Course Number and Name: CSE 4308 Database Management Systems Lab	
Student Name: Nafisa Maliyat	**Student ID:** 200042133
Report Submission Date: 08 September, 2022	**Name of Lab Instructor:** Md. Bakhtiar Hasan, Lecturer, CSE Zannatun Naim Sristy, Lecturer, CSE

Overview:

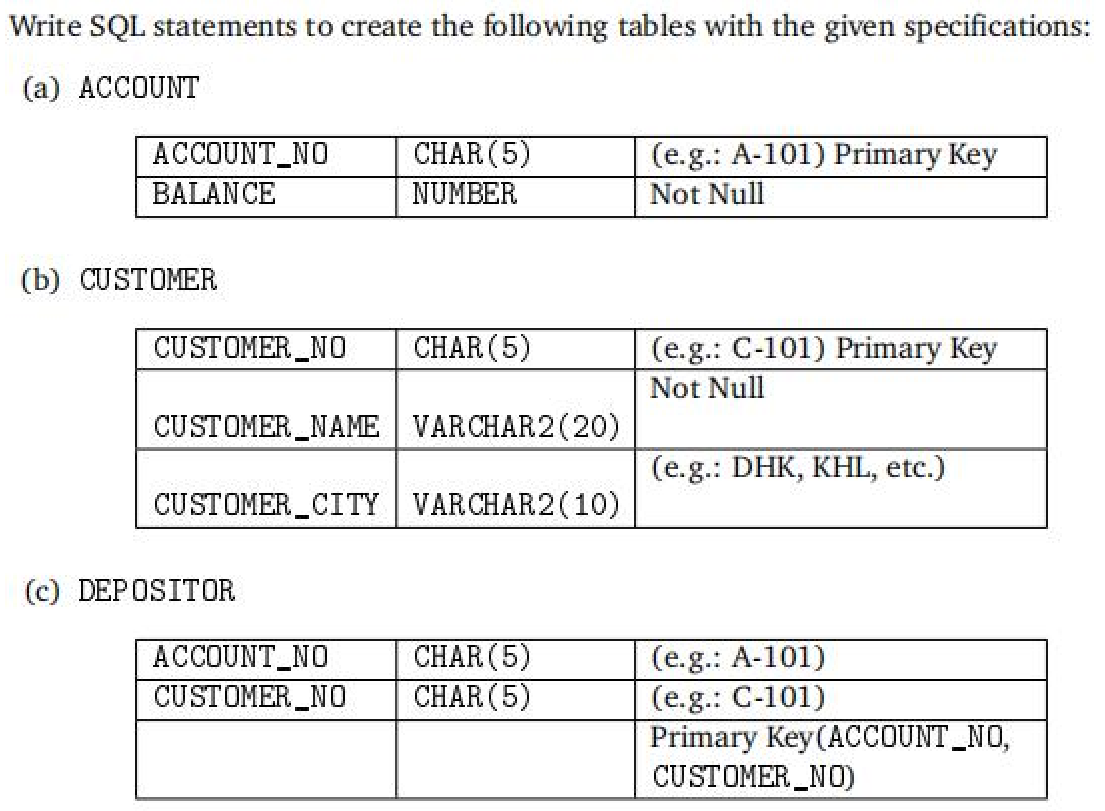
The lab required applying appropriate conditions using different queries to find the information specified in the question. This was done using combination of various clauses and nested queries with help from the lab manual and a few internet resources.

On the next pages, I have mentioned the following :

- the problem statement,
- the problem analysis,
- the code snippets of the queries,
- problems faced (if any) during solution of the tasks, and
- results obtained from the queries.

Task 1

Problem Statement:



Write SQL statements to create the following tables with the given specifications:

(a) ACCOUNT

ACCOUNT_NO	CHAR(5)	(e.g.: A-101) Primary Key
BALANCE	NUMBER	Not Null

(b) CUSTOMER

CUSTOMER_NO	CHAR(5)	(e.g.: C-101) Primary Key
CUSTOMER_NAME	VARCHAR2(20)	Not Null
CUSTOMER_CITY	VARCHAR2(10)	(e.g.: DHK, KHL, etc.)

(c) DEPOSITOR

ACCOUNT_NO	CHAR(5)	(e.g.: A-101)
CUSTOMER_NO	CHAR(5)	(e.g.: C-101)
		Primary Key(ACCOUNT_NO, CUSTOMER_NO)

Analysis of the problem:

Three tables were created with the mentioned attributes and constraints.

SQL Query:

```sql
DROP TABLE DEPOSITOR_INFO;
DROP TABLE ACCOUNT;
DROP TABLE CUSTOMER;


CREATE TABLE ACCOUNT
(
    ACCOUNT_NO CHAR(5),
    BALANCE NUMBER NOT NULL,
    CONSTRAINT PK_ACCOUNT PRIMARY KEY(ACCOUNT_NO)
);
```

```sql
CREATE TABLE CUSTOMER
(
    CUSTOMER_NO CHAR(5),
    CUSTOMER_NAME VARCHAR2(20) NOT NULL,
    CUSTOMER_CITY VARCHAR2(10),
    CONSTRAINT PK_CUSTOMER PRIMARY KEY(CUSTOMER_NO)
);
```

```sql
CREATE TABLE DEPOSITOR
(
    ACCOUNT_NO CHAR(5),
    CUSTOMER_NO CHAR(5),
    CONSTRAINT PK_DEPOSITOR PRIMARY KEY(ACCOUNT_NO, CUSTOMER_NO)
);
```

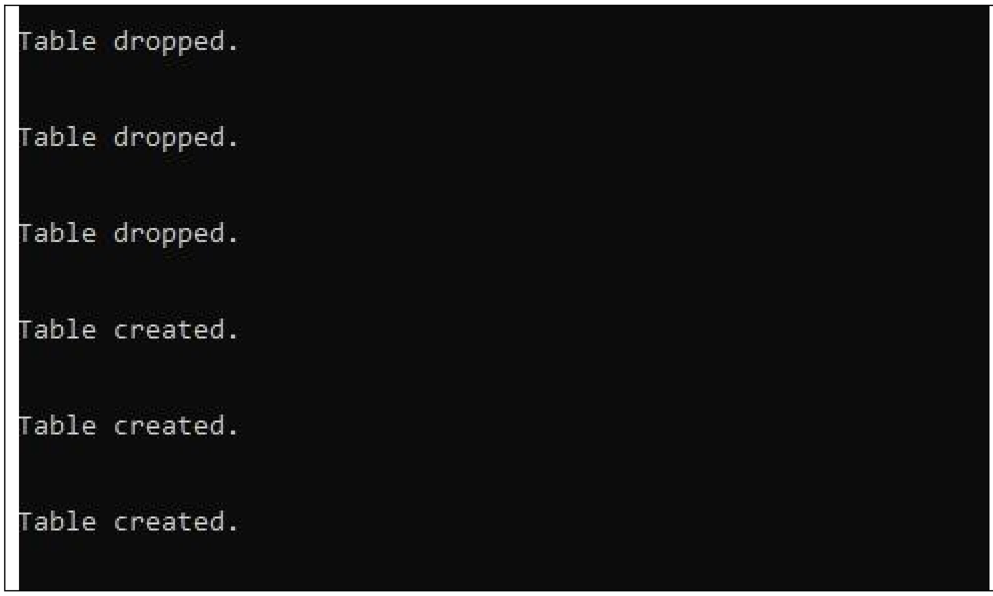
Any problems faced and how it was solved:

When the query was run multiple times, the existing tables created problems. This was solved using Drop Table command at the beginning.

Since the DEPOSITOR table is renamed later on, using the Drop Table for ‘DEPOSITOR’ caused an error since it does not exist. Thus Drop Table was used for ‘DEPOSITOR_INFO’ instead., which solved the issue.

DEPOSTIOR_INFO contains references to ACCOUNT and CUSTOMER tables thus dropping these tables first gave an error. This was resolved by dropping DEPOSITOR_INFO first and then the other two tables.

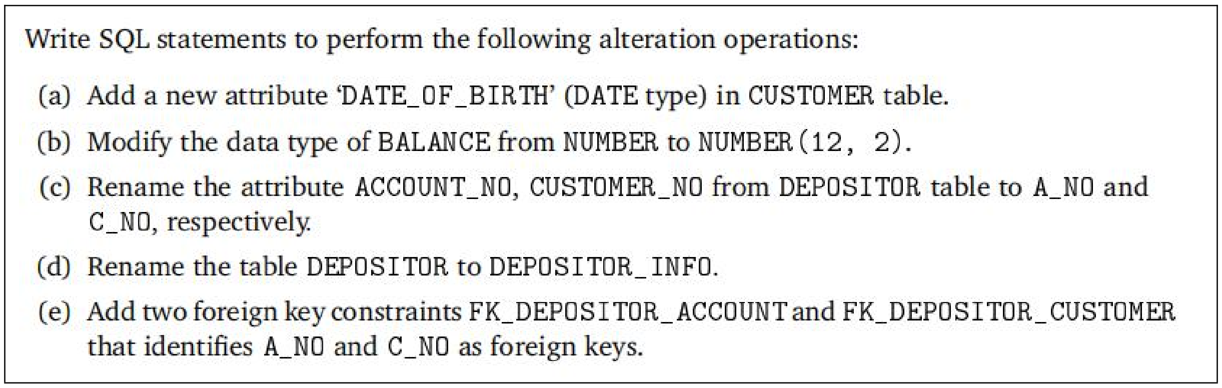
Results:



```
Table dropped.


Table dropped.


Table dropped.


Table created.


Table created.


Table created.
```

Task 2

Problem Statement:


Write SQL statements to perform the following alteration operations:

(a) Add a new attribute 'DATE_OF_BIRTH' (DATE type) in CUSTOMER table.

(b) Modify the data type of BALANCE from NUMBER to NUMBER(12, 2).

(c) Rename the attribute ACCOUNT_NO, CUSTOMER_NO from DEPOSITOR table to A_NO and C_NO, respectively.

(d) Rename the table DEPOSITOR to DEPOSITOR_INFO.

(e) Add two foreign key constraints FK_DEPOSITOR_ACCOUNT and FK_DEPOSITOR_CUSTOMER that identifies A_NO and C_NO as foreign keys.

Analysis of the problem:

The queries for these were written using Alter Table queries, as provided in the lab manual.

(c) required two alter table queries for renaming two attributes, as well as (e).

Task 2 thus required seven query statements.

SQL Query:

```sql
--2A--
ALTER TABLE CUSTOMER ADD DATE_OF_BIRTH DATE;
```

```sql
--2B--
ALTER TABLE ACCOUNT MODIFY BALANCE NUMBER(12,2);
```

```sql
--2C--
ALTER TABLE DEPOSITOR RENAME COLUMN ACCOUNT_NO TO A_NO;
ALTER TABLE DEPOSITOR RENAME COLUMN CUSTOMER_NO TO C_NO;
```

```sql
--2D--
ALTER TABLE DEPOSITOR RENAME TO DEPOSITOR_INFO;
```

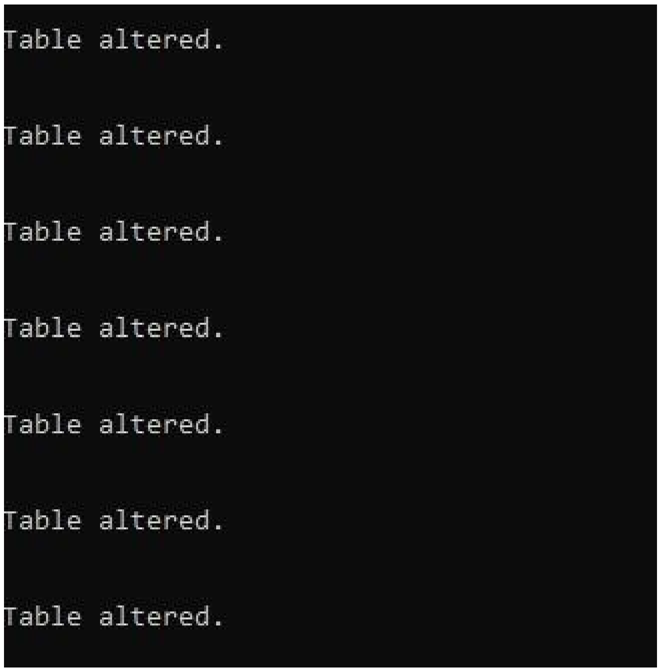
```sql
--2E--
ALTER TABLE DEPOSITOR_INFO
ADD CONSTRAINT FK_DEPOSITOR_ACCOUNT
FOREIGN KEY (A_NO) REFERENCES
ACCOUNT(ACCOUNT_NO);

ALTER TABLE DEPOSITOR_INFO
ADD CONSTRAINT FK_DEPOSITOR_CUSTOMER
```

```
FOREIGN KEY (C_NO) REFERENCES
CUSTOMER(CUSTOMER_NO);
```

Any problems faced and how it was solved:

Since the lab manual provided the necessary instructions, there were no issues encountered.

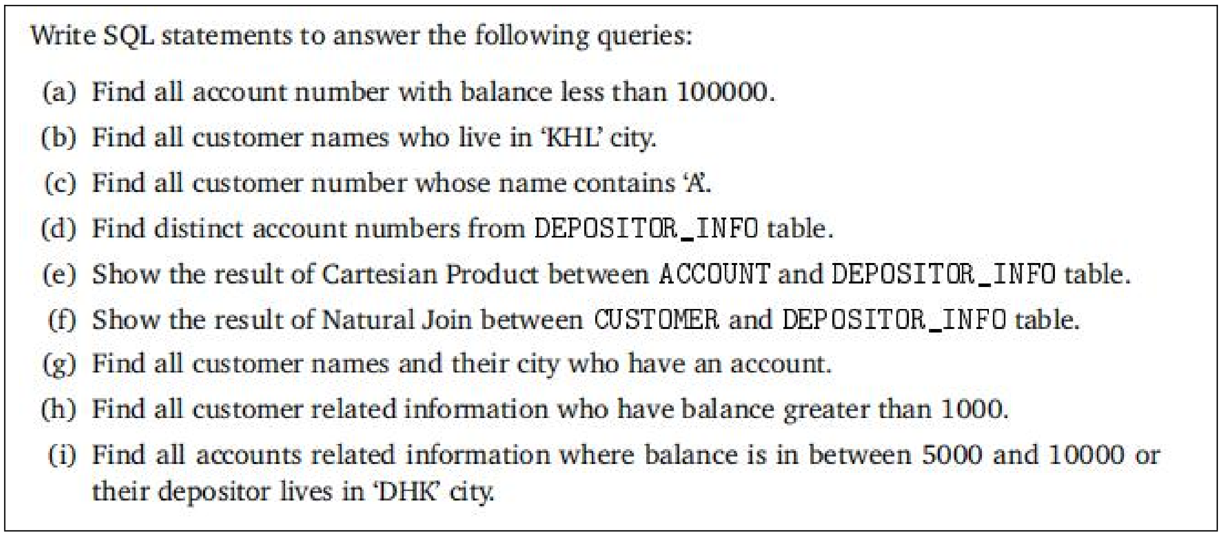
Results:



```
Table altered.


Table altered.


Table altered.


Table altered.


Table altered.


Table altered.


Table altered.
```

Task 3

Problem Statement:



Write SQL statements to answer the following queries:

(a) Find all account number with balance less than 100000.

(b) Find all customer names who live in 'KHL' city.

(c) Find all customer number whose name contains 'A'.

(d) Find distinct account numbers from DEPOSITOR_INFO table.

(e) Show the result of Cartesian Product between ACCOUNT and DEPOSITOR_INFO table.

(f) Show the result of Natural Join between CUSTOMER and DEPOSITOR_INFO table.

(g) Find all customer names and their city who have an account.

(h) Find all customer related information who have balance greater than 1000.

(i) Find all accounts related information where balance is in between 5000 and 10000 or their depositor lives in 'DHK' city.

Analysis of the problem:

From (a) to (f), simple queries in the format of select-from-where were performed on single table and appropriate conditions were applied.

(g) required performing query on both DEPOSITOR_INFO and CUSTOMER since DEPOSITOR_INFO stores the customer number of those customers associated with an account number. CUSTOMER table was required to select customer names and their city.

In (h), DEPOSITOR_INFO and ACCOUNT tables were used to first find the list of customer numbers where the corresponding accounts has a balance greater than 1000. Another query was performed on the obtained result to retrieve the customer information of those selected customers from CUSTOMER table.
'Distinct' keyword was used to ensure even if a customer has more than one account that fulfilled the conditions, their information would be printed just once.

For (i), queries were performed on all three tables. All the conditions were ANDED together; another alternative query could have been nested query similar to (h).
Here, 'distinct' was not used because one account cannot be associated with more than one person logically,

SQL Query:

```sql
--3A--
SELECT ACCOUNT_NO
FROM ACCOUNT
WHERE BALANCE<100000;
```

```
--3B--
SELECT CUSTOMER_NAME
FROM CUSTOMER
WHERE CUSTOMER_CITY='KHL';
```

```
--3C--
SELECT CUSTOMER_NO
FROM CUSTOMER
WHERE CUSTOMER_NAME LIKE '%A%';
```

```
--3D--
SELECT DISTINCT A_NO
FROM DEPOSITOR_INFO;
```

```
--3E--
SELECT *
FROM ACCOUNT, DEPOSITOR_INFO;
```

```
--3F--
ALTER TABLE DEPOSITOR_INFO RENAME COLUMN C_NO TO CUSTOMER_NO;
SELECT *
FROM CUSTOMER NATURAL JOIN DEPOSITOR_INFO;
ALTER TABLE DEPOSITOR_INFO RENAME COLUMN CUSTOMER_NO TO C_NO;
```

```
--3G--
SELECT CUSTOMER.CUSTOMER_NAME, CUSTOMER.CUSTOMER_CITY
FROM DEPOSITOR_INFO, CUSTOMER
WHERE CUSTOMER.CUSTOMER_NO = DEPOSITOR_INFO.C_NO;
```

```
--3H--
SELECT CUSTOMER.CUSTOMER_NO, CUSTOMER.CUSTOMER_NAME, CUSTOMER.CUSTOMER_CITY
FROM CUSTOMER
WHERE CUSTOMER_NO IN
(
    SELECT DISTINCT DEPOSITOR_INFO.C_NO
    FROM DEPOSITOR_INFO, ACCOUNT
    WHERE DEPOSITOR_INFO.A_NO = ACCOUNT.ACCOUNT_NO
        AND ACCOUNT.BALANCE>1000
);
```

```
--3I--
SELECT ACCOUNT.ACCOUNT_NO, ACCOUNT.BALANCE
FROM DEPOSITOR_INFO, ACCOUNT, CUSTOMER
```

```
WHERE DEPOSITOR_INFO.A_NO = ACCOUNT.ACCOUNT_NO
    AND DEPOSITOR_INFO.C_NO = CUSTOMER_NO
    AND ACCOUNT.BALANCE BETWEEN 5000 AND 10000
    AND CUSTOMER.CUSTOMER_CITY='DHK';
```

Problems faced and how they were solved:

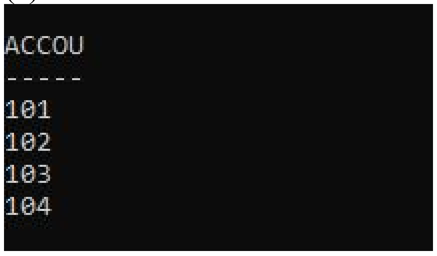
In 3(f), the Natural Join did not occur as expected since DEPOSITOR_INFO and CUSTOMER tables no longer have a commonly named attribute (CUSTOMER_NO was renamed to C_NO in DEPOSITOR_INFO table). Renaming the attribute in DEPOSITOR_INFO table temporarily for the Natural Join query solved the problem.

Test Entries:

```
INSERT INTO ACCOUNT VALUES('101', 1000);
INSERT INTO ACCOUNT VALUES('102', 6000);
INSERT INTO ACCOUNT VALUES('103', 10000);
INSERT INTO ACCOUNT VALUES('104', 3000);

INSERT INTO CUSTOMER VALUES('201', 'NAFISA', 'DHK', null);
INSERT INTO CUSTOMER VALUES('202', 'ANIKA', 'KHL', null);
INSERT INTO CUSTOMER VALUES('203', 'MALIYAT', 'KHL', null);
INSERT INTO CUSTOMER VALUES('204', 'AJAY', 'DHK', null);

INSERT INTO DEPOSITOR_INFO VALUES('101', '201');
INSERT INTO DEPOSITOR_INFO VALUES('102', '202');
INSERT INTO DEPOSITOR_INFO VALUES('104', '203');
INSERT INTO DEPOSITOR_INFO VALUES('101', '204');
```

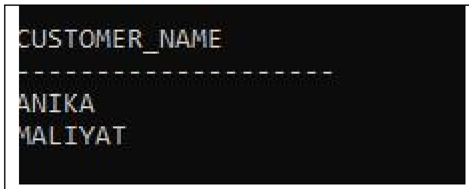
Results:

(a)

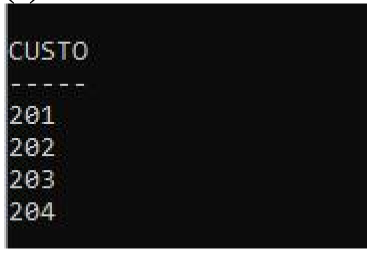


```
ACCOU
-----
101
102
103
104
```

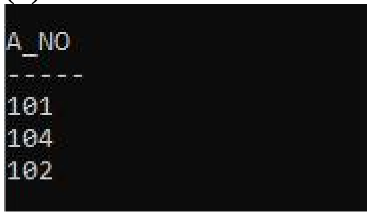
(b)


```
CUSTOMER_NAME
--------------------
ANIKA
MALIYAT
```

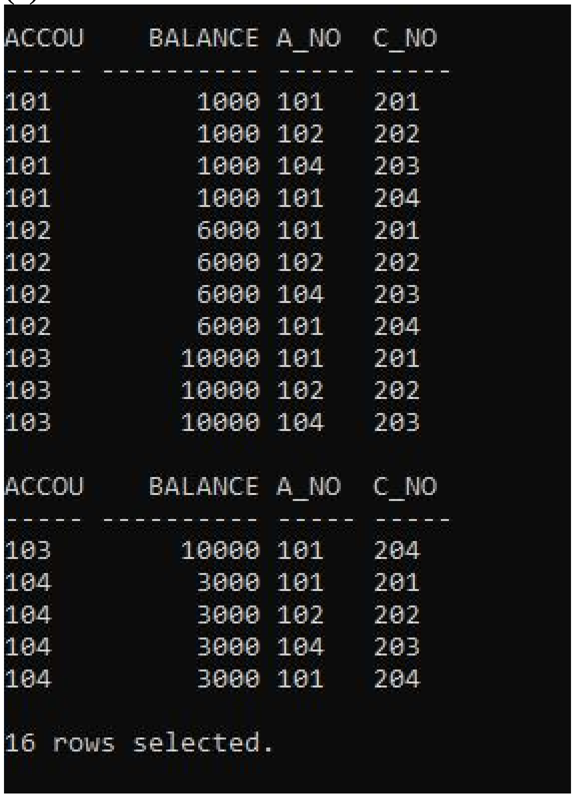
(c)


```
CUSTO
-----
201
202
203
204
```

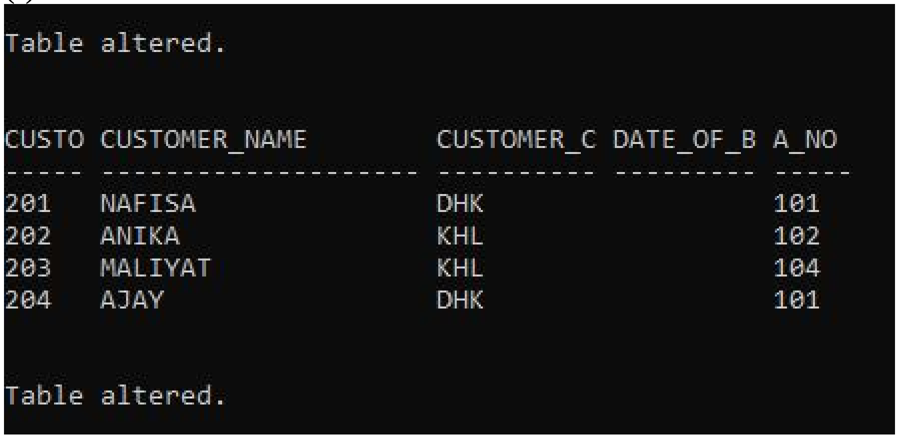
(d)


```
A_NO
-----
101
104
102
```

(e)

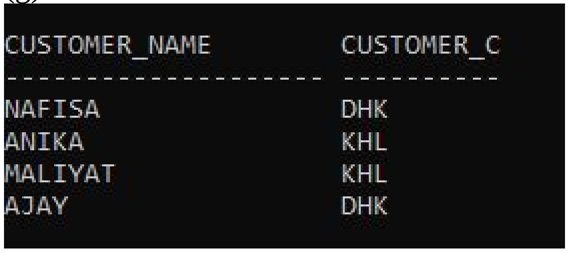

```
ACCOU    BALANCE A_NO  C_NO
----- ---------- ----- -----
101         1000 101   201
101         1000 102   202
101         1000 104   203
101         1000 101   204
102         6000 101   201
102         6000 102   202
102         6000 104   203
102         6000 101   204
103        10000 101   201
103        10000 102   202
103        10000 104   203

ACCOU    BALANCE A_NO  C_NO
----- ---------- ----- -----
103        10000 101   204
104         3000 101   201
104         3000 102   202
104         3000 104   203
104         3000 101   204

16 rows selected.
```

(f)



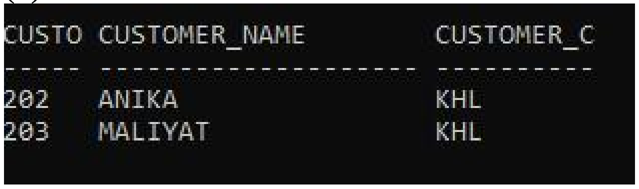
```
Table altered.


CUSTO CUSTOMER_NAME        CUSTOMER_C DATE_OF_B A_NO
----- -------------------- ---------- --------- -----
201   NAFISA               DHK                  101
202   ANIKA                KHL                  102
203   MALIYAT              KHL                  104
204   AJAY                 DHK                  101


Table altered.
```

(g)



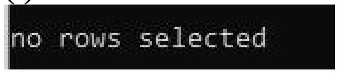
```
CUSTOMER_NAME        CUSTOMER_C
-------------------- ----------
NAFISA               DHK
ANIKA                KHL
MALIYAT              KHL
AJAY                 DHK
```

(h)



```
CUSTO CUSTOMER_NAME        CUSTOMER_C
----- -------------------- ----------
202   ANIKA                KHL
203   MALIYAT              KHL
```

(i)



```
no rows selected
```