Islamic University of Technology (IUT)

CSE 4308: Database Management Systems Lab

Lab Report # 3

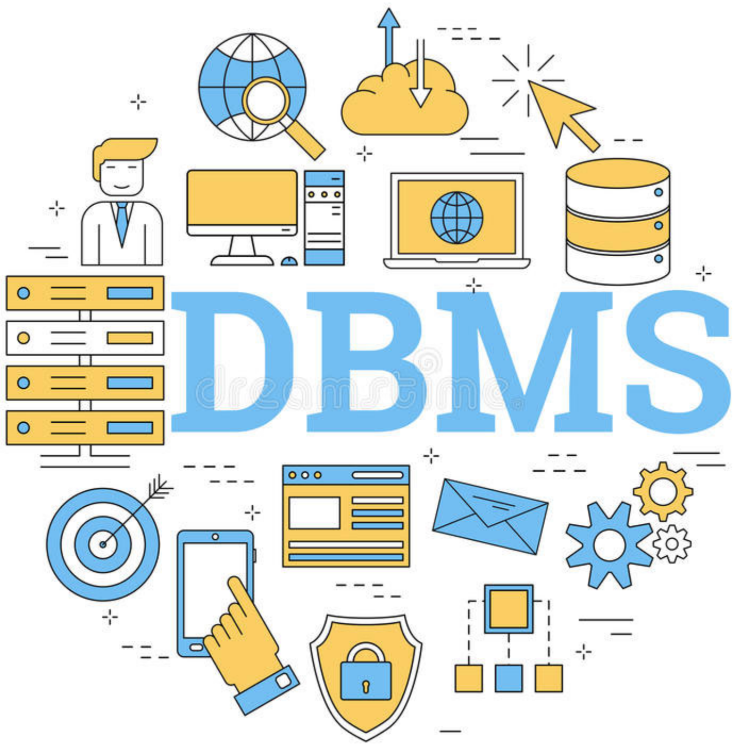
Submitted to:

Md. Bakhtiar Hasan,

Lecturer, CSE.

Zannatun Naim Srsity,

Lecturer, CSE.



Submitted by:

M M Nazmul Hossain
ID 200042118
CSE (SWE)

Submission Date:

07.09.2022

Introduction

The goal of the third database management systems lab was to learn how to create, modify, and alter different parts of a relation in a database.

In this Lab, we learned about the Data Definition Language and the Data Manipulation Language. The different types of statement functions are involved in both.

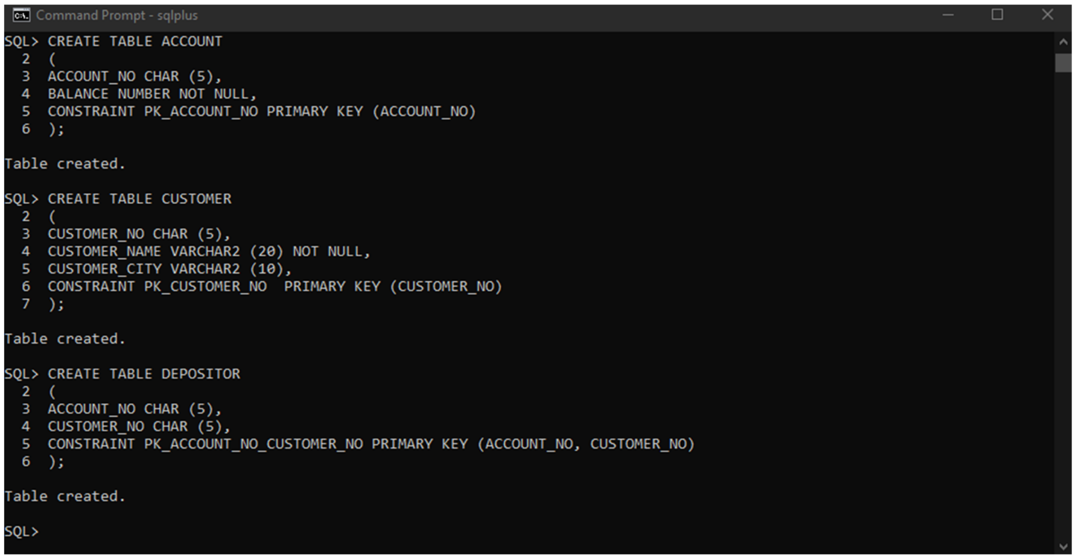
Method

The first task of the lab was to create three tables with specific attributes and domains.

The first task of the Lab was to create three tables with specific attributes and domains.

The first table was an Account table with two attributes. ACCOUNT_NO with char(5) domain, BALANCE with Number domain which can't be NULL, and a constraint where the ACCOUNT_NO was the primary key.

The second table was a Customer table with three attributes. A CUSTOMER_NO attribute with char(5) domain, CUSTOMER_Name attribute with VARCHAR2(20) domain which can't be NULL, and a CUSTOMER_CITY attribute with VARCHAR2(10) attribute. There is an additional constraint where the CUSTOMER_NO attribute is a primary key.

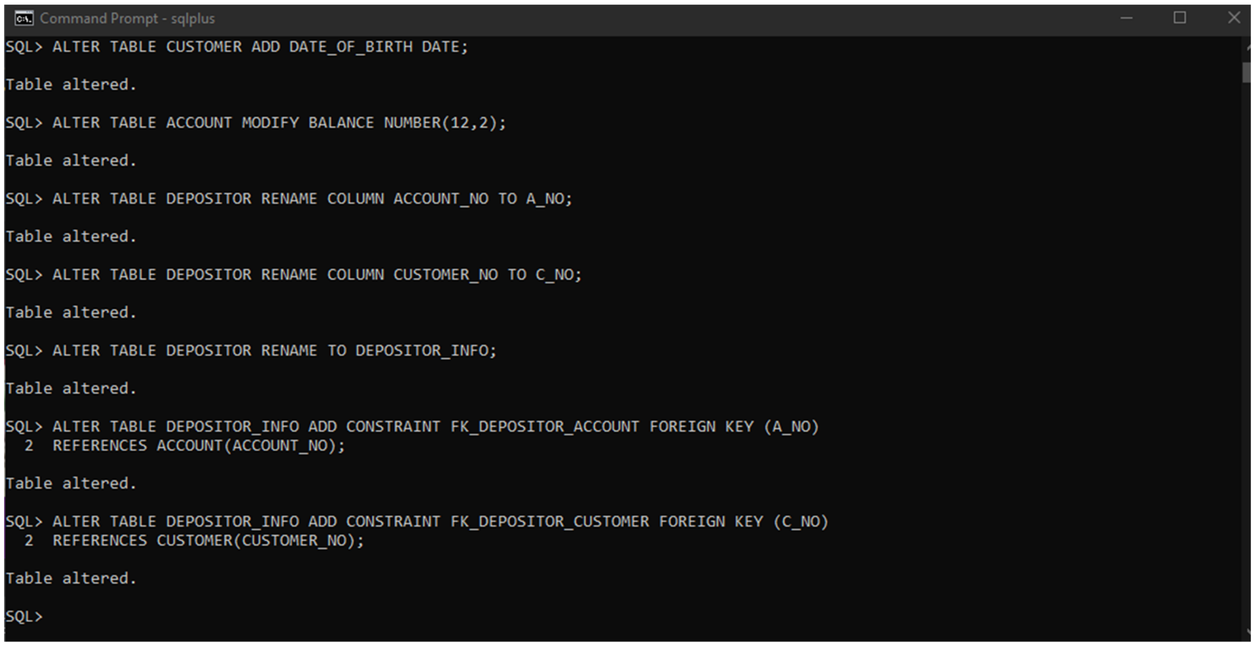


```
SQL> CREATE TABLE ACCOUNT
  2  (
  3  ACCOUNT_NO CHAR (5),
  4  BALANCE NUMBER NOT NULL,
  5  CONSTRAINT PK_ACCOUNT_NO PRIMARY KEY (ACCOUNT_NO)
  6  );

Table created.

SQL> CREATE TABLE CUSTOMER
  2  (
  3  CUSTOMER_NO CHAR (5),
  4  CUSTOMER_NAME VARCHAR2 (20) NOT NULL,
  5  CUSTOMER_CITY VARCHAR2 (10),
  6  CONSTRAINT PK_CUSTOMER_NO  PRIMARY KEY (CUSTOMER_NO)
  7  );

Table created.

SQL> CREATE TABLE DEPOSITOR
  2  (
  3  ACCOUNT_NO CHAR (5),
  4  CUSTOMER_NO CHAR (5),
  5  CONSTRAINT PK_ACCOUNT_NO_CUSTOMER_NO PRIMARY KEY (ACCOUNT_NO, CUSTOMER_NO)
  6  );

Table created.

SQL>
```

The final table is a DEPOSITOR table. There are only two attributes, ACCOUNT_NO with CHAR(5) domain and CUSTOMER_NO with CHAR(5) domain. They are both the primary key to this table.

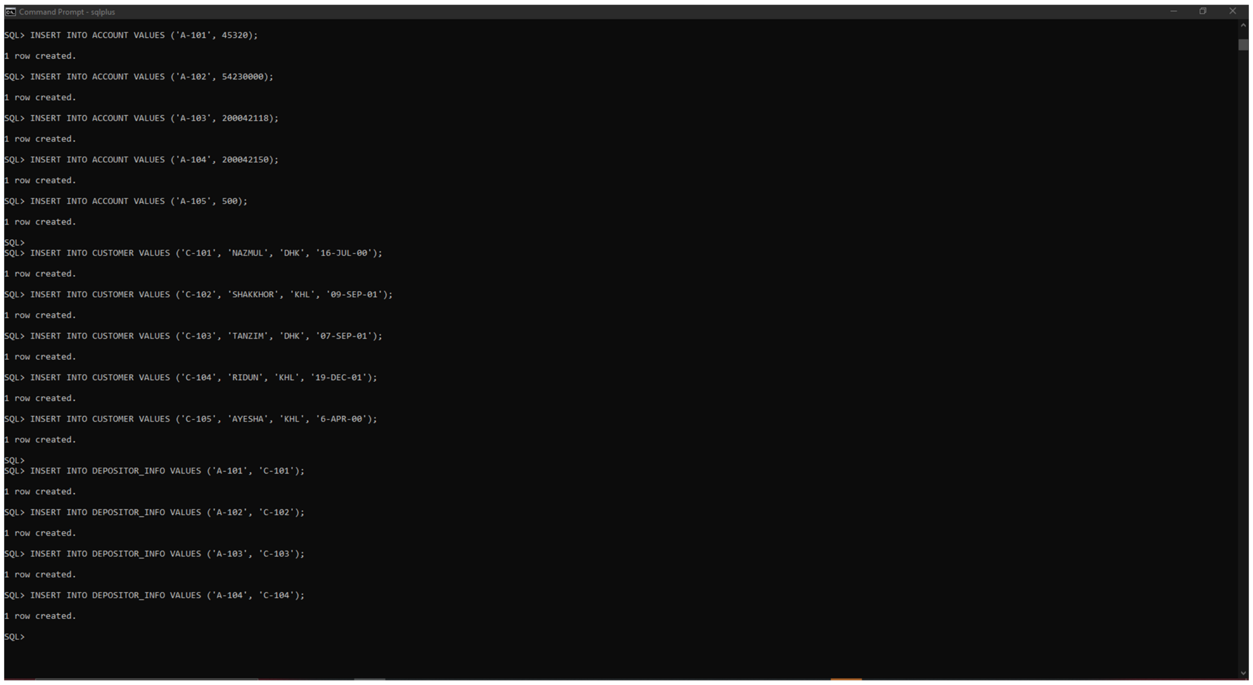
The second task of the lab was to alter these three tables in various ways, using the statement functions, previously shown in the lab.

(a) I altered the CUSTOMER table for it to have another attribute, Date of Birth with Date domain.
(b) I altered the ACCOUNT table and modified the domain of the BALANCE attribute. I changed its precision to NUMBER(12,2).
(c) I altered the names of the attributes of the DEPOSITOR table to A_NO and C_NO.
(d) I also altered the name of the DEPOSITOR table and renamed it to DEPOSITOR_INFO.
(e) I added two foreign key references to the DEPOSITOR_INFO table, where the A_NO foreign key references the ACCOUNT_NO attribute of the ACCOUNT table and the C_NO foreign key references the CUSTOMER_NO of the CUSTOMER table.



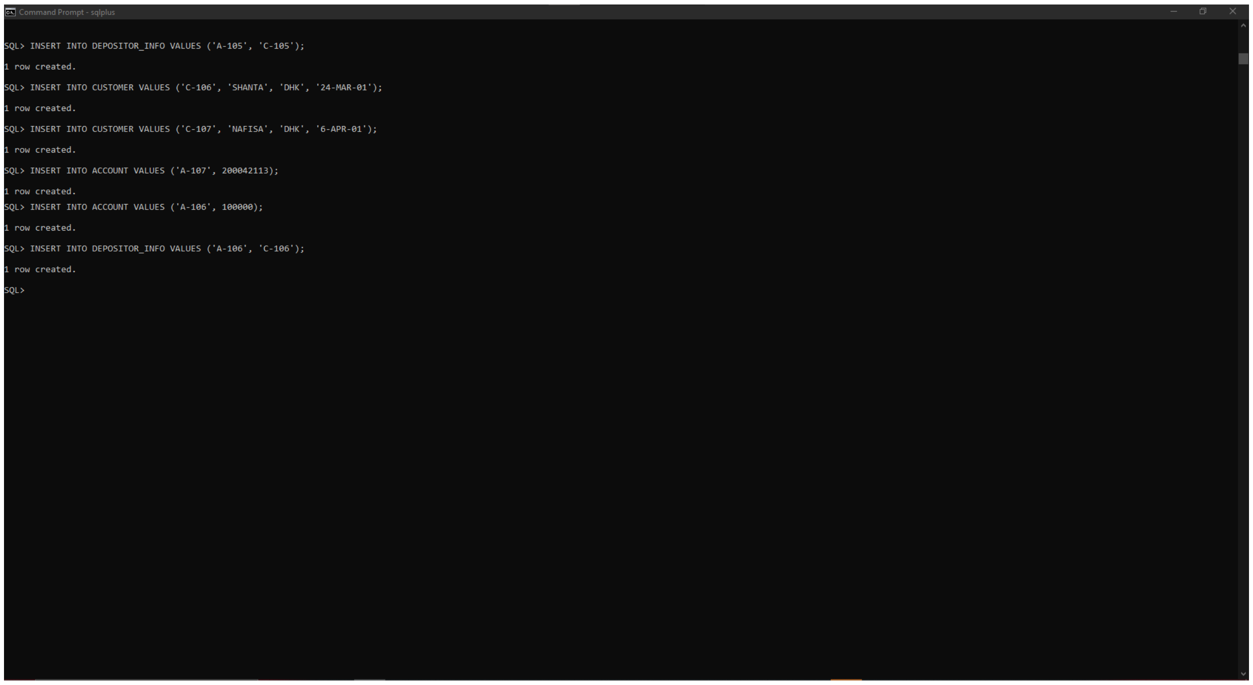
```
SQL> ALTER TABLE CUSTOMER ADD DATE_OF_BIRTH DATE;

Table altered.

SQL> ALTER TABLE ACCOUNT MODIFY BALANCE NUMBER(12,2);

Table altered.

SQL> ALTER TABLE DEPOSITOR RENAME COLUMN ACCOUNT_NO TO A_NO;

Table altered.

SQL> ALTER TABLE DEPOSITOR RENAME COLUMN CUSTOMER_NO TO C_NO;

Table altered.

SQL> ALTER TABLE DEPOSITOR RENAME TO DEPOSITOR_INFO;

Table altered.

SQL> ALTER TABLE DEPOSITOR_INFO ADD CONSTRAINT FK_DEPOSITOR_ACCOUNT FOREIGN KEY (A_NO)
  2  REFERENCES ACCOUNT(ACCOUNT_NO);

Table altered.

SQL> ALTER TABLE DEPOSITOR_INFO ADD CONSTRAINT FK_DEPOSITOR_CUSTOMER FOREIGN KEY (C_NO)
  2  REFERENCES CUSTOMER(CUSTOMER_NO);

Table altered.

SQL>
```

I Input different input values which might satisfy the given problems.



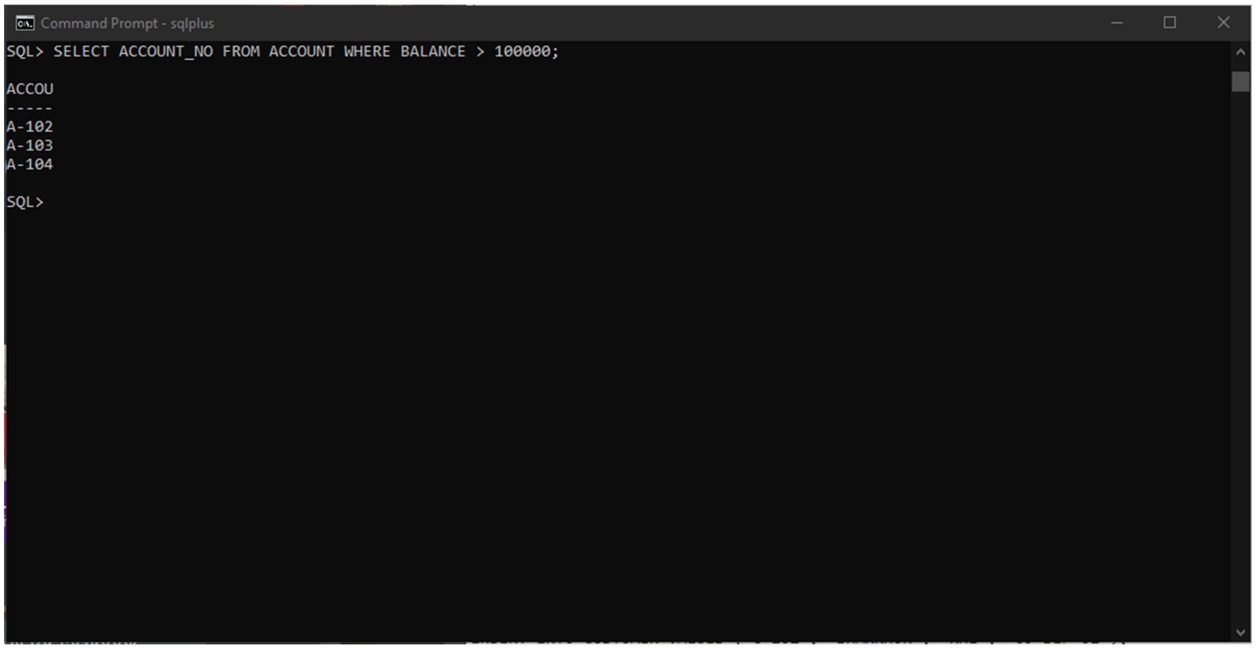
```
SQL> INSERT INTO ACCOUNT VALUES ('A-101', 45320);

1 row created.

SQL> INSERT INTO ACCOUNT VALUES ('A-102', 54230000);

1 row created.

SQL> INSERT INTO ACCOUNT VALUES ('A-103', 200042118);

1 row created.

SQL> INSERT INTO ACCOUNT VALUES ('A-104', 200042150);

1 row created.

SQL> INSERT INTO ACCOUNT VALUES ('A-105', 500);

1 row created.

SQL>
SQL> INSERT INTO CUSTOMER VALUES ('C-101', 'NAZMUL', 'DHK', '16-JUL-00');

1 row created.

SQL> INSERT INTO CUSTOMER VALUES ('C-102', 'SHAKKHOR', 'KHL', '09-SEP-01');

1 row created.

SQL> INSERT INTO CUSTOMER VALUES ('C-103', 'TANZIM', 'DHK', '07-SEP-01');

1 row created.

SQL> INSERT INTO CUSTOMER VALUES ('C-104', 'RIDUN', 'KHL', '19-DEC-01');

1 row created.

SQL> INSERT INTO CUSTOMER VALUES ('C-105', 'AYESHA', 'KHL', '6-APR-00');

1 row created.

SQL>
SQL> INSERT INTO DEPOSITOR_INFO VALUES ('A-101', 'C-101');

1 row created.

SQL> INSERT INTO DEPOSITOR_INFO VALUES ('A-102', 'C-102');

1 row created.

SQL> INSERT INTO DEPOSITOR_INFO VALUES ('A-103', 'C-103');

1 row created.

SQL> INSERT INTO DEPOSITOR_INFO VALUES ('A-104', 'C-104');

1 row created.

SQL>
```

I added these input values later in the lab because the values in previously input wouldn't satisfy all the given tasks.



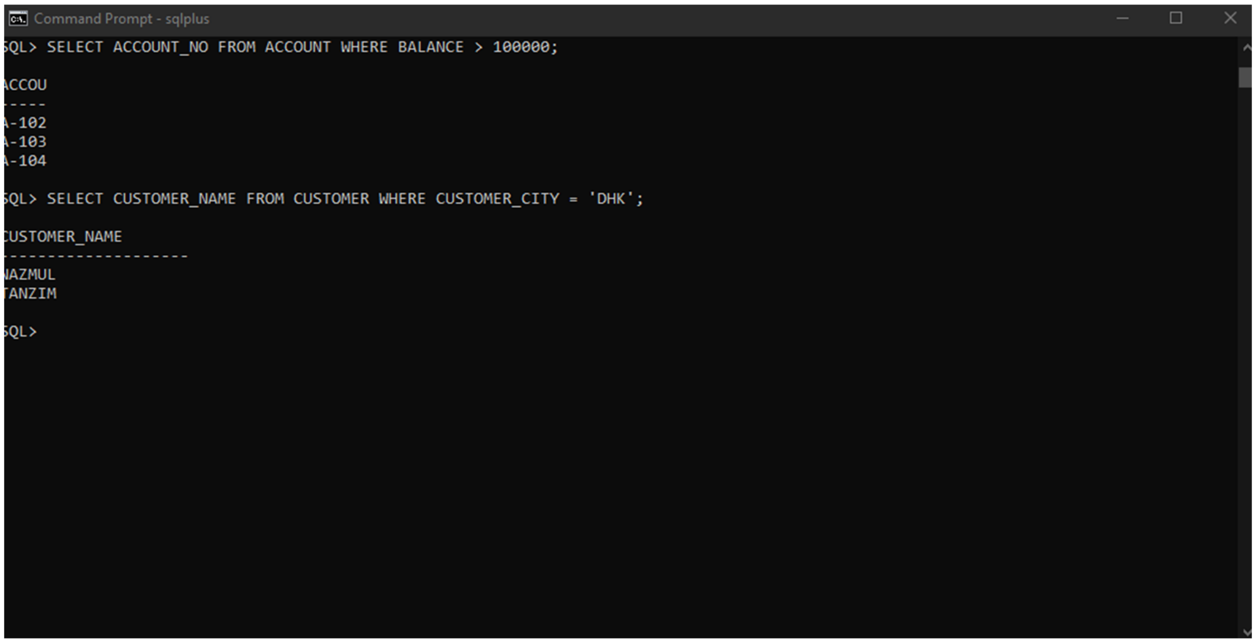
```
SQL> INSERT INTO DEPOSITOR_INFO VALUES ('A-105', 'C-105');

1 row created.

SQL> INSERT INTO CUSTOMER VALUES ('C-106', 'SHANTA', 'DHK', '24-MAR-01');

1 row created.

SQL> INSERT INTO CUSTOMER VALUES ('C-107', 'NAFISA', 'DHK', '6-APR-01');

1 row created.

SQL> INSERT INTO ACCOUNT VALUES ('A-107', 200042113);

1 row created.
SQL> INSERT INTO ACCOUNT VALUES ('A-106', 100000);

1 row created.

SQL> INSERT INTO DEPOSITOR_INFO VALUES ('A-106', 'C-106');

1 row created.

SQL>
```

The first task was, we had to select all the account numbers with a balance greater than 100,000. I used the SELECT statement to display the ACCOUNT_NO and the WHERE statement to state the condition BALANCE > 100,000.



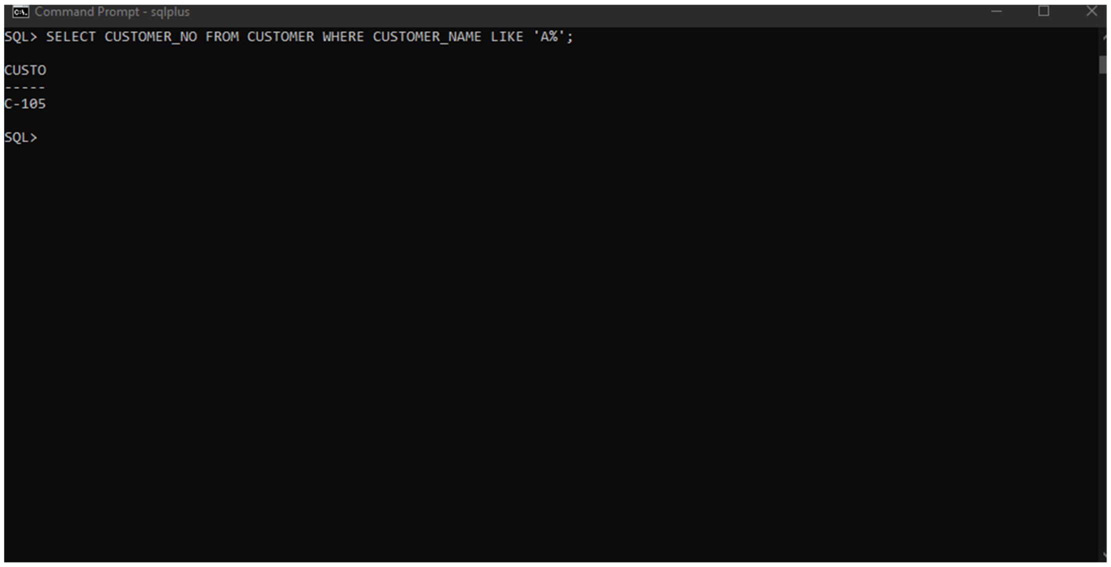
```
SQL> SELECT ACCOUNT_NO FROM ACCOUNT WHERE BALANCE > 100000;

ACCOU
-----
A-102
A-103
A-104

SQL>
```

The second task was to find all the customer names who lived in Dhaka city. I used the SELECT statement to display the CUSTOMER_NAME and the WHERE statement to set the condition of CUSTOMER_CITY is ‘DHK’.



```
SQL> SELECT ACCOUNT_NO FROM ACCOUNT WHERE BALANCE > 100000;

ACCOU
-----
-102
-103
-104

QL> SELECT CUSTOMER_NAME FROM CUSTOMER WHERE CUSTOMER_CITY = 'DHK';

USTOMER_NAME
--------------------
AZMUL
ANZIM

QL>
```

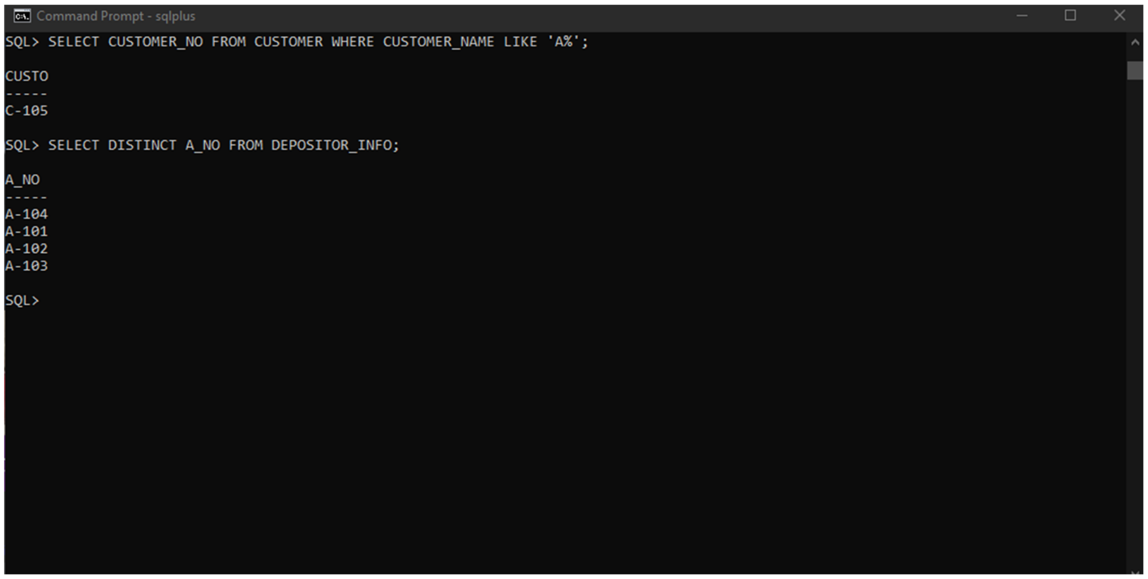
The third task was to find all the customer number whose name started with A. Here I used the SELECT statement to display the CUTOMER_NO and the WHERE statement to choose the records where the CUSTOMER_NAME is LIKE (similar to) ‘A%’(starts with A).



Command Prompt - sqlplus

```
SQL> SELECT CUSTOMER_NO FROM CUSTOMER WHERE CUSTOMER_NAME LIKE 'A%';

CUSTO
-----
C-105

SQL>
```

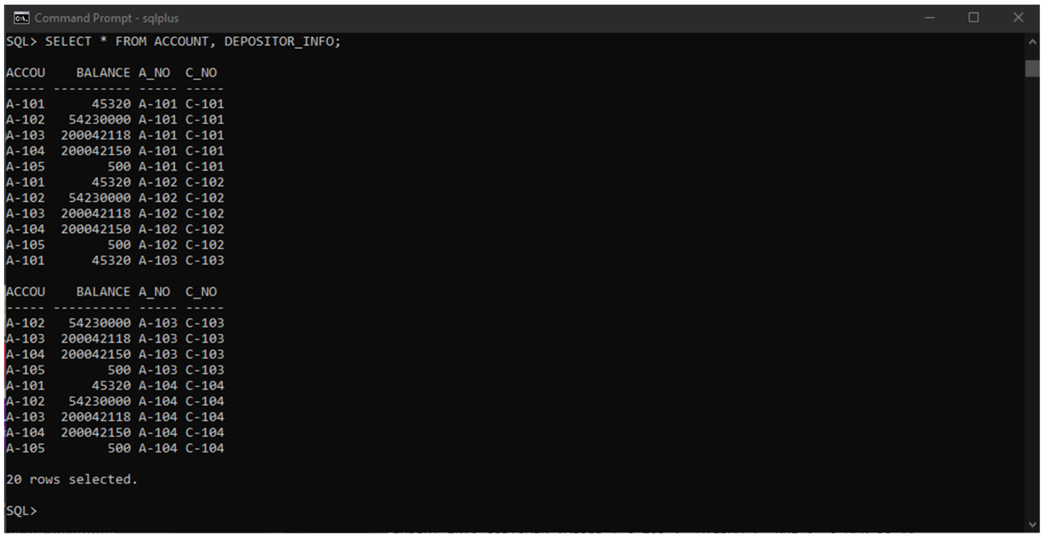
The fourth task was to find all the distinct account numbers from the DEPOSITOR_INFO table. Although this was a bit confusing. Since the A_NO attribute was a primary key, this automatically set the A_NO records to being unique/distinct. Regardless, I used the SELECT DISTINCT A_NO statement to select only unique account no in the DEPOSITOR_INFO table.



Command Prompt - sqlplus

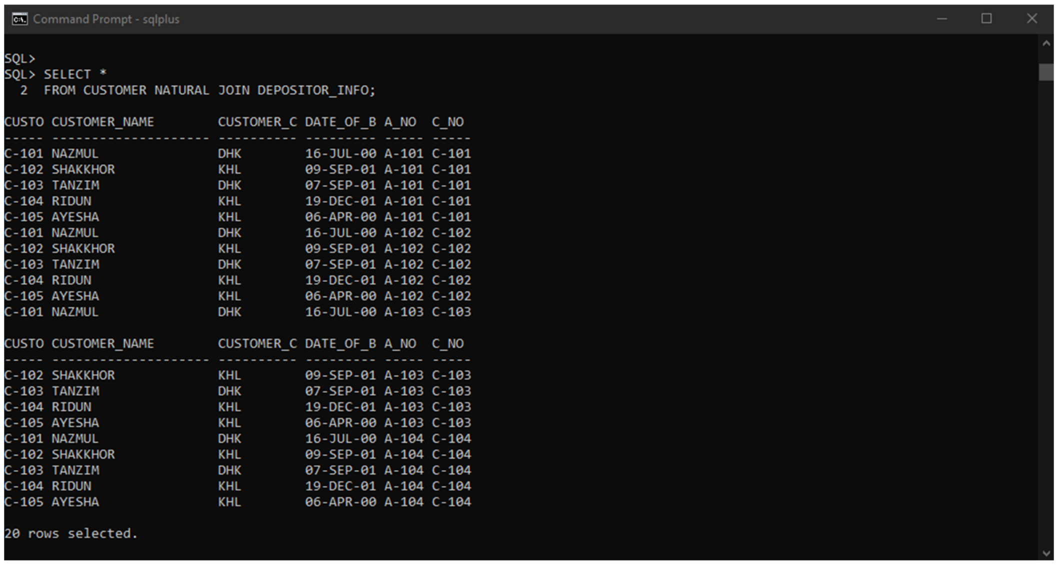
```
SQL> SELECT CUSTOMER_NO FROM CUSTOMER WHERE CUSTOMER_NAME LIKE 'A%';

CUSTO
-----
C-105

SQL> SELECT DISTINCT A_NO FROM DEPOSITOR_INFO;

A_NO
-----
A-104
A-101
A-102
A-103

SQL>
```

The fifth task was to show the cartesian product of the ACOUNT and DEPOSITOR_INFO tables. I used the SELECT * to display all attributes and the FROM ACCOUNT,DEPOSITOR_INFO statement to create the cartesian product table.



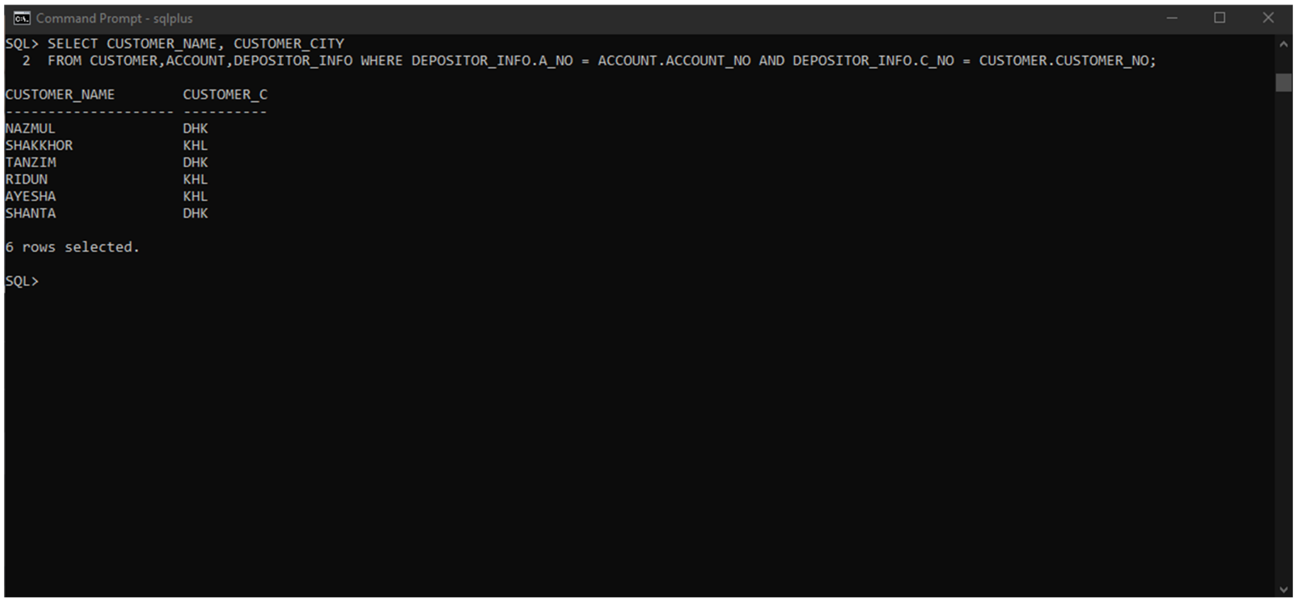
```
SQL> SELECT * FROM ACCOUNT, DEPOSITOR_INFO;

ACCOU    BALANCE A_NO  C_NO
----- ---------- ----- -----
A-101      45320 A-101 C-101
A-102   54230000 A-101 C-101
A-103  200042118 A-101 C-101
A-104  200042150 A-101 C-101
A-105        500 A-101 C-101
A-101      45320 A-102 C-102
A-102   54230000 A-102 C-102
A-103  200042118 A-102 C-102
A-104  200042150 A-102 C-102
A-105        500 A-102 C-102
A-101      45320 A-103 C-103

ACCOU    BALANCE A_NO  C_NO
----- ---------- ----- -----
A-102   54230000 A-103 C-103
A-103  200042118 A-103 C-103
A-104  200042150 A-103 C-103
A-105        500 A-103 C-103
A-101      45320 A-104 C-104
A-102   54230000 A-104 C-104
A-103  200042118 A-104 C-104
A-104  200042150 A-104 C-104
A-105        500 A-104 C-104

20 rows selected.

SQL>
```

The sixth task was to show the natural join between the CUSTOMER and DEPOSITOR_INFO table. I used the SELECT * to display all the attributes and the FROM CUSTOMER NATURAL JOIN DEPOSITOR_INFO statement to create the natural join table.



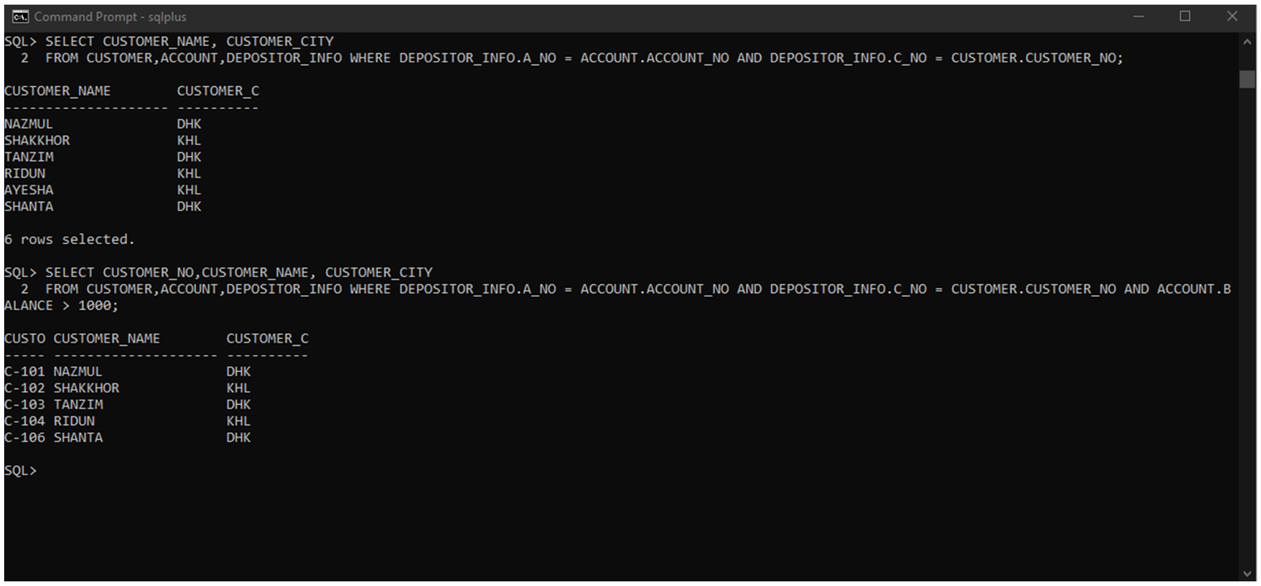
```
SQL>
SQL> SELECT *
  2  FROM CUSTOMER NATURAL JOIN DEPOSITOR_INFO;

CUSTO CUSTOMER_NAME        CUSTOMER_C DATE_OF_B A_NO  C_NO
----- -------------------- ---------- --------- ----- -----
C-101 NAZMUL               DHK        16-JUL-00 A-101 C-101
C-102 SHAKKHOR             KHL        09-SEP-01 A-101 C-101
C-103 TANZIM               DHK        07-SEP-01 A-101 C-101
C-104 RIDUN                KHL        19-DEC-01 A-101 C-101
C-105 AYESHA               KHL        06-APR-00 A-101 C-101
C-101 NAZMUL               DHK        16-JUL-00 A-102 C-102
C-102 SHAKKHOR             KHL        09-SEP-01 A-102 C-102
C-103 TANZIM               DHK        07-SEP-01 A-102 C-102
C-104 RIDUN                KHL        19-DEC-01 A-102 C-102
C-105 AYESHA               KHL        06-APR-00 A-102 C-102
C-101 NAZMUL               DHK        16-JUL-00 A-103 C-103

CUSTO CUSTOMER_NAME        CUSTOMER_C DATE_OF_B A_NO  C_NO
----- -------------------- ---------- --------- ----- -----
C-102 SHAKKHOR             KHL        09-SEP-01 A-103 C-103
C-103 TANZIM               DHK        07-SEP-01 A-103 C-103
C-104 RIDUN                KHL        19-DEC-01 A-103 C-103
C-105 AYESHA               KHL        06-APR-00 A-103 C-103
C-101 NAZMUL               DHK        16-JUL-00 A-104 C-104
C-102 SHAKKHOR             KHL        09-SEP-01 A-104 C-104
C-103 TANZIM               DHK        07-SEP-01 A-104 C-104
C-104 RIDUN                KHL        19-DEC-01 A-104 C-104
C-105 AYESHA               KHL        06-APR-00 A-104 C-104

20 rows selected.
```

Then we had to find the names and cities of the customers who had an account. I created a cartesian product of all three tables. I implemented a WHERE condition which selected only the values where the ACCOUNT_NO attribute of ACCOUNT table and the A_NO attribute of the DEPISITOR_INFO table match and the CUSTOMER_NO attribute of CUSTOMER table and the C_NO attribute of the DEPOSITOR_INFO table match. This eliminates all the illogical records created by the cartesian product. Then I used the SELECT statement to display only the CUSTOMER NAME and the CUSTOMER_CITY.



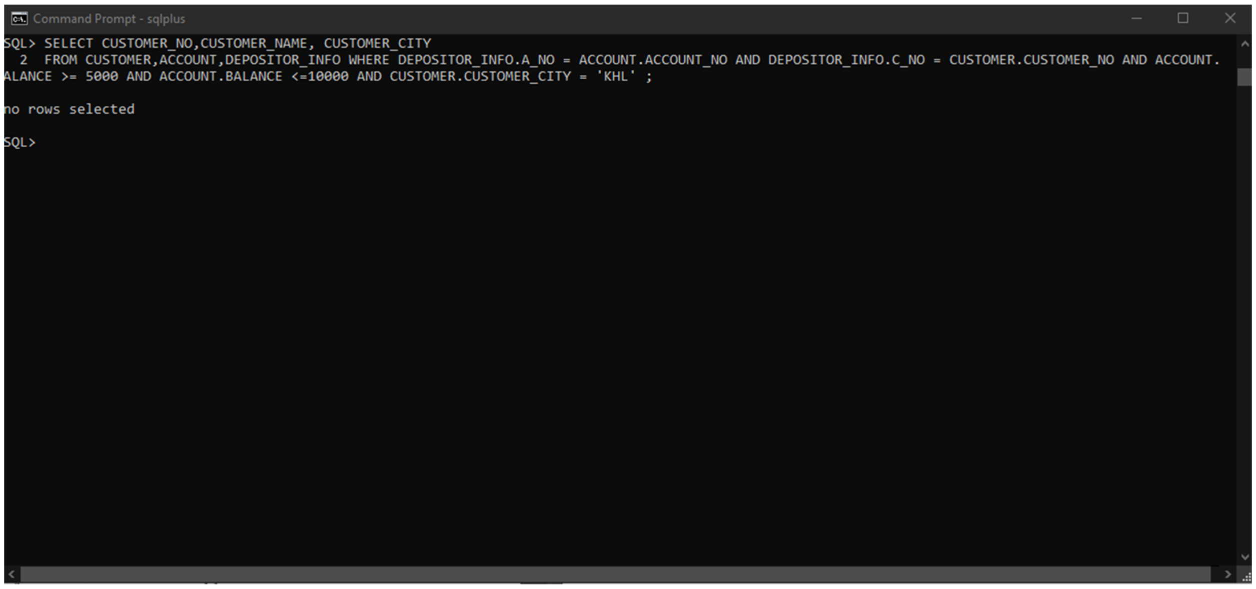
```
SQL> SELECT CUSTOMER_NAME, CUSTOMER_CITY
  2  FROM CUSTOMER,ACCOUNT,DEPOSITOR_INFO WHERE DEPOSITOR_INFO.A_NO = ACCOUNT.ACCOUNT_NO AND DEPOSITOR_INFO.C_NO = CUSTOMER.CUSTOMER_NO;

CUSTOMER_NAME        CUSTOMER_C
-------------------- ----------
NAZMUL               DHK
SHAKKHOR             KHL
TANZIM               DHK
RIDUN                KHL
AYESHA               KHL
SHANTA               DHK

6 rows selected.

SQL>
```

Then we were asked to find all the customer related information who have a balance greater than 1000. I created a cartesian product of all three tables. I implemented a WHERE condition which selected only the values where the ACCOUNT_NO attribute of ACCOUNT table and the A_NO attribute of the DEPISITOR_INFO table match and the CUSTOMER_NO attribute of CUSTOMER table and the C_NO attribute of the DEPOSITOR_INFO table match. This eliminates all the illogical records created by the cartesian product. Then I used the SELECT statement to display only the CUSTOMER_NO, CUSTOMER NAME and the CUSTOMER_CITY. And I added and additional WHERE condition which required the BALANCE to be greater than 10,000.



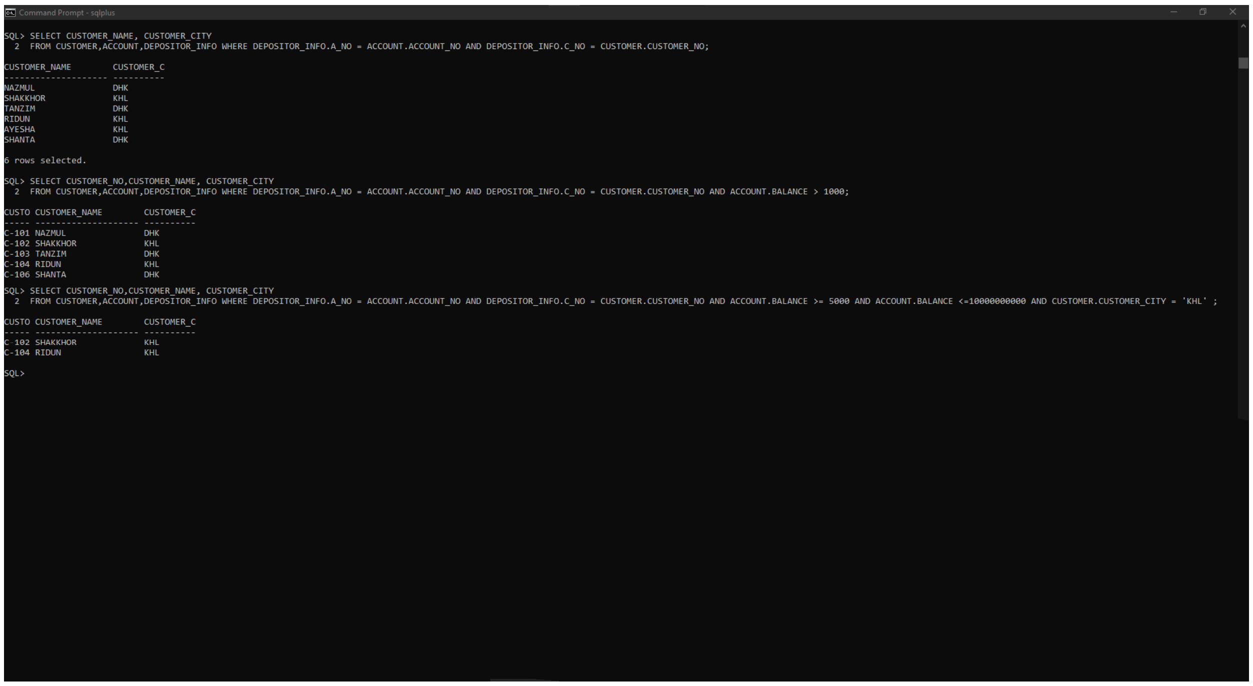
```
SQL> SELECT CUSTOMER_NAME, CUSTOMER_CITY
  2  FROM CUSTOMER,ACCOUNT,DEPOSITOR_INFO WHERE DEPOSITOR_INFO.A_NO = ACCOUNT.ACCOUNT_NO AND DEPOSITOR_INFO.C_NO = CUSTOMER.CUSTOMER_NO;

CUSTOMER_NAME        CUSTOMER_C
-------------------- ----------
NAZMUL               DHK
SHAKKHOR             KHL
TANZIM               DHK
RIDUN                KHL
AYESHA               KHL
SHANTA               DHK

6 rows selected.

SQL> SELECT CUSTOMER_NO,CUSTOMER_NAME, CUSTOMER_CITY
  2  FROM CUSTOMER,ACCOUNT,DEPOSITOR_INFO WHERE DEPOSITOR_INFO.A_NO = ACCOUNT.ACCOUNT_NO AND DEPOSITOR_INFO.C_NO = CUSTOMER.CUSTOMER_NO AND ACCOUNT.B
ALANCE > 1000;

CUSTO CUSTOMER_NAME        CUSTOMER_C
----- -------------------- ----------
C-101 NAZMUL               DHK
C-102 SHAKKHOR             KHL
C-103 TANZIM               DHK
C-104 RIDUN                KHL
C-106 SHANTA               DHK

SQL>
```

Finally, we were asked to show all the customer related information who had a balance between 5000 and 10000 and their depositor lived in Khulna city. . I created a cartesian product of all three tables. I implemented a WHERE condition which selected only the values where the ACCOUNT_NO attribute of ACCOUNT table and the A_NO attribute of the DEPISITOR_INFO table match and the CUSTOMER_NO attribute of CUSTOMER table and the C_NO attribute of the DEPOSITOR_INFO table match. This eliminates all the illogical records created by the cartesian product. Then I used the SELECT statement to display only the CUSTOMER_NO, CUSTOMER NAME and the CUSTOMER_CITY. And I added and additional WHERE condition which required the BALANCE to be between 5000 and 10000 and the CUSTOMER_CITY had to be ‘KHL’.



```
SQL> SELECT CUSTOMER_NO,CUSTOMER_NAME, CUSTOMER_CITY
  2  FROM CUSTOMER,ACCOUNT,DEPOSITOR_INFO WHERE DEPOSITOR_INFO.A_NO = ACCOUNT.ACCOUNT_NO AND DEPOSITOR_INFO.C_NO = CUSTOMER.CUSTOMER_NO AND ACCOUNT.
ALANCE >= 5000 AND ACCOUNT.BALANCE <=10000 AND CUSTOMER.CUSTOMER_CITY = 'KHL' ;

no rows selected

SQL>
```

This however, came with the drawback where, none of my input values matched this requirement, so no records were displayed. I increased the limit of the BALANCE to 10000000000 and then, can see that, my code does indeed work.



```
SQL> SELECT CUSTOMER_NAME, CUSTOMER_CITY
  2  FROM CUSTOMER,ACCOUNT,DEPOSITOR_INFO WHERE DEPOSITOR_INFO.A_NO = ACCOUNT.ACCOUNT_NO AND DEPOSITOR_INFO.C_NO = CUSTOMER.CUSTOMER_NO;

CUSTOMER_NAME        CUSTOMER_C
-------------------- ----------
NAZMUL               DHK
SHAKKHOR             KHL
TANZIM               DHK
RIDUN                KHL
AYESHA               KHL
SHANTA               DHK

6 rows selected.

SQL> SELECT CUSTOMER_NO,CUSTOMER_NAME, CUSTOMER_CITY
  2  FROM CUSTOMER,ACCOUNT,DEPOSITOR_INFO WHERE DEPOSITOR_INFO.A_NO = ACCOUNT.ACCOUNT_NO AND DEPOSITOR_INFO.C_NO = CUSTOMER.CUSTOMER_NO AND ACCOUNT.BALANCE > 1000;

CUSTO CUSTOMER_NAME        CUSTOMER_C
----- -------------------- ----------
C-101 NAZMUL               DHK
C-102 SHAKKHOR             KHL
C-103 TANZIM               DHK
C-104 RIDUN                KHL
C-106 SHANTA               DHK

SQL> SELECT CUSTOMER_NO,CUSTOMER_NAME, CUSTOMER_CITY
  2  FROM CUSTOMER,ACCOUNT,DEPOSITOR_INFO WHERE DEPOSITOR_INFO.A_NO = ACCOUNT.ACCOUNT_NO AND DEPOSITOR_INFO.C_NO = CUSTOMER.CUSTOMER_NO AND ACCOUNT.BALANCE >= 5000 AND ACCOUNT.BALANCE <=10000000000 AND CUSTOMER.CUSTOMER_CITY = 'KHL' ;

CUSTO CUSTOMER_NAME        CUSTOMER_C
----- -------------------- ----------
C-102 SHAKKHOR             KHL
C-104 RIDUN                KHL

SQL>
```

The Code

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

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

CREATE TABLE DEPOSITOR
(
ACCOUNT_NO CHAR (5),
```

```sql
CUSTOMER_NO CHAR (5),
CONSTRAINT PK_ACCOUNT_NO_CUSTOMER_NO PRIMARY KEY (ACCOUNT_NO, CUSTOMER_NO)
);

ALTER TABLE CUSTOMER ADD DATE_OF_BIRTH DATE;

ALTER TABLE ACCOUNT MODIFY BALANCE NUMBER(12,2);

ALTER TABLE DEPOSITOR RENAME COLUMN ACCOUNT_NO TO A_NO;

ALTER TABLE DEPOSITOR RENAME COLUMN CUSTOMER_NO TO C_NO;

ALTER TABLE DEPOSITOR RENAME TO DEPOSITOR_INFO;

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);

INSERT INTO ACCOUNT VALUES ('A-101', 45320);
INSERT INTO ACCOUNT VALUES ('A-102', 54230000);
INSERT INTO ACCOUNT VALUES ('A-103', 200042118);
INSERT INTO ACCOUNT VALUES ('A-104', 200042150);
INSERT INTO ACCOUNT VALUES ('A-105', 500);
INSERT INTO ACCOUNT VALUES ('A-106', 10000);
INSERT INTO ACCOUNT VALUES ('A-107', 200042113);

INSERT INTO CUSTOMER VALUES ('C-101', 'NAZMUL', 'DHK', '16-JUL-00');
INSERT INTO CUSTOMER VALUES ('C-102', 'SHAKKHOR', 'KHL', '09-SEP-01');
INSERT INTO CUSTOMER VALUES ('C-103', 'TANZIM', 'DHK', '07-SEP-01');
INSERT INTO CUSTOMER VALUES ('C-104', 'RIDUN', 'KHL', '19-DEC-01');
INSERT INTO CUSTOMER VALUES ('C-105', 'AYESHA', 'KHL', '6-APR-00');
INSERT INTO CUSTOMER VALUES ('C-106', 'SHANTA', 'DHK', '24-MAR-01');
INSERT INTO CUSTOMER VALUES ('C-107', 'NAFISA', 'DHK', '6-APR-01');

INSERT INTO DEPOSITOR_INFO VALUES ('A-101', 'C-101');
INSERT INTO DEPOSITOR_INFO VALUES ('A-102', 'C-102');
INSERT INTO DEPOSITOR_INFO VALUES ('A-103', 'C-103');
INSERT INTO DEPOSITOR_INFO VALUES ('A-104', 'C-104');
INSERT INTO DEPOSITOR_INFO VALUES ('A-105', 'C-105');
INSERT INTO DEPOSITOR_INFO VALUES ('A-106', 'C-106');
```

```sql
SELECT ACCOUNT_NO FROM ACCOUNT WHERE BALANCE > 100000;

SELECT CUSTOMER_NAME FROM CUSTOMER WHERE CUSTOMER_CITY = 'DHK';

SELECT CUSTOMER_NO FROM CUSTOMER WHERE CUSTOMER_NAME LIKE 'A%';

SELECT DISTINCT A_NO FROM DEPOSITOR_INFO;

SELECT * FROM ACCOUNT, DEPOSITOR_INFO;

SELECT *
FROM CUSTOMER NATURAL JOIN DEPOSITOR_INFO;

SELECT CUSTOMER_NAME FROM DEPOSITOR_INFO,CUSTOMER WHERE DEPOSITOR_INFO.C_NO =
CUSTOMER.CUSTOMER_NO;


SELECT CUSTOMER_NAME, CUSTOMER_CITY
FROM CUSTOMER,ACCOUNT,DEPOSITOR_INFO WHERE DEPOSITOR_INFO.A_NO =
ACCOUNT.ACCOUNT_NO AND DEPOSITOR_INFO.C_NO = CUSTOMER.CUSTOMER_NO;

SELECT CUSTOMER_NO,CUSTOMER_NAME, CUSTOMER_CITY
FROM CUSTOMER,ACCOUNT,DEPOSITOR_INFO WHERE DEPOSITOR_INFO.A_NO =
ACCOUNT.ACCOUNT_NO AND DEPOSITOR_INFO.C_NO = CUSTOMER.CUSTOMER_NO AND
ACCOUNT.BALANCE > 1000;

SELECT CUSTOMER_NO,CUSTOMER_NAME, CUSTOMER_CITY
FROM CUSTOMER,ACCOUNT,DEPOSITOR_INFO WHERE DEPOSITOR_INFO.A_NO =
ACCOUNT.ACCOUNT_NO AND DEPOSITOR_INFO.C_NO = CUSTOMER.CUSTOMER_NO AND
ACCOUNT.BALANCE >= 5000 AND ACCOUNT.BALANCE <=10000 AND CUSTOMER.CUSTOMER_CITY =
'KHL' ;
```

Problems

The third task of the Lab was where we had to rack our brains. The instructions about which type of input we had to put in weren't clear enough. So I spent half the Lab thinking about the different types of inputs that could be possible. The DEPOSITOR_INFO table. I had no idea what type of input should be in the DEPOSITOR_INFO table whatsoever. After figuring out which input values I should put in there, I could finally begin, but it took too long.

Conclusion

With that, I had successfully completed another successful Lab. All in all, it was a pleasant introduction to DDL and DML. I was able to learn a lot about dropping and adding constraints. And the priority order of dropping values. This developed my intrigue towards Database even more than ever before.