**CSE 4308**

**DBMS LAB**

**Documentations**

**Table of Contents**

[1 LAB1 4](#_Toc508481915)

[1.1 SQL BASICS 4](#_Toc508481916)

[2 LAB2: 8](#_Toc508481917)

[2.1 Creating table: 8](#_Toc508481918)

[2.2 How to add foreign key to a table? 11](#_Toc508481919)

[2.3 Alter table 12](#_Toc508481920)

[3 LAB4 & LAB5 13](#_Toc508481921)

[3.1 Select query 13](#_Toc508481922)

[3.2 Adding condition to query 15](#_Toc508481923)

[3.3 SQL USING TWO/MORE TABLES [Natural join] 17](#_Toc508481924)

[3.4 USING AND, OR, NOT IS SQL 18](#_Toc508481925)

[3.4.1 AND: 18](#_Toc508481926)

[3.4.2 OR 18](#_Toc508481927)

[3.4.3 NOT 18](#_Toc508481928)

[3.5 ORDER BY 19](#_Toc508481929)

[3.6 Aggregate Functions 20](#_Toc508481930)

[3.6.1 MIN (): 20](#_Toc508481931)

[3.6.2 MAX (): 20](#_Toc508481932)

[3.6.3 COUNT (): 21](#_Toc508481933)

[3.6.4 AVG (): 21](#_Toc508481934)

[3.6.5 SUM (): 21](#_Toc508481935)

[3.7 Aggregating and Grouping 22](#_Toc508481936)

[3.8 Having Clause 22](#_Toc508481937)

[3.9 USING ROWNUM 22](#_Toc508481938)

[4 Referenced Tables 23](#_Toc508481939)

[4.1 Citizen 23](#_Toc508481940)

[4.2 STD 24](#_Toc508481941)

[4.3 Student 25](#_Toc508481942)

[4.4 DEPT 26](#_Toc508481943)

[4.5 Supervisor 27](#_Toc508481944)

[4.6 Manufacturers 28](#_Toc508481945)

[4.7 Products 29](#_Toc508481946)

# LAB1

## SQL BASICS

* How to open sql?

Just write sqlplus in the command prompt!

Or,

Search ‘run sql command line’ in the program search bar.

* How to connect to a database?

Open sql, write

CONNECT USERNAME/PASSWORD;

Or

Type

CONNECT

Then there will be prompt for username. Input the username. Then it will ask for password. But now if u type the password, it will not show the characters as oracle keeps it hidden. After inputting the password hit enter and see the system will be connected.

* How to see existing users?

First u have to connect to the database.

SELECT USERNAME FROM DBA\_USERS;

To see the existing usernames sorted:

SELECT USERNAME FROM DBA\_USERS ORDER BY USERNAME;

To see in which date the user was created:

SELECT USERNAME, CREATED FROM DBA\_USERS ORDER BY CREATED;

The default order is ascending. Do make it appear in descending order you have to use the following query:

SELECT USERNAME, CREATED FROM DBA\_USERS ORDER BY CREATED DESC;

(Here desc stands for Descending).

When the new user is created, if you try to login with it, Login will be denied as the super user (system) has not given him the privilege to login. There are some privileges in oracle like:

Create session,

Connect,

Resource, (to be able to create table and add data)

DBA, (database admin privilege)

Etc.…

* How to grant session privilege?

GRANT CREATE SESSION TO USERNAME;

Or

GRANT CREATE SESSION TO USERNAME WITH ADMIN OPTION.

To give him both create session and resource privilege:

GRANT CREATE SESSION, RESOURCE TO USERNAME;

To give all possible privilege at a single go, you can write:

GRANT ALL PRIVILEGES TO USERNAME;

(This is strictly NOT RECOMMENDED as the new user might get some unwanted privileges!)

* How to alter the system password in case of u forgotten it?

CONN / AS SYSDBA

ALTER USER USERNAME IDENTIFIED BY NEWPASSWORD;

Or

CONN SYSTEM / PASSWORD AS SYSDBA   
ALTER USER USERNAME IDENTIFIED BY NEWPASSWORD;

* How to create a user in oracle?

CREATE USER username IDENTIFIED BY password

* How to change password?

ALTER USER username IDENTIFIED BY password;

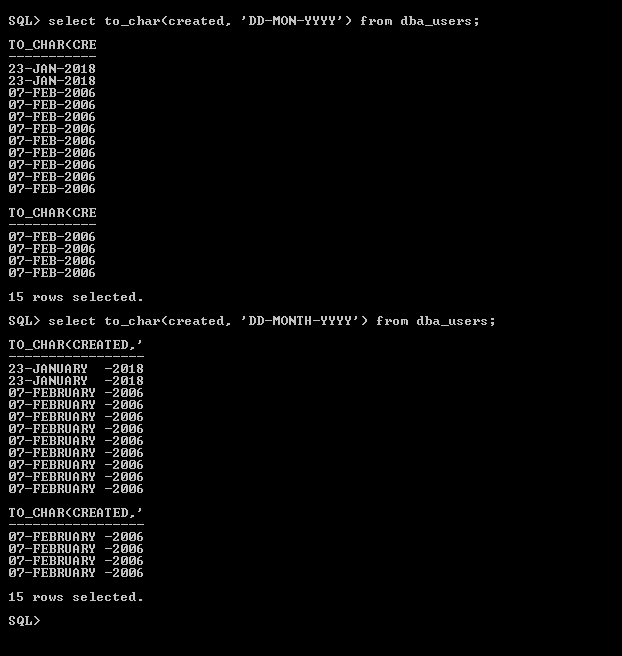
But for that you have to login as superuser! (System/sys/new user who has such privilege!)

* To drop a user:

DROP USER USERNAME CASCADE;

Here cascade means all the objects (for example: tables) should also be deleted with the removal of the user.

* We can format time in different ways using sql.Some examples are showed below:





# LAB2:

## Creating table:

If u want to connect sql from cmd, just type sqlplus!

* How to create a table?

Create table dept (

DeptID number,

DeptName varchar2 (20),

CONSTRAINTS PK\_DEPT PRIMARY KEY (DEPTID)

);

Remember that sql is not case sensitive! But the values are

* To see the column names and their types in sql:

DESC TABLE\_NAME;

* How to show all the tables that are under a particular user?

DESC DBA\_TABLES; [this dba\_tables is a table containing all the tables under each user.]

SELECT TABLE\_NAME

FROM DBA\_TABLES

WHERE OWNER=’USERNAME’;

* Suppose you want to create a new table ‘dept’ but you are not sure whether the table is already created or not. If it is already existing, oracle will not let u create a new table with the same name.

How to be sure that whether a table called ‘dept’ is existing or not?

One solution might be deleting the existing table.

DROP TABLE TABLE\_NAME;

But this is bad practice as the information might be necessary. So we can search

SELECT TABLE\_NAME

FROM DBA\_TABLES;

But it will return a long list of the existing tables. It is difficult to search my desired table\_name from the list.

Another way is:

SELECT TABLE\_NAME

FROM DBA\_TABLES

WHERE TABLE\_NAME=’DEPT’;

IF the dept table is existing, some rows will be selected from the table.

* How to disconnect a user?

Just write DISC

* Syntax of primary key:

CONSTRAINT constraint\_name PRIMARY KEY (column1, column2, … column\_n)

* How to add primary key to a table which is already created, but suppose u have forgotten to add the primary key?

Creating a table without primary key:

CREATE TABLE STD (

SID NUMBER,

SNAME VARCHAR (20),

);

ALTER TABLE TABLE\_NAME ADD CONSTRAINT CONSTRAINT\_NAME PRIMARY KEY (COL\_NAME);

* How to assign a composite primary key?

CONSTRAINT CONSTARINT\_NAME PRIMARY KEY (COL1, COL2…);

* How to insert values to a table?

INSERT INTO table\_name VALUES (value1, value2, value3, ...);

* You can also specify the column names:

INSERT INTO table\_name (column1, column2, column3, ...)   
VALUES (value1, value2, value3, ...);

Remember that sql is not case sensitive. But it is case sensitive in cases like saved values like passwords.

For inserting string/characters you have to put single quotes ‘value’.

* How to update values of a table?

UPDATE table\_name  
SET column1 = value1, column2 = value2, ...  
WHERE condition;

UPDATE Customers  
SET ContactName = 'Alfred Schmidt', City= 'Frankfurt'  
WHERE CustomerID = 1;

* How to delete values from a table?

DELETE FROM table\_name  
WHERE condition;

DELETE FROM Customers  
WHERE CustomerName='Alfreds Futterkiste';

## How to add foreign key to a table?

Suppose we have a table called ‘country’ which has the columns as c\_name, c\_id and c\_id is the primary key.

Now If I want to create a table called ‘citizen’ with the following columns like citizenID, citizenName and countryID we can define citizenId as the primary key of the citizen table. But here countryID should be referencing the ‘country’ table. Thus here countryID in the ‘citizen’ table is acting as the foreign key.

SQL:

CREATE TABLE COUNTRY (

C\_NAME VARCHAR2 (30),

C\_id NUMBER,

CONSTRAINTS PK\_COUNTRY PRIMARY KEY (C\_ID)

);

CREATE TABLE CITIZEN (

CITIZENID NUMBER,

CITIZENNAME VARCHAR2 (20),

COUNTRYID NUMBER,

CONSTRAINTS PK\_CITIZEN PRIMARY KEY (COUNTRYID),

CONSTRAINTS FK\_CITIZEN\_COUNTRY FOREIGN KEY (COUNTRYID)

REFERNECES COUNTRY (C\_ID)

);

REMEMBER THAT THE PARENT TABLE SHOULD ALWAYS BE CREATED FIRST!

And both of the columns (referencing and referenced) should be of the same types.

Now insert some values to both of the table. Here you’ll see that if u want to add a data to citizen table with the c\_id which is not present in the country table, it will show error. This means the foreign key is working properly.

* What will be the constraint if a table contains two foreign keys?

There should be two foreign key statements referencing each to the tables!

NOTE:

* Why do we give constraint names?

Here pk\_dept is the rule/constraint name. It has many uses. You can write anything in the place of pk\_dept. but this is the widely practiced naming convention. For example if table named ‘student’ has the deptID as the foreign key, the name of that foreign key constraint might be given as ‘fk\_student\_dept’.

One of the use of this constraint name is, suppose you have inserted a value to the student table with student\_id=1 where student\_id is the primary key. Now if you add another student with the student\_id=1, the command prompt will show that, ‘pk\_dept’ rule is violated. In another case, suppose we want to add a row to student table where the dept name is ‘abc’ which is not a valid department it is not entered in the dept table. So in this case command prompt will show that the ‘fk\_student\_dept’ constraint is violated.

## Alter table

* How to add column to an existing table?

ALTER TABLE table\_name  
ADD column\_name datatype;

* To delete column from an existing table:

ALTER TABLE table\_name  
DROP COLUMN column\_name;

* How to change the datatype of an existing column?

ALTER TABLE table\_name  
MODIFY column\_name datatype;

# LAB4 & LAB5

## Select query

* How to show all the values?

Select \* from table\_name;

* How to select specific columns from a table?

SELECT column1, column2, ...  
FROM table\_name;

* To select distinct values from a table:

SELECT DISTINCT C\_HOME FROM CITIZEN;

See the difference with

SELECT C\_HOME FROM CITIZEN;

* Renaming Output Column:

In sql, when you write the above query, output will be a column with the name ‘c\_home’ containing all the districts. We can rename the output column name for better visualization!

[Renaming a column in the select query doesn’t have any effect on the main table definition. It is only used for visualization]

SYNTAX:

SELECT COL\_NAME AS NEW\_COL\_NAME FROM TABLE\_NAME;

EXAMPLE:

SELECT C\_HOME AS HOME\_LIST

FROM CITIZEN;

SELECT DISTINCT (C\_HOME) AS UNIQUE\_DISTRICTS

FROM CITIZEN;

* How can I count the number of unique districts?

SELECT COUNT (DISTINCT (C\_HOME) ) FROM CITIZEN;

[In this case the output column name will be ‘COUNT (DISTINCT (C\_HOME) )’ which looks odd! So we can rename the output as

SELECT COUNT (DISTINCT (C\_HOME) ) AS TOTAL\_UNIQUE\_DIST FROM CITIZEN;

Note:

Here count () is an aggregate function which counts the total number of columns presented by the query.

Note:

SELECT COUNT (DISTINCT (C\_HOME) ) FROM CITIZEN;

This query can also be written as

SELECT COUNT (\*) AS TOTAL\_UNIQUE\_HOME

FROM (SELECT DISTINCT (C\_HOME) FROM CITIZEN);

[EXPLAINATION: this sort of query is called NESTED query. Here first the inner query in the FROM section is being calculated. Inner query first returns the names of the distinct district names which are ‘Dhaka’, ’Ctg’, ’Comilla’, ’Khulna’, ’Gazipur’. Then the outer query count the number of rows supplied by the inner query. So output will be 5]

## Adding condition to query

If u want to select some values based on a specific condition, u have to use ‘where’ clause as well!

SELECT COL1, COL2…

FROM TABLE\_NAME

WHERE CONTIDION;

In the where clause you can use operators like, =, >, <, <>, >=, <=, BETWEEN, LIKE, IN etc.

1. Find the citizen names and id who are more than 50 years old

SELECT C\_ID, C\_NAME

FROM CITIZEN

WHERE AGE>50;

1. Find the citizens living in Dhaka

SELECT C\_NAME AS NAME

FROM CITIZEN

WHERE C\_HOME=’Dhaka’;

1. Find the persons not living in Dhaka

SELECT C\_NAME, C\_HOME FROM CITIZEN WHERE C\_HOME<>’Dhaka’;

< > means! =

[NOTE: here Dhaka is CASE SENSETIVE! ]

1. Find the citizens having salary more than 20,000 and less than 50000

SELECT C\_NAME, SALARY

FROM CITIZEN

WHERE SALARY BETWEEN 20000 AND 50000;

Suppose we want to see the values which are greater than 50000 and less than 20000, then we can use the ‘NOT BETWEEN’ clause.

SELECT C\_NAME, SALARY

FROM CITIZEN

WHERE SALARY NOT BETWEEN 20000 AND 50000;

If you want to sort the output based on salary:

SELECT C\_NAME, SALARY

FROM CITIZEN

WHERE SALARY NOT BETWEEN 20000 AND 50000 ORDER BY SALARY;

Default order of sorting is ascending. For Descending:

SELECT C\_NAME, SALARY

FROM CITIZEN

WHERE SALARY NOT BETWEEN 20000 AND 50000

ORDER BY SALARY DESC;

[Note: this query should not necessarily be written in 3 lines, writing like ‘SELECT C\_ID, C\_NAME FROM CITIZEN WHERE AGE>50;’ will work also!]

The select clause may also contain arithmetic expressions involving operations +,-,\*, /.

For example:

* Give the teachers 5% increment and show their new salary?

SELECT C\_ID, C\_NAME, SALARY\*1.05

FROM CITIZEN

WHERE OCCUPATION=’Teacher’;

* How much money should I add to my budget if I want to give the teachers 5% bonus for EID?

SELECT SUM (SALARY\*0.05) AS TOTAL\_ADDITIONAL\_BUDGET

FROM CITIZEN

WHERE OCCUPATION=’Teacher’;

## SQL USING TWO/MORE TABLES [Natural join]

For better understanding, let’s create the following two tables.

CREATE TABLE STD (

STD\_ID NUMBER (3),

STD\_NAME VARCHAR (2),

STD\_DEPT NUMBER (3),

STD\_CG NUMBER (3, 2),

CONSTRAINTS PK\_STD PRIMARY KEY (STD\_ID),

CONSTRAINTS FK\_STD\_DEPT FOREIGN KEY (STD\_DEPT) REFERENCES DEPT (DEPT\_ID)

);

CREATE TABLE DEPT (

DEPT\_ID NUMBER (3),

DEPT\_NAME VARCHAR (3),

DEPT\_BUILDING VARCHAR (3),

DEPT\_ESTD NUMBER (4),

CONSTRAINTS PK\_DEPT PRIMARY KEY (DEPT\_ID)

);

Now suppose you need a list of students along with their dept\_name.

Sql:

SELECT STD.STD\_ID, DEPT.DEPT\_NAME

FROM STD, DEPT

WHERE STD.STD\_DEPT= DEPT.DEPT\_ID;

In sql you can rename the tables as well!

The above query could be written also as:

SELECT S.STD\_ID, D.DEPT\_NAME

FROM STD S, DEPT D

WHERE S.STD\_DEPT= D.DEPT\_ID;

It makes writing queries easier!

## USING AND, OR, NOT IS SQL

### AND:

Syntax:

SELECT column1, column2, ...  
FROM table\_name  
WHERE condition1 AND condition2 AND condition3 ...;

Example:

Show the list who are from ‘DHAKA’ AND salary is a Teacher:

SELECT \*

FROM CITIZEN

WHERE C\_HOME=’Dhaka’ AND OCCUPATION=’Teacher’;

### OR

Syntax:

SELECT column1, column2, ...  
FROM table\_name  
WHERE condition1 OR condition2 OR condition3 ...;

Example:

Show the people who are either a teacher or a doctor

SELECT \*

FROM CITIZEN

WHERE OCCUPATION=’Teacher’ OR OCCUPATION=’Doctor’;

### NOT

SELECT column1, column2, ...  
FROM table\_name  
WHERE NOT condition;

EXAMPLE:

Show the person not from Dhaka:

SELECT \*

FROM CITIZEN

WHERE NOT C\_HOME=’Dhaka’;

LET’S MIX UP THE CLAUSES!

Find the teachers from Dhaka or Ctg:

SELECT \*

FROM CITIZEN

WHERE OCCUPATION=’Teacher’ AND (C\_HOME=’Dhaka’ OR C\_HOME=’Ctg’);

Show the teachers who are not from Dhaka:

SELECT \*

FROM CITIZEN

WHERE OCCUPATION=’Teacher’ AND NOT C\_HOME=’Dhaka’;

## ORDER BY

Used to sort the output into ascending (default) or descending order.

SYNTAX:

SELECT COL1, COL2 . . .

FROMTABLE

WHERE CONDITION

ORDER BY COL1, COL2… ASC|DESC;

Here the where clause is not necessary. In that case:

SELECT COL1, COL2. . .

FROM TABLE

ORDER BY COL1,COL2… ASC|DESC

## Aggregate Functions

### MIN ():

SYNTAX:

SELECT MIN (COL\_NAME)

FROM TABLE\_NAME;

EXAMPLE:

To find the person with the minimum salary:

SELECT MIN (SALARY)

FROM CITIZEN;

### MAX ():

SYNTAX:

SELECT MAX (COL\_NAME)

FROM TABLE\_NAME;

EXAMPLE:

To find the person with maximum salary:

SELECT MAX (SALARY)

FROM CITIZEN;

We can always rename the column from MAX (SALARY) to ‘desired name’

EXAMPLE:

SELECT MAX (SALARY) AS MAXIMUM\_SALARY

FROM CITIZEN;

For selecting the maximum salary along with the salary holders name is quite a bit tricky!

SELECT C\_NAME, MAX (SALARY)

FROM CITIZEN;

Will not work!

Because the c\_name is returning all the c\_name of citizen table and the max (salary) is returning only a single value. Thus oracle can’t relate the two things!

What is the solution?

If we could find the max (salary) first and then match that salary with the table then return the name, the problem will be solved!

For this we need nested query!

SELECT C\_NAME, SALARY

FROM CITIZEN

WHERE SALARY= (SELECT MAX (SALARY) FROM CITIZEN);

Here the (SELECT MAX (SALARY) FROM CITIZEN); executes first. It returns the maximum salary. Then the outer query matches this salary with all the salaries. Then the name is found! Tricky, Right?!

### COUNT ():

COUNT () functions returns the number of rows matching a specific criteria.

EXAMPLE:

Find the number of teachers:

SELECT COUNT (c\_id)

FROM CITIZEN

WHERE OCCUPATION=’Teacher’;

To find the number of rows present in a table:

SELECT COUNT (\*)

FROM CITIZEN;

### AVG ():

Find the average salary of the citizens:

SELECT AVG (SALARY)

FROM CITIZEN;

### SUM ():

SELECT SUM (SALARY)

FROM CITIZEN;

Will return the addition of all the salaries of all citizens.

## Aggregating and Grouping

There are circumstances where we would like to apply the aggregate function not only to a single set of tuples, but also to a group of sets of tuples; we specify this wish in SQL using the **group by** clause. The attribute or attributes given in the **group by** clause are used to form groups. Tuples with the same value on all attributes in the **group by** clause are placed in one group.

Suppose you want the average salary based on occupations!

SELECT OCCUPATION, AVG (SALARY)

FROM CITIZEN

GROUP BY OCCUPATION;

## Having Clause

Sometimes we might work with conditions which apply on a group of tuples but not individual ones. In that case we use the Having clause.

It only works for aggregate functions.

Group the citizens based on living place and show the number of citizens living in each district where at least 2 people belong to that district.

SELECT COUNT (C\_HOME) AS NUMBER\_OF\_CITIZEN, C\_HOME

FROM CITIZEN

GROUP BY C\_HOME

HAVING COUNT (C\_HOME)>2;

Here HAVING clause works similar to where clause. But the difference is ‘where’ clause works on each tuples, but ‘having’ clause works on a group of tuples.

## USING ROWNUM

Suppose you want the top 5 salary holders. But using aggregate functions you can only get the maximum or minimum.

Here we want to sort the table and cut the top 5 rows. Right?

In these cases we use rownum.

SELECT \*

FROM (SELECT \* FROM CITIZEN ORDER BY SALARY)

WHERE ROWNUM<=5;

Here the inner sql in the form section returns a sorted table. The outer sql just takes out the first 5 rows of that table.

# Referenced Tables

## Citizen

CREATE TABLE CITIZEN (

C\_ID NUMBER (3),

C\_NAME VARCHAR2 (10),

C\_HOME VARCHAR2 (10),

AGE NUMBER (2),

OCCUPATION VARCHAR2 (15),

GENDER VARCHAR2 (6),

SALARY NUMBER,

CONSTRAINTS PK\_CITIZEN PRIMARY KEY (C\_ID)

);

INSERT INTO CITIZEN VALUES (1, 'A', 'Dhaka', 25, 'Teacher', 'Male', 50000);

INSERT INTO CITIZEN VALUES (2, 'B', 'Dhaka', 56, 'Service', 'Male', 60000);

INSERT INTO CITIZEN VALUES (3, 'C', 'Ctg', 71, 'Retired', 'Male', 10000);

INSERT INTO CITIZEN VALUES (4, 'D', 'Ctg', 13, 'Student', 'Female', 500);

INSERT INTO CITIZEN VALUES (5, 'E', 'Dhaka', 45, 'Service', 'Male', 40000);

INSERT INTO CITIZEN VALUES (6, 'F', 'Gazipur', 54, 'Doctor', 'Female', 55000);

INSERT INTO CITIZEN VALUES (7, 'G', 'Gazipur', 65, 'Musician', 'Female', 5000);

INSERT INTO CITIZEN VALUES (8, 'H', 'Dhaka', 56, 'Engineer', 'Male', 60000);

INSERT INTO CITIZEN VALUES (9, 'I', 'Ctg', 23, 'Student', 'Male', 1000);

INSERT INTO CITIZEN VALUES (10, 'J', 'Comilla', 32, 'Teacher', 'Male', 45000);

INSERT INTO CITIZEN VALUES (11, 'K', 'Comilla', 51, 'Farmer', 'Male', 20000);

INSERT INTO CITIZEN VALUES (12, 'L', 'Khulna', 15, 'Student', 'Female', 1500);

INSERT INTO CITIZEN VALUES (13, 'M', 'Ctg', 25, 'Business', 'Male', 100000);

INSERT INTO CITIZEN VALUES (14, 'N', 'Comilla', 52, 'Doctor', 'Male', 70000);

INSERT INTO CITIZEN VALUES (15, 'O', 'Gazipur', 53, 'Teacher', 'Male', 50000);

INSERT INTO CITIZEN VALUES (16, 'P', 'Dhaka', 35, 'Musician', 'Female', 50000);

INSERT INTO CITIZEN VALUES (17, 'Q', 'Khulna', 43, 'Service', 'Male', 50000);

INSERT INTO CITIZEN VALUES (18, 'R', 'Khulna', 34, 'Service', 'Female', 45000);

INSERT INTO CITIZEN VALUES (19, 'S', 'Ctg', 16, 'Student', 'Male', 500);

## STD

CREATE TABLE STD (

STD\_ID NUMBER (3),

STD\_NAME VARCHAR (2),

STD\_DEPT NUMBER (3),

STD\_CG NUMBER (3, 2),

CONSTRAINTS PK\_STD PRIMARY KEY (STD\_ID),

CONSTRAINTS FK\_STD\_DEPT FOREIGN KEY (STD\_DEPT) REFERENCES DEPT (DEPT\_ID)

);

INSERT INTO STD VALUES (41,’A’, 101, 3.5);

INSERT INTO STD VALUES (42,’B’, 102, 3.6);

INSERT INTO STD VALUES (43,’C’, 103, 3.7);

INSERT INTO STD VALUES (44,’D’, 101, 3.8);

## Student

CREATE TABLE STUDENT (

S\_ID NUMBER (2),

S\_NAME VARCHAR (3),

S\_HOME VARCHAR (10),

S\_DEPT NUMBER (3),

S\_SUP\_ID NUMBER (4),

S\_CG NUMBER (3, 2),

CONSTRAINTS PK\_STUDENT PRIMARY KEY (S\_ID),

CONSTRAINTS FK\_STUDENT\_DEPT FOREIGN KEY (S\_DEPT) REFERENCES DEPT (DEPT\_ID),

CONSTRAINTS FK\_STUDENT\_SUPERVISOR FOREIGN KEY (S\_SUP\_ID) REFERENCES SUPERVISOR (SUP\_ID)

);

INSERT INTO STUDENT VALUES (1,’SA’,’DHAKA’, 101, 1001, 3.5);

INSERT INTO STUDENT VALUES (2,’SB’,’CTG’, 102, 1002, 3.6);

INSERT INTO STUDENT VALUES (3,’SC’,’DHAKA’, 103, 1003, 3.7);

INSERT INTO STUDENT VALUES (4,’SD’,’COMILLA’, 104, 1004, 3.8);

INSERT INTO STUDENT VALUES (5,’SE’,’SYLHET’, 105, 1005, 3.5);

INSERT INTO STUDENT VALUES (6,’SF’,’DHAKA’, 101, 1006, 3.9);

INSERT INTO STUDENT VALUES (7,’SG’,’RAJSHAHI’, 101, 1001, 3.6);

## DEPT

CREATE TABLE DEPT (

DEPT\_ID NUMBER (3),

DEPT\_NAME VARCHAR (3),

DEPT\_BUILDING VARCHAR (3),

DEPT\_ESTD NUMBER (4),

CONSTRAINTS PK\_DEPT PRIMARY KEY (DEPT\_ID)

);

INSERT INTO DEPT VALUES (101,’CSE’,’NEW’, 1998);

INSERT INTO DEPT VALUES (102,’SWE’,’NEW’, 2018);

INSERT INTO DEPT VALUES (103,’MCE’,’OLD’, 1998);

INSERT INTO DEPT VALUES (104,’EEE’,’OLD’, 1998);

INSERT INTO DEPT VALUES (105,’CEE’,’OLD’, 2009);

INSERT INTO DEPT VALUES (106,’BTM’,’OLD’, 2018);

## Supervisor

CREATE TABLE SUPERVISOR (

SUP\_ID NUMBER (4),

SUP\_STAFF\_ID NUMBER (5),

SUP\_DEPT NUMBER (3),

SUP\_DPET\_ID NUMBER (3),

CONSTRAINTS PK\_SUP PRIMARY KEY (SUP\_ID),

CONSTRAINTS FK\_SUP\_DEPT FOREIGN KEY (SUP\_DEPT) REFERENCES DEPT (DEPT\_ID),

CONSTRAINTS FK\_SUP\_STAFF FOREIGN KEY (SUP\_STAFF\_ID) REFERENCE STAFF (STAFF\_ID)

);

## Manufacturers

CREATE TABLE Manufacturers (

Code INTEGER,

Name VARCHAR (255) NOT NULL,

PRIMARY KEY (Code)

);

INSERT INTO Manufacturers (Codename) VALUES (1,'Sony');

INSERT INTO Manufacturers (Codename) VALUES (2,'Creative Labs');

INSERT INTO Manufacturers (Codename) VALUES (3,'Hewlett-Packard');

INSERT INTO Manufacturers (Codename) VALUES (4,'Iomega');

INSERT INTO Manufacturers (Codename) VALUES (5,'Fujitsu');

INSERT INTO Manufacturers (Codename) VALUES (6,'Winchester');

## Products

CREATE TABLE Products (

Code INTEGER,

Name VARCHAR (255) NOT NULL,

Price DECIMAL NOT NULL,

Manufacturer INTEGER NOT NULL,

PRIMARY KEY (Code),

FOREIGN KEY (Manufacturer) REFERENCES Manufacturers (Code)

);

INSERT INTO Products (Code, Name, Price, and Manufacturer) VALUES (1,'Hard drive', 240, 5);

INSERT INTO Products (Code, Name, Price, and Manufacturer) VALUES (2,'Memory', 120, 6);

INSERT INTO Products(Code,Name,Price,Manufacturer) VALUES(3,'ZIP drive',150,4);

INSERT INTO Products (Code, Name, Price, and Manufacturer) VALUES (4,'Floppy disk', 5, 6);

INSERT INTO Products (Code, Name, Price, and Manufacturer) VALUES (5,'Monitor', 240, 1);

INSERT INTO Products (Code, Name, Price, Manufacturer) VALUES (6,'DVD drive', 180, 2);

INSERT INTO Products (Code, Name, Price, Manufacturer) VALUES (7,'CD drive', 90, 2);

INSERT INTO Products (Code, Name, Price, and Manufacturer) VALUES (8,'Printer', 270, 3);

INSERT INTO Products (Code, Name, Price, and Manufacturer) VALUES (9,'Toner cartridge', 66, 3);

INSERT INTO Products (Code, Name, Price, and Manufacturer) VALUES (10,'DVD burner', 180, 2);

INSERT INTO Products (Code, Name, Price, and Manufacturer) VALUES (11,'Card Reader', 180, 2);