'"+cust_name+"';";

                final Statement statement4 = connection.createStatement();
                    statement4.executeUpdate(Sql4);


                while(count <n) {
                    System.out.println("Enter Process ID - "+(count+1));
                    String process_id = myScan.next();

                    // Executing procedure for Query 4_1
                    final String Sql4_1 = "EXEC qproc_4_1 @asmbly_id =
'"+asmbly_id+"', @process_id = '"+process_id+"';";

                    final Statement statement4_1 = connection.createStatement();
                        statement4_1.executeUpdate(Sql4_1);

                        System.out.println("Manufactured record inserted
successfully.");
                        count++;
                }

                    System.out.println("Assembly record inserted successfully.");

                } catch (Exception e) {
                    System.err.println("Error! Returning to main menu");
                }
                break;



            case 5:
                // try and catch are used to not terminate loop in case of error.
                try {

                    // Declaring Variables
                int acc_no, act_choice;
                String account_start_date;

                // Taking account number from user
                System.out.println("Enter account number");
                acc_no = myScan.nextInt();
                // Taking date account established from user
                System.out.println("Enter the date account established in YYYY-
MM-DD format");
                account_start_date = myScan.next();

                //Executing Procedure for Query 3
                final String Sql3 = "EXEC qproc_5 @account_no = '"+acc_no+"'," +
                        " @account_start_date = '"+account_start_date+"';";

                final Statement statement3 = connection.createStatement();
                    statement3.executeUpdate(Sql3);

                    System.out.println("Account record inserted successfully.");


                // Asking user to provide the type of account
                System.out.println("Choose one of the following type of
account:\n 1. Department Account.\n 2. Assembly Account\n 3.Process Account\n");
                act_choice = myScan.nextInt();


                if (act_choice == 1) {
                    // Declaring Variables

                    String department_no;

                    // Taking details-2 from the user
                    /*
                        * System.out.println("Enter account details"); details2 =
myScan.nextFloat();
                        */

                    // Taking department no from the user
                    System.out.println("Enter Department Number of the account");
                    department_no = myScan.next();

                    // Executing procedure to insert data into Department Account
Table

                    final String Sql5_1 = "EXEC qproc_5_1 @account_no =
'"+acc_no+"',"+
                            " @department_no = '"+department_no+"';";

                    final Statement statement5_1 = connection.createStatement();
                        statement5_1.executeUpdate(Sql5_1);
                        System.out.println("Record inserted successfully in 
Department_Account");

                }

                if (act_choice == 2) {

                    // Declaring Variables

                    String asmbly_id;

                    // Taking details1 from user

                    /*
                        * System.out.println("Enter account details"); details1 = 
mySCan.nextFloat();
                        */

                    // Taking assembly-id from user
                    System.out.println("Enter Assembly id of the account");
                    asmbly_id = myScan.next();

                    // Executing procedure to insert data into Assembly account
                    final String Sql5_2 = "EXEC qproc_5_2 @account_no = 
'"+acc_no+"',"+
                            "@asmbly_id = '"+asmbly_id+"';";

                    final Statement statement5_2 = connection.createStatement();
                        statement5_2.executeUpdate(Sql5_2);

                        System.out.println("Record inserted successfully in 
Assembly Account");

                }

                if (act_choice == 3) {

                    // Declaring Variables

                    String process_id;

                    // Taking details from the user
                    /*
```

```
                         * System.out.println("Enter account details"); details3 =
myScan.nextFloat();
                         */

                     // Taking process-id from the user
                     System.out.println("Enter Process id of the account");
                     process_id = myScan.next();

                     // Executing the procedure to insert data into Process
account
                     final String Sql5_3 = "EXEC qproc_5_3 @account_no =
'"+acc_no+"'," +
                             "@process_id = '"+process_id+"';";

                     final Statement statement5_3 = connection.createStatement();
                         statement5_3.executeUpdate(Sql5_3);

                         System.out.println("Record inserted successfully in
Process Account");

                 }
                 } catch (Exception e) {
                     System.out.println("Error! Returning to main menu");
                 }
                 break;

             case 6:
                 // try and catch are used to not terminate loop in case of error.
                 try {

                     // Declaring Variables
                 String job_start_date, assembly_id, process_id;
                 int job_no;

                 // Taking Job-no from the user
                 System.out.println("Enter Job-no for a new job");
                 job_no = myScan.nextInt();

                 // Taking assembly-id from the user
                 System.out.println("Enter assembly-id for the job");
                 assembly_id = myScan.next();

                 // Taking process-id from user
                 System.out.println("Enter process-id for the job");
                 process_id = myScan.next();
```

```
                // Taking job commenced from user
                System.out.println("Enter job start date in YYYY-MM-DD format");
                job_start_date = myScan.next();

                // Executing Procedure for Query 6
                final String Sql6 = "EXEC qproc_6 @job_no = '"+job_no+"',
@job_start_date = '"+job_start_date+"',"+
                        " @asmbly_id = '"+assembly_id+"', @process_id =
'"+process_id+"';";

                final Statement statement6 = connection.createStatement();
                    statement6.executeUpdate(Sql6);

                    System.out.println("Job record inserted successfully.");

                } catch (Exception e) {
                    System.err.println("Error! Returning to main menu");
                }
                break;

            case 7:
                    // try and catch are used to not terminate loop in case of
error.
                    try {
                        // Declaring Variables
                    String job_end_date;
                    int job_no;

                    // Taking Job-no from user
                    System.out.println("Enter Job-no for the completed job");
                    job_no = myScan.nextInt();

                    // Taking date completed from user
                    System.out.println("Enter job completed date in YYYY-MM-DD
format");

                    job_end_date = myScan.next();

                    // Executing the Procedure for Query 7
                    final String Sql7 = "EXEC qproc_7 @job_no = '"+job_no+"',
@job_end_date = '"+job_end_date+"';";

                    final Statement statement7 = connection.createStatement();
                        statement7.executeUpdate(Sql7);
                        System.out.println("Job Record Updated successfully.");

                        // Declaring Variables
                    int job_choice;

                    // Asking the type of job from the user
                    System.out.println("Choose one of the following type of
job:\n 1. Fit Job.\n 2. Paint Job\n 3.Cut Job\n");
                    job_choice = myScan.nextInt();


                    if (job_choice == 1) {

                        // Declaring Variables
                        String fit_labor_time;

                        // Taking fit labor time from user
                        System.out.println("Enter the fit job labor time in
minutes");

                        fit_labor_time = myScan.next();

                        // Executing procedure to insert data into Job_fit table
                        final String Sql7_1 = "EXEC qproc_7_1 @job_no =
'"+job_no+"', @labor_time = '"+fit_labor_time+"';";

                        final Statement statement7_1 =
connection.createStatement();
                            statement7_1.executeUpdate(Sql7_1);

                            System.out.println("Fit Job record inserted
successfully.");

                    }

                    if (job_choice == 2) {

                        // Declaring Variables
                        String color, paint_labor_time;
                        int volume;

                        // Taking color from user
                        System.out.println("Enter the paint color.");
                        color = myScan.next();

                        // Taking labor time from user
                        System.out.println("Enter the paint job labor time in 
minutes");
                        paint_labor_time = myScan.next();

                        // Taking volume of paint from user
                        System.out.println("Enter the volume of paint.");
                        volume = myScan.nextInt();

                        // Executing the procedure to insert data into Job paint 
table
                        final String Sql7_2 = "EXEC qproc_7_2 @job_no = 
'"+job_no+"', @color = '"+color+"', "+
                                " @volume = '"+volume+"', @labor_time = 
'"+paint_labor_time+"';";

                        final Statement statement7_2 = 
connection.createStatement();
                            statement7_2.executeUpdate(Sql7_2);

                            System.out.println("Paint Job record inserted 
successfully.");

                    }

                    if (job_choice == 3) {

                        // Declaring Variables
                        String job_machine_type, machine_time, material_used, 
cut_labor_time;

                        // Taking machine type from user
                        System.out.println("Enter the Job Machine Type.");
                        job_machine_type = myScan.next();

                        // Taking machine time from user
                        System.out.println("Enter the cut job machine time in 
minutes");
                        machine_time = myScan.next();

                        // Taking material used from the user
                        System.out.println("Enter the material used.");
                        material_used = myScan.next();

                        // Taking labor time from the user
                        System.out.println("Enter the cut job labor time in 
minutes");
                        cut_labor_time = myScan.next();

                        // Executing Procedure to insert data into Job_cut table
                        final String Sql7_3 = "EXEC qproc_7_3 @job_no = 
'"+job_no+"', @machine_type= '"+job_machine_type+"', "+
                        " @machine_time = '"+machine_time+"', @material = 
'"+material_used+"', @labor_time = '"+cut_labor_time+"';";

                        final Statement statement7_3 = 
connection.createStatement();
                            statement7_3.executeUpdate(Sql7_3);

                            System.out.println("Cut Job record inserted 
successfully.");

                    }
                    } catch (Exception e) {
                        System.err.println("Error! Returning to main menu");
                    }
                    break;
            case 8:
                // try and catch are used to not terminate loop in case of error.
                try {

                    // Declaring Variables
                String t_no;
                int job_no;
                float sup_cost;

                // Taking transaction number from the user
                System.out.println("Enter the Transaction Number.");
                t_no = myScan.next();

                // Taking sup-cost from the user
                System.out.println("Enter the cost of the transaction.");
                sup_cost = myScan.nextFloat();

                // Taking job-no from the user
                System.out.println("Enter the Job number related to 
transaction.");
                job_no = myScan.nextInt();

                // Executing procedure for Query 8.
                final String Sql8 = "EXEC qproc_8 @transaction_no = '"+t_no+"',
@sup_cost = '"+sup_cost+"', @job_no = '"+job_no+"';";

                final Statement statement8 = connection.createStatement();
                    statement8.executeUpdate(Sql8);

                    System.out.println("Transaction Record inserted and related
accounts updated successfully.");

                } catch (Exception e) {
                    System.err.println("Error! Returning to main menu");
                }
                break;

            case 9:
                // try and catch are used to not terminate loop in case of error.
                try {
                    // Declaring Variables
                String assembly_id;

                // Taking assembly-id from the user
                System.out.println("Enter Assembly Id to retrieve the cost.");
                assembly_id = myScan.next();

                try {

                // Executing procedure for Query 9
                final String Sql9 = "EXEC qproc_9 @asmbly_id =
'"+assembly_id+"';";

                try (final Statement statement9 = connection.createStatement();
                        final ResultSet resultSet1 =
statement9.executeQuery(Sql9)) {

                        System.out.println(String.format("Cost incurred on
assembly-id %s:", assembly_id));
                        while (resultSet1.next()) {
                            System.out.println(String.format("%f",
                                resultSet1.getFloat(1)));

                            }
                        }
                } catch (Exception e) {
                    System.out.println("Error! Please try again.");
                }
                } catch (Exception e) {
                    System.err.println("Error! Returning to main menu");
                }
                break;

            case 10:
                // try and catch are used to not terminate loop in case of error.
                try {

                    // Declaring Variables
                String dept_no,job_end_date;

                // Taking department-no from user
                System.out.println("Enter Department Number to get the total
labor time.");
                dept_no = myScan.next();

                // Taking job-completed date from user
                System.out.println("Enter Job completed date in YYYY-MM-DD format
to get the total labor time.");
                job_end_date = myScan.next();

                // Executing the procedure for Query 10
                final String Sql10 = "EXEC qproc_10 @department_no =
'"+dept_no+"', @job_end_date = '"+job_end_date+"';";

                try (final Statement statement10 = connection.createStatement();
                        final ResultSet resultSet2 =
statement10.executeQuery(Sql10)) {

                        System.out.println(String.format("Total labor-time in
minutes for department number %s and date" +
                                " job completed %s:", dept_no, job_end_date));
                        while (resultSet2.next()) {

                            System.out.println(String.format("%s",
                                resultSet2.getInt(1)));

                            }
                        }
                } catch (Exception e) {
                    System.err.println("Error! Returning to main menu"+e);
                }
                break;
            case 11:
                // try and catch are used to not terminate loop in case of error.
                try {

                    // Declaring Variables
                String assembly_id;

                // Taking assembly-id from user
                System.out.println("Enter the Assembly Id to retrieve the 
processes.");
                assembly_id = myScan.next();

                // Executing procedure for Query 11d
                final String Sql11 = "EXEC qproc_11 @asmbly_id = 
'"+assembly_id+"';";

                try (final Statement statement11 = connection.createStatement();
                        final ResultSet resultSet3 = 
statement11.executeQuery(Sql11)) {

                        System.out.println(String.format("The processes through 
which Assembly Id %s passed so far:", assembly_id));
                        while (resultSet3.next()) {
                            System.out.println("Process ID | Department No | Job 
Start Date");

                            System.out.println(String.format("%s | %s | %s",
                                resultSet3.getString(1),
                                resultSet3.getString(2),
                                resultSet3.getString(3)));

                            }
                        }
                } catch (Exception e) {
                    System.err.println("Error! Returning to main menu" );
                }
                break;

            case 12:
                // try and catch are used to not terminate loop in case of error.
                try {

                    // Declaring Variables
                String job_end_date, dept_no;

                // Taking date completed from user
                System.out.println("Enter the date job is completed in YYYY-MM-DD
format");
                job_end_date = myScan.next();

                // Taking department number from user
                System.out.println("Enter the department to retrieve the jobs.");
                dept_no = myScan.next();

                // Executing Procedure to retrieve fit jobs
                final String Sql12_1 = "EXEC qproc_12_1 @job_end_date =
'"+job_end_date+"', @department_no = '"+dept_no+"';";

                try (final Statement statement12_1 =
connection.createStatement();
                        final ResultSet resultSet5 =
statement12_1.executeQuery(Sql12_1)) {

                        System.out.println(String.format("The fit jobs completed
on date %s in department %s:", job_end_date, dept_no));
                        while (resultSet5.next()) {
                            System.out.println(String.format("%s | %s | %s",
                                resultSet5.getString(1),
                                resultSet5.getString(2),
                                resultSet5.getString(3)));

                            }
                        }
                // Executing procedure to retrieve paint jobs
                final String Sql12_2 = "EXEC qproc_12_2 @job_end_date =
'"+job_end_date+"', @department_no = '"+dept_no+"';";

                try (final Statement statement12_2 =
connection.createStatement();
                        final ResultSet resultSet6 =
statement12_2.executeQuery(Sql12_2)) {

                        System.out.println(String.format("The Paint jobs
completed on date %s in department %s:", job_end_date, dept_no));
                        while (resultSet6.next()) {
                            System.out.println(String.format("%s | %s | %s | %s |
%s",
                                resultSet6.getString(1),
                                resultSet6.getString(2),
                                resultSet6.getString(3),
                                resultSet6.getString(4),
                                resultSet6.getString(5)));

                            }
                        }


                // Executing procedure to retrieve cut jobs
                final String Sql12_3 = "EXEC qproc_12_3 @job_end_date = 
'"+job_end_date+"', @department_no = '"+dept_no+"';";

                try (final Statement statement12_3 = 
connection.createStatement();
                        final ResultSet resultSet7 = 
statement12_3.executeQuery(Sql12_3)) {

                        System.out.println(String.format("The Cut jobs completed 
on date %s in department %s:", job_end_date, dept_no));
                        while (resultSet7.next()) {
                            System.out.println(String.format("%s | %s | %s | %s | 
%s | %s",
                                resultSet7.getString(1),
                                resultSet7.getString(2),
                                resultSet7.getString(3),
                                resultSet7.getString(4),
                                resultSet7.getString(5),
                                resultSet7.getString(6)));

                            }
                        }
                } catch (Exception e) {
                    System.err.println("Error! Returning to main menu");
                }

                break;

            case 13:
                // try and catch are used to not terminate loop in case of error.
                try {

                    // Declaring Variables
                int lower_b, upper_b;

                // Taking lower bound of category from user
                System.out.println("Enter the lower bound of the category.");
                lower_b = myScan.nextInt();
                // Taking upper bound of category from user
                System.out.println("Enter the upper bound of the category.");
                upper_b = myScan.nextInt();

                // Executing procedure for Query 13
                final String Sql13 = "EXEC qproc_13 @clower = '"+lower_b+"', 
@cupper = '"+upper_b+"';";

                try (final Statement statement13 = connection.createStatement();
                        final ResultSet resultSet4 = 
statement13.executeQuery(Sql13)) {

                        System.out.println("The customers in the given range of 
category are");
                        while (resultSet4.next()) {
                            System.out.println(String.format("%s | %s",
                                resultSet4.getString(1),
                                resultSet4.getString(2)));

                            }
                        }
                } catch (Exception e) {
                    System.err.println("Error! Returning to main menu");
                }
                break;

            case 14:
                // try and catch are used to not terminate loop in case of error.
                try {

                    // Declaring Variables
                int lower_b1, upper_b1;

                // Taking lower bound job-no from user
                System.out.println("Enter the lower bound of the cut job-no.");
                lower_b1 = myScan.nextInt();

                // Taking upper bound job-no from user
                System.out.println("Enter the upper bound of the cut job-no.");
                upper_b1 = myScan.nextInt();

                // Executing procedure for Query 14
                final String Sql14 = "EXEC qproc_14 @jno_lower = '"+lower_b1+"', 
@jno_upper = '"+upper_b1+"';";
```

```
                final Statement statement14 = connection.createStatement();
                    statement14.executeUpdate(Sql14);

                    System.out.println("Deleted all cut jobs in the given range
of job-no.");

                } catch (Exception e) {
                    System.err.println("Error! Returning to main menu");
                }
                break;

            case 15:
                // try and catch are used to not terminate loop in case of error.
                try {

                    // Declaring Variables
                String color;
                int job_no;

                // Taking job-no from user
                System.out.println("Enter the paint job-no.");
                job_no = myScan.nextInt();

                // Taking color from user
                System.out.println("Enter the new color for the job-no.");
                color = myScan.next();

                // Executing procedure for Query 15
                final String Sql15 = "EXEC qproc_15 @job_no = '"+job_no+"',
@color = '"+color+"';";

                final Statement statement15 = connection.createStatement();
                    statement15.executeUpdate(Sql15);

                    System.out.println("Changed the color of a given paint
job.");

                } catch (Exception e) {
                    System.err.println("Error! Returning to main menu");
                }
                break;

            case 16:
                // try and catch are used to not terminate loop in case of error.
                try {
                    // Declaring Variables
                String file_name, line;

                // Taking Input file name from user
                System.out.println("Enter the file-name to Import data.");
                file_name = myScan.next();

                // Creating new File object
                File file = new File(file_name);

                // Creating new Scanner Object
                Scanner sc = new Scanner(file);


                // While loop to read all lines in the input file
                while(sc.hasNextLine()) {
                    line = sc.nextLine();

                    // Dividing line to parts separated by Delimiter (",")
                    String[] parts = line.split(",");

                    String cname = parts[0];
                    String caddress = parts[1];
                    int category = Integer.parseInt(parts[2]);

                    System.out.println(category);
                    // Execute procedure for Query 16
                    final String Sql16 = "EXEC qproc_1 @cust_name = '"+cname+"',
@cust_address = '"+caddress+"', @category = '"+category+"';";

                    final Statement statement16 = connection.createStatement();
                        statement16.executeUpdate(Sql16);

                }
                sc.close();

                } catch (Exception e) {
                    System.err.print("Error! Returning to main menu \r\n"+e);
                }
                break;
            case 17:
                // try and catch are used to not terminate loop in case of error.
                try {
                    // Declaring Variables
                    String file_name1, lower_b, upper_b;

                    // Enter the filename to output result
                    System.out.println("Enter the file-name to Export data.");
                    file_name1 = myScan.next();

                    // Taking lower bound of category from user
                    System.out.println("Enter the lower bound of category.");
                    lower_b = myScan.next();

                    // Taking upper bound of category from user
                    System.out.println("Enter the upper bound of category.");
                    upper_b = myScan.next();

                    // Creating new file writer Object
                    FileWriter fw = new FileWriter(file_name1);

                    // Executing Procedure(for Query 13) to get output
                    final String Sql17 = "EXEC qproc_13 @clower = '"+lower_b+"',
@cupper = '"+upper_b+"';";


                    try (final Statement statement17 =
connection.createStatement();
                            final ResultSet resultSet4 =
statement17.executeQuery(Sql17)) {

                            //System.out.println("The customers in the given
range of category are");
                            while (resultSet4.next()) {

                                //fw.write(String.format("%s,%s",
resultSet4.getString(1), resultSet4.getString(2)));
                                fw.write(resultSet4.getString(1) + "," +
resultSet4.getString(2) + "\n");
                                }
                            fw.close();
                            } catch (SQLException e) {
                                e.getCause().getMessage();
                            }


                    } catch (Exception e) {
                        System.err.println("Error! Returning to main menu");
```

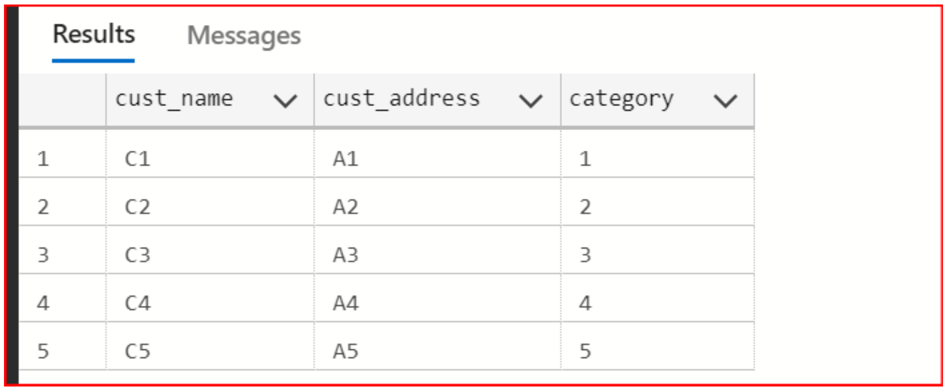
```java
                    }
                break;

            case 18:
                System.out.println("You choose to Quit! Bye!");

            }
            myScan.close();
        }
    }
    }
}
```



```
Successful connection - Schema:dbo
You have the Following Options to Choose:
1. Enter a new Customer (Option 1)
2. Enter a new Department (Option 2)
3. Enter a new process-id and its department together with its type and information relevant to
the type (Option 3)
4. Enter a new assembly with its customer-name, assembly-details, assembly-id, and date-
ordered and associate it with one or more processes (Option 4)
5. Create a new account and associate it with the process, assembly, or department to which it is
applicable (Option 5)
6. Enter a new Job (Option 6)
7. Enter the date job is completed and information related to type of job (Option 7)
8. Enter a transaction-no and its sup-cost (Option 8)
9. Retrieve the cost incurred on an assembly-id (Option 9)
10. Retrieve the total labor time within a department for jobs completed during a given date (Option 10)
11. Retrieve the processes through which a given assembly-id has passed so far and the department responsible for the process (Option 11)
12. Retrieve all jobs completed during the given date in a given department (Option 12)
13. Retrieve the customers whose category is in a given range (Option 13)
14. Delete all cut-jobs whose job-no is in a given range (Option 14)
15. Change the color of a given paint job (Option 15)
16. Import: enter new customers from a data file until the file is empty (Option 16).
17. Export: Retrieve the customers whose category is in a given range and output them to a data file(Option 17).
18. QUIT (Option 18)
=========================================
```

Task 6. Java program Execution

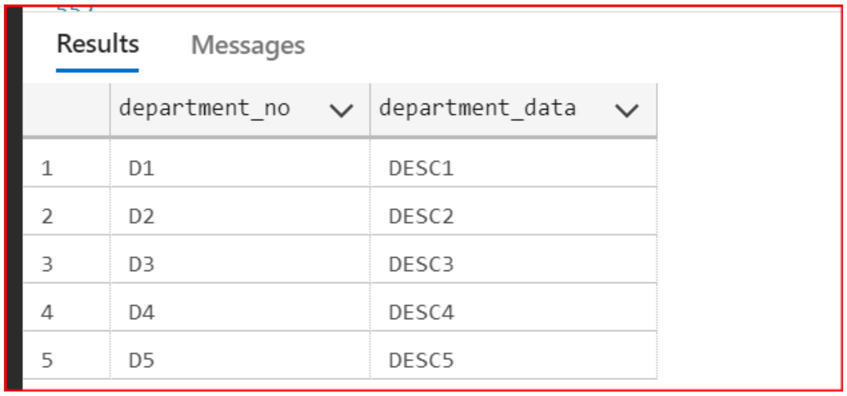
6.1. Screenshots showing the testing of query 1



Results Messages

	cust_name	cust_address	category
1	C1	A1	1
2	C2	A2	2
3	C3	A3	3
4	C4	A4	4
5	C5	A5	5

6.2 Screenshots showing the testing of query 2

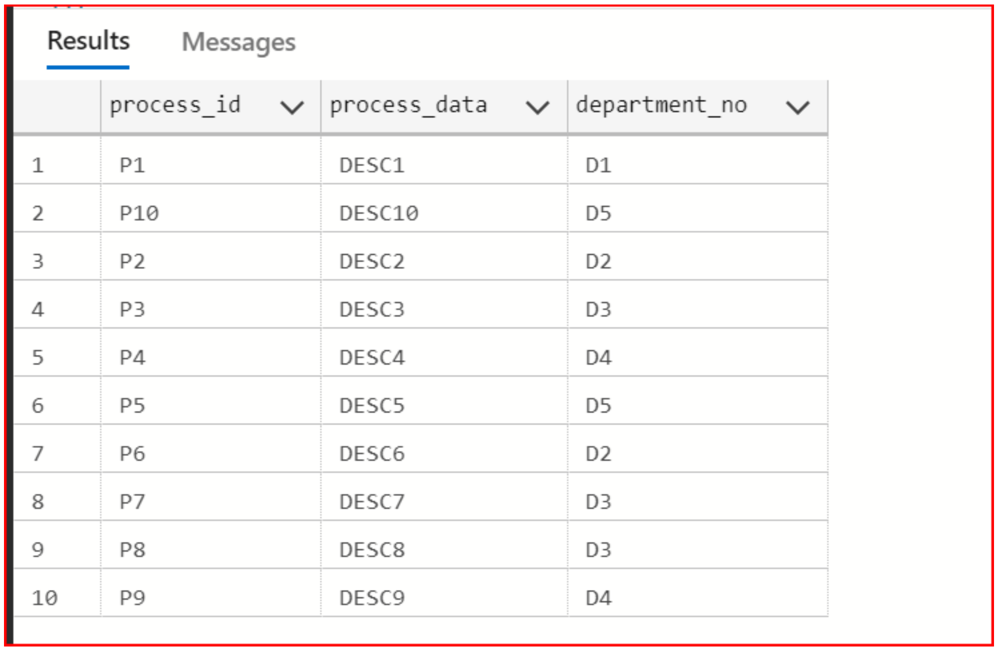


Results Messages

	department_no	department_data
1	D1	DESC1
2	D2	DESC2
3	D3	DESC3
4	D4	DESC4
5	D5	DESC5

6.3 Screenshots showing the testing of query 3

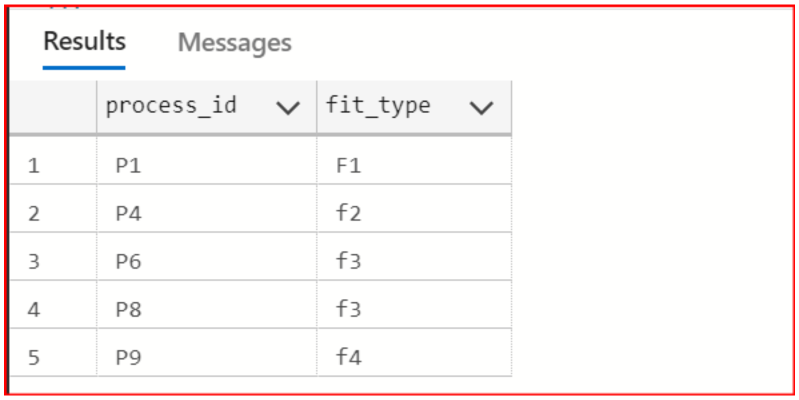
Process Table



Results Messages

	process_id	process_data	department_no
1	P1	DESC1	D1
2	P10	DESC10	D5
3	P2	DESC2	D2
4	P3	DESC3	D3
5	P4	DESC4	D4
6	P5	DESC5	D5
7	P6	DESC6	D2
8	P7	DESC7	D3
9	P8	DESC8	D3
10	P9	DESC9	D4

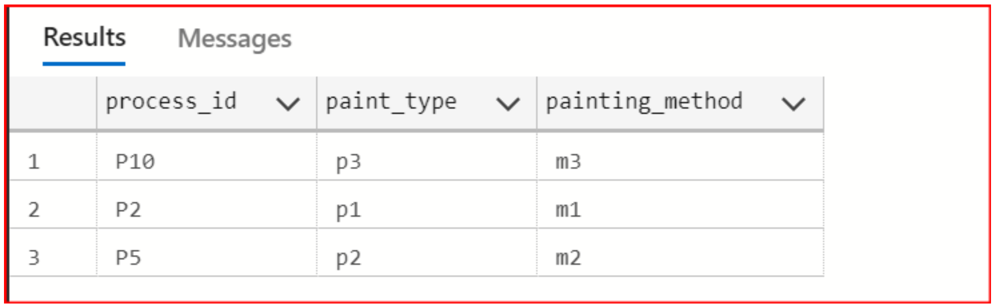
Fit_Process Table



Results Messages

	process_id	fit_type
1	P1	F1
2	P4	f2
3	P6	f3
4	P8	f3
5	P9	f4

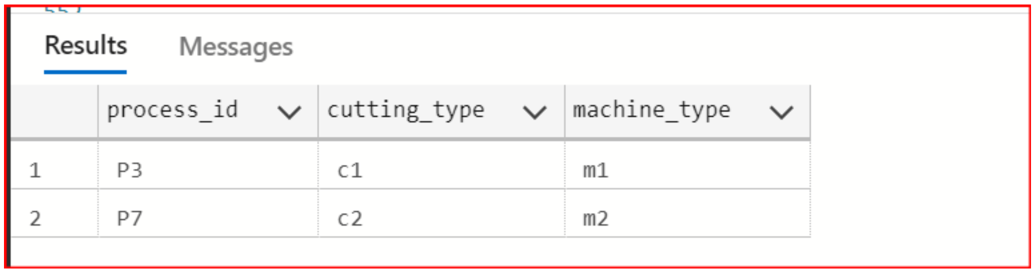
Paint_Process Table



Results Messages

	process_id	paint_type	painting_method
1	P10	p3	m3
2	P2	p1	m1
3	P5	p2	m2

Cut_Process Table

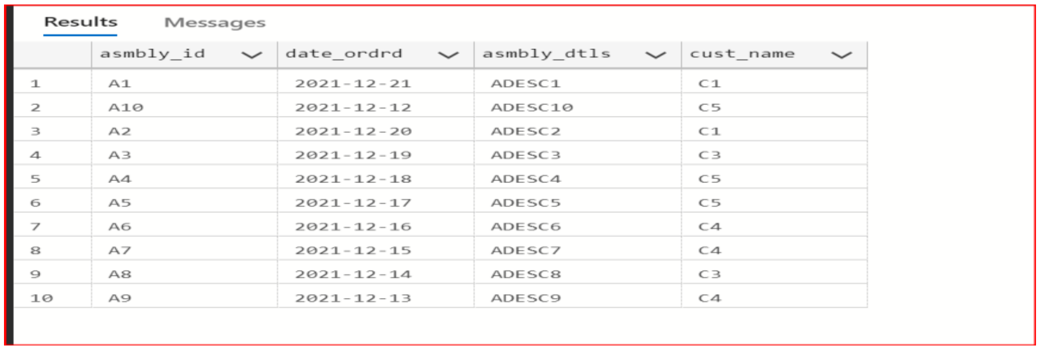


Results Messages

	process_id	cutting_type	machine_type
1	P3	c1	m1
2	P7	c2	m2

6.4 Screenshots showing the testing of query 4

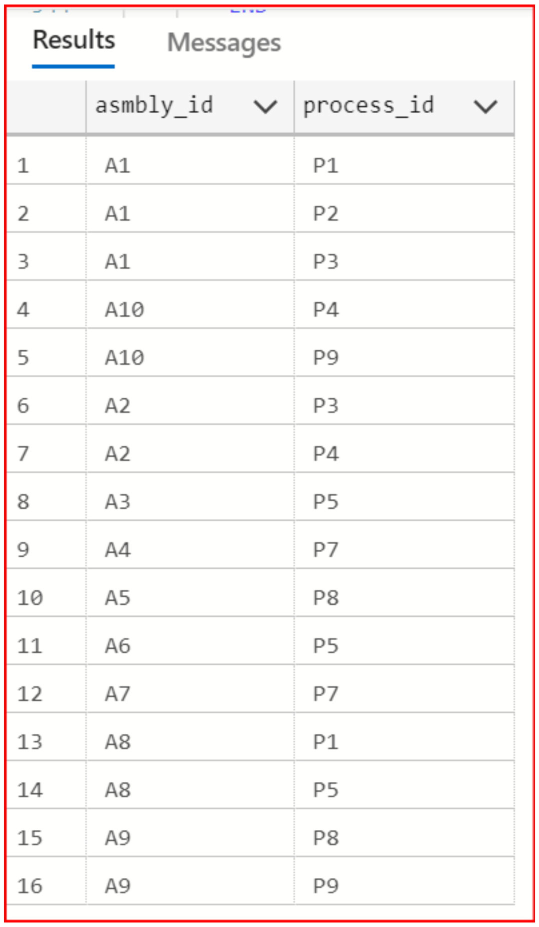
Asmbly Table



Results Messages

	asmbly_id	date_ordrd	asmbly_dtls	cust_name
1	A1	2021-12-21	ADESC1	C1
2	A10	2021-12-12	ADESC10	C5
3	A2	2021-12-20	ADESC2	C1
4	A3	2021-12-19	ADESC3	C3
5	A4	2021-12-18	ADESC4	C5
6	A5	2021-12-17	ADESC5	C5
7	A6	2021-12-16	ADESC6	C4
8	A7	2021-12-15	ADESC7	C4
9	A8	2021-12-14	ADESC8	C3
10	A9	2021-12-13	ADESC9	C4

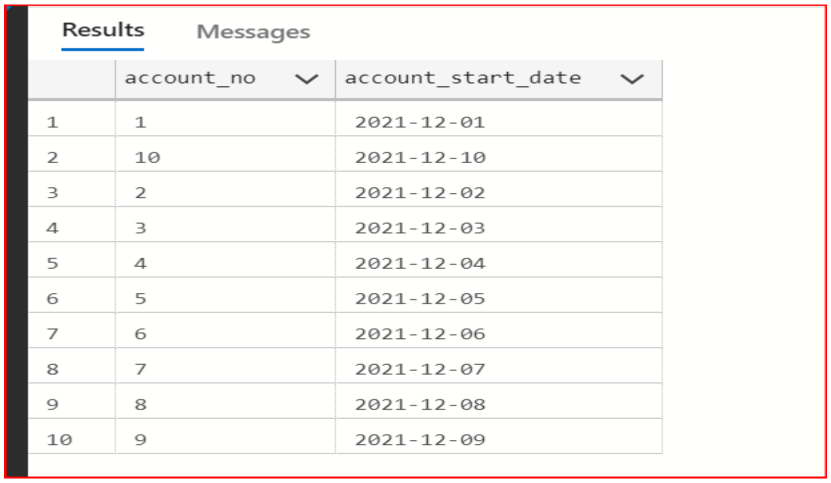
Manufactured Table


Results Messages

	asmbly_id	process_id
1	A1	P1
2	A1	P2
3	A1	P3
4	A10	P4
5	A10	P9
6	A2	P3
7	A2	P4
8	A3	P5
9	A4	P7
10	A5	P8
11	A6	P5
12	A7	P7
13	A8	P1
14	A8	P5
15	A9	P8
16	A9	P9

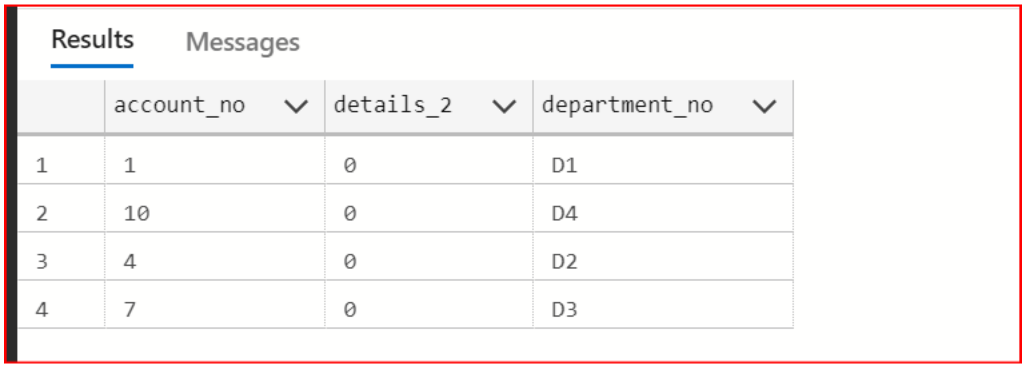
6.5 Screenshots showing the testing of query 5

Account Table


Results Messages

	account_no	account_start_date
1	1	2021-12-01
2	10	2021-12-10
3	2	2021-12-02
4	3	2021-12-03
5	4	2021-12-04
6	5	2021-12-05
7	6	2021-12-06
8	7	2021-12-07
9	8	2021-12-08
10	9	2021-12-09

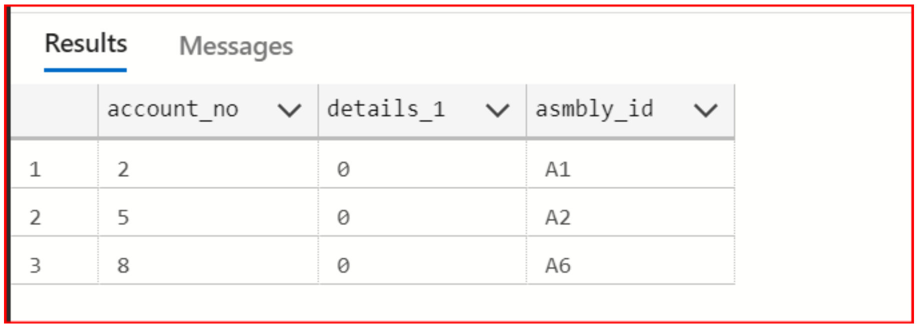
Department_Account Table



Results Messages

	account_no	details_2	department_no
1	1	0	D1
2	10	0	D4
3	4	0	D2
4	7	0	D3

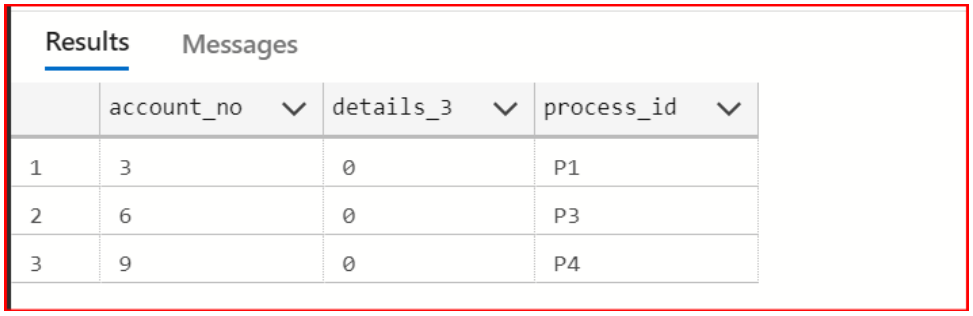
Asmbly_Account Table



Results Messages

	account_no	details_1	asmbly_id
1	2	0	A1
2	5	0	A2
3	8	0	A6

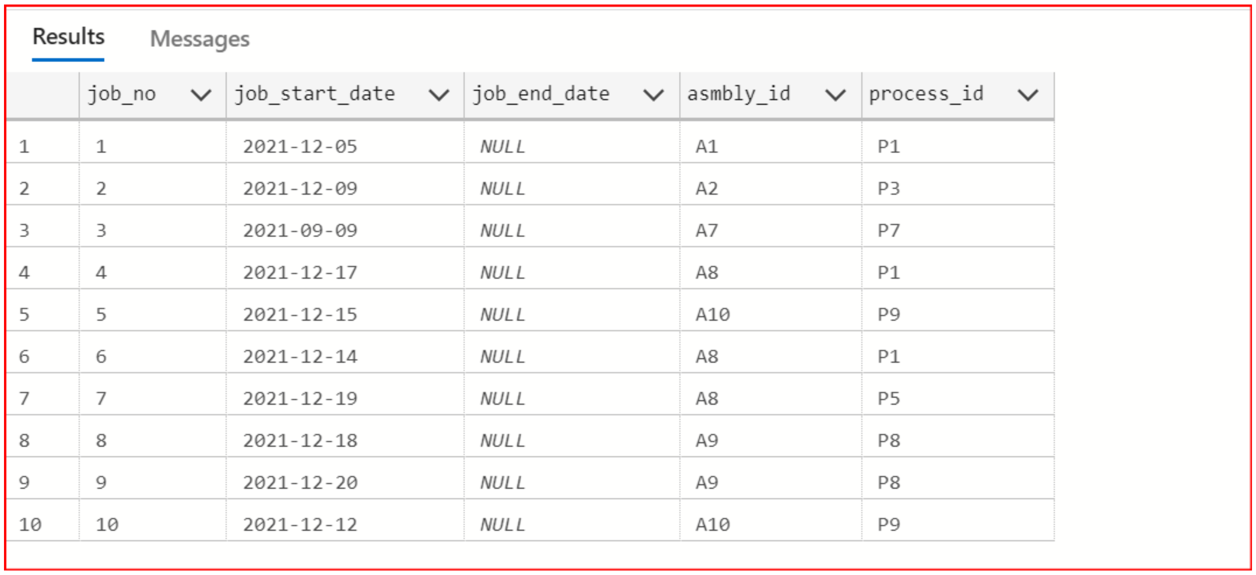
Process_Account Table



Results Messages

	account_no	details_3	process_id
1	3	0	P1
2	6	0	P3
3	9	0	P4

6.6 Screenshots showing the testing of query 6

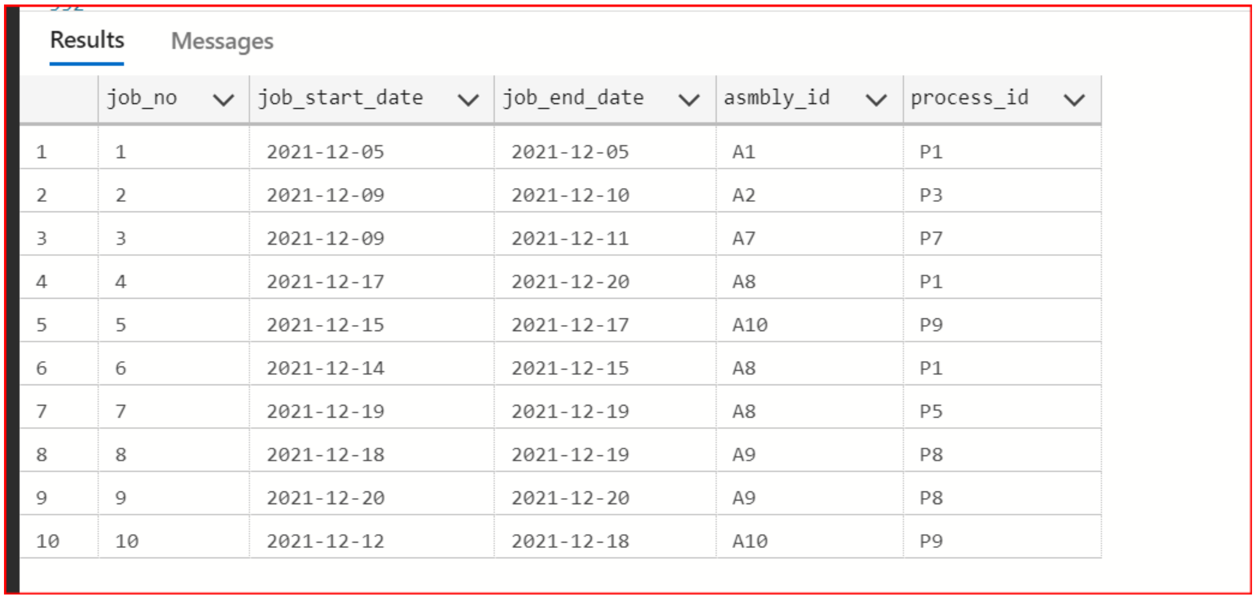


Results Messages

	job_no	job_start_date	job_end_date	asmbly_id	process_id
1	1	2021-12-05	NULL	A1	P1
2	2	2021-12-09	NULL	A2	P3
3	3	2021-09-09	NULL	A7	P7
4	4	2021-12-17	NULL	A8	P1
5	5	2021-12-15	NULL	A10	P9
6	6	2021-12-14	NULL	A8	P1
7	7	2021-12-19	NULL	A8	P5
8	8	2021-12-18	NULL	A9	P8
9	9	2021-12-20	NULL	A9	P8
10	10	2021-12-12	NULL	A10	P9

6.7 Screenshots showing the testing of query 7

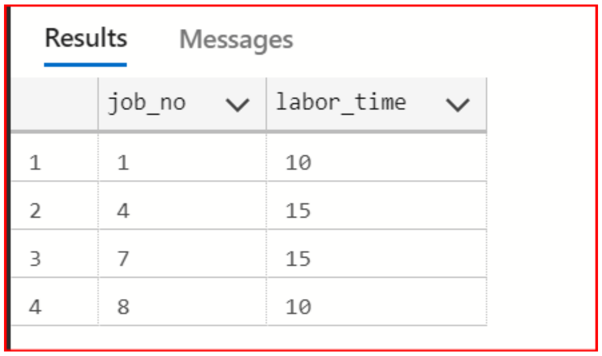
Job Table



Results Messages

	job_no	job_start_date	job_end_date	asmbly_id	process_id
1	1	2021-12-05	2021-12-05	A1	P1
2	2	2021-12-09	2021-12-10	A2	P3
3	3	2021-12-09	2021-12-11	A7	P7
4	4	2021-12-17	2021-12-20	A8	P1
5	5	2021-12-15	2021-12-17	A10	P9
6	6	2021-12-14	2021-12-15	A8	P1
7	7	2021-12-19	2021-12-19	A8	P5
8	8	2021-12-18	2021-12-19	A9	P8
9	9	2021-12-20	2021-12-20	A9	P8
10	10	2021-12-12	2021-12-18	A10	P9

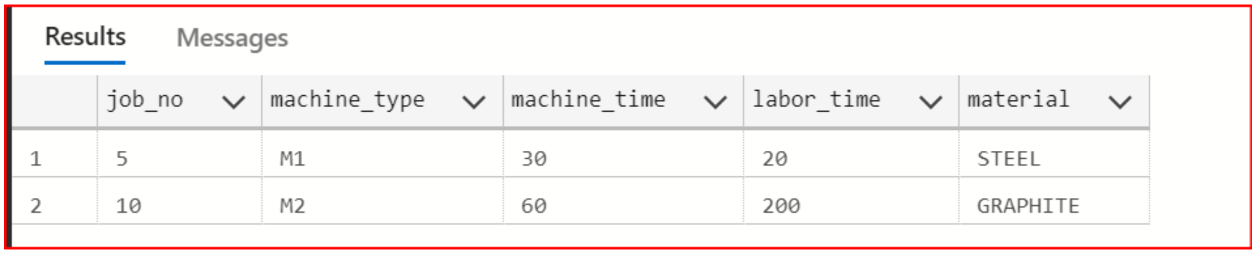
Fit_Job Table



Results Messages

	job_no	labor_time
1	1	10
2	4	15
3	7	15
4	8	10

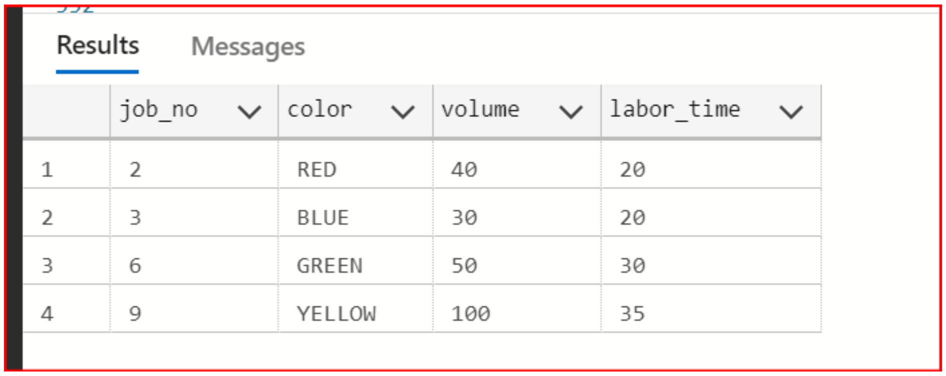
Cut_Job Table



Results Messages

	job_no	machine_type	machine_time	labor_time	material
1	5	M1	30	20	STEEL
2	10	M2	60	200	GRAPHITE

Paint_Job

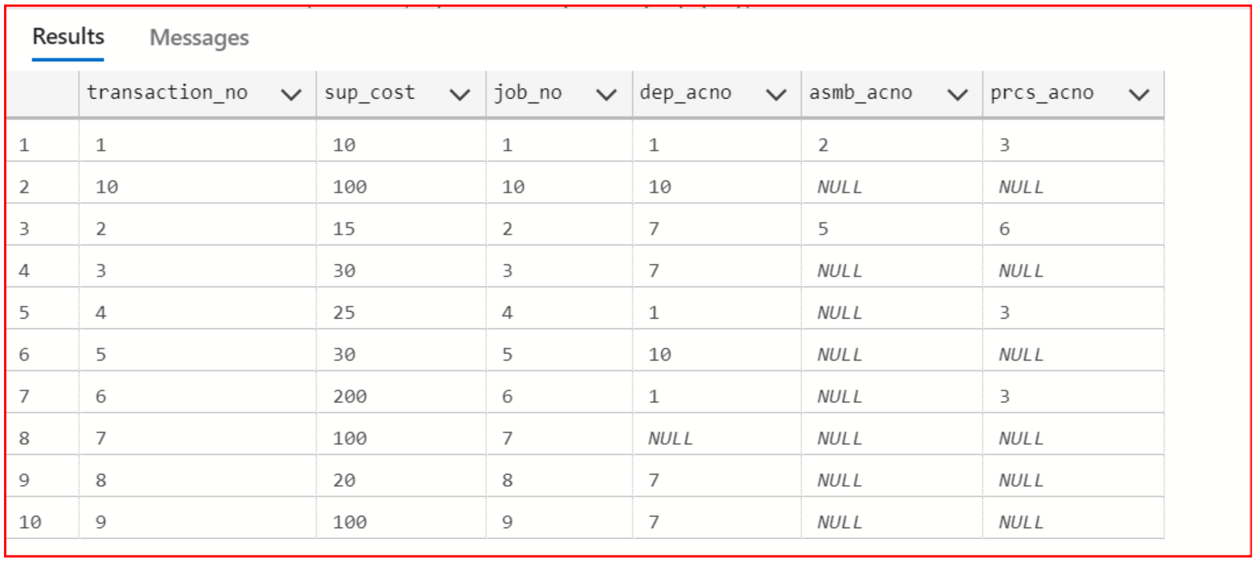


Results Messages

	job_no	color	volume	labor_time
1	2	RED	40	20
2	3	BLUE	30	20
3	6	GREEN	50	30
4	9	YELLOW	100	35

6.8 Screenshots showing the testing of query 8

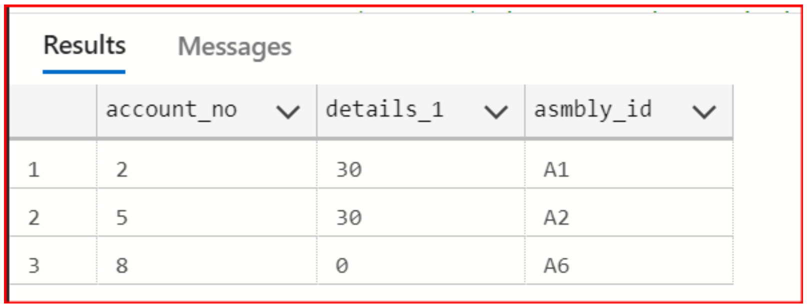
Some values for assembly account no and process account no are null because accounts are not created for those assemblies and processes in the previous steps.



Results Messages

	transaction_no	sup_cost	job_no	dep_acno	asmb_acno	prcs_acno
1	1	10	1	1	2	3
2	10	100	10	10	NULL	NULL
3	2	15	2	7	5	6
4	3	30	3	7	NULL	NULL
5	4	25	4	1	NULL	3
6	5	30	5	10	NULL	NULL
7	6	200	6	1	NULL	3
8	7	100	7	NULL	NULL	NULL
9	8	20	8	7	NULL	NULL
10	9	100	9	7	NULL	NULL

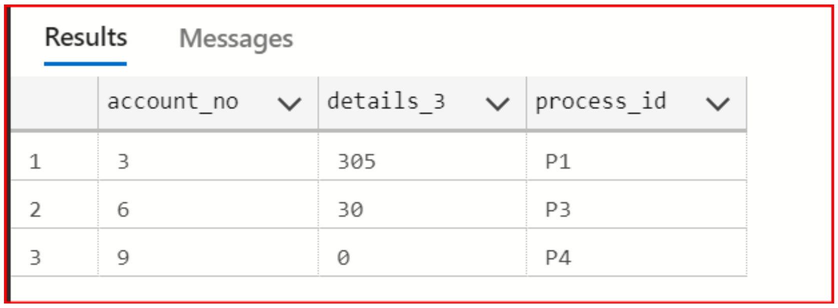
Asmbly_Account Table



Results Messages

	account_no	details_1	asmbly_id
1	2	30	A1
2	5	30	A2
3	8	0	A6

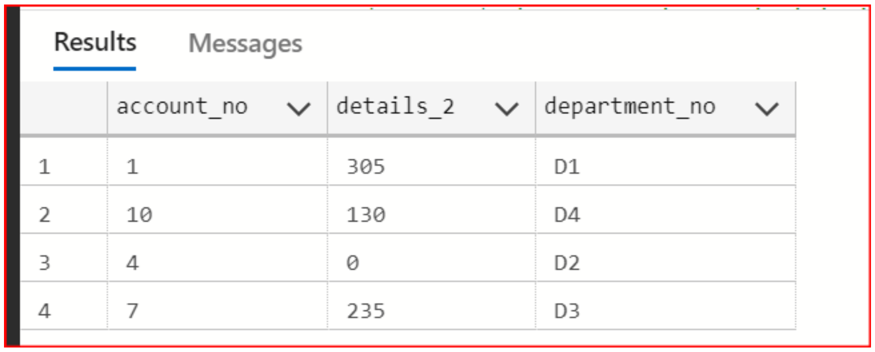
Process_Account Table



Results Messages

	account_no	details_3	process_id
1	3	305	P1
2	6	30	P3
3	9	0	P4

Department_Account table



Results Messages

	account_no	details_2	department_no
1	1	305	D1
2	10	130	D4
3	4	0	D2
4	7	235	D3

6.9 Screenshots showing the testing of query 9

```
****************** START *************************
You have the Following Options to Choose:
1. Enter a new Customer (Option 1)
2. Enter a new Department (Option 2)
3. Enter a new process-id and its department together with its type and
information relevant to
the type (Option 3)
4. Enter a new assembly with its customer-name, assembly-details, assembly-id,
and date-
ordered and associate it with one or more processes (Option 4)
5. Create a new account and associate it with the process, assembly, or
department to which it is
applicable (Option 5)
6. Enter a new Job (Option 6)
7. Enter the date job is completed and information related to type of job
(Option 7)
8. Enter a transaction-no and its sup-cost (Option 8)
9. Retrieve the cost incurred on an assembly-id (Option 9)
10. Retrieve the total labor time within a department for jobs completed
during a given date (Option 10)
11. Retrieve the processes through which a given assembly-id has passed so far
and the department responsible for the process (Option 11)
12. Retrieve all jobs completed during the given date in a given department
(Option 12)
13. Retrieve the customers whose category is in a given range (Option 13)
14. Delete all cut-jobs whose job-no is in a given range (Option 14)
15. Change the color of a given paint job (Option 15)
16. Import: enter new customers from a data file until the file is empty
(Option 16).
17. Export: Retrieve the customers whose category is in a given range and
output them to a data file(Option 17).
18. QUIT (Option 18)

 ****************** END *************************

9
Enter Assembly Id to retrieve the cost.
A1
Cost incurred on assembly-id A1: 30.000000
```

```
****************** START *************************
You have the Following Options to Choose:
1. Enter a new Customer (Option 1)
2. Enter a new Department (Option 2)
3. Enter a new process-id and its department together with its type and
information relevant to
the type (Option 3)
4. Enter a new assembly with its customer-name, assembly-details, assembly-id,
and date-
ordered and associate it with one or more processes (Option 4)
5. Create a new account and associate it with the process, assembly, or
department to which it is
applicable (Option 5)
6. Enter a new Job (Option 6)
7. Enter the date job is completed and information related to type of job
(Option 7)
8. Enter a transaction-no and its sup-cost (Option 8)
9. Retrieve the cost incurred on an assembly-id (Option 9)
10. Retrieve the total labor time within a department for jobs completed
during a given date (Option 10)
11. Retrieve the processes through which a given assembly-id has passed so far
and the department responsible for the process (Option 11)
12. Retrieve all jobs completed during the given date in a given department
(Option 12)
13. Retrieve the customers whose category is in a given range (Option 13)
14. Delete all cut-jobs whose job-no is in a given range (Option 14)
15. Change the color of a given paint job (Option 15)
16. Import: enter new customers from a data file until the file is empty
(Option 16).
17. Export: Retrieve the customers whose category is in a given range and
output them to a data file(Option 17).
18. QUIT (Option 18)

 ****************** END *************************

9
Enter Assembly Id to retrieve the cost.
A2
Cost incurred on assembly-id A2:
30.000000


       ****************** START *************************
       You have the Following Options to Choose:
       1. Enter a new Customer (Option 1)
       2. Enter a new Department (Option 2)
       3. Enter a new process-id and its department together with its type and
       information relevant to
       the type (Option 3)
       4. Enter a new assembly with its customer-name, assembly-details,
       assembly-id, and date-
```

```
ordered and associate it with one or more processes (Option 4)
5. Create a new account and associate it with the process, assembly, or
department to which it is
applicable (Option 5)
6. Enter a new Job (Option 6)
7. Enter the date job is completed and information related to type of
job (Option 7)
8. Enter a transaction-no and its sup-cost (Option 8)
9. Retrieve the cost incurred on an assembly-id (Option 9)
10. Retrieve the total labor time within a department for jobs completed
during a given date (Option 10)
11. Retrieve the processes through which a given assembly-id has passed
so far and the department responsible for the process (Option 11)
12. Retrieve all jobs completed during the given date in a given
department (Option 12)
13. Retrieve the customers whose category is in a given range (Option
13)
14. Delete all cut-jobs whose job-no is in a given range (Option 14)
15. Change the color of a given paint job (Option 15)
16. Import: enter new customers from a data file until the file is empty
(Option 16).
17. Export: Retrieve the customers whose category is in a given range
and output them to a data file(Option 17).
18. QUIT (Option 18)

 ****************** END *************************

9
Enter Assembly Id to retrieve the cost.
A6
Cost incurred on assembly-id A6:
       0.000000
```

6.10 Screenshots showing the testing of query 10

```
       ****************** START *************************
You have the Following Options to Choose:
1. Enter a new Customer (Option 1)
2. Enter a new Department (Option 2)
3. Enter a new process-id and its department together with its type and information
relevant to
the type (Option 3)
4. Enter a new assembly with its customer-name, assembly-details, assembly-id, and
date-
ordered and associate it with one or more processes (Option 4)
5. Create a new account and associate it with the process, assembly, or department to
which it is
applicable (Option 5)
6. Enter a new Job (Option 6)
7. Enter the date job is completed and information related to type of job (Option 7)
8. Enter a transaction-no and its sup-cost (Option 8)
9. Retrieve the cost incurred on an assembly-id (Option 9)
```

```
10. Retrieve the total labor time within a department for jobs completed during a
given date (Option 10)
11. Retrieve the processes through which a given assembly-id has passed so far and
the department responsible for the process (Option 11)
12. Retrieve all jobs completed during the given date in a given department (Option
12)
13. Retrieve the customers whose category is in a given range (Option 13)
14. Delete all cut-jobs whose job-no is in a given range (Option 14)
15. Change the color of a given paint job (Option 15)
16. Import: enter new customers from a data file until the file is empty (Option 16).
17. Export: Retrieve the customers whose category is in a given range and output them
to a data file(Option 17).
18. QUIT (Option 18)

 ****************** END *************************

10
Enter Department Number to get the total labor time.
D4
Enter Job completed date in YYYY-MM-DD format to get the total labor time.
2021-12-18
Total labor-time in minutes for department number D4 and date job completed 2021-12-
18:
200

---------------------------------------------------------------------------------


****************** START *************************
You have the Following Options to Choose:
1. Enter a new Customer (Option 1)
2. Enter a new Department (Option 2)
3. Enter a new process-id and its department together with its type and information
relevant to
the type (Option 3)
4. Enter a new assembly with its customer-name, assembly-details, assembly-id, and
date-
ordered and associate it with one or more processes (Option 4)
5. Create a new account and associate it with the process, assembly, or department to
which it is
applicable (Option 5)
6. Enter a new Job (Option 6)
7. Enter the date job is completed and information related to type of job (Option 7)
8. Enter a transaction-no and its sup-cost (Option 8)
9. Retrieve the cost incurred on an assembly-id (Option 9)
10. Retrieve the total labor time within a department for jobs completed during a
given date (Option 10)
11. Retrieve the processes through which a given assembly-id has passed so far and
the department responsible for the process (Option 11)
12. Retrieve all jobs completed during the given date in a given department (Option
12)
13. Retrieve the customers whose category is in a given range (Option 13)
14. Delete all cut-jobs whose job-no is in a given range (Option 14)
15. Change the color of a given paint job (Option 15)
16. Import: enter new customers from a data file until the file is empty (Option 16).
```

```
17. Export: Retrieve the customers whose category is in a given range and output them
to a data file(Option 17).
18. QUIT (Option 18)

 ****************** END *************************

10
Enter Department Number to get the total labor time.
D3
Enter Job completed date in YYYY-MM-DD format to get the total labor time.
2021-12-19
Total labor-time in minutes for department number D3 and date job completed 2021-12-
19:
10

-------------------------------------------------------------------------------------

****************** START *************************
You have the Following Options to Choose:
1. Enter a new Customer (Option 1)
2. Enter a new Department (Option 2)
3. Enter a new process-id and its department together with its type and information
relevant to
the type (Option 3)
4. Enter a new assembly with its customer-name, assembly-details, assembly-id, and
date-
ordered and associate it with one or more processes (Option 4)
5. Create a new account and associate it with the process, assembly, or department to
which it is
applicable (Option 5)
6. Enter a new Job (Option 6)
7. Enter the date job is completed and information related to type of job (Option 7)
8. Enter a transaction-no and its sup-cost (Option 8)
9. Retrieve the cost incurred on an assembly-id (Option 9)
10. Retrieve the total labor time within a department for jobs completed during a
given date (Option 10)
11. Retrieve the processes through which a given assembly-id has passed so far and
the department responsible for the process (Option 11)
12. Retrieve all jobs completed during the given date in a given department (Option
12)
13. Retrieve the customers whose category is in a given range (Option 13)
14. Delete all cut-jobs whose job-no is in a given range (Option 14)
15. Change the color of a given paint job (Option 15)
16. Import: enter new customers from a data file until the file is empty (Option 16).
17. Export: Retrieve the customers whose category is in a given range and output them
to a data file(Option 17).
18. QUIT (Option 18)

 ****************** END *************************

10
Enter Department Number to get the total labor time.
D5
Enter Job completed date in YYYY-MM-DD format to get the total labor time.
2021-12-19
```

```
Total labor-time in minutes for department number D5 and date job completed 2021-12-
19:
15

-------------------------------------------------------------------------------------
```

6.11 Screenshots showing the testing of query 11

```
****************** START *************************
You have the Following Options to Choose:
1. Enter a new Customer (Option 1)
2. Enter a new Department (Option 2)
3. Enter a new process-id and its department together with its type and information
relevant to
the type (Option 3)
4. Enter a new assembly with its customer-name, assembly-details, assembly-id, and
date-
ordered and associate it with one or more processes (Option 4)
5. Create a new account and associate it with the process, assembly, or department to
which it is
applicable (Option 5)
6. Enter a new Job (Option 6)
7. Enter the date job is completed and information related to type of job (Option 7)
8. Enter a transaction-no and its sup-cost (Option 8)
9. Retrieve the cost incurred on an assembly-id (Option 9)
10. Retrieve the total labor time within a department for jobs completed during a
given date (Option 10)
11. Retrieve the processes through which a given assembly-id has passed so far and
the department responsible for the process (Option 11)
12. Retrieve all jobs completed during the given date in a given department (Option
12)
13. Retrieve the customers whose category is in a given range (Option 13)
14. Delete all cut-jobs whose job-no is in a given range (Option 14)
15. Change the color of a given paint job (Option 15)
16. Import: enter new customers from a data file until the file is empty (Option 16).
17. Export: Retrieve the customers whose category is in a given range and output them
to a data file(Option 17).
18. QUIT (Option 18)

 ****************** END *************************

11
Enter the Assembly Id to retrieve the processes.
A1
The processes through which Assembly Id A1 passed so far:
Process ID | Department No | Job Start Date
P1 | D1 | 2021-12-05



---------------------------------------------------------------------------------

****************** START *************************
You have the Following Options to Choose:
1. Enter a new Customer (Option 1)
2. Enter a new Department (Option 2)
```

```
3. Enter a new process-id and its department together with its type and information
relevant to
the type (Option 3)
4. Enter a new assembly with its customer-name, assembly-details, assembly-id, and
date-
ordered and associate it with one or more processes (Option 4)
5. Create a new account and associate it with the process, assembly, or department to
which it is
applicable (Option 5)
6. Enter a new Job (Option 6)
7. Enter the date job is completed and information related to type of job (Option 7)
8. Enter a transaction-no and its sup-cost (Option 8)
9. Retrieve the cost incurred on an assembly-id (Option 9)
10. Retrieve the total labor time within a department for jobs completed during a
given date (Option 10)
11. Retrieve the processes through which a given assembly-id has passed so far and
the department responsible for the process (Option 11)
12. Retrieve all jobs completed during the given date in a given department (Option
12)
13. Retrieve the customers whose category is in a given range (Option 13)
14. Delete all cut-jobs whose job-no is in a given range (Option 14)
15. Change the color of a given paint job (Option 15)
16. Import: enter new customers from a data file until the file is empty (Option 16).
17. Export: Retrieve the customers whose category is in a given range and output them
to a data file(Option 17).
18. QUIT (Option 18)

 ****************** END *************************

11
Enter the Assembly Id to retrieve the processes.
A2
The processes through which Assembly Id A2 passed so far:
Process ID | Department No | Job Start Date
P3 | D3 | 2021-12-09

-----------------------------------------------------------------------------------

****************** START *************************
You have the Following Options to Choose:
1. Enter a new Customer (Option 1)
2. Enter a new Department (Option 2)
3. Enter a new process-id and its department together with its type and information
relevant to
the type (Option 3)
4. Enter a new assembly with its customer-name, assembly-details, assembly-id, and
date-
ordered and associate it with one or more processes (Option 4)
5. Create a new account and associate it with the process, assembly, or department to
which it is
applicable (Option 5)
6. Enter a new Job (Option 6)
7. Enter the date job is completed and information related to type of job (Option 7)
8. Enter a transaction-no and its sup-cost (Option 8)
9. Retrieve the cost incurred on an assembly-id (Option 9)
```

```
10. Retrieve the total labor time within a department for jobs completed during a
given date (Option 10)
11. Retrieve the processes through which a given assembly-id has passed so far and
the department responsible for the process (Option 11)
12. Retrieve all jobs completed during the given date in a given department (Option
12)
13. Retrieve the customers whose category is in a given range (Option 13)
14. Delete all cut-jobs whose job-no is in a given range (Option 14)
15. Change the color of a given paint job (Option 15)
16. Import: enter new customers from a data file until the file is empty (Option 16).
17. Export: Retrieve the customers whose category is in a given range and output them
to a data file(Option 17).
18. QUIT (Option 18)

 ****************** END *************************

11
Enter the Assembly Id to retrieve the processes.
A7
The processes through which Assembly Id A7 passed so far:
Process ID | Department No | Job Start Date
P7 | D3 | 2021-12-09

-------------------------------------------------------------------------------------
```

6.12 Screenshots showing the testing of query 12

```
****************** START *************************
You have the Following Options to Choose:
1. Enter a new Customer (Option 1)
2. Enter a new Department (Option 2)
3. Enter a new process-id and its department together with its type and information
relevant to
the type (Option 3)
4. Enter a new assembly with its customer-name, assembly-details, assembly-id, and
date-
ordered and associate it with one or more processes (Option 4)
5. Create a new account and associate it with the process, assembly, or department to
which it is
applicable (Option 5)
6. Enter a new Job (Option 6)
7. Enter the date job is completed and information related to type of job (Option 7)
8. Enter a transaction-no and its sup-cost (Option 8)
9. Retrieve the cost incurred on an assembly-id (Option 9)
10. Retrieve the total labor time within a department for jobs completed during a
given date (Option 10)
11. Retrieve the processes through which a given assembly-id has passed so far and
the department responsible for the process (Option 11)
12. Retrieve all jobs completed during the given date in a given department (Option
12)
13. Retrieve the customers whose category is in a given range (Option 13)
14. Delete all cut-jobs whose job-no is in a given range (Option 14)
15. Change the color of a given paint job (Option 15)
16. Import: enter new customers from a data file until the file is empty (Option 16).
```

```
17. Export: Retrieve the customers whose category is in a given range and output them
to a data file(Option 17).
18. QUIT (Option 18)

 ****************** END *************************

12
Enter the date job is completed in YYYY-MM-DD format
2021-12-19
Enter the department to retrieve the jobs.
D5
The fit jobs completed on date 2021-12-19 in department D5:
7 | A8 | 15
8 | A9 | 10
The Paint jobs completed on date 2021-12-19 in department D5:
The Cut jobs completed on date 2021-12-19 in department D5:

-------------------------------------------------------------------------------------

****************** START *************************
You have the Following Options to Choose:
1. Enter a new Customer (Option 1)
2. Enter a new Department (Option 2)
3. Enter a new process-id and its department together with its type and information
relevant to
the type (Option 3)
4. Enter a new assembly with its customer-name, assembly-details, assembly-id, and
date-
ordered and associate it with one or more processes (Option 4)
5. Create a new account and associate it with the process, assembly, or department to
which it is
applicable (Option 5)
6. Enter a new Job (Option 6)
7. Enter the date job is completed and information related to type of job (Option 7)
8. Enter a transaction-no and its sup-cost (Option 8)
9. Retrieve the cost incurred on an assembly-id (Option 9)
10. Retrieve the total labor time within a department for jobs completed during a
given date (Option 10)
11. Retrieve the processes through which a given assembly-id has passed so far and
the department responsible for the process (Option 11)
12. Retrieve all jobs completed during the given date in a given department (Option
12)
13. Retrieve the customers whose category is in a given range (Option 13)
14. Delete all cut-jobs whose job-no is in a given range (Option 14)
15. Change the color of a given paint job (Option 15)
16. Import: enter new customers from a data file until the file is empty (Option 16).
17. Export: Retrieve the customers whose category is in a given range and output them
to a data file(Option 17).
18. QUIT (Option 18)

 ****************** END *************************

12
Enter the date job is completed in YYYY-MM-DD format
2021-12-18
```

```
Enter the department to retrieve the jobs.
D4
The fit jobs completed on date 2021-12-18 in department D4:
The Paint jobs completed on date 2021-12-18 in department D4:
The Cut jobs completed on date 2021-12-18 in department D4:
10 | A10 | M2 | 60 | GRAPHITE | 200

-------------------------------------------------------------------------------------



****************** START *************************
You have the Following Options to Choose:
1. Enter a new Customer (Option 1)
2. Enter a new Department (Option 2)
3. Enter a new process-id and its department together with its type and information
relevant to
the type (Option 3)
4. Enter a new assembly with its customer-name, assembly-details, assembly-id, and
date-
ordered and associate it with one or more processes (Option 4)
5. Create a new account and associate it with the process, assembly, or department to
which it is
applicable (Option 5)
6. Enter a new Job (Option 6)
7. Enter the date job is completed and information related to type of job (Option 7)
8. Enter a transaction-no and its sup-cost (Option 8)
9. Retrieve the cost incurred on an assembly-id (Option 9)
10. Retrieve the total labor time within a department for jobs completed during a
given date (Option 10)
11. Retrieve the processes through which a given assembly-id has passed so far and
the department responsible for the process (Option 11)
12. Retrieve all jobs completed during the given date in a given department (Option
12)
13. Retrieve the customers whose category is in a given range (Option 13)
14. Delete all cut-jobs whose job-no is in a given range (Option 14)
15. Change the color of a given paint job (Option 15)
16. Import: enter new customers from a data file until the file is empty (Option 16).
17. Export: Retrieve the customers whose category is in a given range and output them
to a data file(Option 17).
18. QUIT (Option 18)

 ****************** END *************************

12
Enter the date job is completed in YYYY-MM-DD format
2021-12-19
Enter the department to retrieve the jobs.
D3
The fit jobs completed on date 2021-12-19 in department D3:
7 | A8 | 15
8 | A9 | 10
The Paint jobs completed on date 2021-12-19 in department D3:
The Cut jobs completed on date 2021-12-19 in department D3:
```

6.13 Screenshots showing the testing of query 13

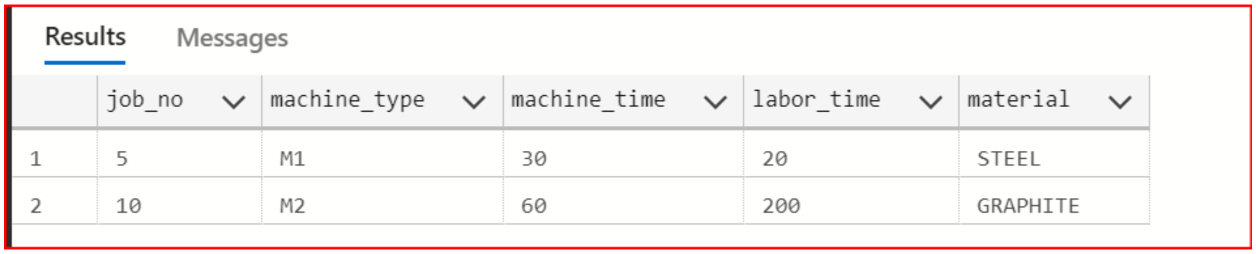
```
      ****************** START *************************
You have the Following Options to Choose:
1. Enter a new Customer (Option 1)
2. Enter a new Department (Option 2)
3. Enter a new process-id and its department together with its type and information
relevant to
the type (Option 3)
4. Enter a new assembly with its customer-name, assembly-details, assembly-id, and
date-
ordered and associate it with one or more processes (Option 4)
5. Create a new account and associate it with the process, assembly, or department to
which it is
applicable (Option 5)
6. Enter a new Job (Option 6)
7. Enter the date job is completed and information related to type of job (Option 7)
8. Enter a transaction-no and its sup-cost (Option 8)
9. Retrieve the cost incurred on an assembly-id (Option 9)
10. Retrieve the total labor time within a department for jobs completed during a
given date (Option 10)
11. Retrieve the processes through which a given assembly-id has passed so far and
the department responsible for the process (Option 11)
12. Retrieve all jobs completed during the given date in a given department (Option
12)
13. Retrieve the customers whose category is in a given range (Option 13)
14. Delete all cut-jobs whose job-no is in a given range (Option 14)
15. Change the color of a given paint job (Option 15)
16. Import: enter new customers from a data file until the file is empty (Option 16).
17. Export: Retrieve the customers whose category is in a given range and output them
to a data file(Option 17).
18. QUIT (Option 18)

 ****************** END *************************

13
Enter the lower bound of the category.
1
Enter the upper bound of the category.
4
The customers in the given range of category are
C1 | A1
C2 | A2
C3 | A3
C4 | A4


-------------------------------------------------------------

****************** START *************************
You have the Following Options to Choose:
1. Enter a new Customer (Option 1)
2. Enter a new Department (Option 2)
```

```
3. Enter a new process-id and its department together with its type and information
relevant to
the type (Option 3)
4. Enter a new assembly with its customer-name, assembly-details, assembly-id, and
date-
ordered and associate it with one or more processes (Option 4)
5. Create a new account and associate it with the process, assembly, or department to
which it is
applicable (Option 5)
6. Enter a new Job (Option 6)
7. Enter the date job is completed and information related to type of job (Option 7)
8. Enter a transaction-no and its sup-cost (Option 8)
9. Retrieve the cost incurred on an assembly-id (Option 9)
10. Retrieve the total labor time within a department for jobs completed during a
given date (Option 10)
11. Retrieve the processes through which a given assembly-id has passed so far and
the department responsible for the process (Option 11)
12. Retrieve all jobs completed during the given date in a given department (Option
12)
13. Retrieve the customers whose category is in a given range (Option 13)
14. Delete all cut-jobs whose job-no is in a given range (Option 14)
15. Change the color of a given paint job (Option 15)
16. Import: enter new customers from a data file until the file is empty (Option 16).
17. Export: Retrieve the customers whose category is in a given range and output them
to a data file(Option 17).
18. QUIT (Option 18)

 ****************** END *************************

13
Enter the lower bound of the category.
4
Enter the upper bound of the category.
7
The customers in the given range of category are
C4 | A4
C5 | A5

----------------------------------------------------------------

****************** START *************************
You have the Following Options to Choose:
1. Enter a new Customer (Option 1)
2. Enter a new Department (Option 2)
3. Enter a new process-id and its department together with its type and information
relevant to
the type (Option 3)
4. Enter a new assembly with its customer-name, assembly-details, assembly-id, and
date-
ordered and associate it with one or more processes (Option 4)
5. Create a new account and associate it with the process, assembly, or department to
which it is
applicable (Option 5)
6. Enter a new Job (Option 6)
7. Enter the date job is completed and information related to type of job (Option 7)
```

```
8. Enter a transaction-no and its sup-cost (Option 8)
9. Retrieve the cost incurred on an assembly-id (Option 9)
10. Retrieve the total labor time within a department for jobs completed during a
given date (Option 10)
11. Retrieve the processes through which a given assembly-id has passed so far and
the department responsible for the process (Option 11)
12. Retrieve all jobs completed during the given date in a given department (Option
12)
13. Retrieve the customers whose category is in a given range (Option 13)
14. Delete all cut-jobs whose job-no is in a given range (Option 14)
15. Change the color of a given paint job (Option 15)
16. Import: enter new customers from a data file until the file is empty (Option 16).
17. Export: Retrieve the customers whose category is in a given range and output them
to a data file(Option 17).
18. QUIT (Option 18)

 ****************** END *************************

13
Enter the lower bound of the category.
2
Enter the upper bound of the category.
5
The customers in the given range of category are
C2 | A2
C3 | A3
C4 | A4
C5 | A5

---------------------------------------------------------------------------------
```

6.14 Screenshots showing the testing of query 14

Before executing query 14



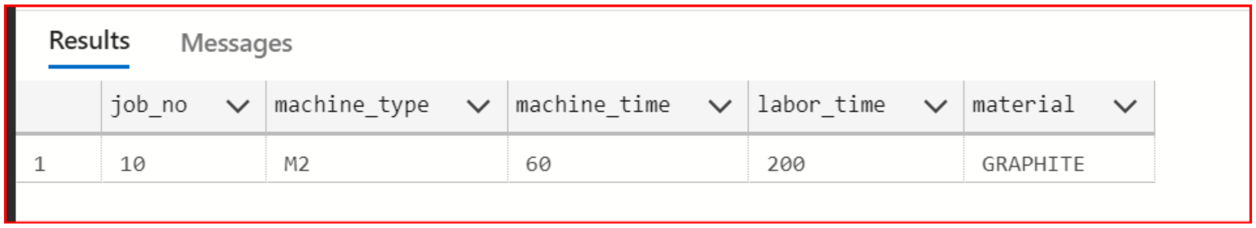
Results Messages

	job_no	machine_type	machine_time	labor_time	material
1	5	M1	30	20	STEEL
2	10	M2	60	200	GRAPHITE

```
      ****************** START *************************
You have the Following Options to Choose:
1. Enter a new Customer (Option 1)
2. Enter a new Department (Option 2)
3. Enter a new process-id and its department together with its type and information
relevant to
the type (Option 3)
```

```
4. Enter a new assembly with its customer-name, assembly-details, assembly-id, and
date-
ordered and associate it with one or more processes (Option 4)
5. Create a new account and associate it with the process, assembly, or department to
which it is
applicable (Option 5)
6. Enter a new Job (Option 6)
7. Enter the date job is completed and information related to type of job (Option 7)
8. Enter a transaction-no and its sup-cost (Option 8)
9. Retrieve the cost incurred on an assembly-id (Option 9)
10. Retrieve the total labor time within a department for jobs completed during a
given date (Option 10)
11. Retrieve the processes through which a given assembly-id has passed so far and
the department responsible for the process (Option 11)
12. Retrieve all jobs completed during the given date in a given department (Option
12)
13. Retrieve the customers whose category is in a given range (Option 13)
14. Delete all cut-jobs whose job-no is in a given range (Option 14)
15. Change the color of a given paint job (Option 15)
16. Import: enter new customers from a data file until the file is empty (Option 16).
17. Export: Retrieve the customers whose category is in a given range and output them
to a data file(Option 17).
18. QUIT (Option 18)

 ****************** END *************************

14
Enter the lower bound of the cut job-no.
5
Enter the upper bound of the cut job-no.
7
```

After Query Execution

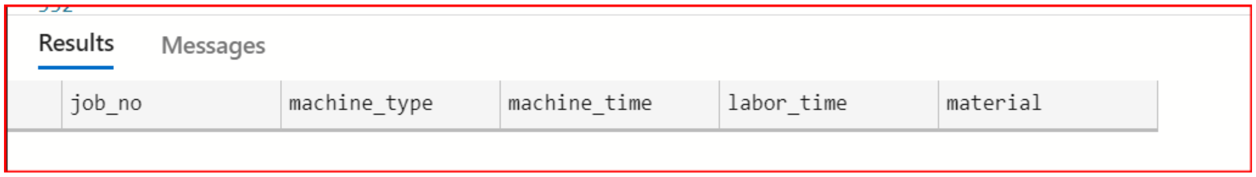


	job_no	machine_type	machine_time	labor_time	material
1	10	M2	60	200	GRAPHITE

```
****************** START *************************
You have the Following Options to Choose:
1. Enter a new Customer (Option 1)
2. Enter a new Department (Option 2)
3. Enter a new process-id and its department together with its type and information
relevant to
the type (Option 3)
4. Enter a new assembly with its customer-name, assembly-details, assembly-id, and
date-
```

```
ordered and associate it with one or more processes (Option 4)
5. Create a new account and associate it with the process, assembly, or department to
which it is
applicable (Option 5)
6. Enter a new Job (Option 6)
7. Enter the date job is completed and information related to type of job (Option 7)
8. Enter a transaction-no and its sup-cost (Option 8)
9. Retrieve the cost incurred on an assembly-id (Option 9)
10. Retrieve the total labor time within a department for jobs completed during a
given date (Option 10)
11. Retrieve the processes through which a given assembly-id has passed so far and
the department responsible for the process (Option 11)
12. Retrieve all jobs completed during the given date in a given department (Option
12)
13. Retrieve the customers whose category is in a given range (Option 13)
14. Delete all cut-jobs whose job-no is in a given range (Option 14)
15. Change the color of a given paint job (Option 15)
16. Import: enter new customers from a data file until the file is empty (Option 16).
17. Export: Retrieve the customers whose category is in a given range and output them
to a data file(Option 17).
18. QUIT (Option 18)

 ****************** END *************************

14
Enter the lower bound of the cut job-no.
1
Enter the upper bound of the cut job-no.
4
Deleted all cut jobs in the given range of job-no.

-------------------------------------------------------------------------------------

****************** START *************************
You have the Following Options to Choose:
1. Enter a new Customer (Option 1)
2. Enter a new Department (Option 2)
3. Enter a new process-id and its department together with its type and information
relevant to
the type (Option 3)
4. Enter a new assembly with its customer-name, assembly-details, assembly-id, and
date-
ordered and associate it with one or more processes (Option 4)
5. Create a new account and associate it with the process, assembly, or department to
which it is
applicable (Option 5)
6. Enter a new Job (Option 6)
7. Enter the date job is completed and information related to type of job (Option 7)
8. Enter a transaction-no and its sup-cost (Option 8)
9. Retrieve the cost incurred on an assembly-id (Option 9)
10. Retrieve the total labor time within a department for jobs completed during a
given date (Option 10)
11. Retrieve the processes through which a given assembly-id has passed so far and
the department responsible for the process (Option 11)
```

```
12. Retrieve all jobs completed during the given date in a given department (Option
12)
13. Retrieve the customers whose category is in a given range (Option 13)
14. Delete all cut-jobs whose job-no is in a given range (Option 14)
15. Change the color of a given paint job (Option 15)
16. Import: enter new customers from a data file until the file is empty (Option 16).
17. Export: Retrieve the customers whose category is in a given range and output them
to a data file(Option 17).
18. QUIT (Option 18)

 ****************** END *************************

14
Enter the lower bound of the cut job-no.
9
Enter the upper bound of the cut job-no.
10
Deleted all cut jobs in the given range of job-no.
```

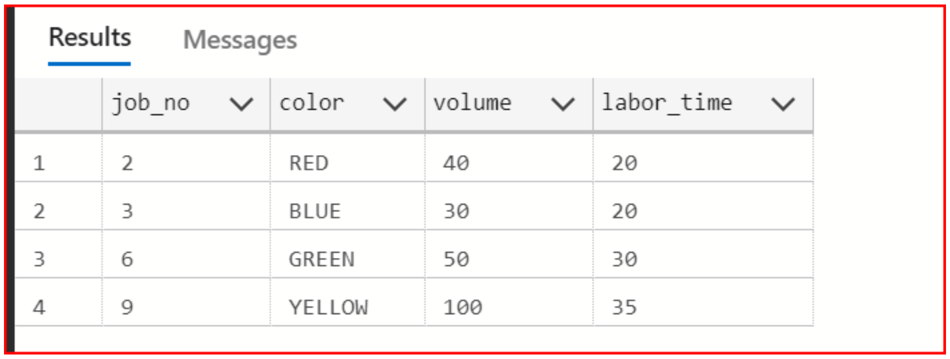


Results Messages

	job_no	machine_type	machine_time	labor_time	material

6.15 Screenshots showing the testing of query 15

```
Before Query execution
```

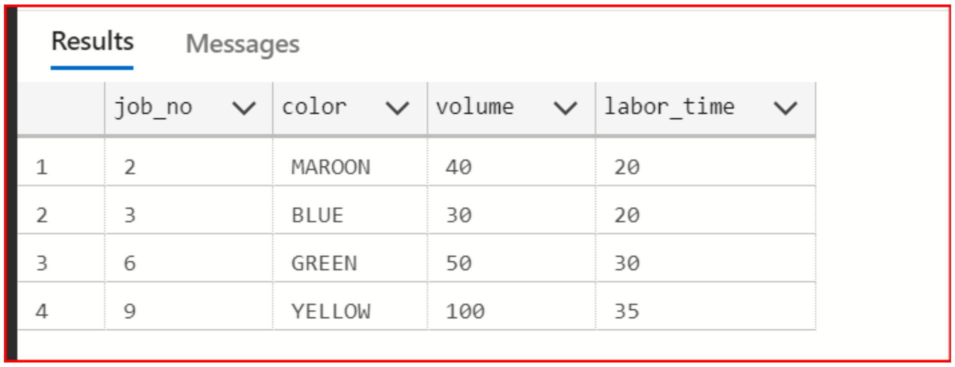


Results Messages

	job_no	color	volume	labor_time
1	2	RED	40	20
2	3	BLUE	30	20
3	6	GREEN	50	30
4	9	YELLOW	100	35

```
****************** START *************************
You have the Following Options to Choose:
1. Enter a new Customer (Option 1)
2. Enter a new Department (Option 2)
3. Enter a new process-id and its department together with its type and information
relevant to
the type (Option 3)
```

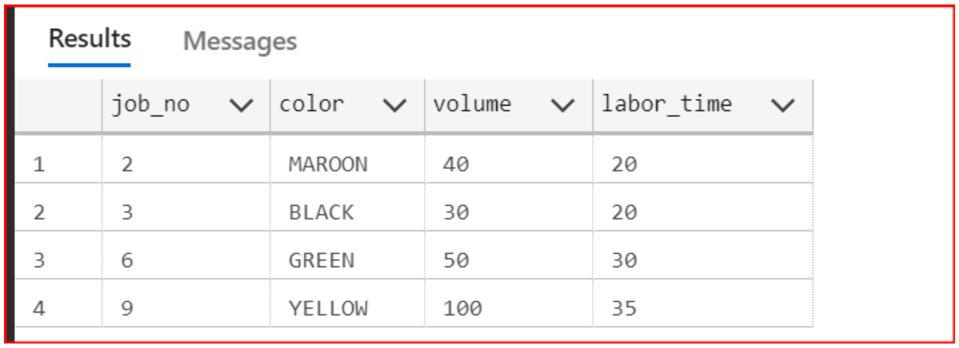
```
4. Enter a new assembly with its customer-name, assembly-details, assembly-id, and
date-
ordered and associate it with one or more processes (Option 4)
5. Create a new account and associate it with the process, assembly, or department to
which it is
applicable (Option 5)
6. Enter a new Job (Option 6)
7. Enter the date job is completed and information related to type of job (Option 7)
8. Enter a transaction-no and its sup-cost (Option 8)
9. Retrieve the cost incurred on an assembly-id (Option 9)
10. Retrieve the total labor time within a department for jobs completed during a
given date (Option 10)
11. Retrieve the processes through which a given assembly-id has passed so far and
the department responsible for the process (Option 11)
12. Retrieve all jobs completed during the given date in a given department (Option
12)
13. Retrieve the customers whose category is in a given range (Option 13)
14. Delete all cut-jobs whose job-no is in a given range (Option 14)
15. Change the color of a given paint job (Option 15)
16. Import: enter new customers from a data file until the file is empty (Option 16).
17. Export: Retrieve the customers whose category is in a given range and output them
to a data file(Option 17).
18. QUIT (Option 18)

 ****************** END *************************

15
Enter the paint job-no.
2
Enter the new color for the job-no.
MAROON
Changed the color of a given paint job.
```


Results Messages

	job_no	color	volume	labor_time
1	2	MAROON	40	20
2	3	BLUE	30	20
3	6	GREEN	50	30
4	9	YELLOW	100	35

```
****************** START *************************
You have the Following Options to Choose:
1. Enter a new Customer (Option 1)
2. Enter a new Department (Option 2)
```

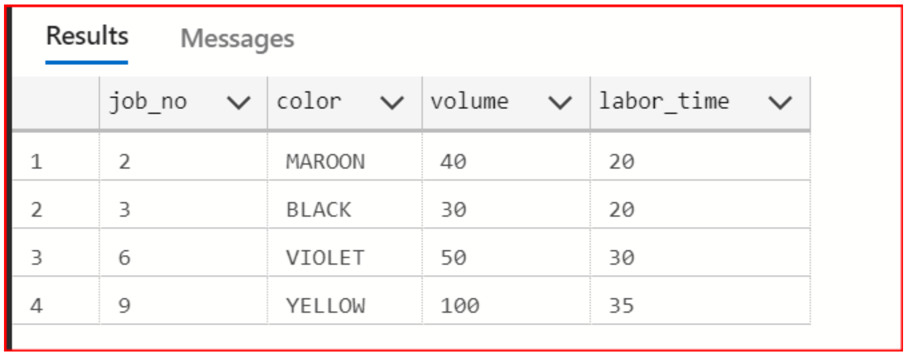
```
3. Enter a new process-id and its department together with its type and information
relevant to
the type (Option 3)
4. Enter a new assembly with its customer-name, assembly-details, assembly-id, and
date-
ordered and associate it with one or more processes (Option 4)
5. Create a new account and associate it with the process, assembly, or department to
which it is
applicable (Option 5)
6. Enter a new Job (Option 6)
7. Enter the date job is completed and information related to type of job (Option 7)
8. Enter a transaction-no and its sup-cost (Option 8)
9. Retrieve the cost incurred on an assembly-id (Option 9)
10. Retrieve the total labor time within a department for jobs completed during a
given date (Option 10)
11. Retrieve the processes through which a given assembly-id has passed so far and
the department responsible for the process (Option 11)
12. Retrieve all jobs completed during the given date in a given department (Option
12)
13. Retrieve the customers whose category is in a given range (Option 13)
14. Delete all cut-jobs whose job-no is in a given range (Option 14)
15. Change the color of a given paint job (Option 15)
16. Import: enter new customers from a data file until the file is empty (Option 16).
17. Export: Retrieve the customers whose category is in a given range and output them
to a data file(Option 17).
18. QUIT (Option 18)

 ****************** END *************************

15
Enter the paint job-no.
3
Enter the new color for the job-no.
BLACK
Changed the color of a given paint job.
```



	job_no	color	volume	labor_time
1	2	MAROON	40	20
2	3	BLACK	30	20
3	6	GREEN	50	30
4	9	YELLOW	100	35

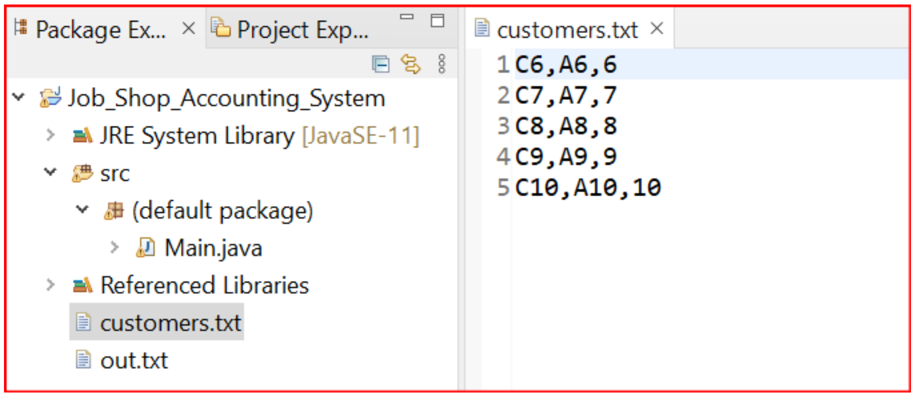
```
---------------------------------------------------------------------------

****************** START *************************
You have the Following Options to Choose:
1. Enter a new Customer (Option 1)
2. Enter a new Department (Option 2)
3. Enter a new process-id and its department together with its type and information
relevant to
```

```
the type (Option 3)
4. Enter a new assembly with its customer-name, assembly-details, assembly-id, and
date-
ordered and associate it with one or more processes (Option 4)
5. Create a new account and associate it with the process, assembly, or department to
which it is
applicable (Option 5)
6. Enter a new Job (Option 6)
7. Enter the date job is completed and information related to type of job (Option 7)
8. Enter a transaction-no and its sup-cost (Option 8)
9. Retrieve the cost incurred on an assembly-id (Option 9)
10. Retrieve the total labor time within a department for jobs completed during a
given date (Option 10)
11. Retrieve the processes through which a given assembly-id has passed so far and
the department responsible for the process (Option 11)
12. Retrieve all jobs completed during the given date in a given department (Option
12)
13. Retrieve the customers whose category is in a given range (Option 13)
14. Delete all cut-jobs whose job-no is in a given range (Option 14)
15. Change the color of a given paint job (Option 15)
16. Import: enter new customers from a data file until the file is empty (Option 16).
17. Export: Retrieve the customers whose category is in a given range and output them
to a data file(Option 17).
18. QUIT (Option 18)

 ****************** END *************************

15
Enter the paint job-no.
6
Enter the new color for the job-no.
VIOLET
Changed the color of a given paint job.
```



	job_no	color	volume	labor_time
1	2	MAROON	40	20
2	3	BLACK	30	20
3	6	VIOLET	50	30
4	9	YELLOW	100	35

6.16 Screenshots showing the testing of query 16

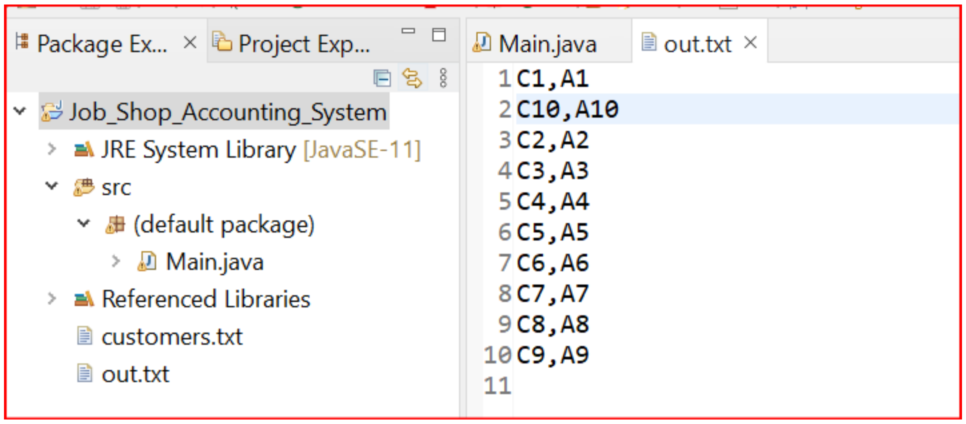


```
C6,A6,6
C7,A7,7
C8,A8,8
C9,A9,9
C10,A10,10
```

```
****************** START *************************
You have the Following Options to Choose:
1. Enter a new Customer (Option 1)
2. Enter a new Department (Option 2)
3. Enter a new process-id and its department together with its type and information
relevant to
the type (Option 3)
4. Enter a new assembly with its customer-name, assembly-details, assembly-id, and
date-
ordered and associate it with one or more processes (Option 4)
5. Create a new account and associate it with the process, assembly, or department to
which it is
applicable (Option 5)
6. Enter a new Job (Option 6)
7. Enter the date job is completed and information related to type of job (Option 7)
8. Enter a transaction-no and its sup-cost (Option 8)
9. Retrieve the cost incurred on an assembly-id (Option 9)
10. Retrieve the total labor time within a department for jobs completed during a
given date (Option 10)
11. Retrieve the processes through which a given assembly-id has passed so far and
the department responsible for the process (Option 11)
12. Retrieve all jobs completed during the given date in a given department (Option
12)
13. Retrieve the customers whose category is in a given range (Option 13)
14. Delete all cut-jobs whose job-no is in a given range (Option 14)
15. Change the color of a given paint job (Option 15)
16. Import: enter new customers from a data file until the file is empty (Option 16).
17. Export: Retrieve the customers whose category is in a given range and output them
to a data file(Option 17).
18. QUIT (Option 18)

 ****************** END *************************
16
Enter the file-name to Import data.
customers.txt
6
7
8
9
10
```

6.17 Screenshots showing the testing of query 17

```
****************** START *************************
You have the Following Options to Choose:
1. Enter a new Customer (Option 1)
2. Enter a new Department (Option 2)
3. Enter a new process-id and its department together with its type and information
relevant to
the type (Option 3)
4. Enter a new assembly with its customer-name, assembly-details, assembly-id, and
date-
ordered and associate it with one or more processes (Option 4)
5. Create a new account and associate it with the process, assembly, or department to
which it is
applicable (Option 5)
6. Enter a new Job (Option 6)
7. Enter the date job is completed and information related to type of job (Option 7)
8. Enter a transaction-no and its sup-cost (Option 8)
9. Retrieve the cost incurred on an assembly-id (Option 9)
10. Retrieve the total labor time within a department for jobs completed during a
given date (Option 10)
11. Retrieve the processes through which a given assembly-id has passed so far and
the department responsible for the process (Option 11)
12. Retrieve all jobs completed during the given date in a given department (Option
12)
13. Retrieve the customers whose category is in a given range (Option 13)
14. Delete all cut-jobs whose job-no is in a given range (Option 14)
15. Change the color of a given paint job (Option 15)
16. Import: enter new customers from a data file until the file is empty (Option 16).
17. Export: Retrieve the customers whose category is in a given range and output them
to a data file(Option 17).
18. QUIT (Option 18)

 ****************** END *************************

17
Enter the file-name to Export data.
OUT.TXT
Enter the lower bound of category.
1
Enter the upper bound of category.
10
```



```
C1,A1
C10,A10
C2,A2
C3,A3
C4,A4
C5,A5
C6,A6
C7,A7
C8,A8
C9,A9
```

6.18 Screenshots showing the testing of query 18

```
       ****************** START *************************
You have the Following Options to Choose:
1. Enter a new Customer (Option 1)
2. Enter a new Department (Option 2)
3. Enter a new process-id and its department together with its type and information
relevant to
the type (Option 3)
4. Enter a new assembly with its customer-name, assembly-details, assembly-id, and
date-
ordered and associate it with one or more processes (Option 4)
5. Create a new account and associate it with the process, assembly, or department to
which it is
applicable (Option 5)
6. Enter a new Job (Option 6)
7. Enter the date job is completed and information related to type of job (Option 7)
8. Enter a transaction-no and its sup-cost (Option 8)
9. Retrieve the cost incurred on an assembly-id (Option 9)
10. Retrieve the total labor time within a department for jobs completed during a
given date (Option 10)
11. Retrieve the processes through which a given assembly-id has passed so far and
the department responsible for the process (Option 11)
12. Retrieve all jobs completed during the given date in a given department (Option
12)
13. Retrieve the customers whose category is in a given range (Option 13)
14. Delete all cut-jobs whose job-no is in a given range (Option 14)
15. Change the color of a given paint job (Option 15)
16. Import: enter new customers from a data file until the file is empty (Option 16).
17. Export: Retrieve the customers whose category is in a given range and output them
to a data file(Option 17).
18. QUIT (Option 18)

 ****************** END *************************

18
You chose to Quit! Bye!
```

- To demonstrate that Azure SQL Database can detect errors, you also need to perform 3 queries of different types that contain some errors.

```
****************** START *************************
You have the Following Options to Choose:
1. Enter a new Customer (Option 1)
2. Enter a new Department (Option 2)
3. Enter a new process-id and its department together with its type and
information relevant to
the type (Option 3)
4. Enter a new assembly with its customer-name, assembly-details,
assembly-id, and date-
ordered and associate it with one or more processes (Option 4)
5. Create a new account and associate it with the process, assembly, or
department to which it is
applicable (Option 5)
6. Enter a new Job (Option 6)
7. Enter the date job is completed and information related to type of
job (Option 7)
8. Enter a transaction-no and its sup-cost (Option 8)
9. Retrieve the cost incurred on an assembly-id (Option 9)
10. Retrieve the total labor time within a department for jobs completed
during a given date (Option 10)
11. Retrieve the processes through which a given assembly-id has passed
so far and the department responsible for the process (Option 11)
12. Retrieve all jobs completed during the given date in a given
department (Option 12)
13. Retrieve the customers whose category is in a given range (Option
13)
14. Delete all cut-jobs whose job-no is in a given range (Option 14)
15. Change the color of a given paint job (Option 15)
16. Import: enter new customers from a data file until the file is empty
(Option 16).
17. Export: Retrieve the customers whose category is in a given range
and output them to a data file(Option 17).
18. QUIT (Option 18)

 ****************** END *************************

1
Enter the Customer Name
C1
Enter the Customer Address
A11
Enter the Customer Category
7
             Error! Returning to main menu
```

➔ **Primary key violation in Customer Table is handled entirely by Azure SQL database and error is printed by java program.**

```
****************** START *************************
You have the Following Options to Choose:
1. Enter a new Customer (Option 1)
2. Enter a new Department (Option 2)
3. Enter a new process-id and its department together with its type and information
relevant to
the type (Option 3)
4. Enter a new assembly with its customer-name, assembly-details, assembly-id, and
date-
ordered and associate it with one or more processes (Option 4)
5. Create a new account and associate it with the process, assembly, or department to
which it is
applicable (Option 5)
6. Enter a new Job (Option 6)
7. Enter the date job is completed and information related to type of job (Option 7)
8. Enter a transaction-no and its sup-cost (Option 8)
9. Retrieve the cost incurred on an assembly-id (Option 9)
10. Retrieve the total labor time within a department for jobs completed during a
given date (Option 10)
11. Retrieve the processes through which a given assembly-id has passed so far and
the department responsible for the process (Option 11)
12. Retrieve all jobs completed during the given date in a given department (Option
12)
13. Retrieve the customers whose category is in a given range (Option 13)
14. Delete all cut-jobs whose job-no is in a given range (Option 14)
15. Change the color of a given paint job (Option 15)
16. Import: enter new customers from a data file until the file is empty (Option 16).
17. Export: Retrieve the customers whose category is in a given range and output them
to a data file(Option 17).
18. QUIT (Option 18)

 ****************** END *************************

4
Enter Assembly ID
A11
Enter Date Ordered in YYYY-MM-DD format


NULL
Enter Assembly Details
ADESC11
Enter the Customer Name
C1
Enter number of processes associated with this Assembly:

1
      Error! Returning to main menu
```

➔ **Tried to insert null value in date_ordrd column in Asmbly table shows error is thrown and handled by azure sql database**

```
****************** START *************************
You have the Following Options to Choose:
1. Enter a new Customer (Option 1)
2. Enter a new Department (Option 2)
3. Enter a new process-id and its department together with its type and information
relevant to
the type (Option 3)
4. Enter a new assembly with its customer-name, assembly-details, assembly-id, and
date-
ordered and associate it with one or more processes (Option 4)
5. Create a new account and associate it with the process, assembly, or department to
which it is
applicable (Option 5)
6. Enter a new Job (Option 6)
7. Enter the date job is completed and information related to type of job (Option 7)
8. Enter a transaction-no and its sup-cost (Option 8)
9. Retrieve the cost incurred on an assembly-id (Option 9)
10. Retrieve the total labor time within a department for jobs completed during a
given date (Option 10)
11. Retrieve the processes through which a given assembly-id has passed so far and
the department responsible for the process (Option 11)
12. Retrieve all jobs completed during the given date in a given department (Option
12)
13. Retrieve the customers whose category is in a given range (Option 13)
14. Delete all cut-jobs whose job-no is in a given range (Option 14)
15. Change the color of a given paint job (Option 15)
16. Import: enter new customers from a data file until the file is empty (Option 16).
17. Export: Retrieve the customers whose category is in a given range and output them
to a data file(Option 17).
18. QUIT (Option 18)

 ****************** END *************************

2
Enter the Department Number

D1
Enter the Department Data

DESC00
Error! Returning to main menu
```

➔ **Tried to insert duplicate value in primary key column of department table. Handled by azure sql databse and error message is shown to user**

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

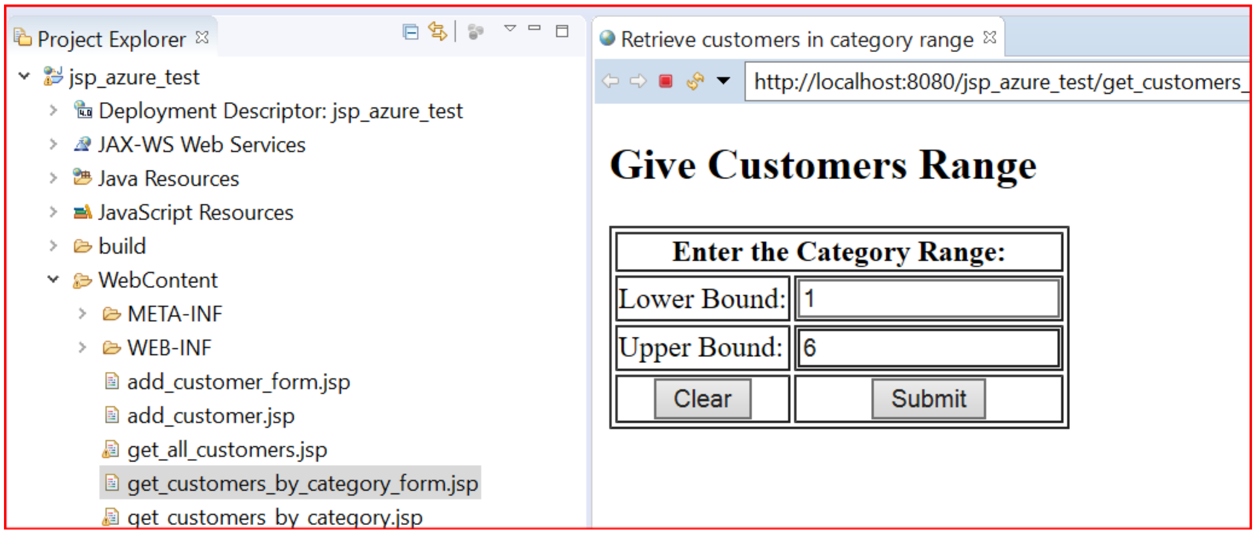
Jsp form to get details from front end html page for query 13

get_customers_by_category_form.jsp

```
      <!DOCTYPE html>

<html>
<head>
<meta charset="UTF-8">
<title>Retrieve customers in category range</title>
</head>
<body>
<h2>Give Customers Range</h2>
<!--
Form for collecting user input for the new customer record.
Upon form submission, retrieve_customers.jsp file will be invoked.
-->
<form action="get_customers_by_category.jsp">
<!-- The form organized in an HTML table for better clarity. -->
<table border=1>
<tr>
<th colspan="2">Enter the Category Range:</th>
</tr>
<tr>
<td>Lower Bound:</td>
<td><div style="text-align: center;">
<input type=text name=lower_b>
</div></td>
</tr>
<tr>
<td>Upper Bound:</td>
<td><div style="text-align: center;">
<input type=text name=upper_b>
</div></td>
</tr>
<tr>
<td><div style="text-align: center;">
<input type=reset value=Clear>
</div></td>
<td><div style="text-align: center;">
<input type=submit value=Submit>
</div></td>
</tr>
</table>
</form>
</body>
</html>
```

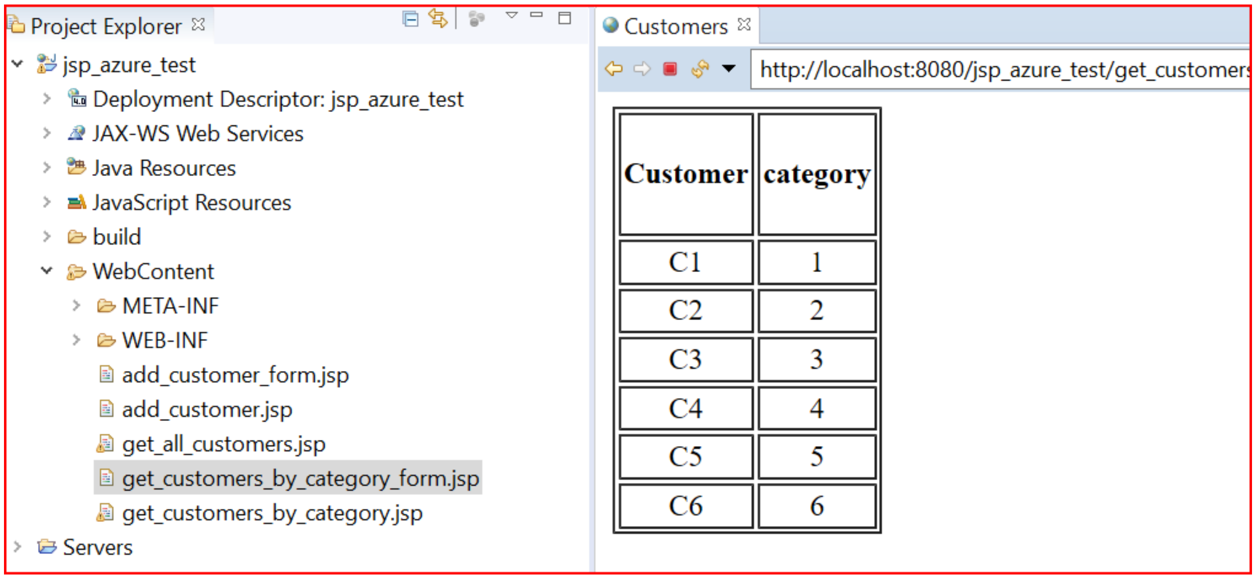
get_customers_by_category.jsp
This jsp file displays the results for the previous query through form file

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
    <meta charset="UTF-8">
        <title>Customers</title>
    </head>
    <body>
    <%@page import="jsp_azure_test.DataHandler"%>
    <%@page import="java.sql.ResultSet"%>
    <%
        // We instantiate the data handler here, and get all the customers from 
the database
        final DataHandler handler = new DataHandler();
    // Get the attribute values passed from the input form.
        String lower_b = request.getParameter("lower_b");
        String upper_b = request.getParameter("upper_b");


        if (lower_b.equals("") || upper_b.equals("")) {
        response.sendRedirect("get_customers_by_category_form.jsp");
        } else {
        int lower_b1 = Integer.parseInt(lower_b);
        int upper_b1 = Integer.parseInt(upper_b);
        // Now perform the query with the data from the form.
        final ResultSet customers = handler.retrieveCustomers(lower_b1, 
upper_b1);
    %>
    <!-- The table for displaying all the Customer records -->
    <table cellspacing="2" cellpadding="2" border="1">
        <tr> <!-- The table headers row -->
            <td align="center">
                <h4>Customer</h4>
            </td>
            <td align="center">
                <h4>category</h4>
            </td>
        <%
            while(customers.next()) { // For each Customer record returned...
                // Extract the attribute values for every row returned
                final String cname = customers.getString("Name");
                final String category = customers.getString("category");
                out.println("<tr>"); // Start printing out the new table row
                out.println( // Print each attribute value
                    "<td align=\"center\">" + cname +
                    "</td><td align=\"center\"> " + category + "</td>");
                out.println("</tr>");
            }
        }
            %>
        </table>
    </body>
</html>
```


Project Explorer
jsp_azure_test
Deployment Descriptor: jsp_azure_test
JAX-WS Web Services
Java Resources
JavaScript Resources
build
WebContent
META-INF
WEB-INF
add_customer_form.jsp
add_customer.jsp
get_all_customers.jsp
get_customers_by_category_form.jsp
get_customers_by_category.jsp
Retrieve customers in category range
http://localhost:8080/jsp_azure_test/get_customers_

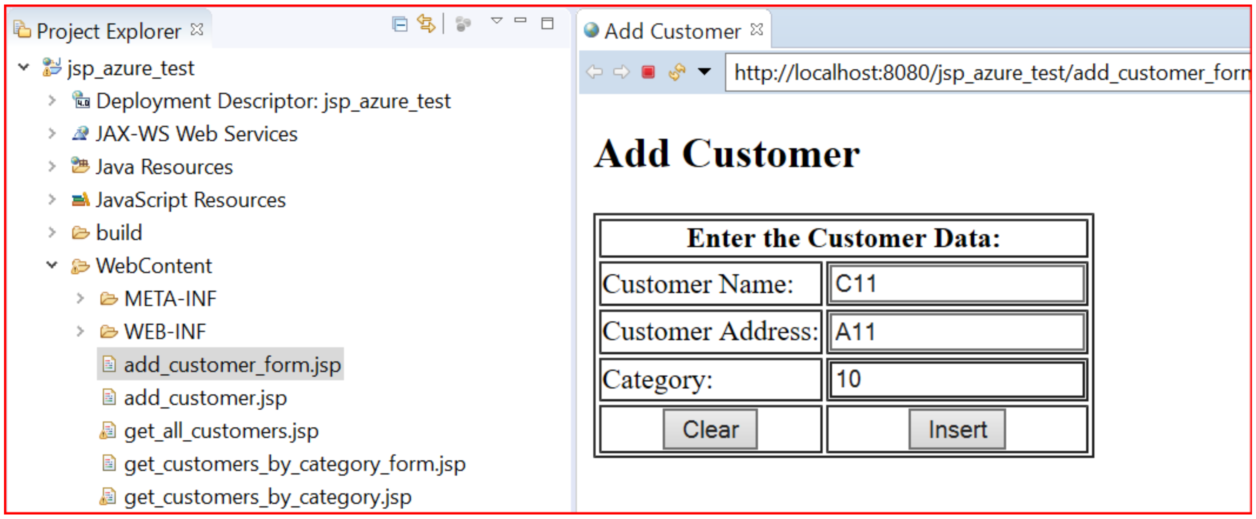
Give Customers Range

Enter the Category Range:	
Lower Bound:	1
Upper Bound:	6
Clear	Submit



Customer	category
C1	1
C2	2
C3	3
C4	4
C5	5
C6	6

add_customer_form.jsp to add customers to the database – fetches details from the user.



http://localhost:8080/jsp_azure_test/add_customer_form

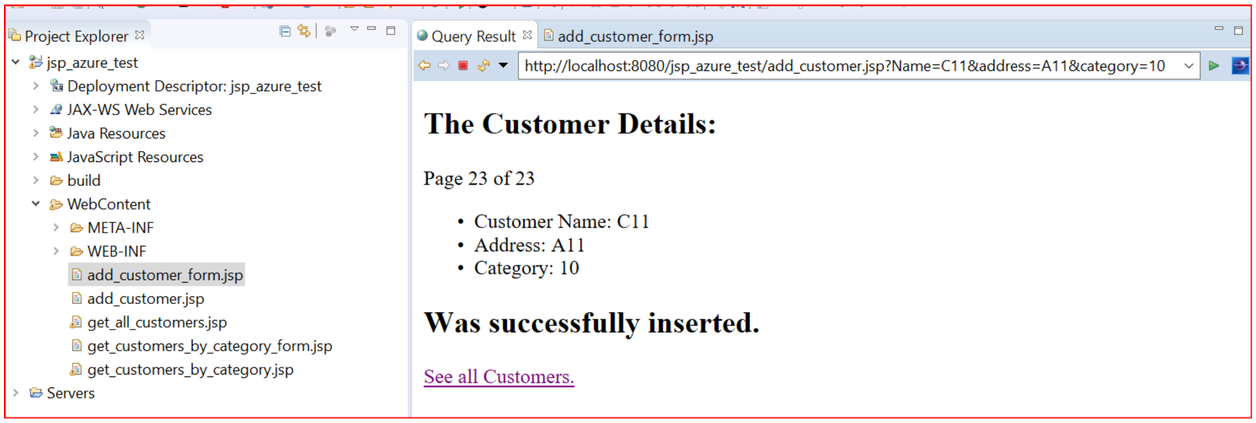
Add Customer

Enter the Customer Data:	
Customer Name:	C11
Customer Address:	A11
Category:	10

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

```
<input type=text name=address>
</div></td>
</tr>
<tr>
<td>Category:</td>
<td><div style="text-align: center;">
<input type=text name=category>
</div></td>
</tr>
<tr>
<td><div style="text-align: center;">
<input type=reset value=Clear>
</div></td>
<td><div style="text-align: center;">
<input type=submit value=Insert>
</div></td>
</tr>
</table>
</form>
</body>
</html>
```

add_customer.jsp



http://localhost:8080/jsp_azure_test/add_customer.jsp?Name=C11&address=A11&category=10

The Customer Details:

Page 23 of 23

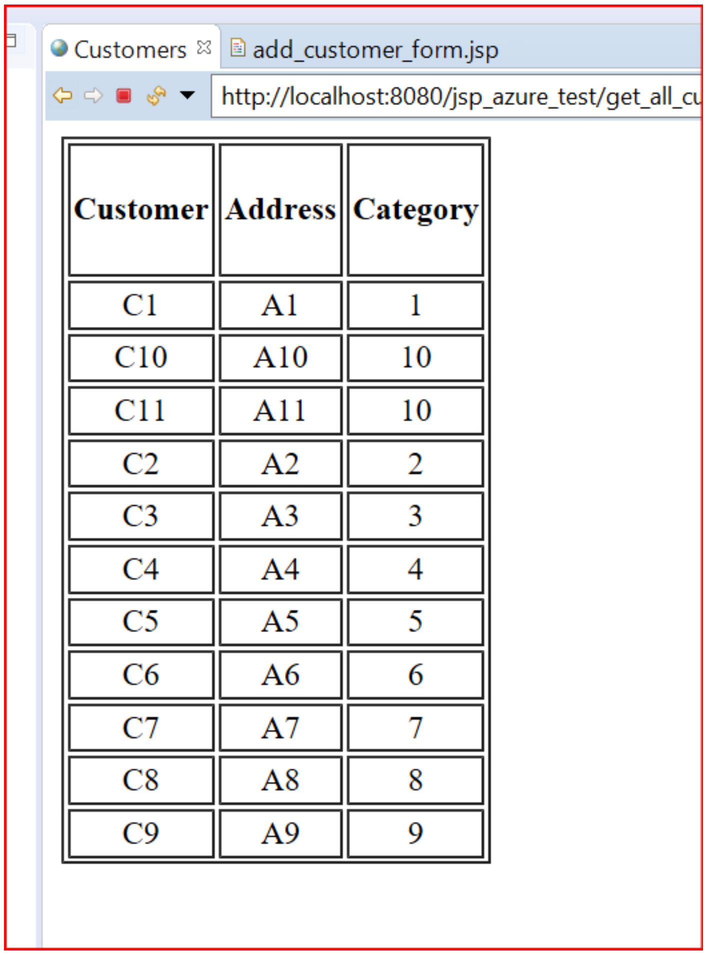
- Customer Name: C11
- Address: A11
- Category: 10

Was successfully inserted.

See all Customers.

```
Customer is inserted into the database
```

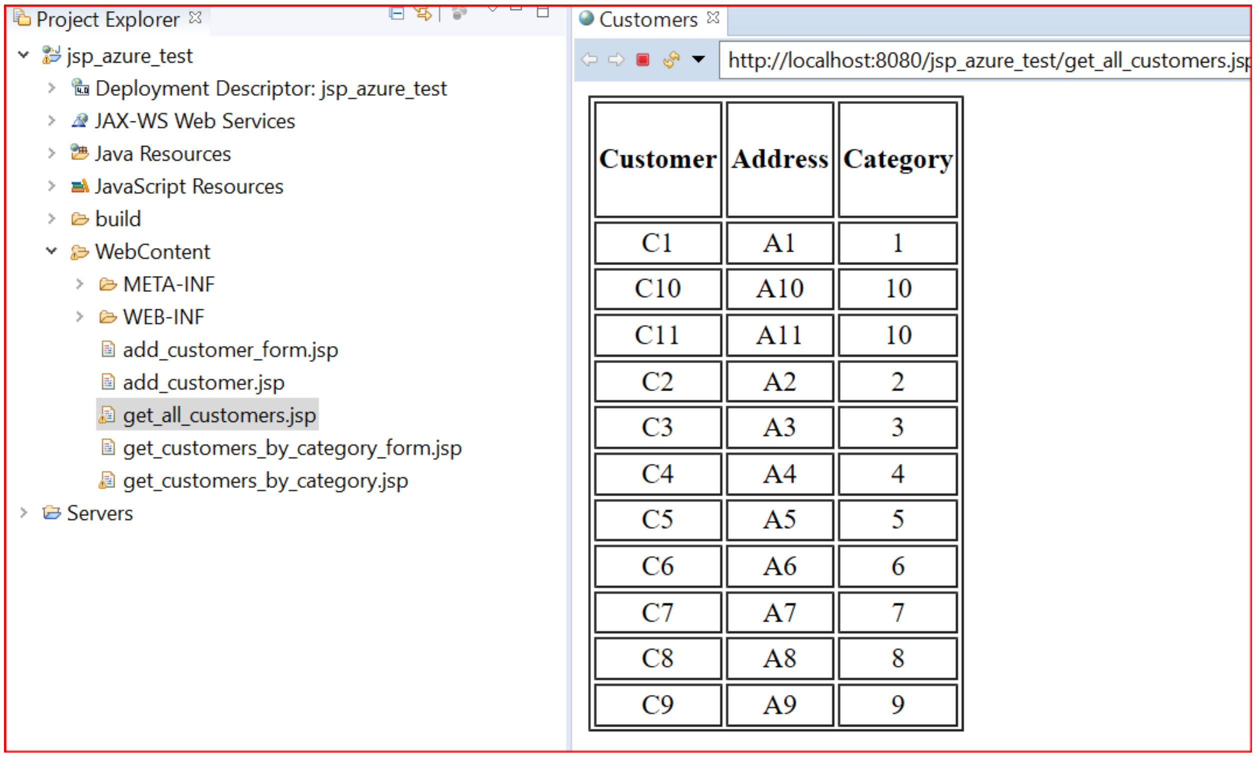
```
See all customers
```



Customer	Address	Category
C1	A1	1
C10	A10	10
C11	A11	10
C2	A2	2
C3	A3	3
C4	A4	4
C5	A5	5
C6	A6	6
C7	A7	7
C8	A8	8
C9	A9	9

7.2. Screenshots showing the testing of the Web database application

Getting all customers


Project Explorer

- jsp_azure_test
  - Deployment Descriptor: jsp_azure_test
  - JAX-WS Web Services
  - Java Resources
  - JavaScript Resources
  - build
  - WebContent
    - META-INF
    - WEB-INF
    - add_customer_form.jsp
    - add_customer.jsp
    - get_all_customers.jsp
    - get_customers_by_category_form.jsp
    - get_customers_by_category.jsp
- Servers

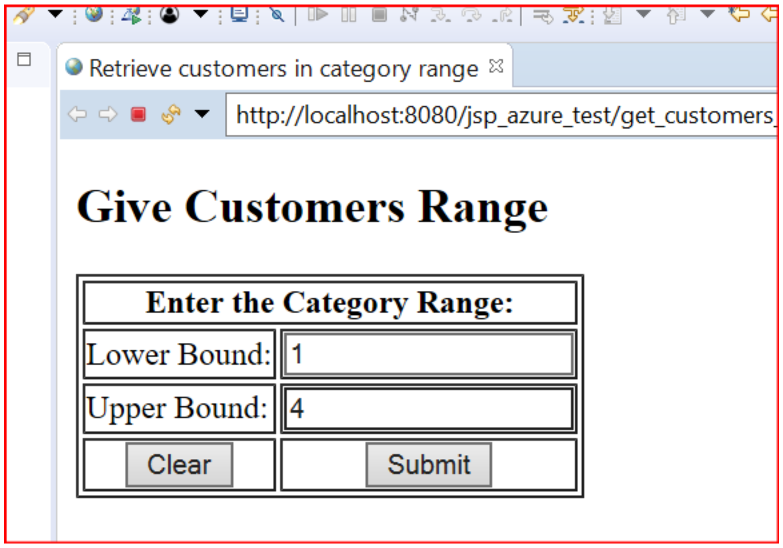
Customers

http://localhost:8080/jsp_azure_test/get_all_customers.jsp

Customer	Address	Category
C1	A1	1
C10	A10	10
C11	A11	10
C2	A2	2
C3	A3	3
C4	A4	4
C5	A5	5
C6	A6	6
C7	A7	7
C8	A8	8
C9	A9	9

Get customers by category range – Enter lower bound and upper bound , Submit

Click on clear to clear the form

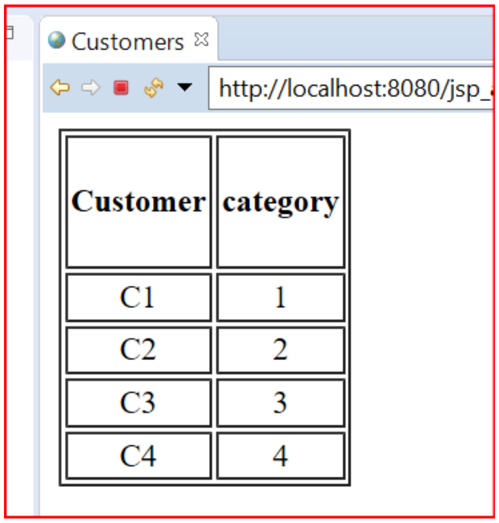

Retrieve customers in category range

http://localhost:8080/jsp_azure_test/get_customers_

Give Customers Range

Enter the Category Range:	
Lower Bound:	1
Upper Bound:	4
Clear	Submit

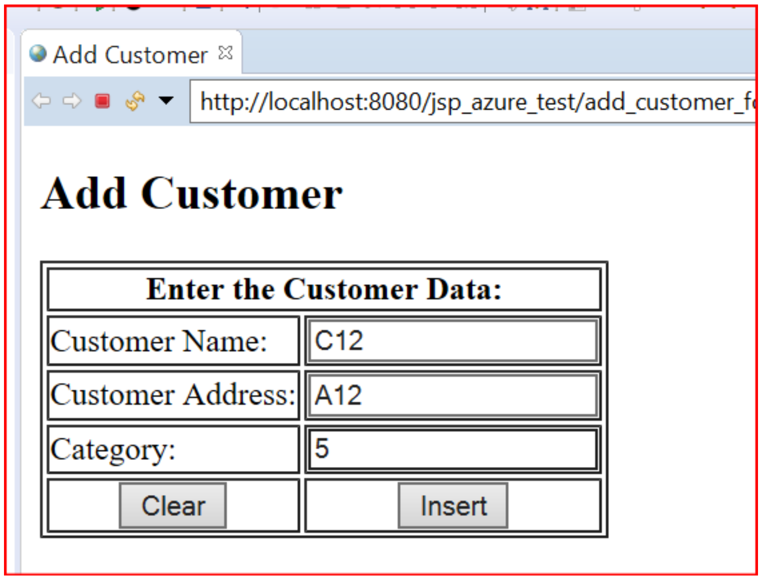
```
Clicked on submit
```



Customer	category
C1	1
C2	2
C3	3
C4	4

Results are shown above

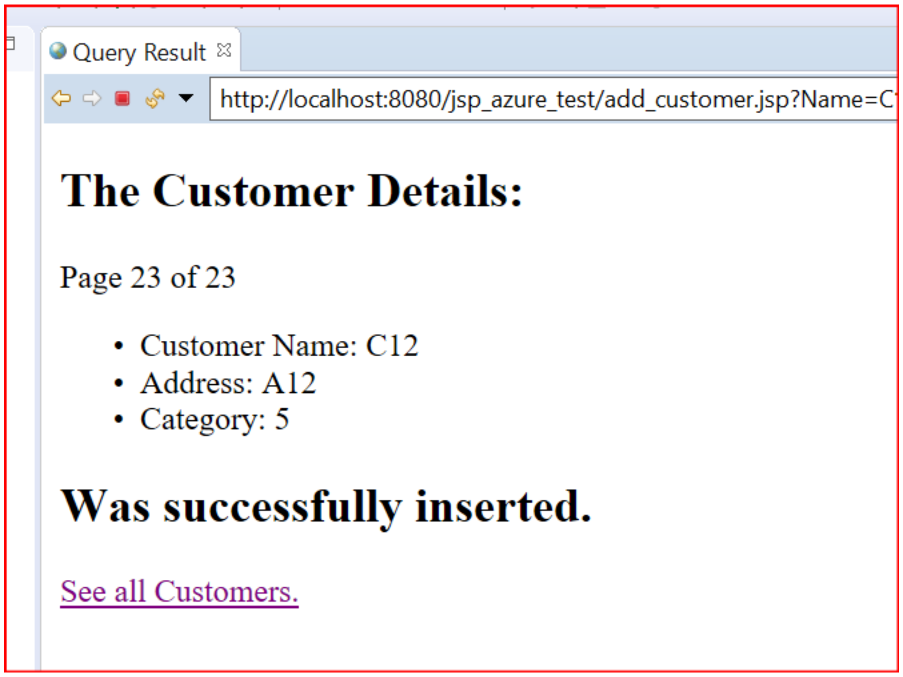
Adding Customer



Add Customer

Enter the Customer Data:	
Customer Name:	C12
Customer Address:	A12
Category:	5
Clear	Insert

Click on insert to insert the record into database


Query Result

http://localhost:8080/jsp_azure_test/add_customer.jsp?Name=C

The Customer Details:

Page 23 of 23

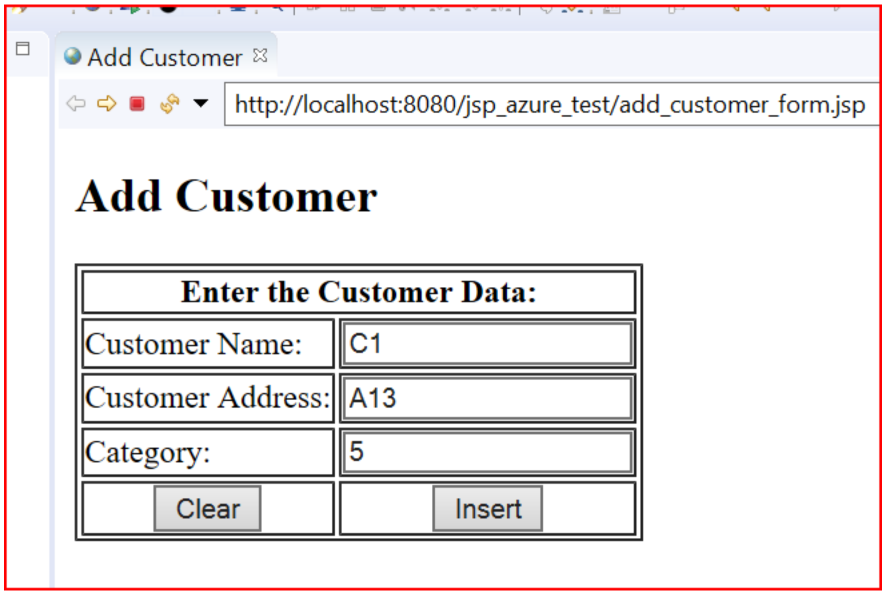
- Customer Name: C12
- Address: A12
- Category: 5

Was successfully inserted.

See all Customers.

Record inserted successfully

Lets do Primary key error

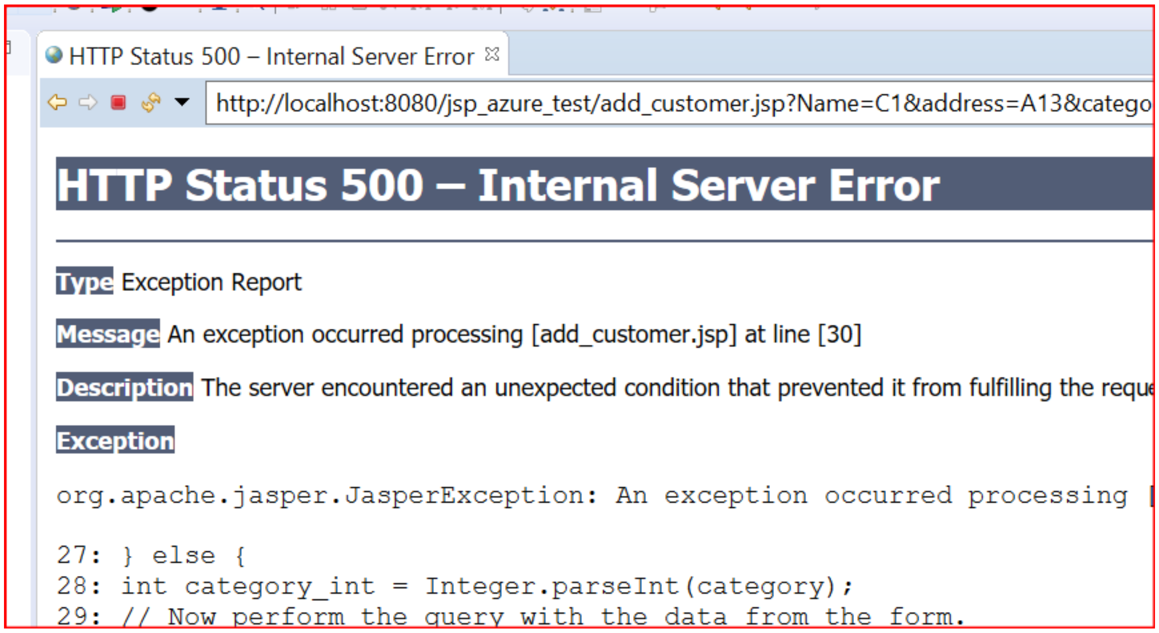

Add Customer

http://localhost:8080/jsp_azure_test/add_customer_form.jsp

Add Customer

Enter the Customer Data:	
Customer Name:	C1
Customer Address:	A13
Category:	5
Clear	Insert

Trying to insert customer C1 again which should throw error


HTTP Status 500 – Internal Server Error

http://localhost:8080/jsp_azure_test/add_customer.jsp?Name=C1&address=A13&catego

HTTP Status 500 – Internal Server Error

Type Exception Report

Message An exception occurred processing [add_customer.jsp] at line [30]

Description The server encountered an unexpected condition that prevented it from fulfilling the reque

Exception

```
org.apache.jasper.JasperException: An exception occurred processing 

27: } else {
28: int category_int = Integer.parseInt(category);
29: // Now perform the query with the data from the form.
```

Ah ha! Throws internal server error as azure database throws an error for primary key violation.