CSE – 302
Database Management System Sessional

SQL
(STRUCTURED QUERY LANGUAGE)

Database

- A database is a collection of information, that is organized so that it can be accessed, managed, and updated easily

Database

ID	NAME	DIVISION
201414020	Md. Jahidul Islam	Dhaka
201414021	Sanjida Akter Sharna	Khulna
201414022	Suzzana Rafi	Chittagong
201414023	Iffat Tamanna	Barishal
201414106	Arnab Debnath	Dhaka
201414107	S. M. Faisal Rahman	Dhaka
201414108	Sazid Al Ahsan	Barishal

Table 1

DIVISION	DESCRIPTION
Dhaka	Dhaka, set beside the Buriganga River, is the capital of Bangladesh. It’s a hub for trade and culture, with a long history as a seat of government.
Khulna	Khulna is the third-largest city in Bangladesh. It is the administrative seat of Khulna District and Khulna Division
Chittagong	Chittagong is a major coastal seaport city and financial centre in southeastern Bangladesh.
Barisal	Barisal is a major city that lies on the bank of Kirtankhola river in south-central Bangladesh.

Table 2

Steps

- Create table
  - Specify columns
  - Specify column datatypes
- Insert data
- Display data
- Modify data
- Delete data

Character Datatypes 1/2

Datatype	Description
VARCHAR2 (size)	Variable-length character string having maximum length ***size***. Maximum size is 4000 bytes or characters, and minimum is 1 byte or 1 character. **You must specify size for VARCHAR2.**
NVARCHAR2 (size)	Variable-length Unicode character string having maximum length ***size*** characters. The NVARCHAR2 datatype was introduced by Oracle for databases that want to use Unicode for some columns while keeping another character set for the rest of the database (which uses VARCHAR2). The NVARCHAR2 is a **Unicode-only** datatype.

Character Datatypes 2/2

Datatype	Description
CHAR2 (size)	Fixed-length character data of length ***size*** bytes or characters. Maximum size is 2000 bytes or characters. Default and minimum size is 1 byte.
NCHAR2 (size)	Fixed-length character data of length ***size*** bytes or characters. Maximum size is 2000 bytes or characters. Default and minimum size is 1 byte. **The only difference is, nchar store Unicode characters**

Sum Up (nchar, nvarchar, char, varchar)

- nchar and nvarchar can store Unicode characters
- char and varchar cannot store Unicode characters
- char and nchar are fixed-length which will reserve storage space for number of characters you specify, even if you don't use up all that space
- varchar and nvarchar are variable-length which will only use up spaces for the characters you store. It will not reserve storage like char or nchar

Numeric Datatypes

Datatype	Description
NUMBER (precision, scale)	**Precision** is the total number of digits and **Scale** is the number of digits to the right (positive) or left (negative) of the decimal point. Precision can range from 1 to 38. Scale can range from -84 to 127. For example, number(7,2) is a number that has 5 digits before the decimal and 2 digits after the decimal.
Float, Decimal	This datatypes are subclasses of Number datatype

Example:
Precision 4, scale 2: 99.99
Precision 10, scale 0: 9999999999
Precision 8, scale 3: 99999.999
Precision 5, scale -3: 99999000

Date/Time Datatypes

Datatype	Description
Date	Use the DATE data type to store point-in-time values (dates and times) in a table. The DATE data type stores the century, year, month, day, hours, minutes, and seconds. Valid date range from January 1, 4712 BC to December 31, 9999 AD.

- Follow the link Oracle Built-in Datatypes for more datatypes of Oracle database.

Table Design

- Before creating a table, the user should examine what type of data it will contain
- The actual data values to be stored in the table to determine the data type and width to be assigned to each column

When You Create a Table (1/3)

- The table must be assigned a unique name
- The name of a table can be no longer than **30** characters
- At least one column must be defined
- The columns within each table must be unique
- Each column within the table must be assigned a column name and a data type.
- The name of a column can be no longer than 30 characters

When You Create a Table (2/3)

- The **underscore symbol (_) and the number sign (#)** are allowed in table and column names
- Data type specifies what type of data will be stored in that column
- The width of the column can also be stated
- A table can be created based on data retrieved through a subquery

When You Create a Table (3/3)

- Once a table has been created, the structure of the table can be changed using the ALTER TABLE command with the appropriate clause
- To change the name of an existing table, the RENAME command is used

Keep in Mind While Defining Columns

- A table can have a maximum of 1,000 columns

- The column list must be enclosed within parentheses

- For each column, specify the name, datatype (including the width, if necessary)

Design a Table: Customer (1/5)

Cust_id				

<u>Column 1</u>

Purpose: To uniquely identify each Customer

Field name: **Cust_id**

Datatype: VARCHAR2 (because column will consist of both letters and numbers)

Width: 12

Design a Table: Customer (2/5)

Cust_id	Cust_name			

<u>Column 2</u>

Purpose: To store the first and last name of each Customer

Field name: Cust_name

Contents: text data

Datatype: VARCHAR2

Width: 20 (20 characters probably enough; can easily increase size if necessary)

Design a Table: Customer (3/5)

Cust_id	Cust_name	Cust_dob		

Column 3

Purpose: To store the date of birth of each Customer

Field name: Cust_dob

Contents: Date of Birth

Datatype: DATE

Width: (automatically handled by Oracle)

Design a Table: Customer (4/5)

Cust_id	Cust_name	Cust_dob	Cust_street	

Column 4

Purpose: To store the street address of each Customer

Field name: Cust_street

Contents: Street Address

Datatype: VARCHAR2

Width: 12

Design a Table: Customer (5/5)

Cust_id	Cust_name	Cust_dob	Cust_street	Cust_city

<u>Column 5</u>

Purpose: To store the CITY of each Customer

Field name: Cust_city

Contents: City of Customers

Datatype: VARCHAR2

Width: 12

Create Table

Basic syntax:

CREATE TABLE *table_name*

(*column_name1* datatype [DEFAULT value],

column_name2 datatype [DEFAULT value],

column_name3 datatype [DEFAULT value], …);

Create a Table *Customer*

```
Create table Customer
(
Cust_id VARCHAR2(12) NOT NULL,
Cust_name VARCHAR2(20),
Cust_dob DATE,
Cust_street VARCHAR2(12),
Cust_city VARCHAR2(12) DEFAULT 'DHAKA'
);
```

Insert Data

Form of INSERT Command

- Single-Row Insert
- Multi Row Insert

Single Row INSERT Command (1/4)

Basic Syntax:

INSERT INTO *table_name* (*column1*,*column2*,*column3*,...) VALUES (*value1*,*value2*,*value3*,...);

Table Name: *Customer*

Cust_id	Cust_name	Cust_dob	Cust_street	Cust_city
C00000000001	C_A	11-JAN-1982	c_street_006	c_city_001

Single Row INSERT Command (2/4)

INSERT INTO *CUSTOMER*

(*Cust_id*, *Cust_name*, *Cust_dob*, *Cust_street*, *Cust_city*) VALUES

(*'C00000000001'*, *'C_A'*, *'11-JAN-1982'*, *'c_street_006'*, *'c_city_001'*);

Executing this one will see the message:

"1 row created"

Single Row INSERT Command (3/4)

INSERT INTO *CUSTOMER*

(*Cust_id*, *Cust_name*, *Cust_city*) VALUES

(*'C00000000002'*, *'C_B'*, *'c_city_002'*);

ERROR????

Single Row INSERT Command (3/4)

```
INSERT INTO CUSTOMER
(Cust_id, Cust_name, Cust_city) VALUES
('C00000000002', 'C_B', 'c_city_002');
```

Executing this one will see the message:

"1 row created"

Single Row INSERT Command (4/4)

- Any missing values will be NULL, unless a DEFAULT value is provided in the table definition
- Column list is optional

Multi-Row INSERT Command

- Basic Syntax

  Uses a sub query allowing zero, one or more rows to insert

Example:

Create a new table named *NEW_CUSTOMER* using similar columns of table *CUSTOMER*

Multi-Row INSERT Command

- INSERT INTO *NEW_CUSTOMER*
  SELECT * FROM *CUSTOMER*;

Multi-Row INSERT Command

- INSERT INTO *NEW_CUSTOMER*
  SELECT * FROM *CUSTOMER*;

Now Write,

SELECT * FROM *NEW_CUSTOMER*

Cust_id	Cust_name	Cust_dob	Cust_street	Cust_city
C00000000001	C_A	11-JAN-1982	c_street_006	c_city_001
C00000000002	C_B			c_city_002

Problems!

- What'll we do if any one inserts wrong data by mistake?

- How can we insert the missing data of row2?

Problems!

- What'll we do if any one inserts wrong data by mistake?
- How can we insert the missing data of row2?

UPDATE Command!!!

UPDATE

- Use UPDATE command to -
  - Change existing values
  - Add values to an existing row

Basic Syntax:

UPDATE *tablename*

SET *columnname* = *newvalue*

[WHERE condition];

UPDATE

- UPDATE clause identifies the table
- SET clause identifies the column(s) being changed and new value(s)
- Optional WHERE clause specifies row(s) to be changed;
  if omitted, will update all rows

Example:

UPDATE *CUSTOMER*

SET *Cust_name* = ‘Suzzana Rafi’

Where *Cust_id* = ‘C00000000001’ ;

Adding Values to an Existing Row

- Add street address 'c_street_002' to the customer who has a customer ID C00000000002

Cust_id	Cust_name	Cust_dob	Cust_street	Cust_city
C00000000001	C_A	11-JAN-1982	c_street_006	c_city_001
C00000000002	C_B		c_street_002	c_city_002

Changing Existing Values

- Change street address 'c_street_006' to 'c_street_007'

Cust_id	Cust_name	Cust_dob	Cust_street	Cust_city
C00000000001	C_A	11-JAN-1982	c_street_007	c_city_001
C00000000002	C_B		c_street_002	c_city_002

Changing Existing Values

- Change the street address of the customer who has a customer ID 'C00000000001' to 'c_street_001'

Cust_id	Cust_name	Cust_dob	Cust_street	Cust_city
C00000000001	C_A	11-JAN-1982	c_street_001	c_city_001
C00000000002	C_B		c_street_002	c_city_002

Problems!

- How can we show the inserted data from the database?

Problems!

- How can we show the inserted data from the database?

SELECT Command!!!

- SELECT statements are used to retrieve data from the database

SELECT Statement

Basic Syntax:

- Every SELECT statement is required to have a SELECT and FROM clause. A clause always begins with a keyword.

  - ➢ The SELECT clause is used to identify the column or columns to be retrieved from a table

  - ➢ The name of the table is identified in the FROM clause.

SELECT Statement

- Select all of the data (i.e., all rows and columns) in a table

  SELECT * FROM CUSTOMER;

- Select cust_name column from the Customer table
- Select cust_id, cust_name, cust_city columns from the Customer table

DELETE

- DELETE command removes entire rows from a table
- It is not permanent until a **COMMIT** command is issued
- Cannot be applied for specific column
- If no WHERE clause is specified, all rows will be deleted

DELETE ROWS

Basic Syntax

DELETE FROM *tablename*
[WHERE condition];

Example:
DELETE FROM *CUSTOMER*;

DELETE FROM *CUSTOMER* WHERE Cust_id = ‘C00000000001’

DELETE COLUMN VALUES**

- You **cannot** directly delete all **values** from a particular column without deleting that column

- But you can use **UPDATE** query to make a column empty –

UPDATE *Table_name* SET *Column_name*=NULL

Example: UPDATE *Customer* SET *Cust_city*=NULL

**Added Later

DELETE TABLE

- To delete the whole table

  DROP TABLE *table_name*;

Example:

DROP TABLE **NEW_CUSTOMER**;

COMMIT

- Use the COMMIT statement to end a current transaction and make **permanent** all changes performed in the transaction

- You can see any changes you have made during the transaction by querying the modified tables, but other users cannot see the changes. After you commit the transaction, the changes are visible to other users' statements that execute after the commit

COMMIT

- Types of COMMIT
  - Explicit COMMIT:
    When you type COMMIT; at the SQL prompt;
  - Implicit COMMIT:
    At the end of the SQL session by typing EXIT;

Practice 1

1. Create an *EMPLOYEE* table which consists
   - Employee_id, datatype VARCHAR2, size 20
   - Employee_name, datatype VARCHAR2, size 20
   - Employee_dob, datatype DATE
   - Employee_street, datatype VARCHAR2, size 20
   - Employee_city, datatype VARCHAR2, size 20
   - Employee_startdate, datatype DATE

Practice 1

2. Insert the following data into the EMPLOYEE table

E_id	E_name	E_dob	E_street	E_city	E_startdate
E01	Pious	11-JAN-1996	e_s_001	Dhaka	1-JAN-2012
E02	Sadia	06-FEB-1996	e_s_002	Khulna	1-JAN-2013
E03	Ashraf	08-MAR-1996	e_s_003	Dhaka	1-JAN-2014
E04	Rana	1-JUN-1996	e_s_004	Dhaka	1-JAN-2012
E05	Shovon	1-JAN-1996	e_s_005	Barisal	1-JAN-2013
E06	Iffat	1-DEC-1996	e_s_006	Khulna	1-JAN-2015

Practice 1

3. Display all of the records in the *EMPLOYEE* table

4. Display the employee id and employee city names for all records in the *EMPLOYEE* Table

5. Display the name, living street, and date of birth of the employees for all records in the *EMPLOYEE* table.

Operations within SELECT Statement

- Column alias can be used for column headings
- Suppress duplicates
- Concatenate data

Case - 01

- Customized Column Name

Employee ID	Name	Date of Birth	Street	City	Job Startdate
E01	Pious	11-JAN-1996	e_s_001	Dhaka	1-JAN-2012
E02	Sadia	06-FEB-1996	e_s_002	Khulna	1-JAN-2013
E03	Ashraf	08-MAR-1996	e_s_003	Dhaka	1-JAN-2014
E04	Rana	1-JUN-1996	e_s_004	Dhaka	1-JAN-2012
E05	Shovon	1-JAN-1996	e_s_005	Barisal	1-JAN-2013
E06	Iffat	1-DEC-1996	e_s_006	Khulna	1-JAN-2015

Column Alias (1/2)

- To display customized column name
- Optional use of the keyword **AS**

Column Alias (2/2)

- Basic Syntax:

  SELECT *column_name* "*The name you want to display*" FROM *table_name*

  OR

  SELECT *column_name* **AS** "*The name you want to display*" FROM *table_name*

Case - 02

- Name the cities where employees live.

Employee City
Dhaka
Khulna
Dhaka
Dhaka
Barisal
Khulna

There are unnecessary duplicate values!!!!

Suppressing Duplicates

- To suppress duplicate values, enter ***DISTINCT*** or ***UNIQUE*** after the SELECT keyword

- SELECT ***DISTINCT*** employee_city FROM EMPLOYEE;

Case - 03

- Sadia has a Employee ID e_01.

Concatenation

- Can combine data with a string literal
- Use the concatenation operator ||
- Allows use of column aliasing

Concatenation

- Basic Syntax

  SELECT *Column_Name* || ' *The string you want to concate* ' || *Column_Name* FROM *table_name*;

Example:

SELECT *Employee_name* || ' has Employee ID ' || *Employee_id* FROM *EMPLOYEE*;

Concatenation AND Aliasing

- SELECT *Employee_name* || ' has Employee ID ' || *Employee_id* " Employee Name AND Employee ID " FROM *Employee* ;

$$\downarrow$$

$\boxed{\textbf{ALIASING}}$

Date Format

- To show any DATE in a specific format –

```
SELECT to_char(Employee_dob, 'mm/dd/yyyy') as "Date of Birth" FROM Employee;
```

TO_DATE() Function

- TO_DATE converts *char* of CHAR, VARCHAR2, NCHAR, or NVARCHAR2 data type to a value of DATE data type

- Syntax:

```
SELECT TO_DATE('1999-JAN-05', 'YYYY-
        MON-DD') FROM Customer;
```

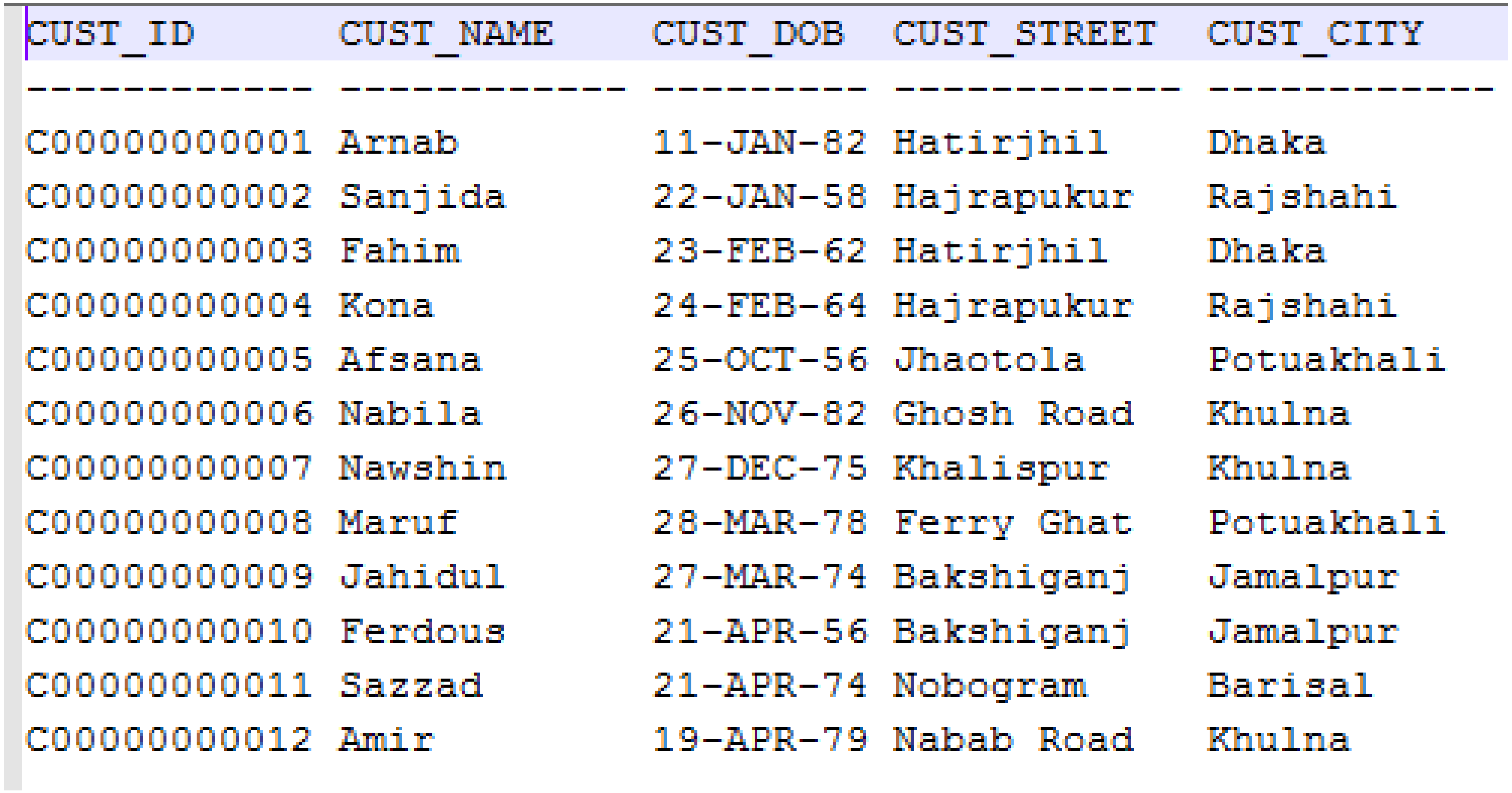
Practice 02

1. Display the Employee_name and Employee_id for all of the records in the Employee table. There should be a column alias “Name of the Employee” for the Employee_name.

2. Display a list of unique Employee_city records within the Employee table

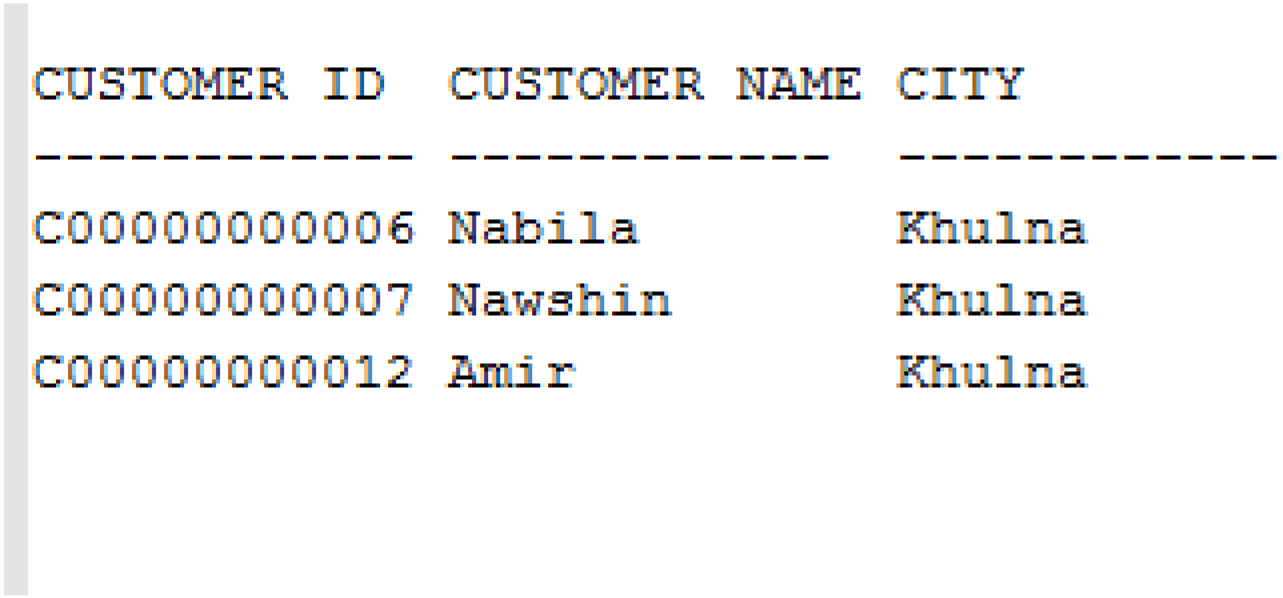
3. Display the concatenation of the Employee_name, “lives in” and Employee_city for all of the records in the Employee table.

Customer table



```
CUST_ID      CUST_NAME    CUST_DOB  CUST_STREET  CUST_CITY
------------ ------------ --------- ------------ ------------
C00000000001 Arnab        11-JAN-82 Hatirjhil    Dhaka
C00000000002 Sanjida      22-JAN-58 Hajrapukur   Rajshahi
C00000000003 Fahim        23-FEB-62 Hatirjhil    Dhaka
C00000000004 Kona         24-FEB-64 Hajrapukur   Rajshahi
C00000000005 Afsana       25-OCT-56 Jhaotola     Potuakhali
C00000000006 Nabila       26-NOV-82 Ghosh Road   Khulna
C00000000007 Nawshin      27-DEC-75 Khalispur    Khulna
C00000000008 Maruf        28-MAR-78 Ferry Ghat   Potuakhali
C00000000009 Jahidul      27-MAR-74 Bakshiganj   Jamalpur
C00000000010 Ferdous      21-APR-56 Bakshiganj   Jamalpur
C00000000011 Sazzad       21-APR-74 Nobogram     Barisal
C00000000012 Amir         19-APR-79 Nabab Road   Khulna
```

WHERE Clause

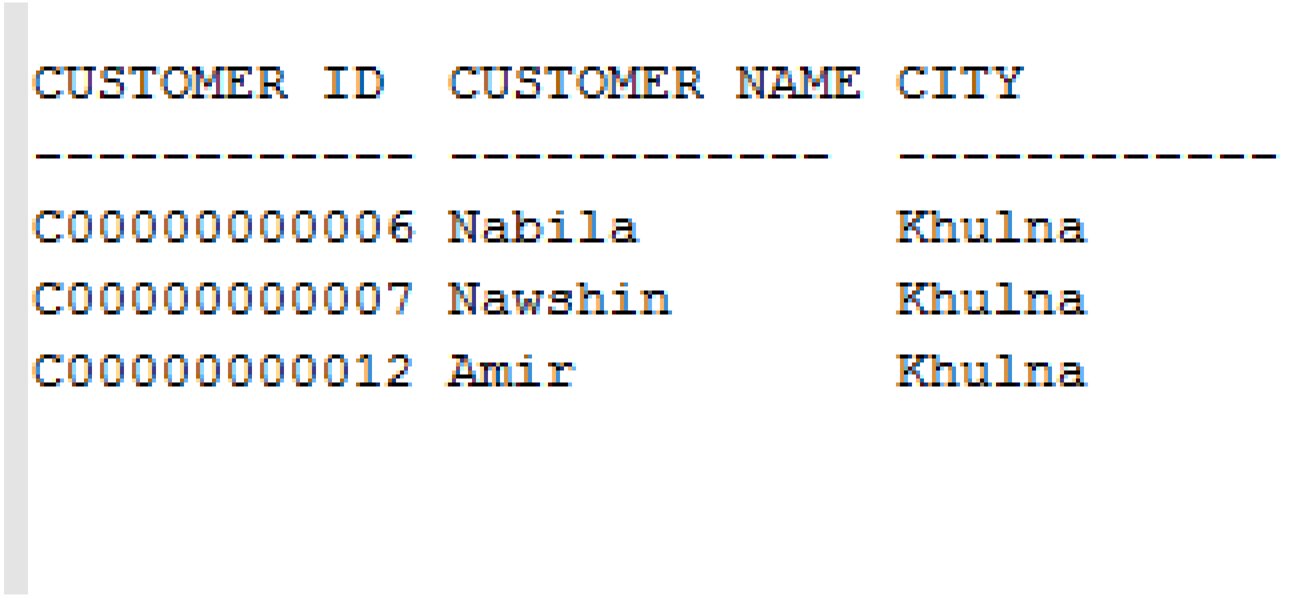


```
CUSTOMER ID  CUSTOMER NAME CITY
------------ ------------  ------------
C00000000006 Nabila        Khulna
C00000000007 Nawshin       Khulna
C00000000012 Amir          Khulna
```

- Syntax:

WHERE <column name> <relational operator> <value>

WHERE Clause



```
CUSTOMER ID  CUSTOMER NAME CITY
------------ ------------  ------------
C00000000006 Nabila        Khulna
C00000000007 Nawshin       Khulna
C00000000012 Amir          Khulna
```

SELECT *CUST_ID* "CUSTOMER ID", *CUST_NAME* "CUSTOMER NAME", *CUST_CITY* "CITY"

FROM CUSTOMER

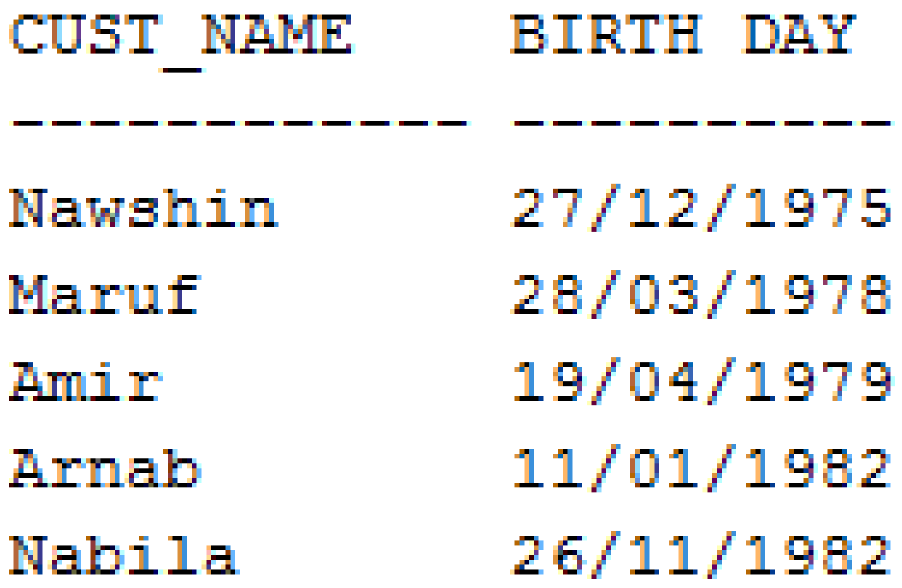
WHERE *CUST_CITY* = 'Khulna';

Relational Operator

- = equal to
- > greater than
- < less than
- < > not equal to
- ! = not equal to
- ^ = not equal to
- < = less than or equal to
- > = greater than or equal to

BETWEEN …AND Operator

Select name and date of birth of those customers, who has birth day **between 27-12-1975 and 26-11-1982.**



```
CUST_NAME      BIRTH DAY
-------------- -----------
Nawshin        27/12/1975
Maruf          28/03/1978
Amir           19/04/1979
Arnab          11/01/1982
Nabila         26/11/1982
```

BETWEEN …AND

SELECT ……

FROM *table_name*

WHERE *Column_name*

BETWEEN *Value1* AND *Value2* ;

BETWEEN …AND

```
SELECT CUST_NAME, to_char(CUST_DOB,
   'DD/MM/YYYY') "BIRTH DAY"
FROM CUSTOMER
WHERE CUST_DOB BETWEEN
TO_DATE('27-12-1975','DD-MM-YYYY') AND
TO_DATE('26-NOV-1982','DD-MON-YYYY')
ORDER BY CUST_DOB;
```

BETWEEN …AND

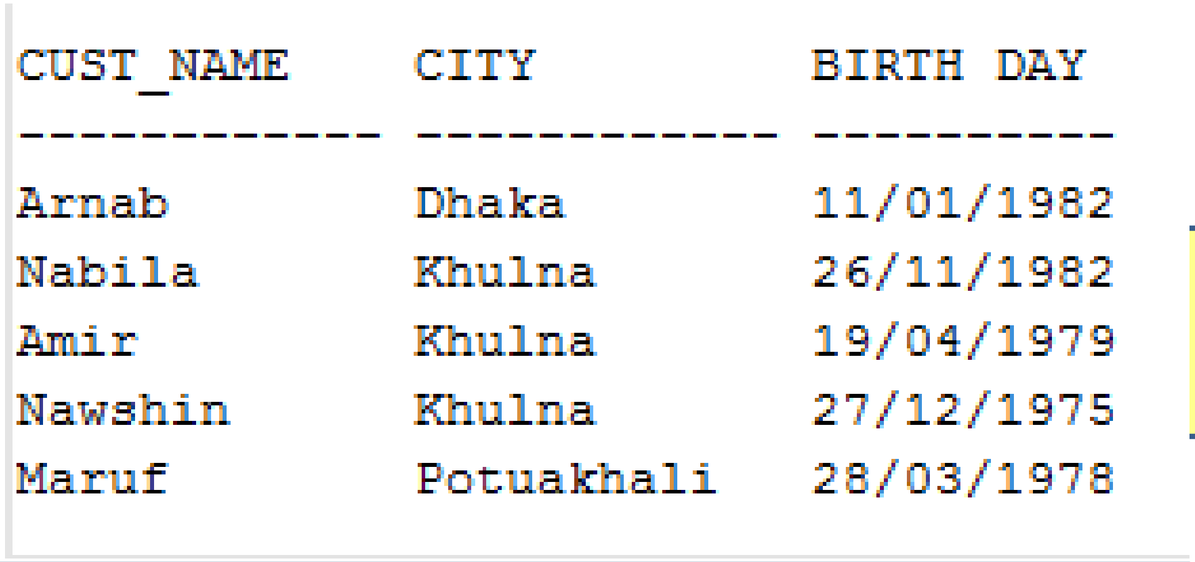
```
SELECT CUST_NAME, to_char(CUST_DOB,
   'DD/MM/YYYY') "BIRTH DAY"
FROM CUSTOMER
WHERE CUST_DOB BETWEEN
TO_DATE('27-12-1975','DD-MM-YYYY') AND
TO_DATE('26-NOV-1982','DD-MON-YYYY')
ORDER BY CUST_DOB ASC;
```

ORDER BY

```
SELECT CUST_NAME, to_char(CUST_DOB,
    'DD/MM/YYYY') "BIRTH DAY"
FROM CUSTOMER
WHERE CUST_DOB BETWEEN
TO_DATE('27-12-1975','DD-MM-YYYY') AND
TO_DATE('26-NOV-1982','DD-MON-YYYY')
ORDER BY CUST_DOB DESC;
```

PRIMARY

ORDER BY



```
CUST_NAME      CITY           BIRTH DAY
------------   ------------   ----------
Arnab          Dhaka          11/01/1982
Nabila         Khulna         26/11/1982
Amir           Khulna         19/04/1979
Nawshin        Khulna         27/12/1975
Maruf          Potuakhali     28/03/1978
```

Secondary Sorts

First order by city, then DOB

ORDER BY

SELECT *CUST_NAME*, *CUST_CITY* "CITY", to_char(*CUST_DOB*, 'DD/MM/YYYY') "BIRTH DAY"

FROM CUSTOMER

WHERE *CUST_DOB* **BETWEEN**

TO_DATE('27-12-1975','DD-MM-YYYY') **AND**

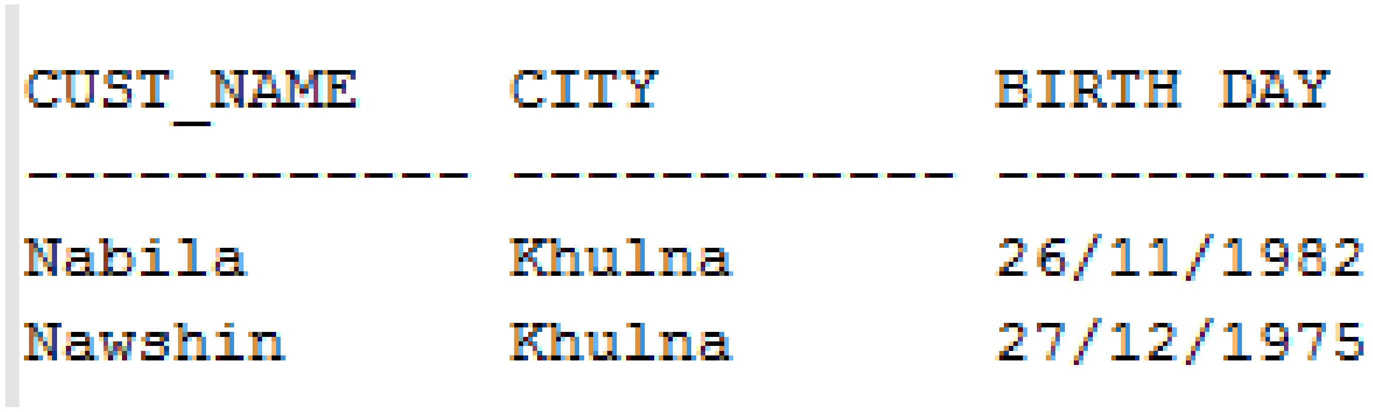
TO_DATE('26-NOV-1982','DD-MON-YYYY')

ORDER BY *CUST_CITY* **ASC**, *CUST_DOB* **DESC**;

Select name, city and date of birth of those customers, who has birth day from any of the following dates

27-12-1975

26-11-1982



```
CUST_NAME     CITY          BIRTH DAY
------------- ------------- ----------
Nabila        Khulna        26/11/1982
Nawshin       Khulna        27/12/1975
```

IN operator

- To search for values that match one of the values given in the listed values

- Values are in parentheses and are separated by commas

IN ('27-DEC-1975', '26-NOV-1982')

IN operator

SELECT *CUST_NAME*, *CUST_CITY* "CITY", to_char(*CUST_DOB*, 'DD/MM/YYYY') "BIRTH DAY"

FROM CUSTOMER

WHERE CUST_DOB **IN ('27-DEC-1975', '26-NOV-1982')**

ORDER BY CUST_CITY, *CUST_DOB* DESC;

LIKE

- **LIKE** performs pattern matching

- **An** underline character (_) represents **exactly one** character, **n** underline character (_) represents **exactly n** characters

- A percent sign (%) represents any number of characters, including zero characters

  $\text{Fi\%} \xrightarrow{\qquad\qquad} \text{Fish}$

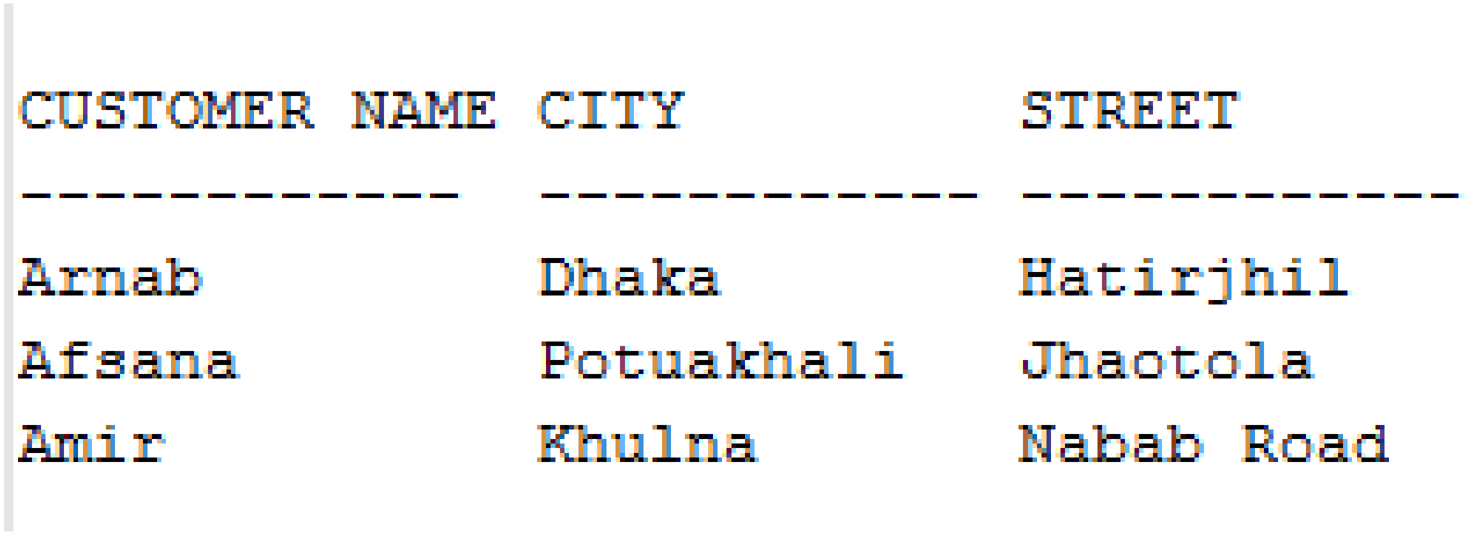
  Fighter

  Finger

LIKE

- Lists the city and street of all customers whose name starts with the “A”

  - WHERE *CUST_NAME* **LIKE** 'A%'



```
CUSTOMER NAME CITY           STREET
------------- -------------- -------------
Arnab         Dhaka          Hatirjhil
Afsana        Potuakhali     Jhaotola
Amir          Khulna         Nabab Road
```

LIKE

- %A
- %R%
- %oho% : ohona???
- %a_a
- %a%a
- a__a : abba

Logical Operators: AND / OR

- **<condition 1> AND <condition 2>**
- List all the records for customers whose City is Jamalpur **AND** Street is Bakshiganj.

- **<condition 1> OR <condition 2>**
- List all the records for customers whose City is Jamalpur **OR** Street is Hatirjhil.

ALTER TABLE

- Table altering means –
  - Adding columns
  - Deleting columns
  - Changing datatype or name of a column
  - Changing contraints etc

Add Columns

- Basic Syntax:

  ALTER TABLE *table_name* **ADD** *column_name datatype*

- *Example –*

  ALTER TABLE *Customer* ***ADD*** *Cust_Address Varchar2(20)*

  ** Add Multiple Columns In Table - Self

Delete Columns

- Basic Syntax:

  ALTER TABLE *table_name* **DROP COLUMN** *column_name*

- *Example –*

  ALTER TABLE *Customer* **DROP COLUMN** *Cust_Address;*

Change Column Name in the DATABASE

- Permanent Change in the database, not like aliasing.

- Basic Syntax:

  ALTER TABLE *table_name* **RENAME COLUMN** *old_name* **TO** *new_name*;

- Example –

  ALTER TABLE *customer* **RENAME COLUMN** *c_name* **TO** *cust_name*;

Change Column Datatype

- Permanent Change in the database
- Column **MUST BE** empty to change the datatype.

- Basic Syntax:

  ALTER TABLE *Table_name* **MODIFY** *Column_name Datatype*;

- Example -

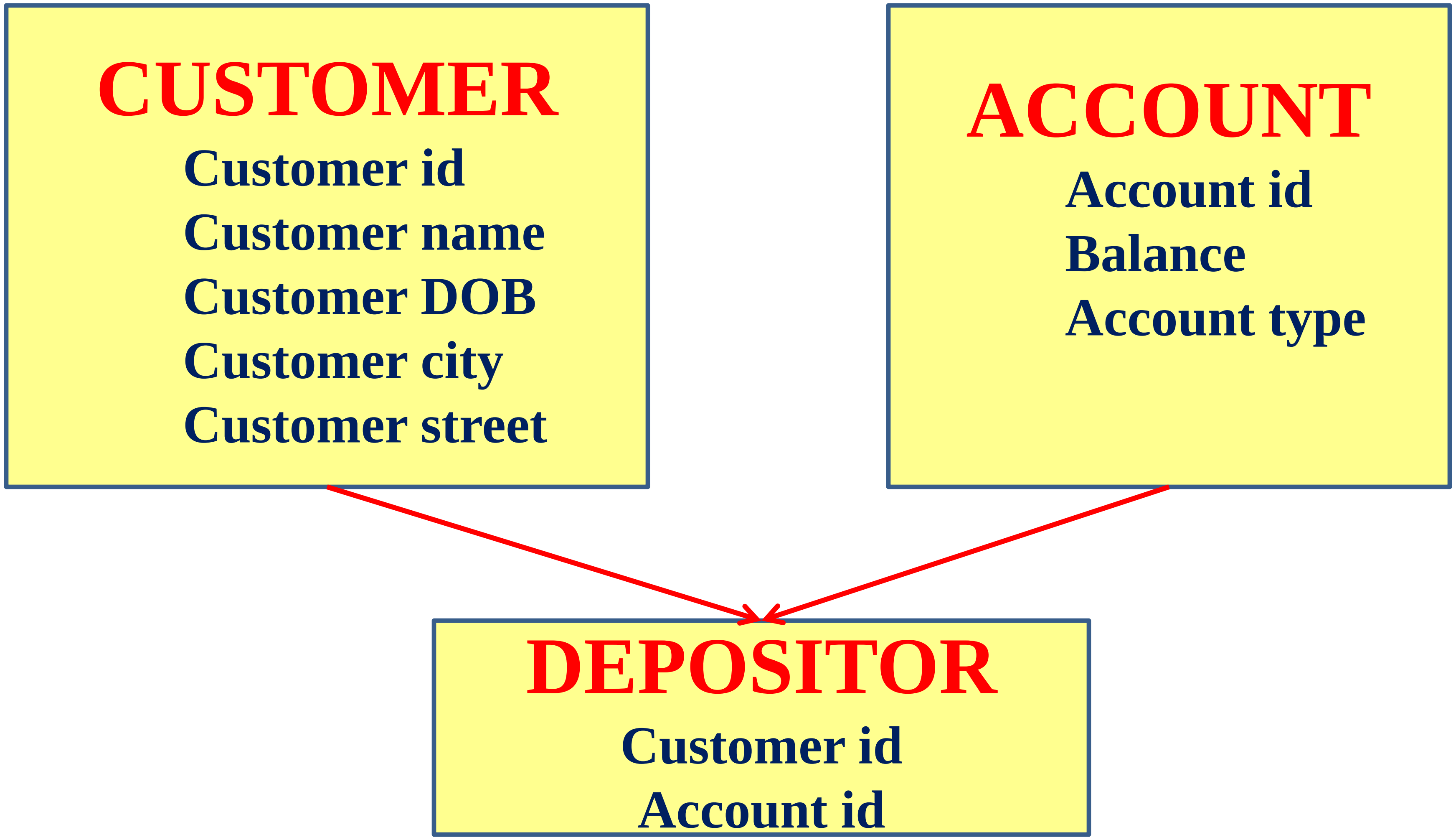
  ALTER TABLE *Customer* **MODIFY** *Cust_address Char(20)*;

CUSTOMER

Customer id
Customer name
Customer DOB
Customer city
Customer street

ACCOUNT

Account id
Balance
Account type


CUSTOMER
Customer id
Customer name
Customer DOB
Customer city
Customer street
ACCOUNT
Account id
Balance
Account type
DEPOSITOR
Customer id
Account id

Joining Multiple Tables

- Use a Join to query data from more than one table
  - Cartesian Join OR Cross Join
  - Equality Join

Cartesian Join / Cross Join

- Also referred to as Cartesian Product or Cross Join

- A Cartesian Join is useful for performing certain statistical procedures for data analysis. Otherwise, it is very rarely used. You probably will have no reason to create it

- In most cases, a Cartesian join is the result of accidentally not including the joining condition in either the WHERE or FROM clause.

Cartesian Join / Cross Join

- If one table has 20 records and the other table has 5 records, the number of rows returned is the product of 20 * 5. This results in the display of 100 records!!

```
SELECT table1.col, table2.col
FROM table1,table2
```

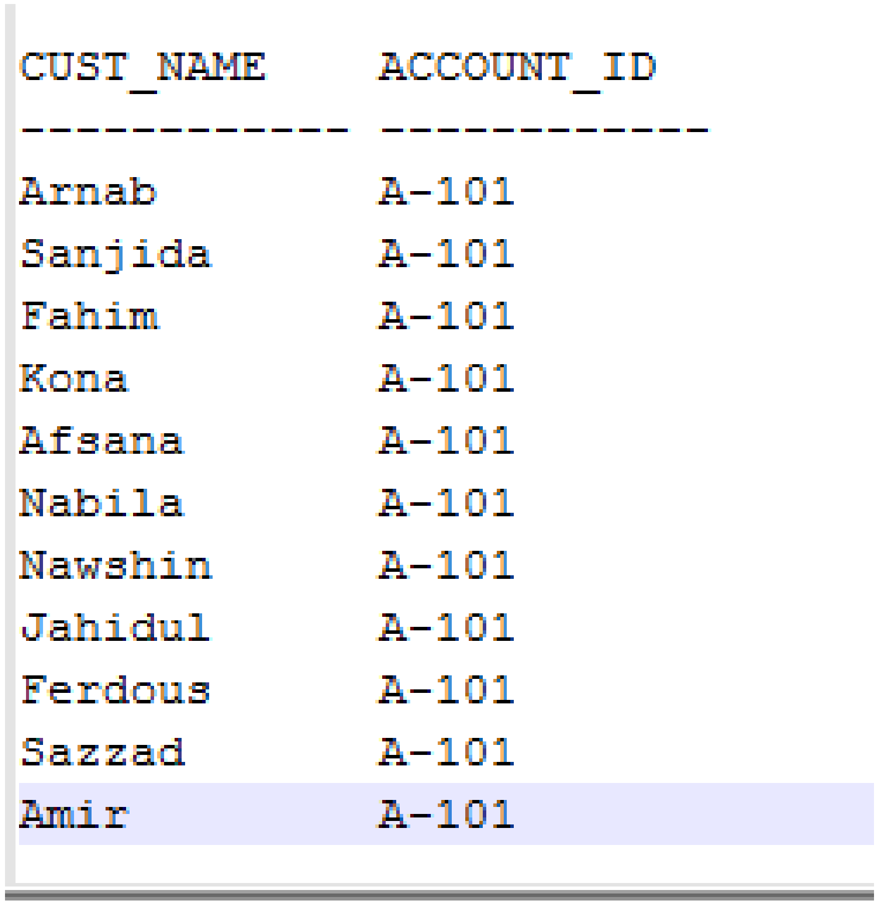
Cartesian Join / Cross Join

- SELECT *cust_name*, *account_id* FROM CUSTOMER , DEPOSITOR;

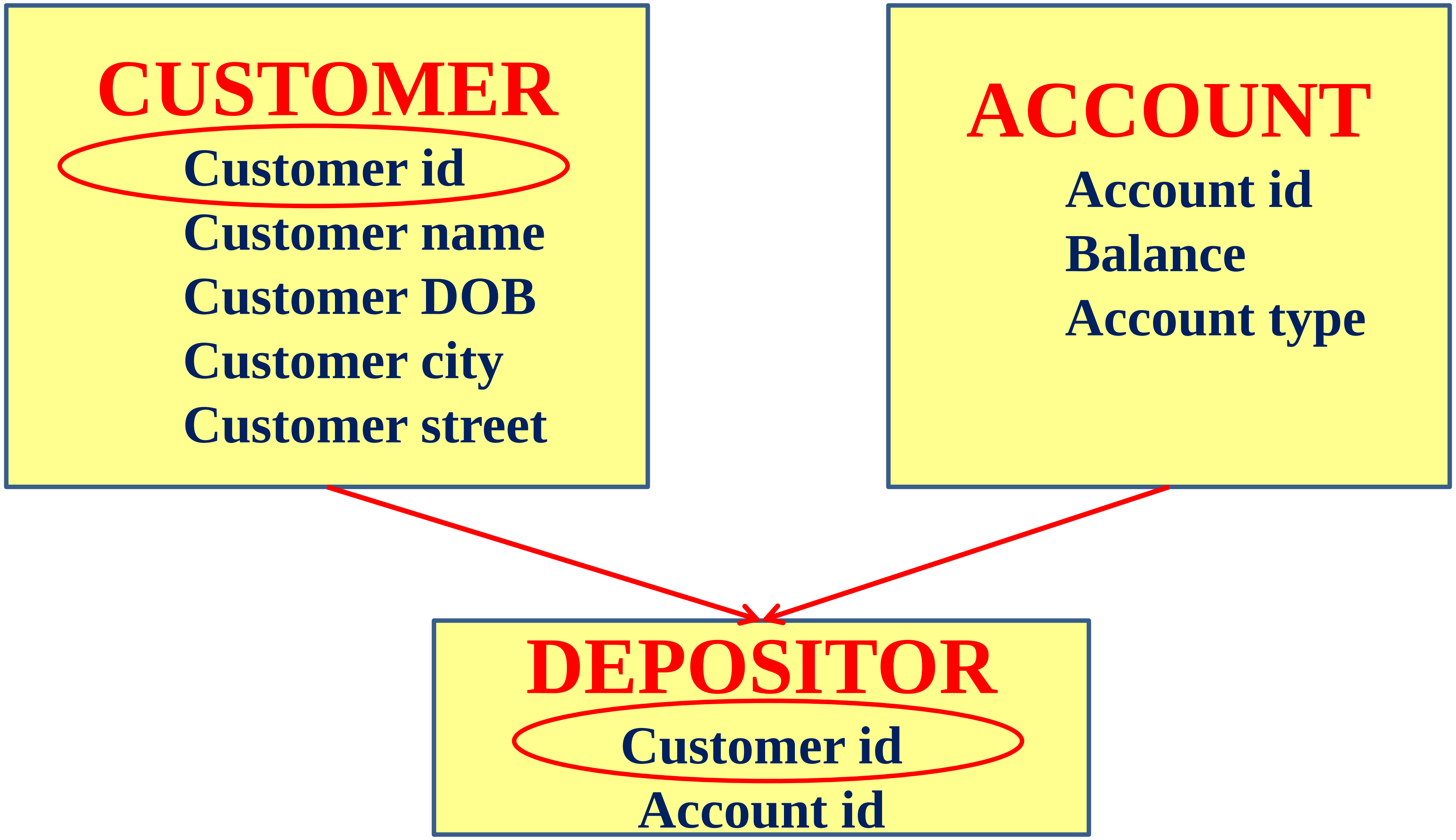
OR

- SELECT *cust_name*, *account_id* FROM CUSTOMER **CROSS JOIN** DEPOSITOR;

Cartesian Join / Cross Join



```
CUST_NAME    ACCOUNT_ID
------------ ------------
Arnab        A-101
Sanjida      A-101
Fahim        A-101
Kona         A-101
Afsana       A-101
Nabila       A-101
Nawshin      A-101
Jahidul      A-101
Ferdous      A-101
Sazzad       A-101
Amir         A-101
```


CUSTOMER
Customer id
Customer name
Customer DOB
Customer city
Customer street
ACCOUNT
Account id
Balance
Account type
DEPOSITOR
Customer id
Account id

Equijoins

- Also known as Equijoins, Inner Joins, or Simple Joins
- Based upon two (or more) tables having equivalent data stored in a "**common column**"
- Example:

  Both the Customer table and the Depositor table have *Cust_ID* as a common column.

  Cust_ID contains an identification code that is assigned to each Customer.

Equijoins

- Display a list showing the **Customer Name from Customer Table and the Account No from Depositor Table**

```
SELECT Cust_name, Account_id
FROM CUSTOMER, DEPOSITOR
WHERE CUSTOMER.Cust_id = DEPOSITOR.Cust_id;
```

Equijoins

- Include the *Cust_id* field in the displayed rows?

```
SELECT Cust_name, Cust_id, Account_id
FROM CUSTOMER, DEPOSITOR
WHERE CUSTOMER.Cust_id = DEPOSITOR.Cust_id;
```

- Add **Customer or Depositor as a column qualifier**

Table Aliases

```
SELECT CUST_NAME, ACCOUNT_ID
FROM CUSTOMER C, DEPOSITOR D
WHERE C.CUST_ID = D.CUST_ID;
```

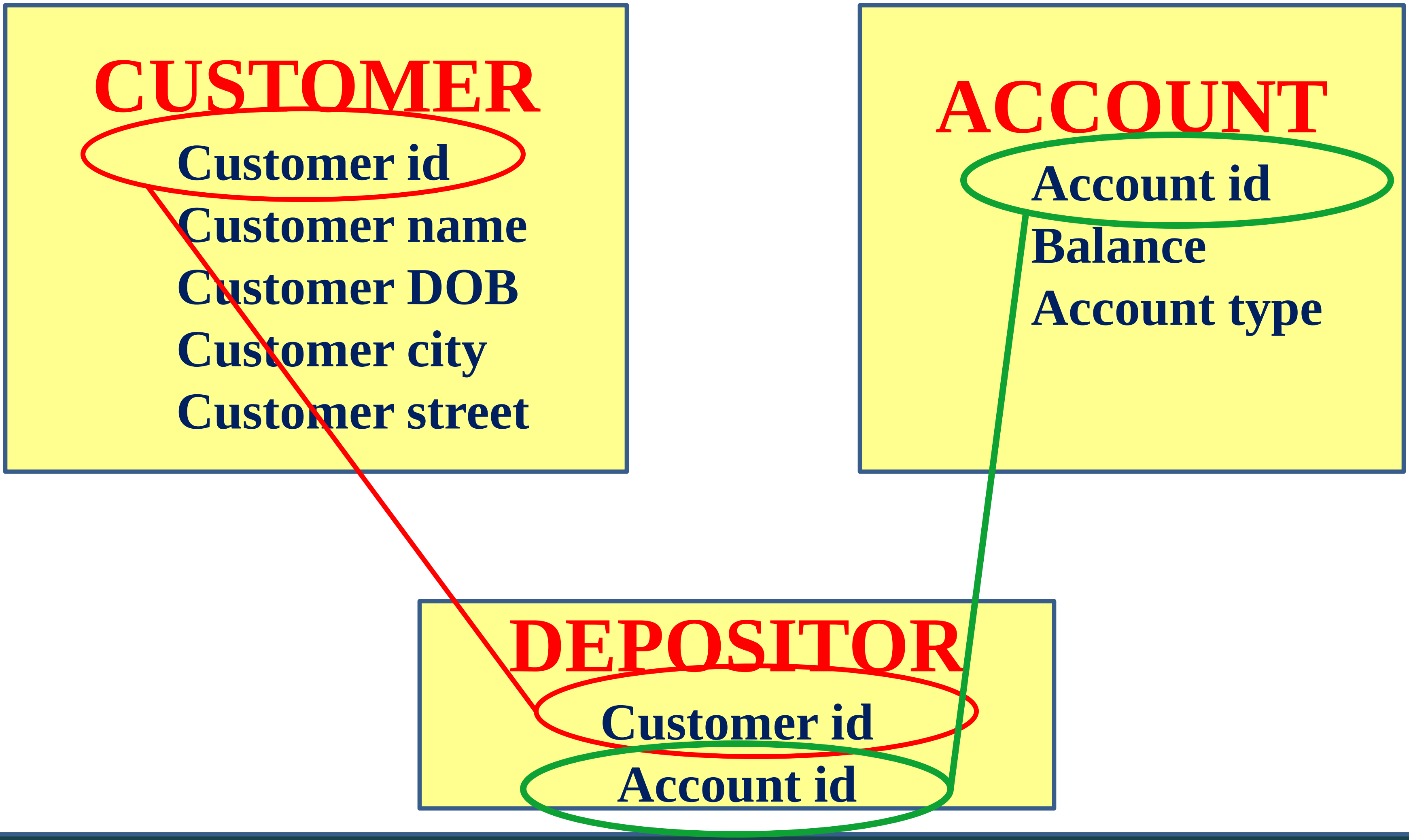
Additional Search Condition

Display **customer name** and **account ID** of those customer who lives in ‘Dhaka’ city

Problem!

- Display the **Customer Name, Account ID and Balance**

Joining More Than Two Tables



Some Keywords

- Feedback
- numwidth

References

1. Oracle_Database_11g_The_Complete Reference
2. http://www.w3schools.com/sql/
3. Oracle Built-in Datatypes
4. Stackoverflow
5. Book: Database System Concepts written by Avi Silberschatz, Henry F. Korth, S. Sudarshan

Thank You