MADRAS INSTITUTE OF TECHNOLOGY

ANNA UNIVERSITY

DEPARTMENT OF INFORMATION TECHNOLOGY

IT5312 - DATABASE MANAGEMENT SYSTEMS LABORATORY

LAB MANUAL

Vision of the Department

To educate students with conceptual knowledge and technical skills in the field of Information Technology with moral and ethical values to achieve excellence in an academic, industry and research centric environment.

Mission of the Department

1. To inculcate in students a firm foundation in theory and practice of IT skills coupled with the thought process for disruptive innovation and research methodologies, to keep pace with emerging technologies.
2. To provide a conducive environment for all academic, administrative, and interdisciplinary research activities using state-of-the-art technologies.
3. To stimulate the growth of graduates and doctorates, who will enter the workforce as productive IT engineers, researchers, and entrepreneurs with necessary soft skills, and continue higher professional education with competence in the global market.
4. To enable seamless collaboration with the IT industry and Government for consultancy and sponsored research.
5. To cater to cross-cultural, multinational, and demographic diversity of students.
6. To educate the students on the social, ethical, and moral values needed to make significant contributions to society.

Program Educational Objectives (PEOs)

PEO1: Demonstrate core competence in basic engineering and mathematics to design, formulate, analyze, and solve hardware/software engineering problems.

PEO2: Develop insights in foundational areas of Information Technology and related engineering to address real-world problems using digital and cognitive technologies.

PEO3: Collaborate with industry, academic and research institutions for state-of-the-art product development and research.

PEO4: Inculcate a high degree of professionalism, effective communication skills and team spirit to work on multidisciplinary projects in diverse environments.

PEO5: Practice high ethical values and technical standards.

Program Specific Outcomes (PSOs):

PSO1: Ability to apply programming principles and practices for the design of software solutions in an internet-enabled world of business and social activities.

PSO2: Ability to identify the resources to build and manage the IT infrastructure using the current technologies in order to solve real world problems with an understanding of the tradeoffs involved in the design choices.

PSO3: Ability to plan, design and execute projects for the development of intelligent systems with a focus on the future.

IT5312 DATABASE MANAGEMENT SYSTEMS LABORATORY L T P C

0 0 4 2

OBJECTIVES

- To learn and implement important commands in SQL.
- To learn the usage of nested and joint queries.
- To understand functions, procedures and procedural extensions of databases.
- To understand design and implementation of typical database applications.
- To be familiar with the use of a front end tool for GUI based application development.

LABORATORY EXERCISES

1. Create a database table, add constraints (primary key, unique, check, not null), insert rows, update and delete rows using SQL DDL and DML commands.
2. Create set of tables, add foreign key constraints and incorporate referential integrity.
3. Query the database tables using different 'where' clause conditions and also implement aggregate functions.
4. Query the database tables and explore sub queries and simple join operations.
5. Query the database tables and explore natural, equi and outer joins.
6. Write user defined functions and stored procedures in SQL.
7. Execute complex transactions and realize DCL and TCL commands.
8. Write SQL Triggers for insert, delete, and update operations in database table.
9. Create View and index for database tables with large number of records.
10. Create a XML database and validate it using XML schema.
11. Create Document, column and graph based data using NOSQL database tools.
12. Develop a simple GUI based database application and incorporate all the above-mentioned features.

OUTCOMES

On completion of the course, the student will be able to:

- Create databases with different types of key constraints.
- Write simple and complex SQL queries using DML and DCL commands.
- Realize database design using 3NF and BCNF.
- Use advanced features such as stored procedures and triggers and incorporate in GUI based application development.
- Create XML database and validate with meta-data (XML schema).
- Create and manipulate data using NOSQL database.

Mapping of Course Outcomes (COs) with Program Outcomes (POs)

Course Outcomes (COs)	**Program Outcomes (POs)**											
	1	2	3	4	5	6	7	8	9	10	11	12
Create databases with different types of key constraints	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓
Write simple and complex SQL queries using DML and DCL commands	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Realize database design using 3NF and BCNF	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Use advanced features such as stored procedures and triggers and incorporate in GUI based application development	✓	✓	✓	✓	✓		✓	✓	✓		✓	✓
Create XML database and valida with meta-data (XML schema)	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓
Create and manipulate data using NOSQL database	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

GRADING RUBRIC FOR LABORATORY COURSES

	Good Marks (81%-100%)	**Average Marks (50%-80%)**	**Satisfactory Marks (< 50%)**
Continuous Assessment (Covers Preparedness, Basic implementation, Ability to adapt additional Features and coding standards) **(Max Marks:30)**	Presence of detailed procedure, coding samples with proper implementation. Able to adapt the changes in the code quickly. Proper Coding Style.	Clarity of the procedure and coding samples are average with partial implementation. Able to understand the changes but unable to implement it. Fairly presented code with medium standards.	Lack of detailed procedure as well as coding samples with incorrect implementation. Unable to adapt the change in coding. Coding standards are not followed. Code is messy.
Laboratory Test (Covers Understanding problem, Basic Problem Solving and Ability to code, test, run and debug within the stipulated time) **(Max Marks:25)**	Problem understood clearly and solved. Complete implementation with proper test data within the stipulated time.	Problem understood but problem solving is not full- fledged. Completion of three fourths of the implementation with proper test data.	Lack of understanding and problem-solving ability is poor. Implementation not completed/Partial implementation within the stipulated time.
Course Oriented Laboratory Project (Covers Problem Selection, Demonstration of the Project, Wide coverage concepts in the target language) **(Max Marks:20)**	Selection of good real time problem with complete implementation with in-depth understanding on the concepts implemented. Wide coverage of concepts in the target language.	Selection of good real time problem with partially complete implementation and good knowledge on the concepts implemented. Moderate coverage of concepts.	Selection of fair problem with incomplete implementation. Lack of proper knowledge and understanding on the concepts implemented. Limited coverage of concepts.

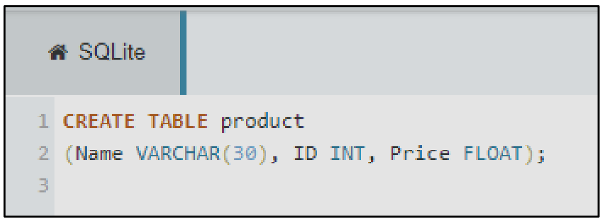
S. No.	Name of the Experiment
1.	SQL CRUD (Table Creation, Deletion, Updation and Retrieval)
2.	SQL Aggregate Functions
3.	SQL Operators
4.	SQL Constraints
5.	SQL Sub-Queries
6.	SQL Join Queries
7.	PL/SQL Anonymous and Named Block
8.	PL/SQL Cursors
9.	PL/SQL Triggers
10.	GUI Based Database Application

Exp. No.: 01	**SQL CRUD (Table Creation, Deletion, Updation and Retrieval)**

Aim: To create a table with fields of your choice and insert 4-5 records into it. Also perform the basic operations such as selecting record(s), updating table and deletion of record.

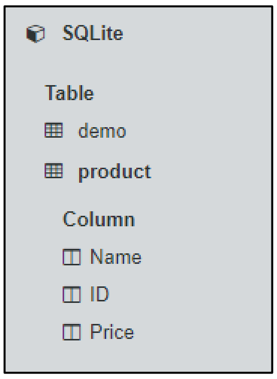
Queries and Corresponding Outputs:

a) To create the table 'product':

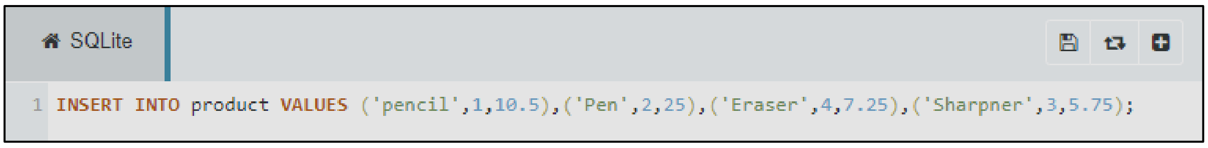


```
CREATE TABLE product
(Name VARCHAR(30), ID INT, Price FLOAT);
```

Result:



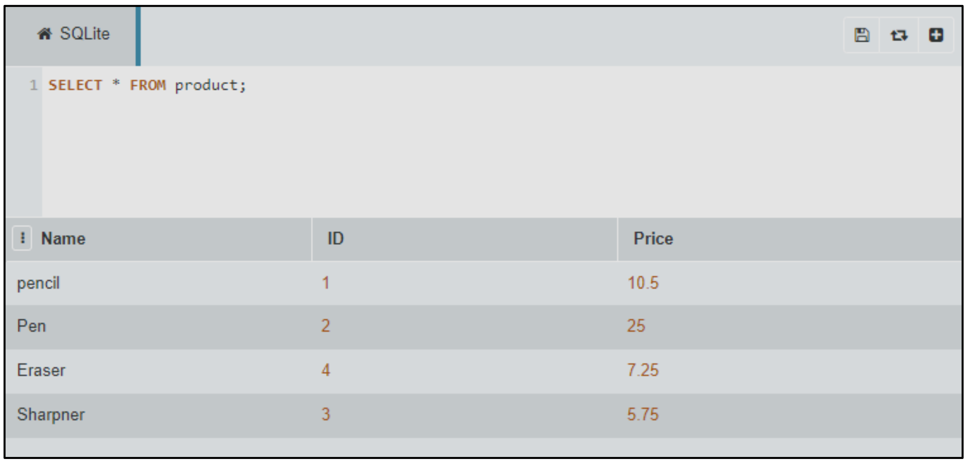
b) Insert records into the table:



```
INSERT INTO product VALUES ('pencil',1,10.5),('Pen',2,25),('Eraser',4,7.25),('Sharpner',3,5.75);
```

c) Select records from the table:

To select all the records in 'product'

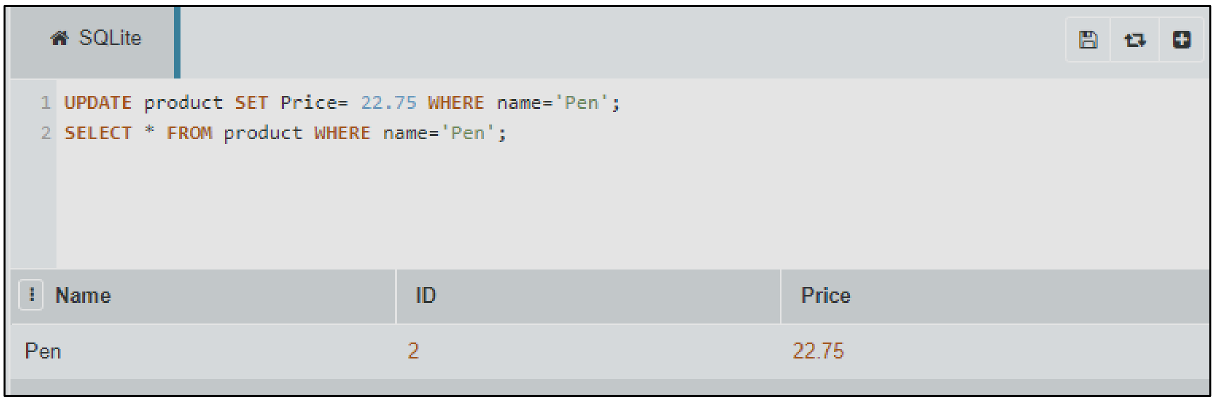


```
SELECT * FROM product;
```

Name	ID	Price
pencil	1	10.5
Pen	2	25
Eraser	4	7.25
Sharpner	3	5.75

d) Update table:

To update the value of Price to 22.75 for the record that has the value of name as 'Pen'

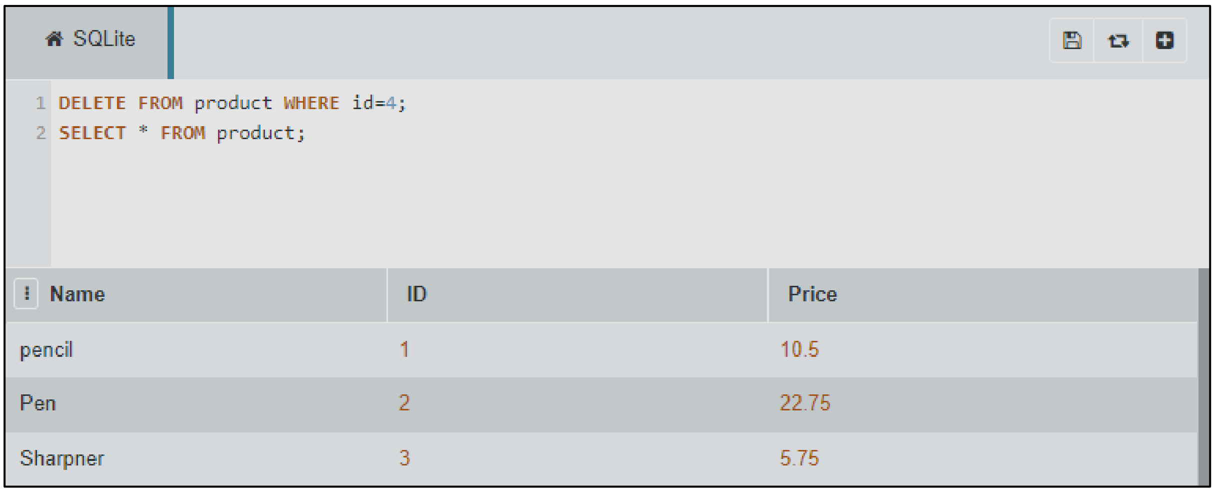


```
UPDATE product SET Price= 22.75 WHERE name='Pen';
SELECT * FROM product WHERE name='Pen';
```

Name	ID	Price
Pen	2	22.75

e) Delete record from table:

To delete the record whose ID is 4 from the table



```
DELETE FROM product WHERE id=4;
SELECT * FROM product;
```

Name	ID	Price
pencil	1	10.5
Pen	2	22.75
Sharpner	3	5.75

RESULT: Thus the operation of database table CRUD was successfully implemented using SQL query.

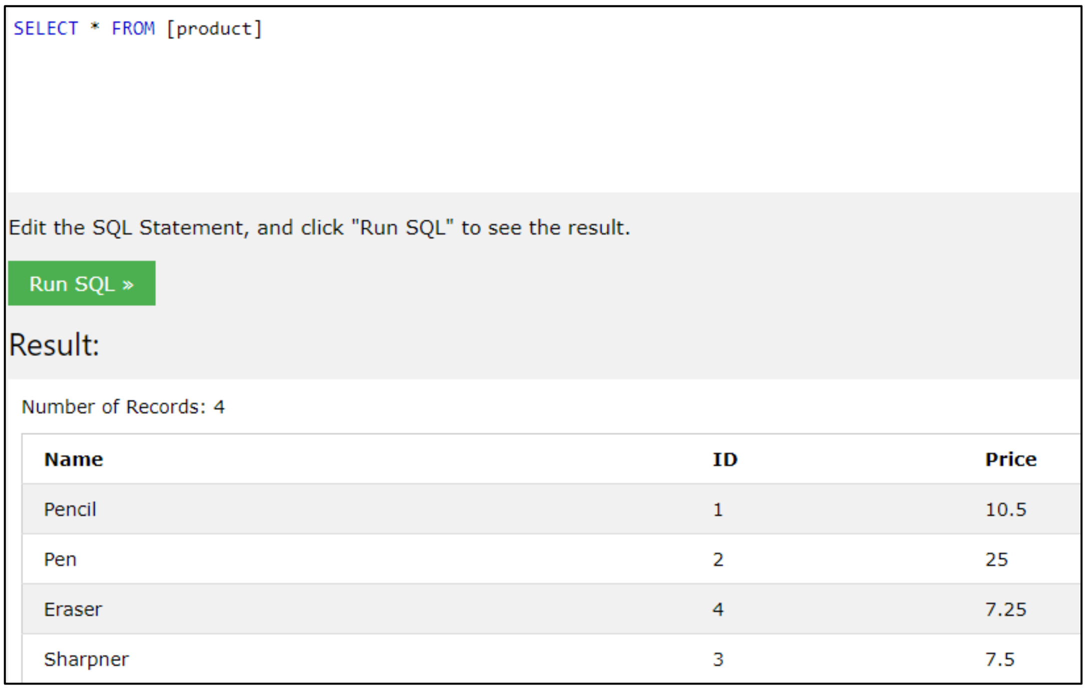
------ X ------

Exp. No.: 02	**SQL Aggregate Functions**

Aim: To demonstrate the working of SQL aggregate functions

Queries and Corresponding Outputs:

➔ Existing table ‘product’



```
SELECT * FROM [product]
```

Edit the SQL Statement, and click "Run SQL" to see the result.

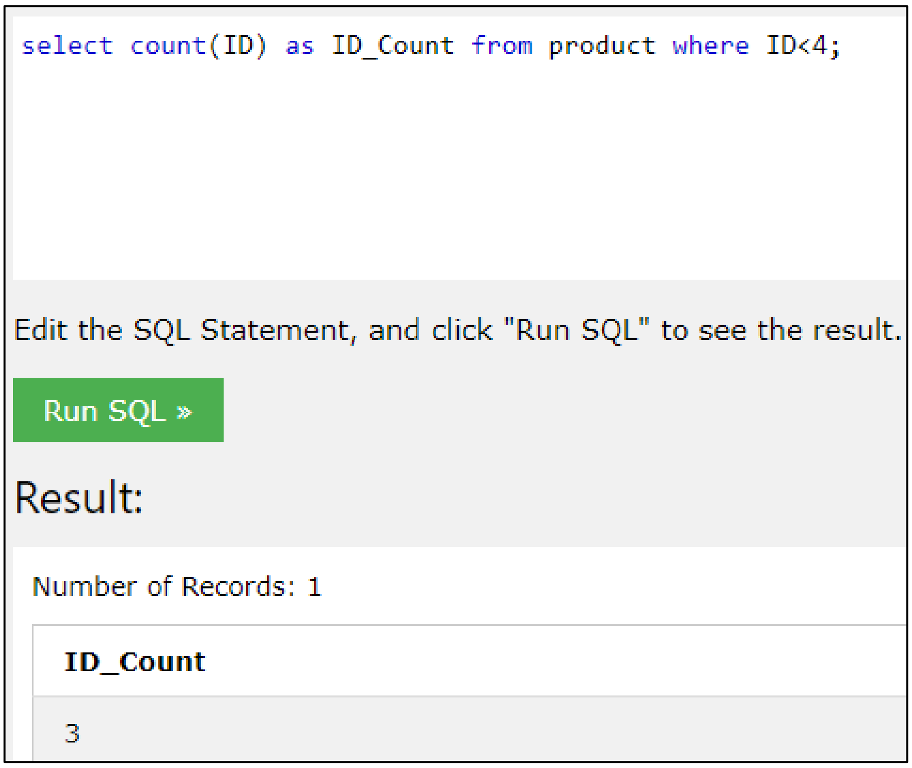
Result:

Number of Records: 4

Name	ID	Price
Pencil	1	10.5
Pen	2	25
Eraser	4	7.25
Sharpner	3	7.5

a) Count function, count( )

To count the number of values of ID that are less than 4 in the given table



```
select count(ID) as ID_Count from product where ID<4;
```

Edit the SQL Statement, and click "Run SQL" to see the result.

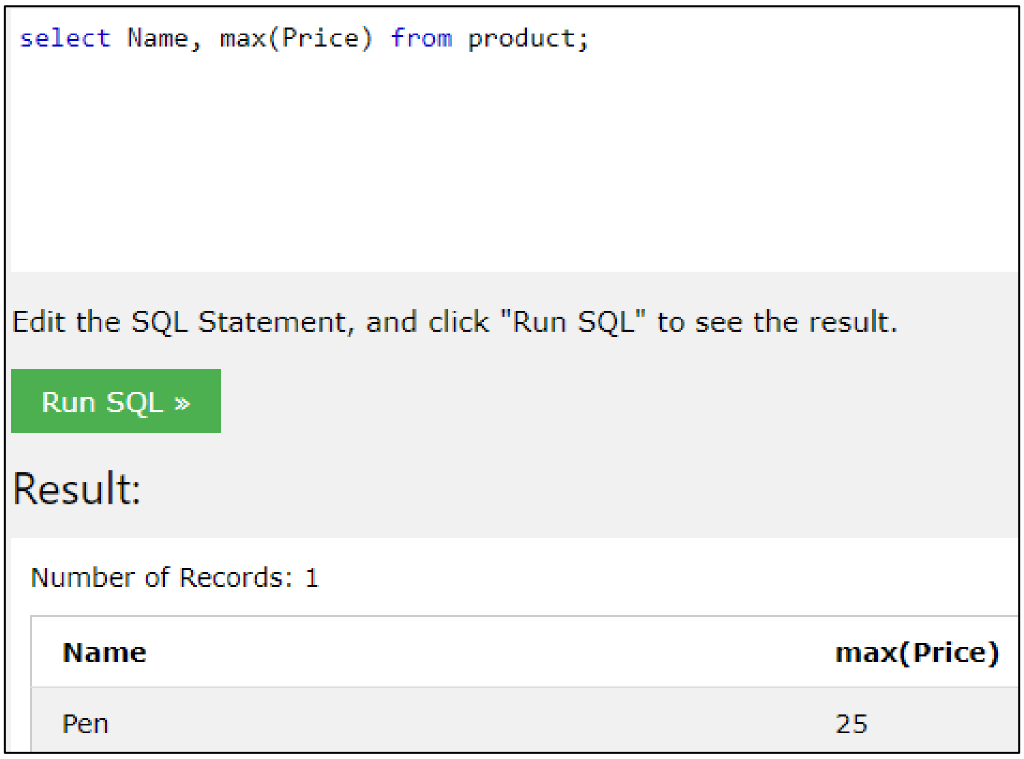
Result:

Number of Records: 1

ID_Count
3

b) Maximum Function, max( )

To find the maximum value of Price and the corresponding Name from the table



```
select Name, max(Price) from product;
```

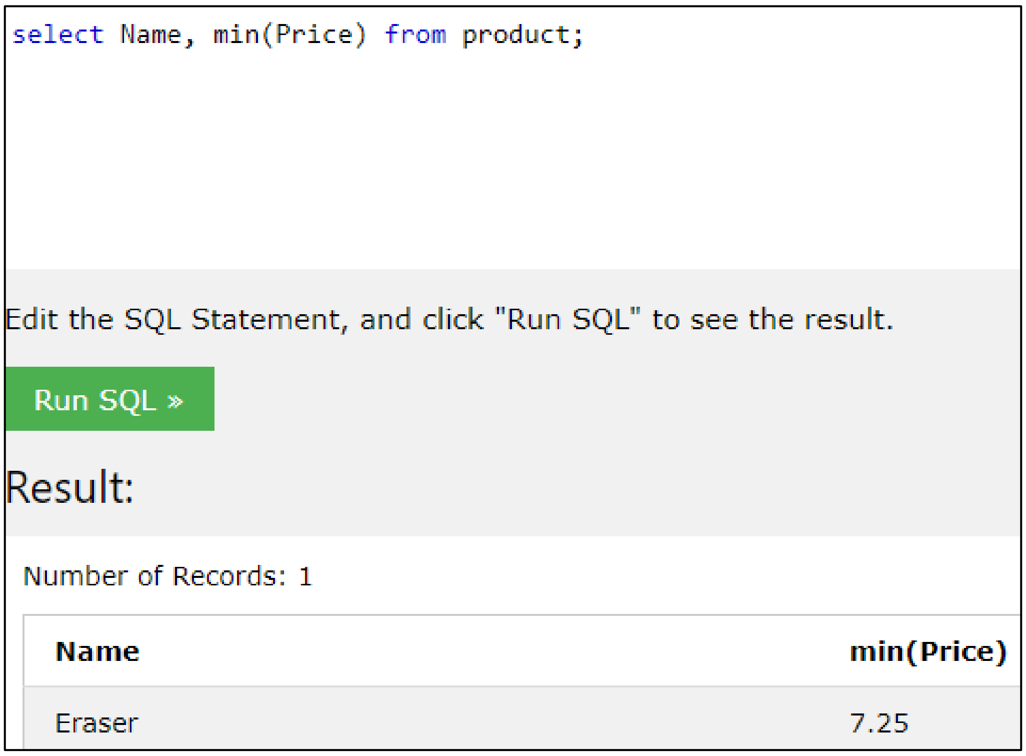
Edit the SQL Statement, and click "Run SQL" to see the result.
Run SQL »
Result:

Number of Records: 1

Name	max(Price)
Pen	25

c) Minimum Function, min( )

To find the minimum value of Price and the corresponding Name from the table



```
select Name, min(Price) from product;
```

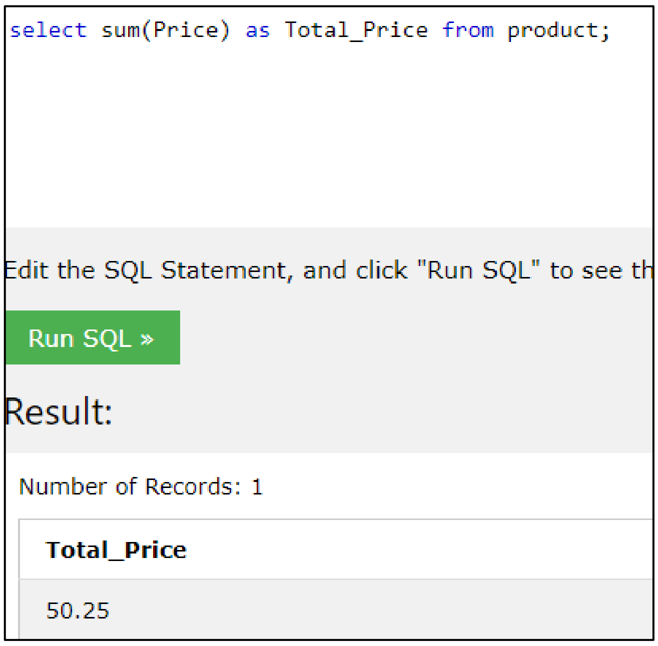
Edit the SQL Statement, and click "Run SQL" to see the result.
Run SQL »
Result:

Number of Records: 1

Name	min(Price)
Eraser	7.25

d) Sum Function, sum( )

To find the sum of all the values of Price in the given table



```
select sum(Price) as Total_Price from product;
```

Edit the SQL Statement, and click "Run SQL" to see th

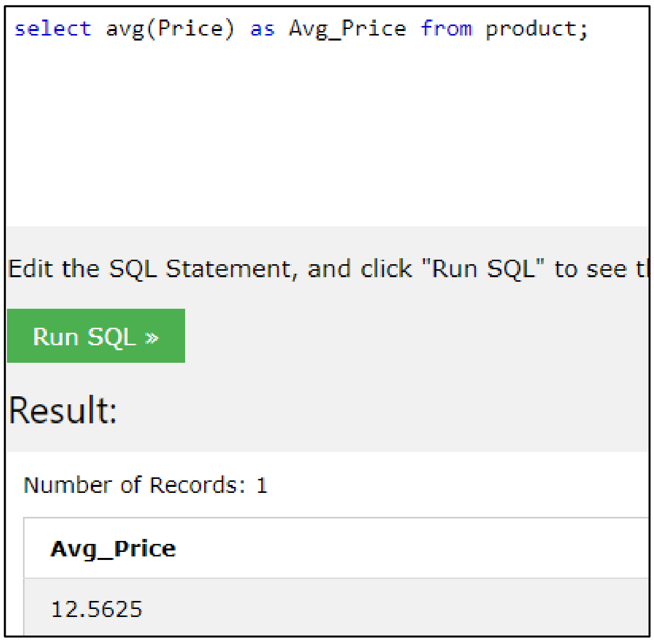
Result:

Number of Records: 1

Total_Price
50.25

e) Average Function, avg( )

To find the average value of Price in the given table



```
select avg(Price) as Avg_Price from product;
```

Edit the SQL Statement, and click "Run SQL" to see t

Result:

Number of Records: 1

Avg_Price
12.5625

RESULT: Thus the operation of database aggregate functions was successfully implemented using SQL query.

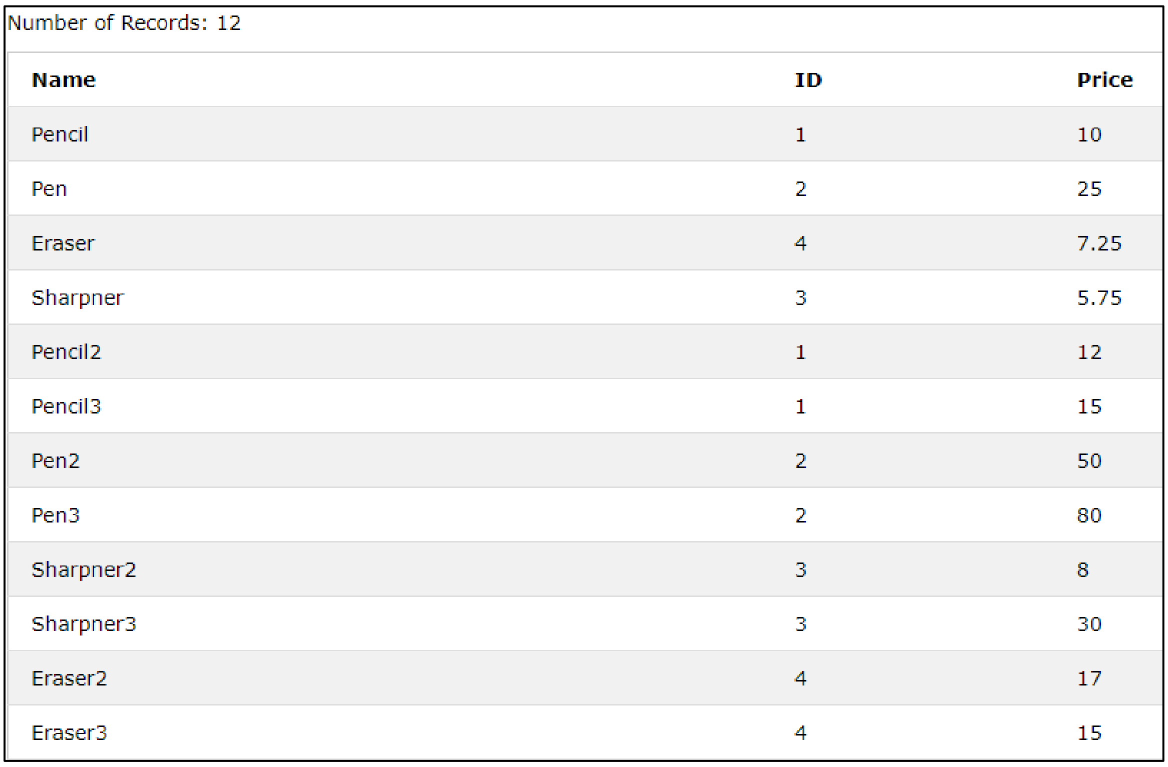
------ X ------

Exp. No.: 03	**SQL Operators**

Aim: To demonstrate the working of SQL Operators

Queries and Corresponding Outputs:

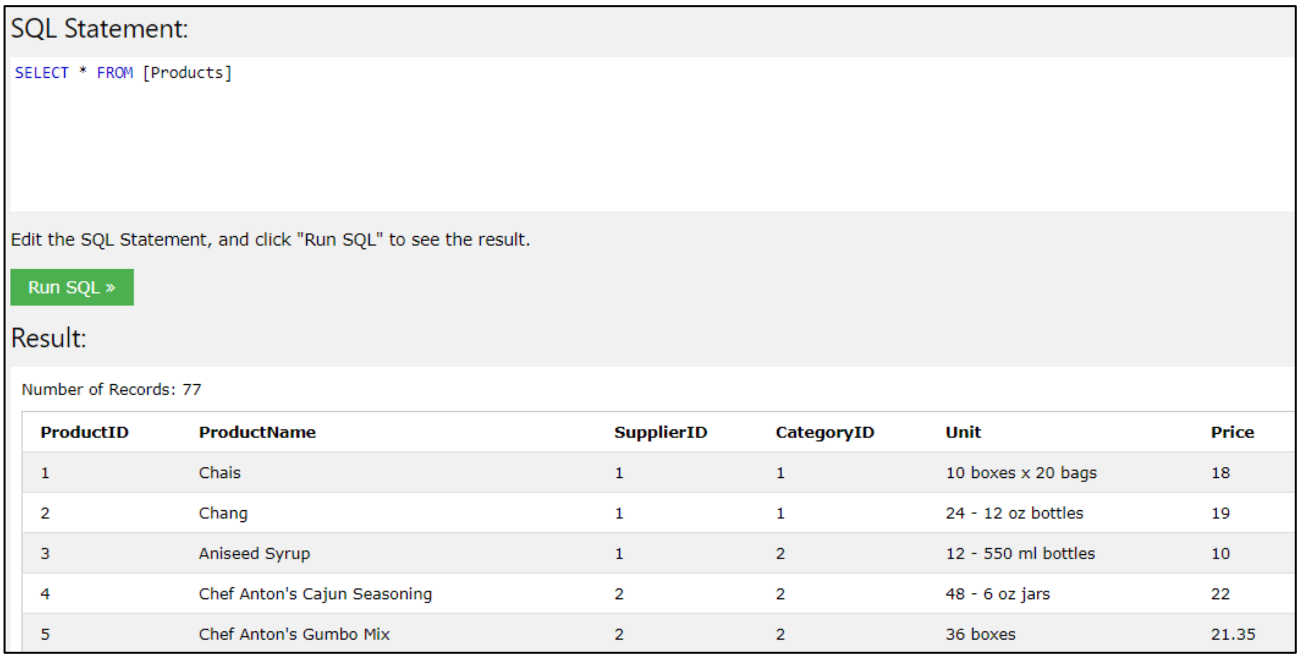
➔ Existing table 'product'



Number of Records: 12

Name	ID	Price
Pencil	1	10
Pen	2	25
Eraser	4	7.25
Sharpner	3	5.75
Pencil2	1	12
Pencil3	1	15
Pen2	2	50
Pen3	2	80
Sharpner2	3	8
Sharpner3	3	30
Eraser2	4	17
Eraser3	4	15

➔ Existing table 'Products'



SQL Statement:

```
SELECT * FROM [Products]
```

Edit the SQL Statement, and click "Run SQL" to see the result.

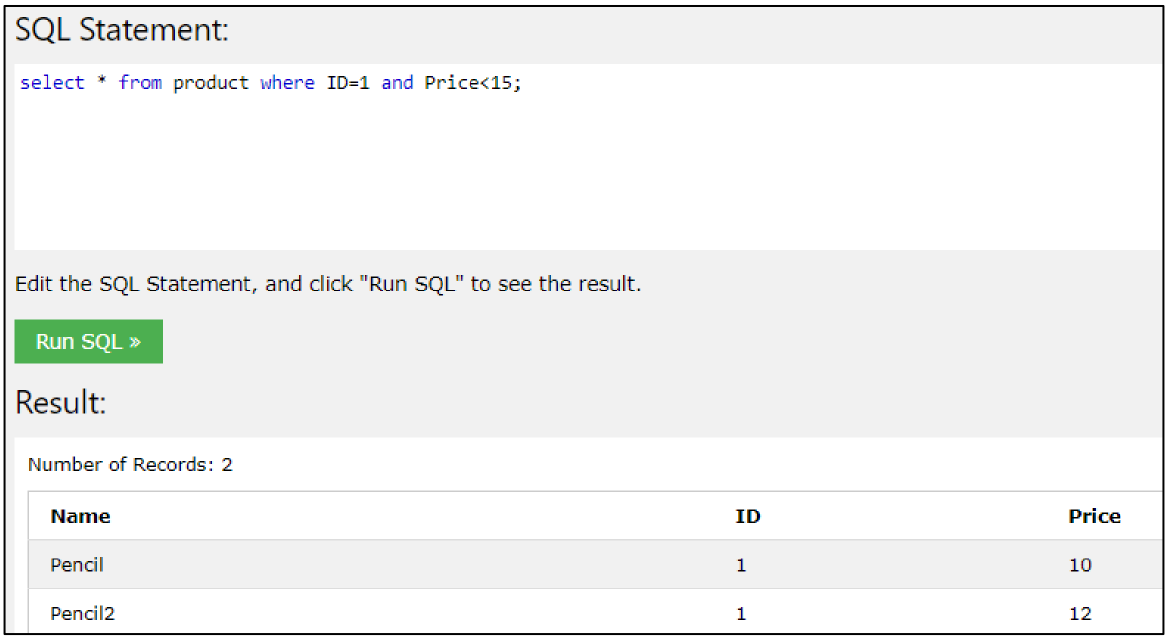
Result:

Number of Records: 77

ProductID	ProductName	SupplierID	CategoryID	Unit	Price
1	Chais	1	1	10 boxes x 20 bags	18
2	Chang	1	1	24 - 12 oz bottles	19
3	Aniseed Syrup	1	2	12 - 550 ml bottles	10
4	Chef Anton's Cajun Seasoning	2	2	48 - 6 oz jars	22
5	Chef Anton's Gumbo Mix	2	2	36 boxes	21.35

a) AND operator

To select those records from product that have ID as 1 and also have price less than 15


SQL Statement:
Edit the SQL Statement, and click "Run SQL" to see the result.
Run SQL »
Result:

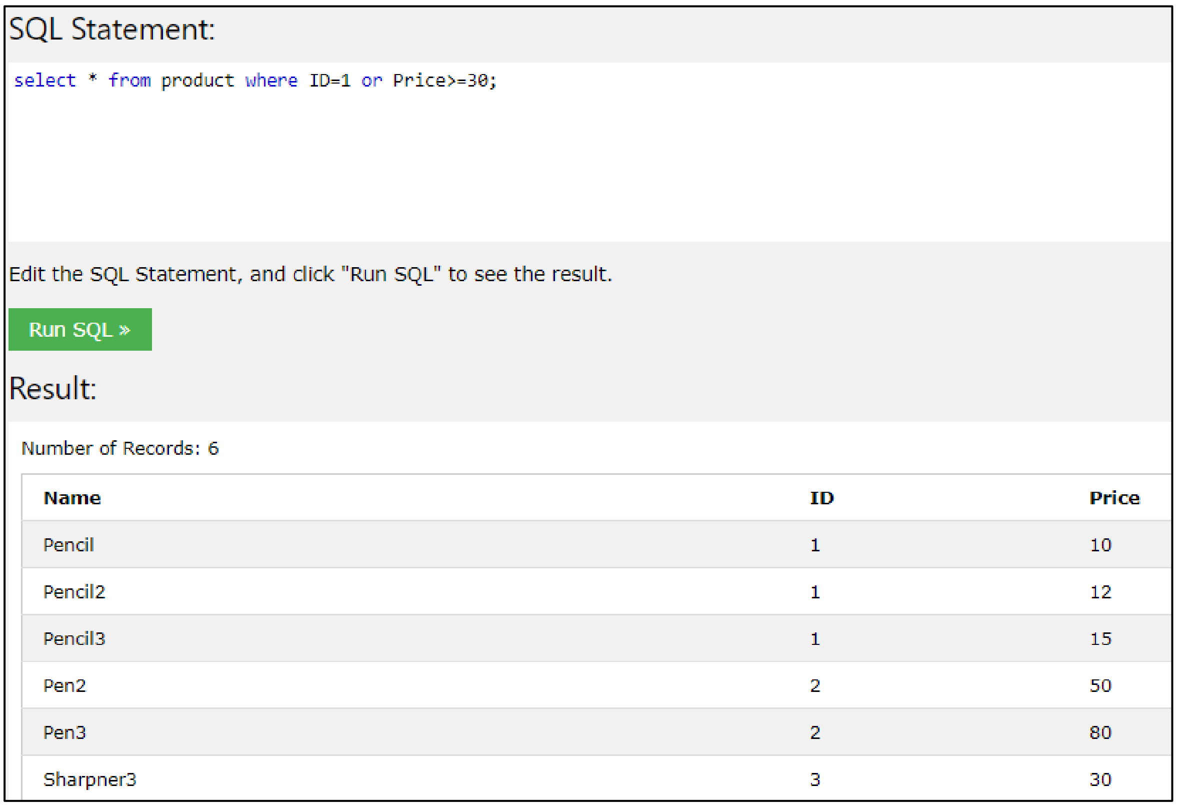
```
select * from product where ID=1 and Price<15;
```

Number of Records: 2

Name	ID	Price
Pencil	1	10
Pencil2	1	12

b) OR operator

To select those records from product that have ID as 1 or have price more than or equal to 30


SQL Statement:
Edit the SQL Statement, and click "Run SQL" to see the result.
Run SQL »
Result:

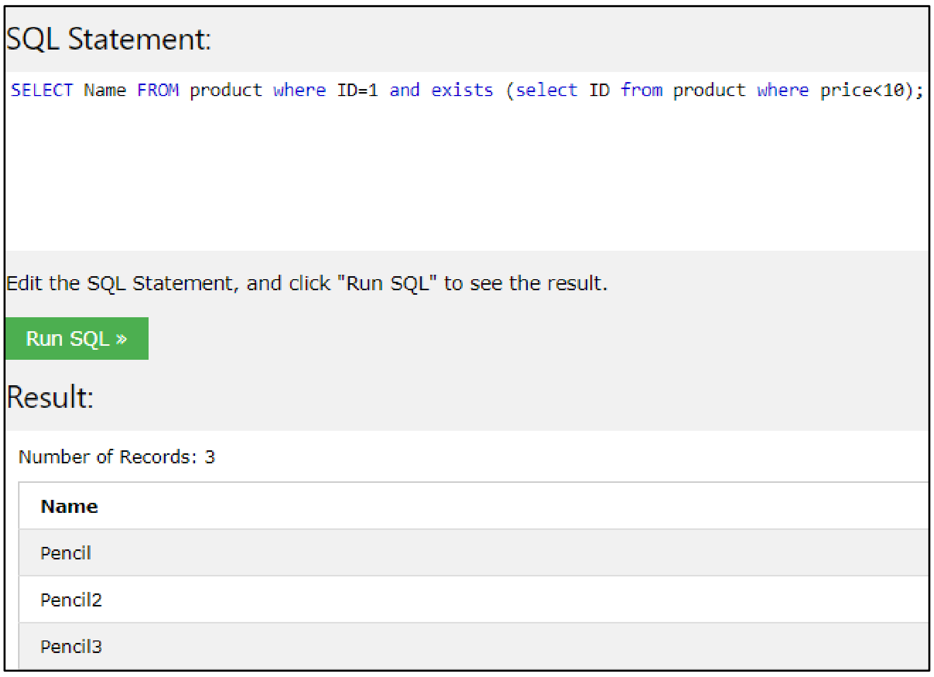
```
select * from product where ID=1 or Price>=30;
```

Number of Records: 6

Name	ID	Price
Pencil	1	10
Pencil2	1	12
Pencil3	1	15
Pen2	2	50
Pen3	2	80
Sharpner3	3	30

c) EXISTS operator

1. To select the Names from the table product whose corresponding ID is 1 given that there exists at least one record in the same table with Price less than 10


SQL Statement:

```
SELECT Name FROM product where ID=1 and exists (select ID from product where price<10);
```

Edit the SQL Statement, and click "Run SQL" to see the result.

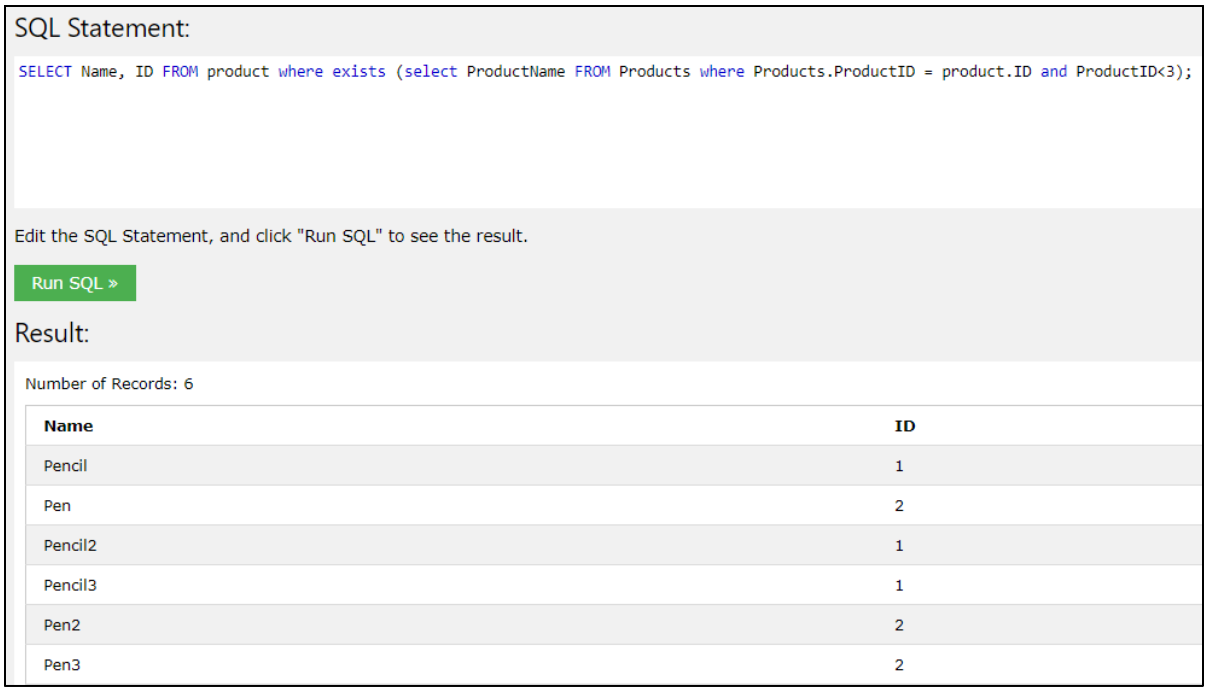
Result:

Number of Records: 3

Name
Pencil
Pencil2
Pencil3

2. To select the Name and ID from product given that there exists records in Products such that the value of ProductID of its record and that of ID of record in product are the same and are less than 3

Here, the 2 tables ‘product’ and ‘Product’ are joined using the attributes ProductID and ID respectively. The Name and ID of the records of product that have the ID value as less than 3 and present in domain of ProductID are displayed


SQL Statement:

```
SELECT Name, ID FROM product where exists (select ProductName FROM Products where Products.ProductID = product.ID and ProductID<3);
```

Edit the SQL Statement, and click "Run SQL" to see the result.

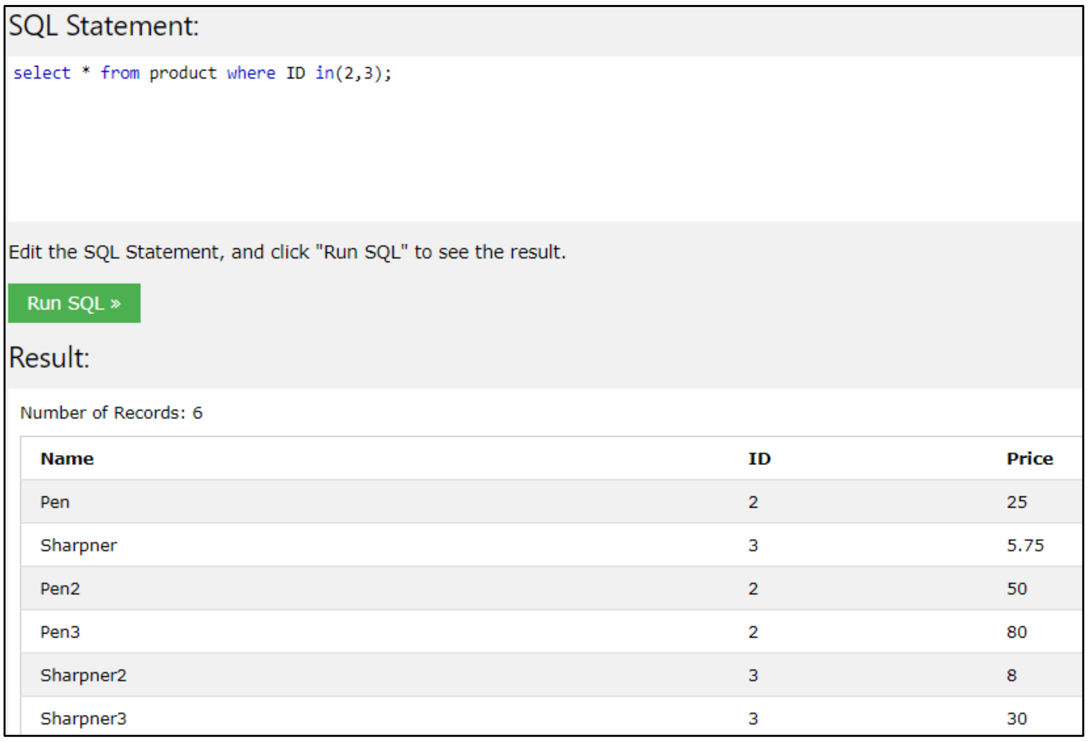
Result:

Number of Records: 6

Name	ID
Pencil	1
Pen	2
Pencil2	1
Pencil3	1
Pen2	2
Pen3	2

d) IN operator

To select the records of product that have ID value in the set {2.3}, i.e. ID value is 2 or 3



```
select * from product where ID in(2,3);
```

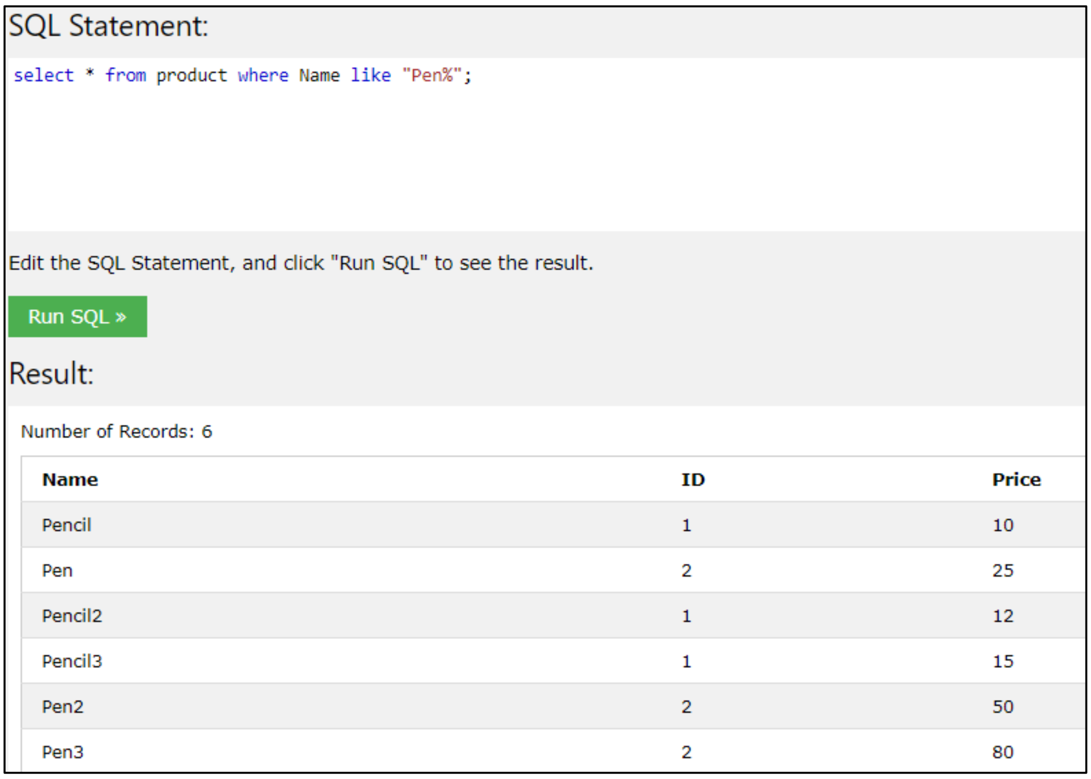
Edit the SQL Statement, and click "Run SQL" to see the result.
Run SQL »
Result:

Number of Records: 6

Name	ID	Price
Pen	2	25
Sharpner	3	5.75
Pen2	2	50
Pen3	2	80
Sharpner2	3	8
Sharpner3	3	30

e) LIKE operator

To select the records of product that have the value of Name starting with the substring "Pen"



```
select * from product where Name like "Pen%";
```

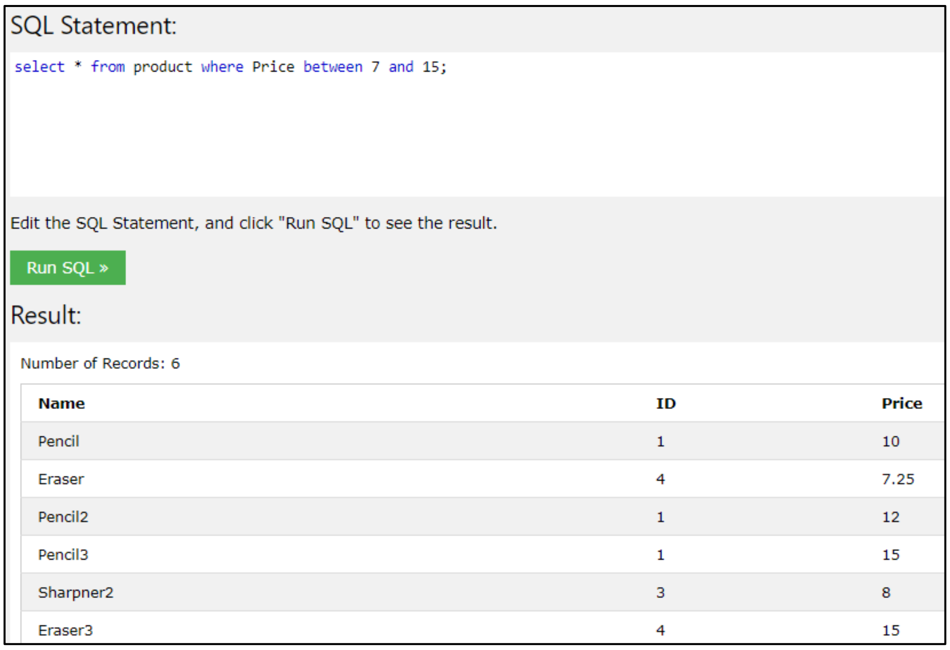
Edit the SQL Statement, and click "Run SQL" to see the result.
Run SQL »
Result:

Number of Records: 6

Name	ID	Price
Pencil	1	10
Pen	2	25
Pencil2	1	12
Pencil3	1	15
Pen2	2	50
Pen3	2	80

f) BETWEEN operator

To select the records of product that have the value of Price in the range 7-15 (inclusive)



SQL Statement:

```
select * from product where Price between 7 and 15;
```

Edit the SQL Statement, and click "Run SQL" to see the result.

Run SQL »

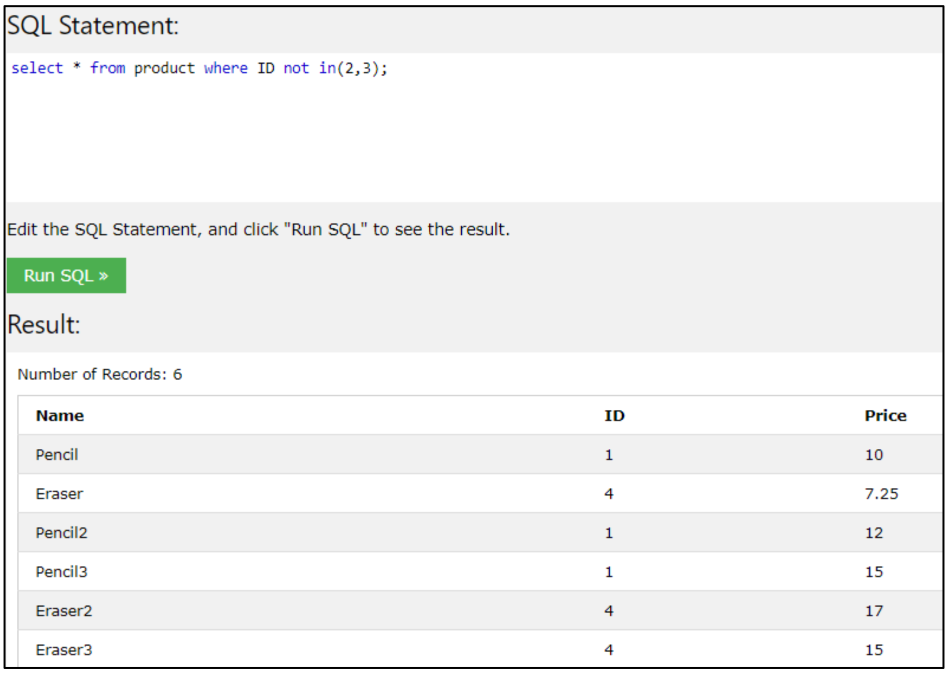
Result:

Number of Records: 6

Name	ID	Price
Pencil	1	10
Eraser	4	7.25
Pencil2	1	12
Pencil3	1	15
Sharpner2	3	8
Eraser3	4	15

g) NOT operator

To select the records of product that don't have the value of ID in the set {2,3}, i.e. ID is neither 2 nor 3



SQL Statement:

```
select * from product where ID not in(2,3);
```

Edit the SQL Statement, and click "Run SQL" to see the result.

Run SQL »

Result:

Number of Records: 6

Name	ID	Price
Pencil	1	10
Eraser	4	7.25
Pencil2	1	12
Pencil3	1	15
Eraser2	4	17
Eraser3	4	15

RESULT: Thus the operation of database operators was successfully implemented using SQL query.

------ X ------

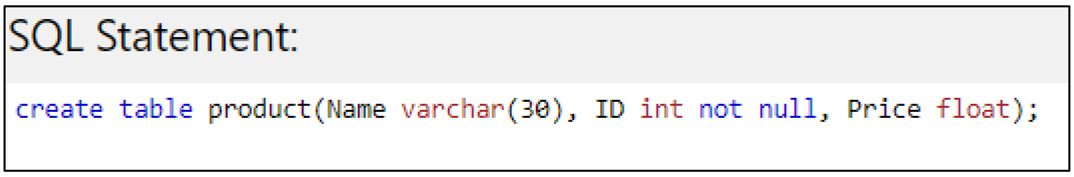
Exp. No.: 04	**SQL Constraints**

Aim: To demonstrate the working of SQL Constraints

Queries and Corresponding Outputs:

a) NOT NULL constraint

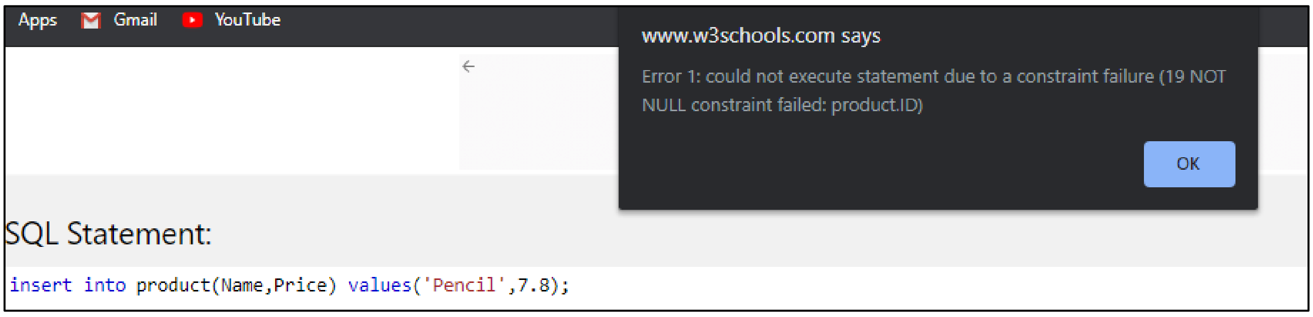
To create a table 'product' whose attribute 'ID' has the constraint 'NOT NULL'



SQL Statement:

```
create table product(Name varchar(30), ID int not null, Price float);
```

- Insertion of a record that has no value for ID, i.e. NULL



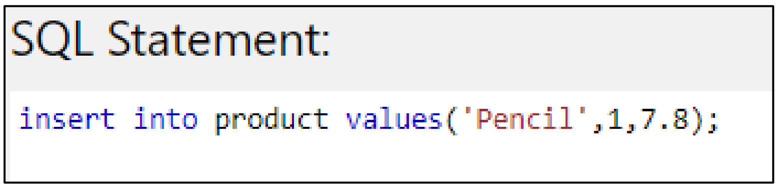
www.w3schools.com says

Error 1: could not execute statement due to a constraint failure (19 NOT NULL constraint failed: product.ID)

SQL Statement:

```
insert into product(Name,Price) values('Pencil',7.8);
```

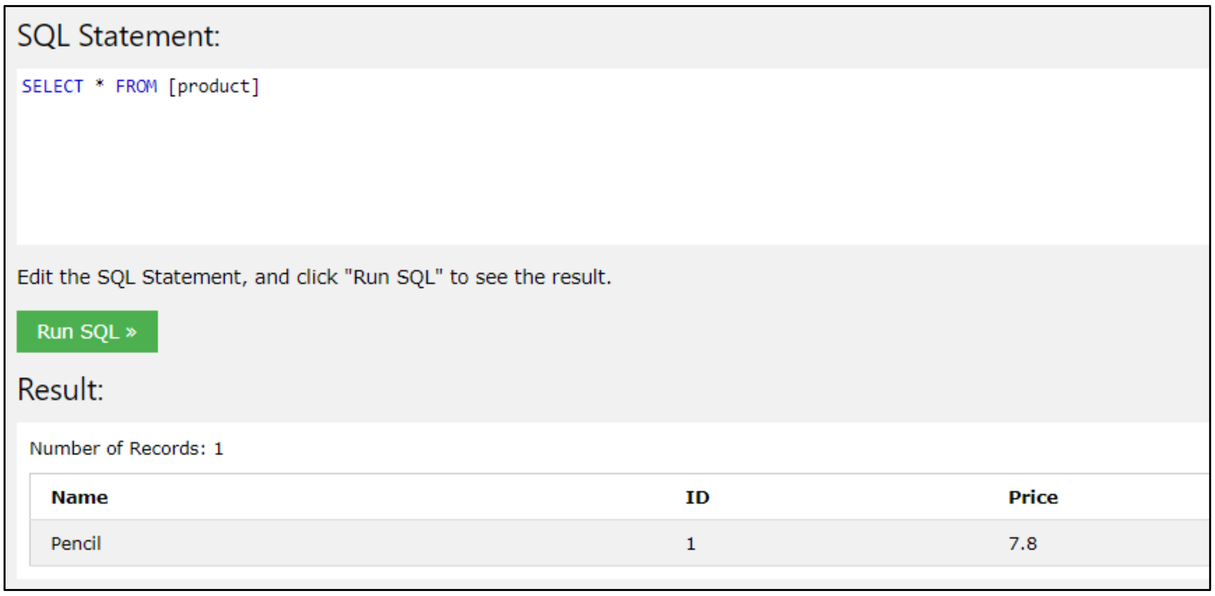
- Insertion of a record where ID value is given



SQL Statement:

```
insert into product values('Pencil',1,7.8);
```

- Table 'product'



SQL Statement:

```
SELECT * FROM [product]
```

Edit the SQL Statement, and click "Run SQL" to see the result.

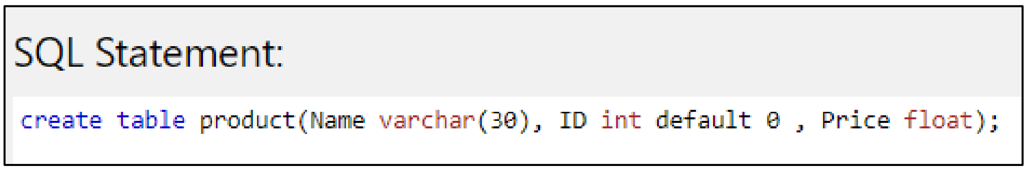
Result:

Number of Records: 1

Name	ID	Price
Pencil	1	7.8

b) DEFAULT constraint

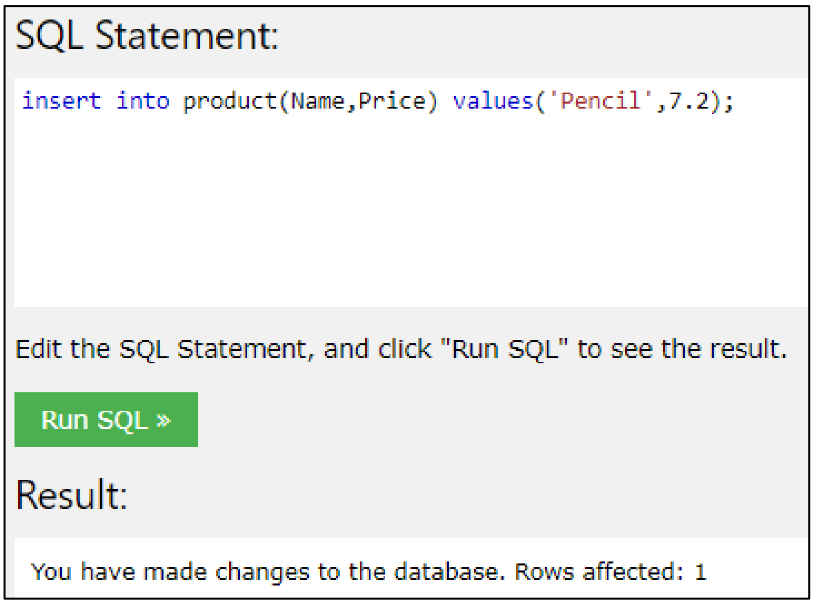
To create a table ‘product’ whose attribute ‘ID’ has the constraint ‘default’ with the value 0



SQL Statement:

```
create table product(Name varchar(30), ID int default 0 , Price float);
```

- Insertion of a record that has no explicitly mentioned value for ID



SQL Statement:

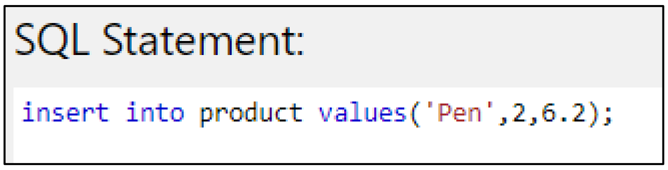
```
insert into product(Name,Price) values('Pencil',7.2);
```

Edit the SQL Statement, and click "Run SQL" to see the result.

Result:

```
You have made changes to the database. Rows affected: 1
```

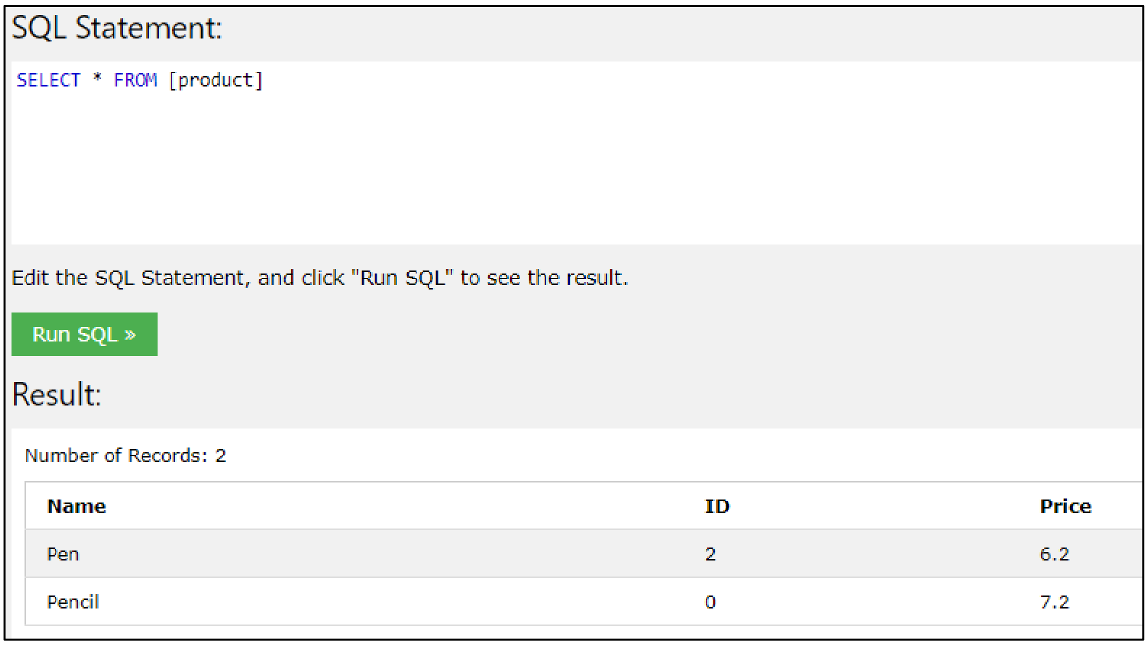
- Insertion of a record that has values of all attributes mentioned in the query



SQL Statement:

```
insert into product values('Pen',2,6.2);
```

- Table ‘product’



SQL Statement:

```
SELECT * FROM [product]
```

Edit the SQL Statement, and click "Run SQL" to see the result.

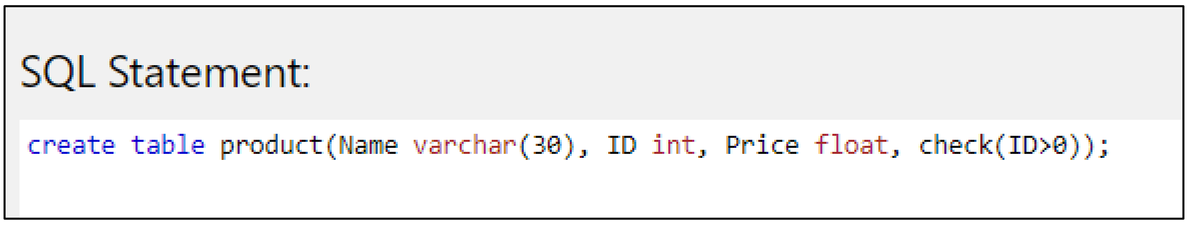
Result:

Number of Records: 2

Name	ID	Price
Pen	2	6.2
Pencil	0	7.2

c) CHECK constraint

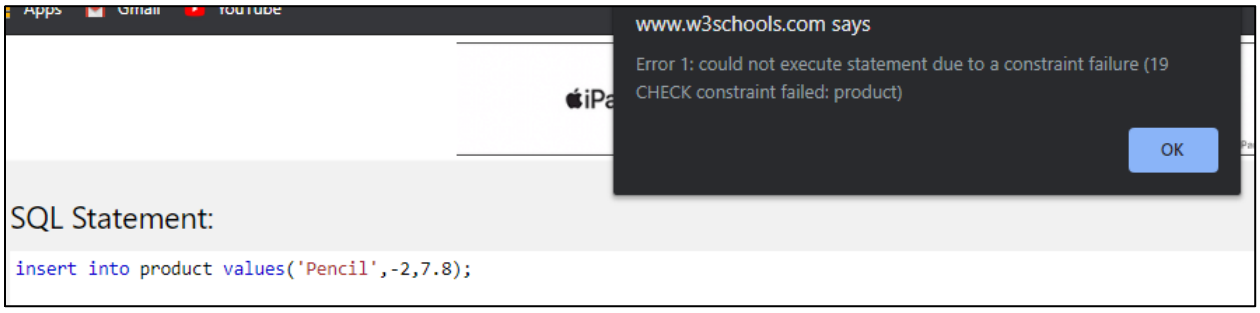
To create a table 'product' that accepts only those records that have a positive value of 'ID'



SQL Statement:

```
create table product(Name varchar(30), ID int, Price float, check(ID>0));
```

- Insertion of a record that has negative value of 'ID'



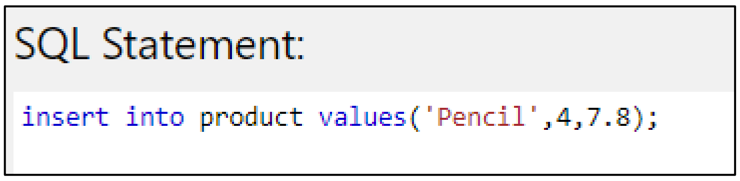
www.w3schools.com says

Error 1: could not execute statement due to a constraint failure (19 CHECK constraint failed: product)

SQL Statement:

```
insert into product values('Pencil',-2,7.8);
```

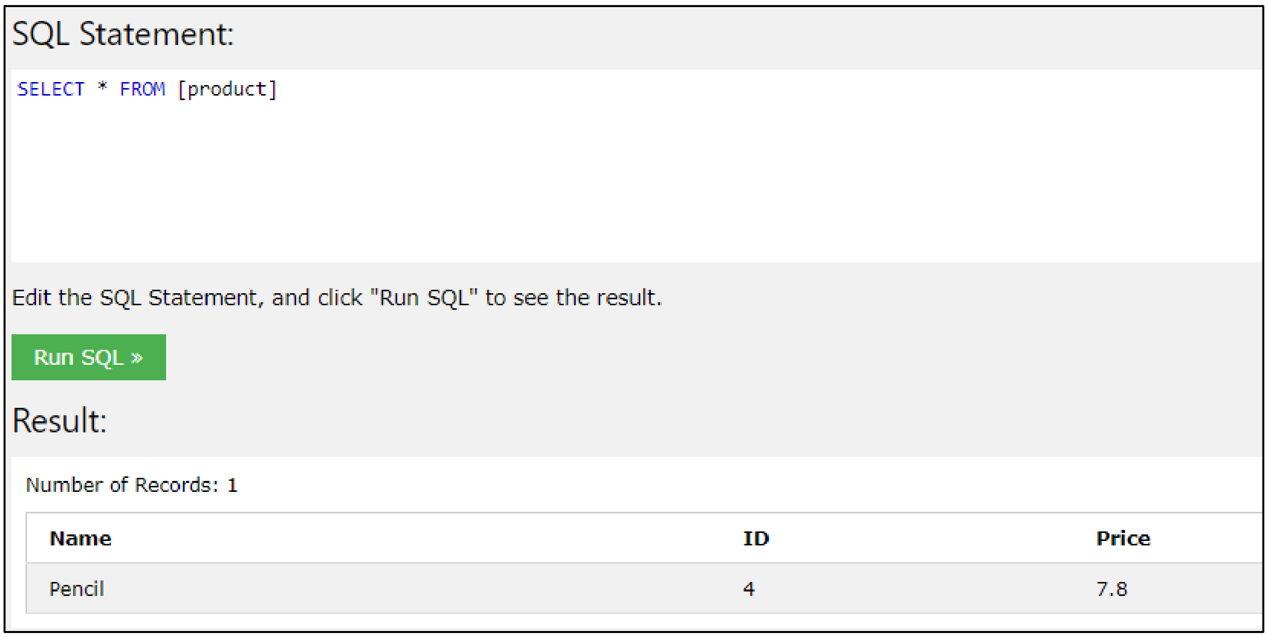
- Insertion of a record that has positive value of 'ID'



SQL Statement:

```
insert into product values('Pencil',4,7.8);
```

- Table 'product'



SQL Statement:

```
SELECT * FROM [product]
```

Edit the SQL Statement, and click "Run SQL" to see the result.

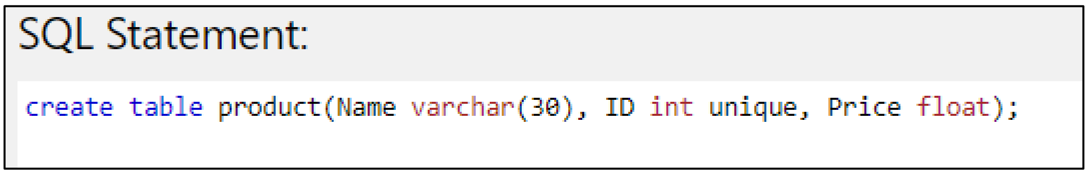
Result:

Number of Records: 1

Name	ID	Price
Pencil	4	7.8

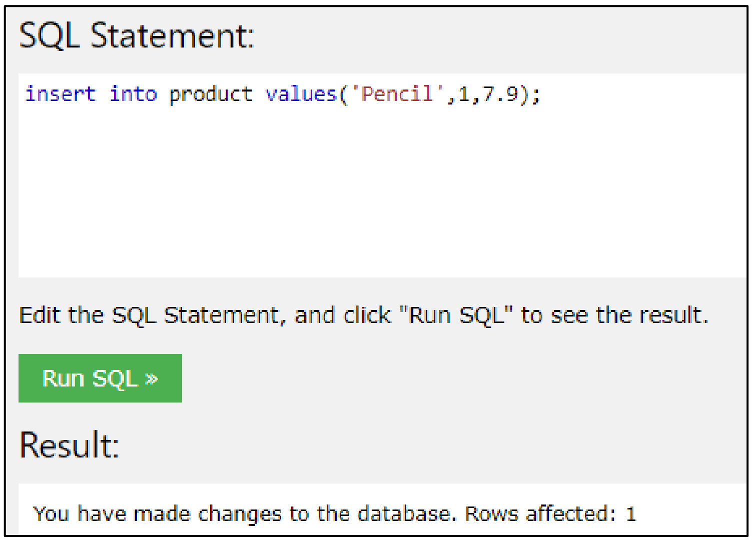
d) UNIQUE constraint

To create a table 'product' whose attribute 'ID' has the constraint 'unique'


SQL Statement:

```
create table product(Name varchar(30), ID int unique, Price float);
```

- Inserting a record that has value of 'ID' as 1


SQL Statement:

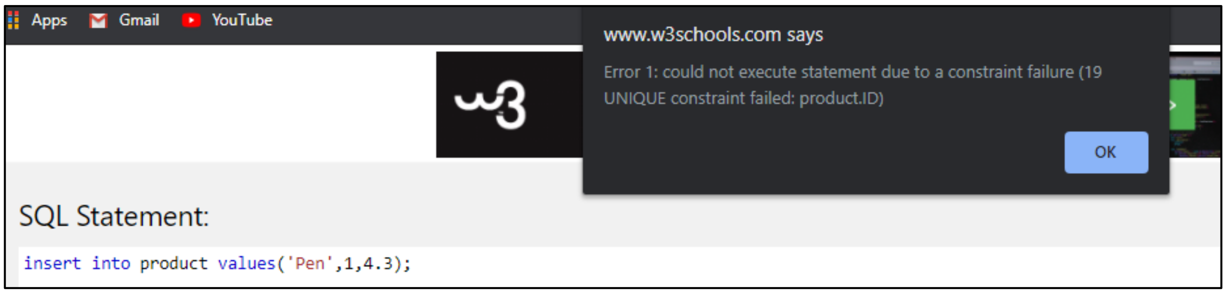
```
insert into product values('Pencil',1,7.9);
```

Edit the SQL Statement, and click "Run SQL" to see the result.

Result:

```
You have made changes to the database. Rows affected: 1
```

- Insertion of another record that has value of 'ID' as 1



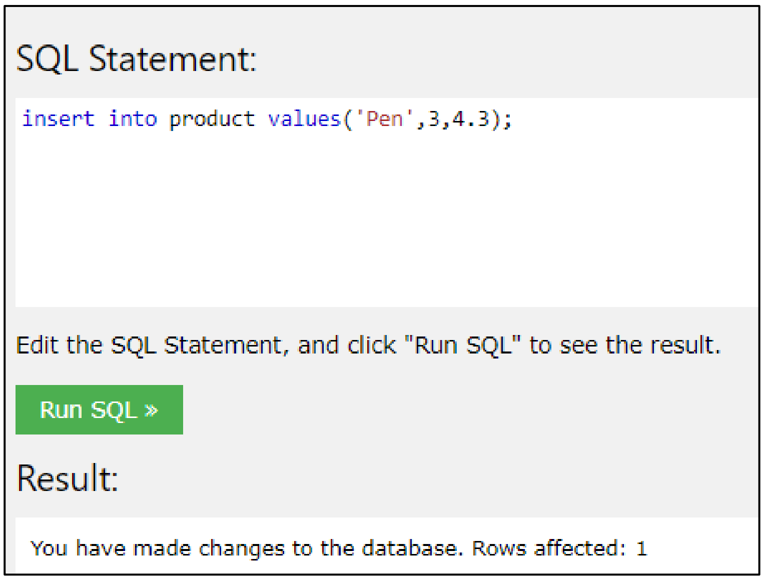
www.w3schools.com says

```
Error 1: could not execute statement due to a constraint failure (19 UNIQUE constraint failed: product.ID)
```

SQL Statement:

```
insert into product values('Pen',1,4.3);
```

- Insertion of a record that has value of 'ID' as 3


SQL Statement:

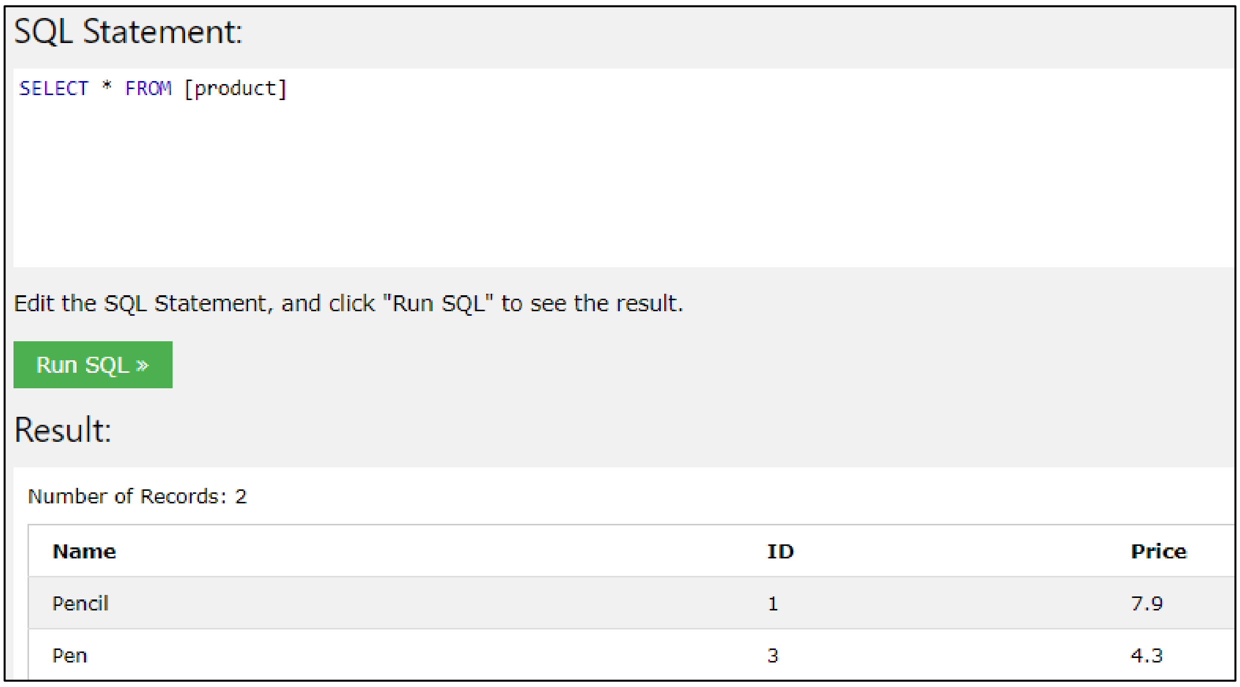
```
insert into product values('Pen',3,4.3);
```

Edit the SQL Statement, and click "Run SQL" to see the result.

Result:

```
You have made changes to the database. Rows affected: 1
```

- Table ‘product’



```
SELECT * FROM [product]
```

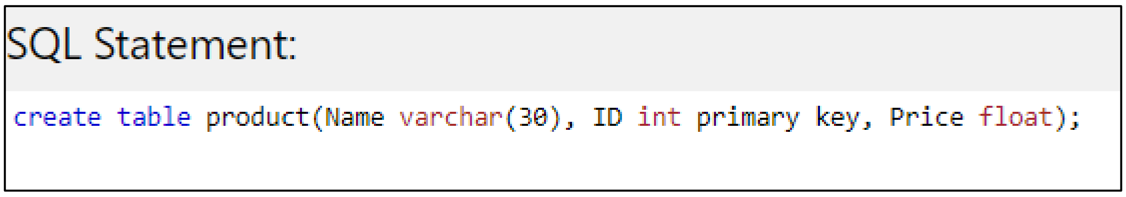
Edit the SQL Statement, and click "Run SQL" to see the result.

Number of Records: 2

Name	ID	Price
Pencil	1	7.9
Pen	3	4.3

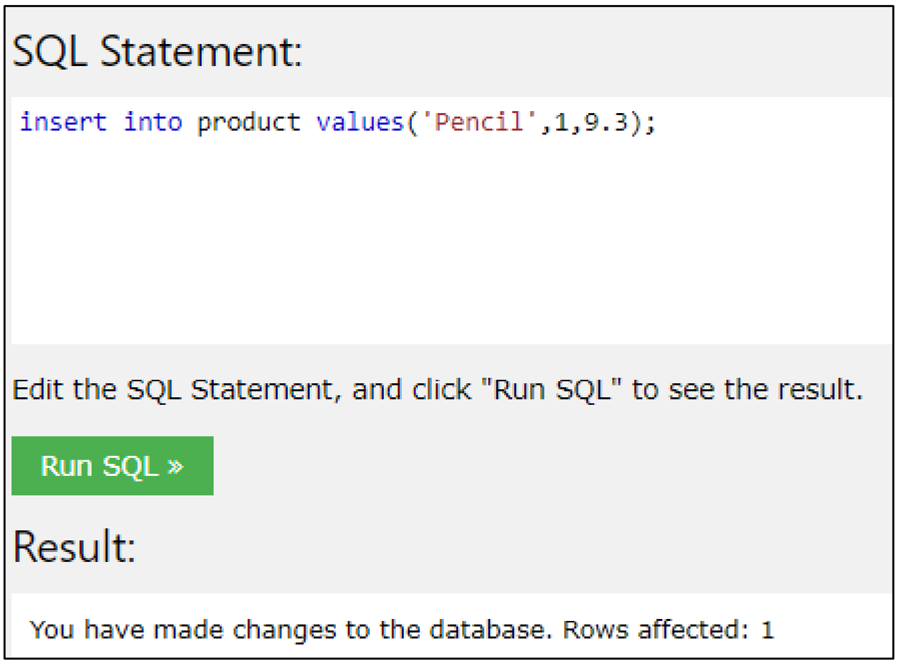
e) PRIMARY KEY constraint

To create a table ‘product’ with the attribute ‘ID’ as its primary key



```
create table product(Name varchar(30), ID int primary key, Price float);
```

- Inserting a record that has value of ‘ID’ as 1

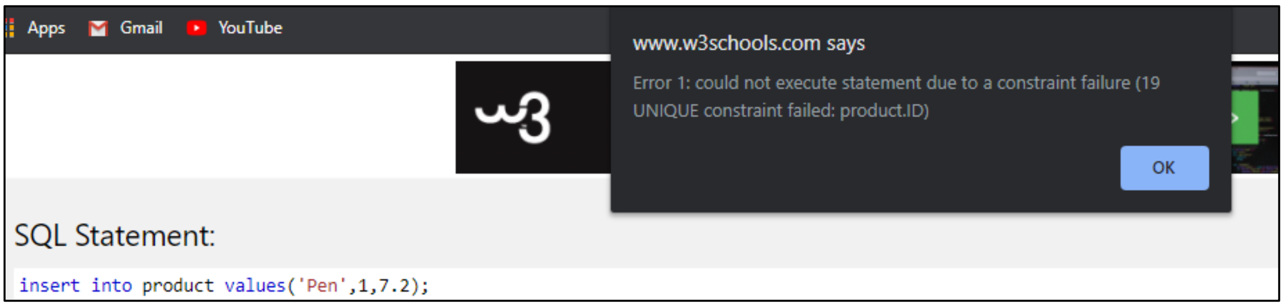


```
insert into product values('Pencil',1,9.3);
```

Edit the SQL Statement, and click "Run SQL" to see the result.

You have made changes to the database. Rows affected: 1

- Insertion of another record that has value of 'ID' as 1



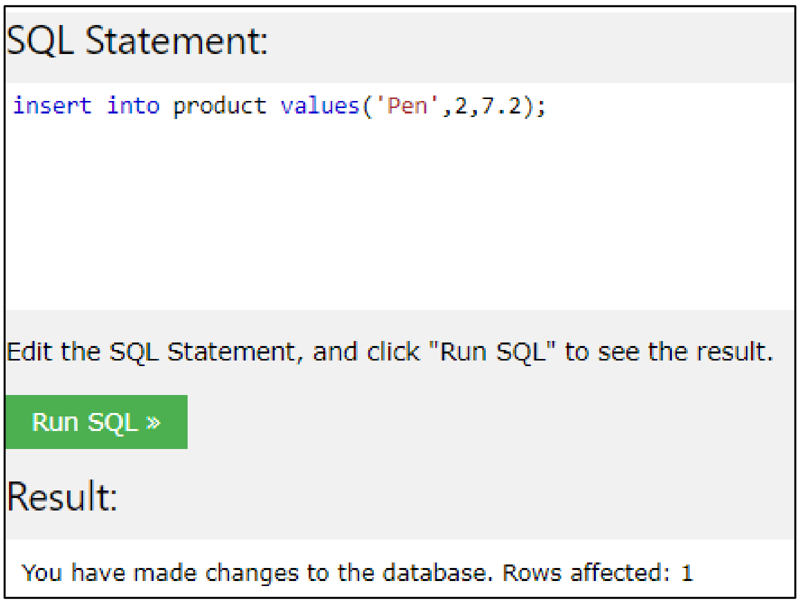
www.w3schools.com says

Error 1: could not execute statement due to a constraint failure (19 UNIQUE constraint failed: product.ID)

SQL Statement:

```
insert into product values('Pen',1,7.2);
```

- Insertion of a record that has value of 'ID' as 2



SQL Statement:

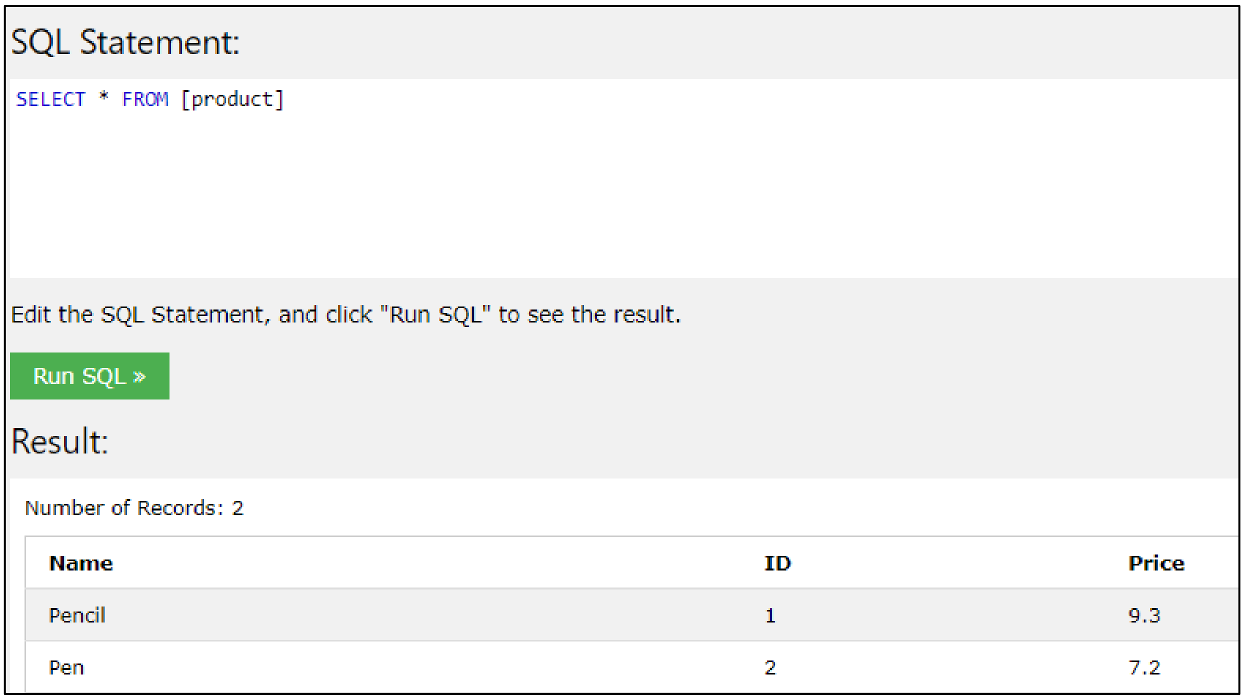
```
insert into product values('Pen',2,7.2);
```

Edit the SQL Statement, and click "Run SQL" to see the result.

Result:

You have made changes to the database. Rows affected: 1

- Table 'product'



SQL Statement:

```
SELECT * FROM [product]
```

Edit the SQL Statement, and click "Run SQL" to see the result.

Result:

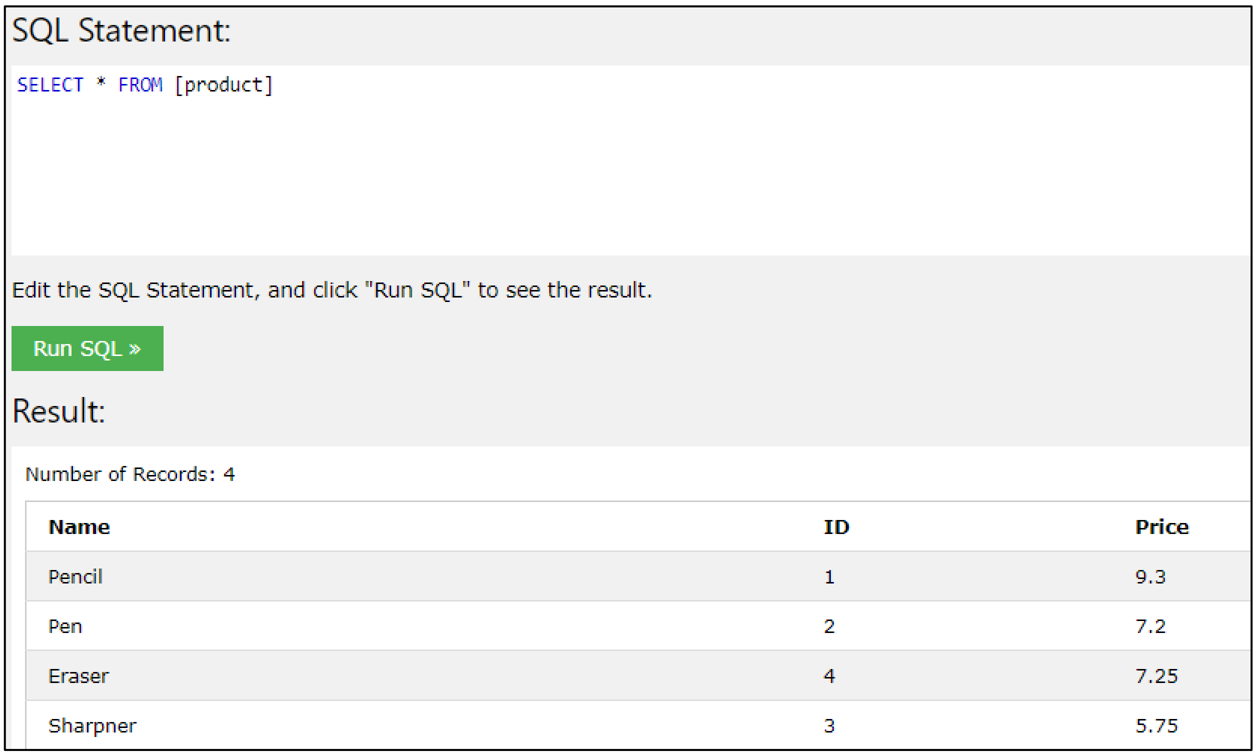
Number of Records: 2

Name	ID	Price
Pencil	1	9.3
Pen	2	7.2

f) INDEX constraint

To create an index on the attribute 'ID' of table 'product'

- Existing table



SQL Statement:

```
SELECT * FROM [product]
```

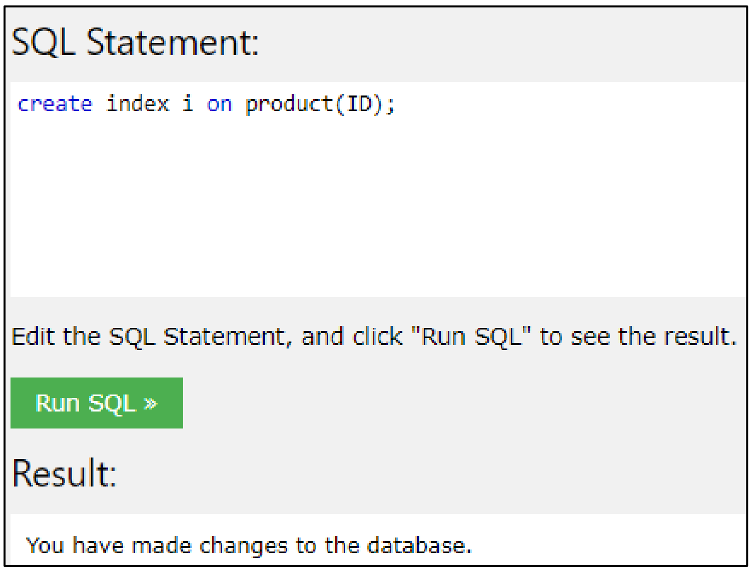
Edit the SQL Statement, and click "Run SQL" to see the result.

Result:

Number of Records: 4

Name	ID	Price
Pencil	1	9.3
Pen	2	7.2
Eraser	4	7.25
Sharpner	3	5.75

- Creating index on 'ID' of 'product'



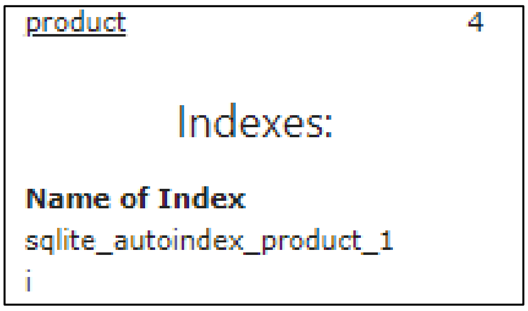
SQL Statement:

```
create index i on product(ID);
```

Edit the SQL Statement, and click "Run SQL" to see the result.

Result:

You have made changes to the database.

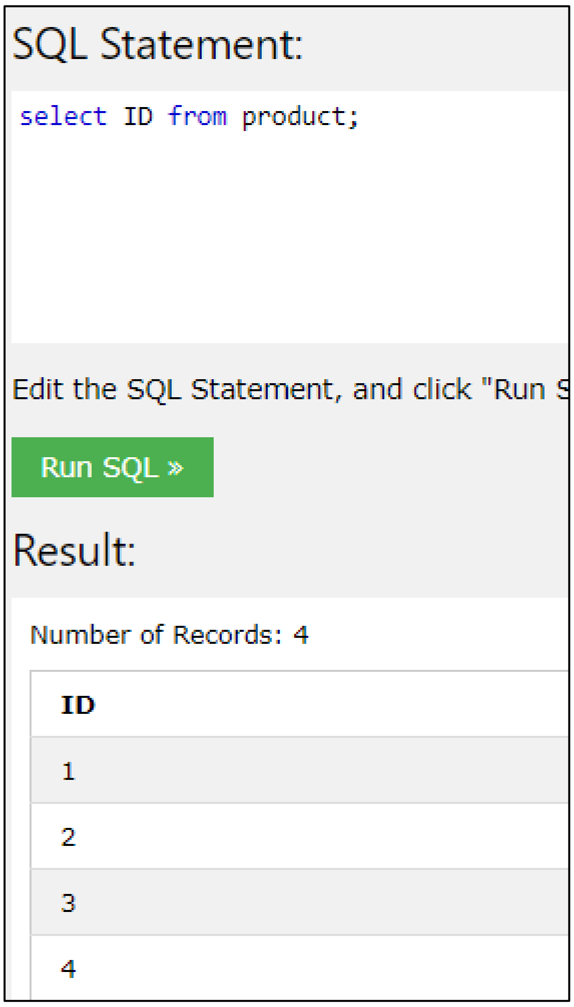


product 4

Indexes:

Name of Index
sqlite_autoindex_product_1
i

- Selecting values of 'ID' from 'product'



SQL Statement:

```
select ID from product;
```

Edit the SQL Statement, and click "Run S

Result:

Number of Records: 4

ID
1
2
3
4

RESULT: Thus the operation of database constraints was successfully implemented using SQL query.

------ X ------

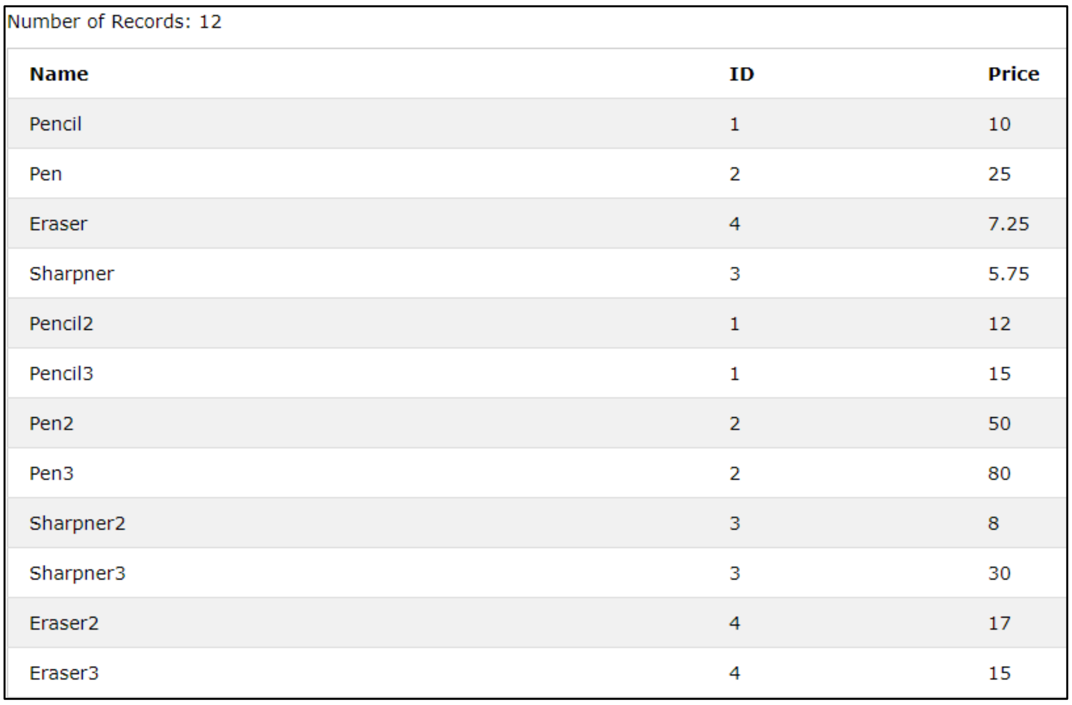
Exp. No.: 05

SQL Sub-Queries

Aim: To demonstrate the working of SQL Sub-Queries

Queries and Corresponding Outputs:

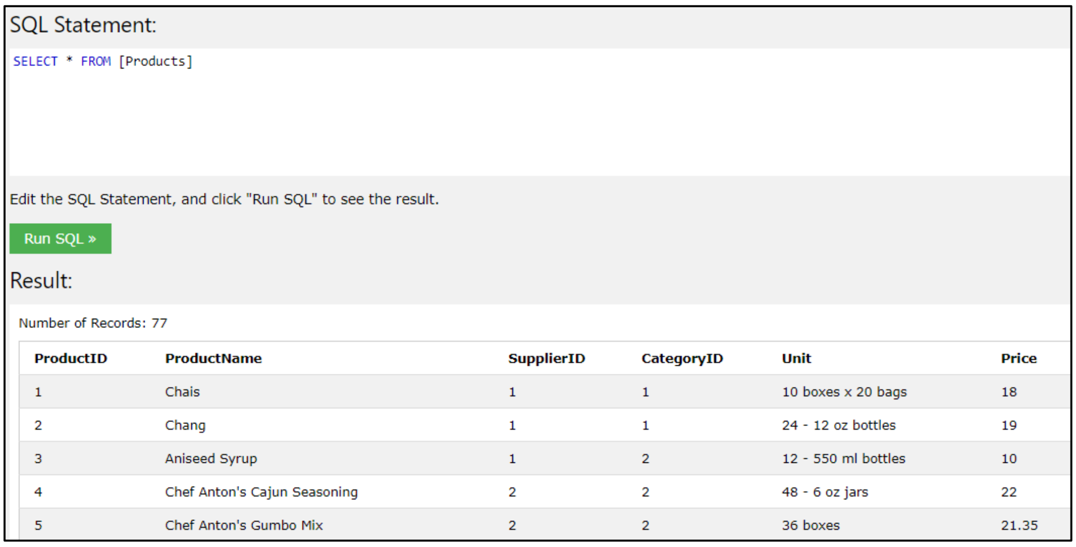
➔ Existing table 'product'



Number of Records: 12

Name	ID	Price
Pencil	1	10
Pen	2	25
Eraser	4	7.25
Sharpner	3	5.75
Pencil2	1	12
Pencil3	1	15
Pen2	2	50
Pen3	2	80
Sharpner2	3	8
Sharpner3	3	30
Eraser2	4	17
Eraser3	4	15

➔ Existing table 'Products'



SQL Statement:

```
SELECT * FROM [Products]
```

Edit the SQL Statement, and click "Run SQL" to see the result.

Result:

Number of Records: 77

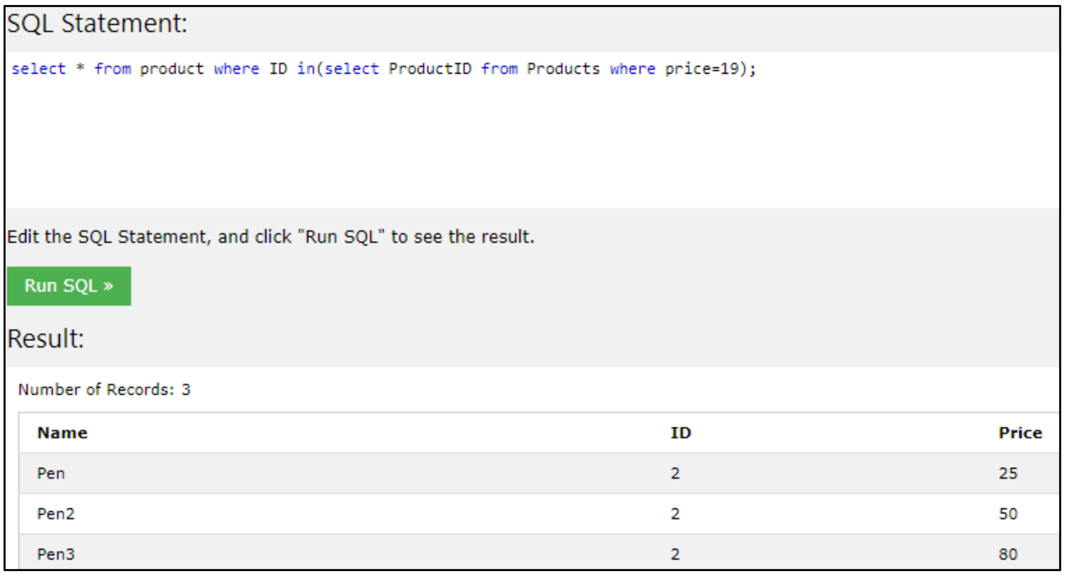
ProductID	ProductName	SupplierID	CategoryID	Unit	Price
1	Chais	1	1	10 boxes x 20 bags	18
2	Chang	1	1	24 - 12 oz bottles	19
3	Aniseed Syrup	1	2	12 - 550 ml bottles	10
4	Chef Anton's Cajun Seasoning	2	2	48 - 6 oz jars	22
5	Chef Anton's Gumbo Mix	2	2	36 boxes	21.35

a) Single Row Sub-Query

To select all the records from 'product' that have the value of 'ID' same as 'ProductID' of records of 'Products' whose 'price' is 19

Main Query: `select * from product where ID in (select ProductID from Products where price=19);`

Sub-Query: `select ProductID from Products where price=19;`


SQL Statement:

```
select * from product where ID in(select ProductID from Products where price=19);
```

Edit the SQL Statement, and click "Run SQL" to see the result.

Result:

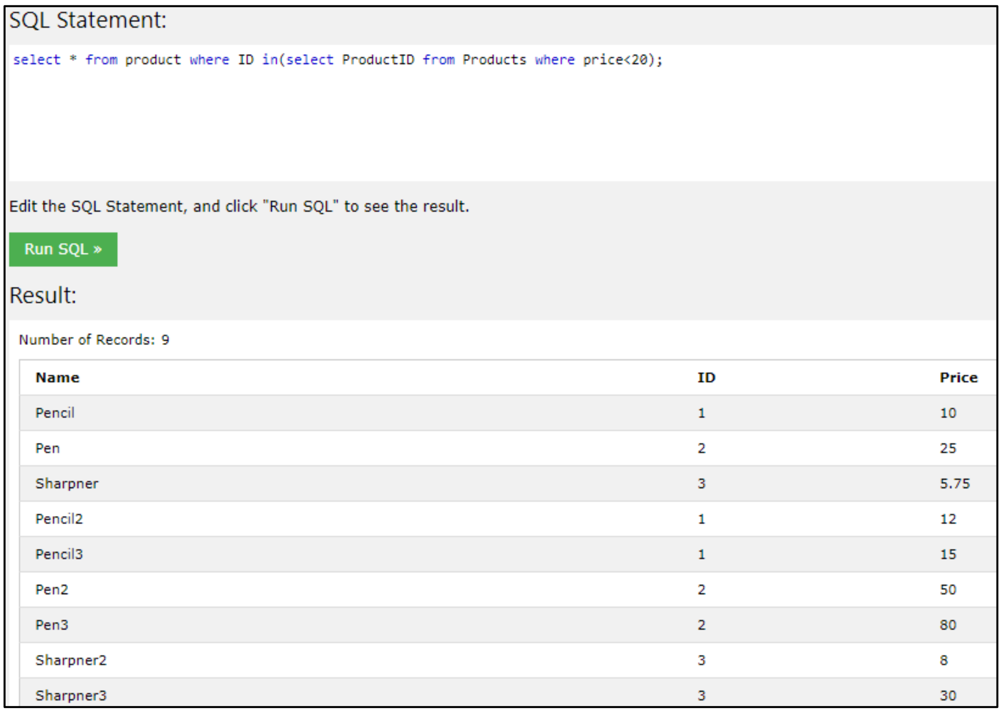
Number of Records: 3

Name	ID	Price
Pen	2	25
Pen2	2	50
Pen3	2	80

b) Multiple Row Sub-Query

To select all the records from 'product' that have the value of 'ID' same as 'ProductID' of records of 'Products' whose 'price' is less than 20

Main Query: `select * from product where ID in (select ProductID from Products where price<20);`
Sub-Query: `select ProductID from Products where price<20;`


SQL Statement:

```
select * from product where ID in(select ProductID from Products where price<20);
```

Edit the SQL Statement, and click "Run SQL" to see the result.

Result:

Number of Records: 9

Name	ID	Price
Pencil	1	10
Pen	2	25
Sharpner	3	5.75
Pencil2	1	12
Pencil3	1	15
Pen2	2	50
Pen3	2	80
Sharpner2	3	8
Sharpner3	3	30

RESULT: Thus the operation of database sub queries was successfully implemented using SQL query.

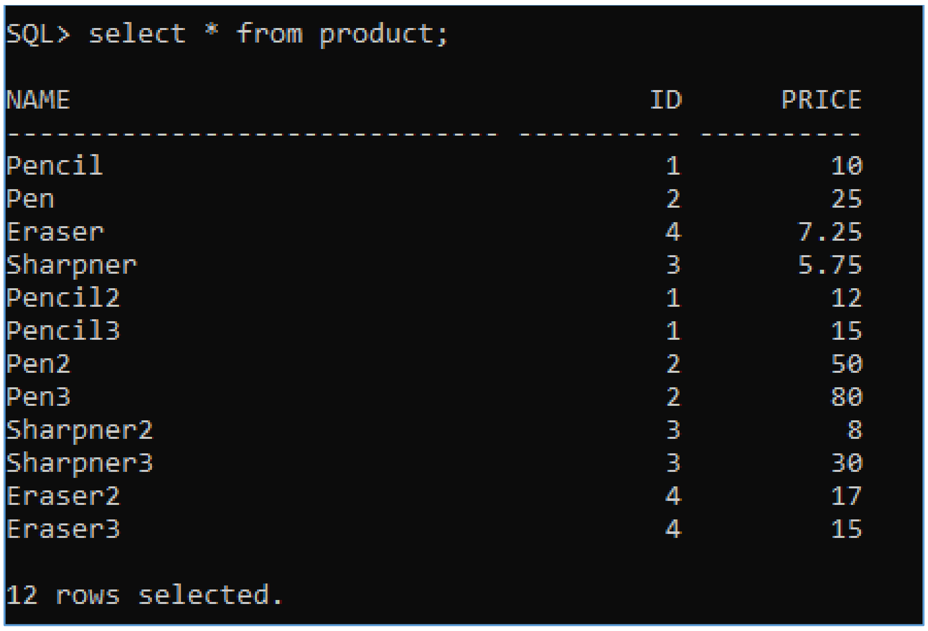
------ X ------

Exp. No.: 06	**SQL Join Queries**

Aim: To demonstrate the working of SQL Join Queries

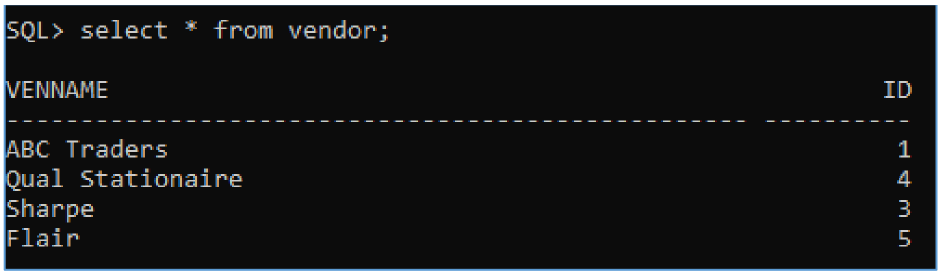
Queries and Corresponding Outputs:

➔ Existing table ‘product’



```
SQL> select * from product;

NAME                                   ID      PRICE
------------------------------ ---------- ----------
Pencil                                  1         10
Pen                                     2         25
Eraser                                  4       7.25
Sharpner                                3       5.75
Pencil2                                 1         12
Pencil3                                 1         15
Pen2                                    2         50
Pen3                                    2         80
Sharpner2                               3          8
Sharpner3                               3         30
Eraser2                                 4         17
Eraser3                                 4         15

12 rows selected.
```

➔ Existing table ‘vendor’

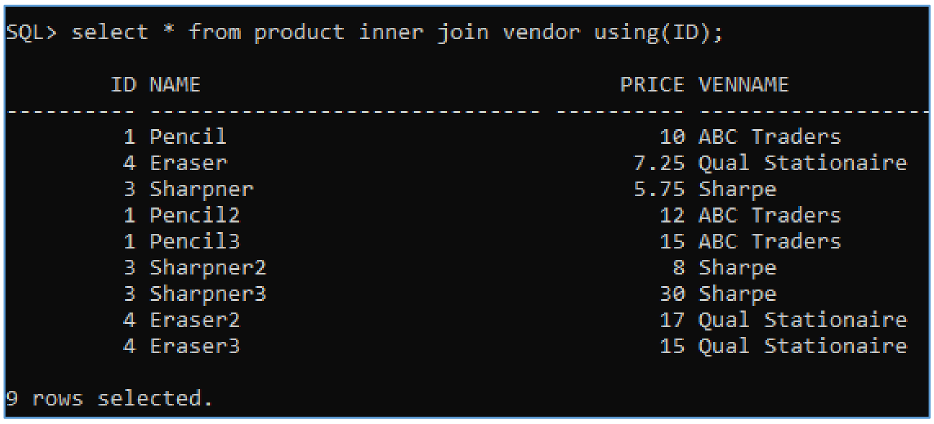


```
SQL> select * from vendor;

VENNAME                                                    ID
-------------------------------------------------- ----------
ABC Traders                                                 1
Qual Stationaire                                            4
Sharpe                                                      3
Flair                                                       5
```

a) Inner Join

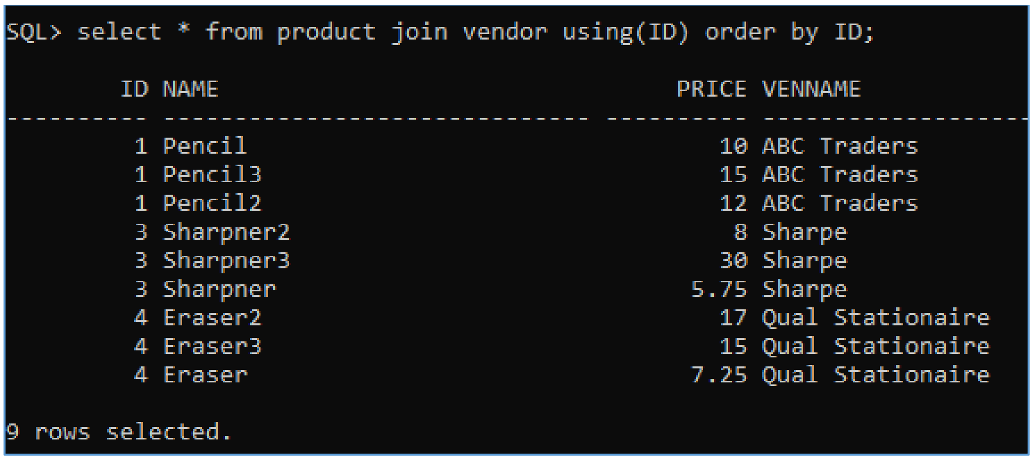
i. Keyword: Using()

To select all the records from the inner join of ‘product’ and ‘vendor’ using attribute ‘ID’



```
SQL> select * from product inner join vendor using(ID);

        ID NAME                                PRICE VENNAME
---------- ------------------------------ ---------- -----------------
         1 Pencil                                 10 ABC Traders
         4 Eraser                               7.25 Qual Stationaire
         3 Sharpner                             5.75 Sharpe
         1 Pencil2                                12 ABC Traders
         1 Pencil3                                15 ABC Traders
         3 Sharpner2                               8 Sharpe
         3 Sharpner3                              30 Sharpe
         4 Eraser2                                17 Qual Stationaire
         4 Eraser3                                15 Qual Stationaire

9 rows selected.
```

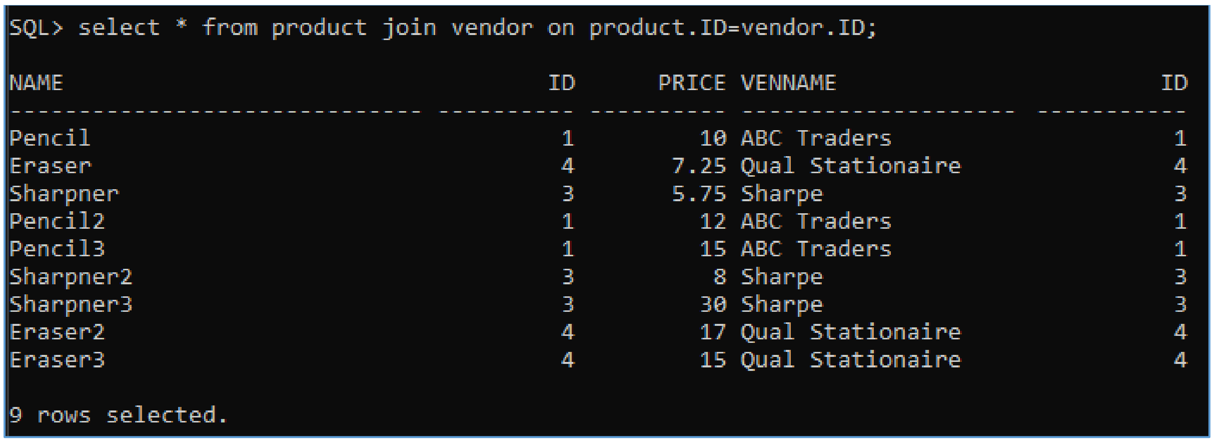
To select all the records from the inner join of ‘product’ and ‘vendor’ using attribute ‘ID’ and arrange the records of the result table in increasing order of ‘ID’



```
SQL> select * from product join vendor using(ID) order by ID;

        ID NAME                                PRICE VENNAME
---------- ------------------------------ ---------- -------------------
         1 Pencil                                 10 ABC Traders
         1 Pencil3                                15 ABC Traders
         1 Pencil2                                12 ABC Traders
         3 Sharpner2                               8 Sharpe
         3 Sharpner3                              30 Sharpe
         3 Sharpner                             5.75 Sharpe
         4 Eraser2                                17 Qual Stationaire
         4 Eraser3                                15 Qual Stationaire
         4 Eraser                               7.25 Qual Stationaire

9 rows selected.
```

ii. Keyword: on

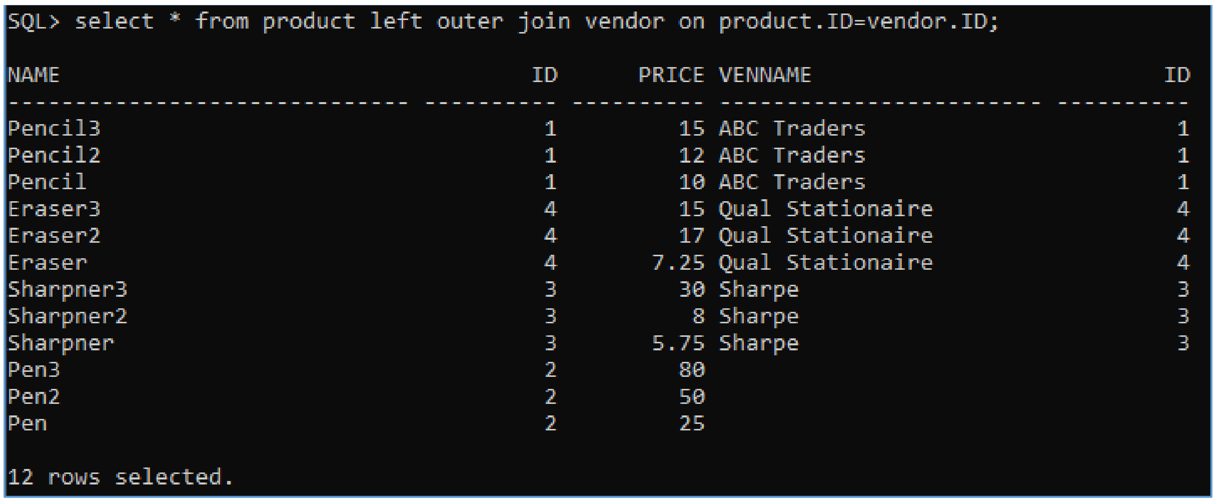
To select all the records from the inner join of ‘product’ and ‘vendor’ where they are joined on the attribute ‘ID’



```
SQL> select * from product join vendor on product.ID=vendor.ID;

NAME                                   ID      PRICE VENNAME                      ID
------------------------------ ---------- ---------- -------------------- ----------
Pencil                                  1         10 ABC Traders                   1
Eraser                                  4       7.25 Qual Stationaire              4
Sharpner                                3       5.75 Sharpe                        3
Pencil2                                 1         12 ABC Traders                   1
Pencil3                                 1         15 ABC Traders                   1
Sharpner2                               3          8 Sharpe                        3
Sharpner3                               3         30 Sharpe                        3
Eraser2                                 4         17 Qual Stationaire              4
Eraser3                                 4         15 Qual Stationaire              4

9 rows selected.
```

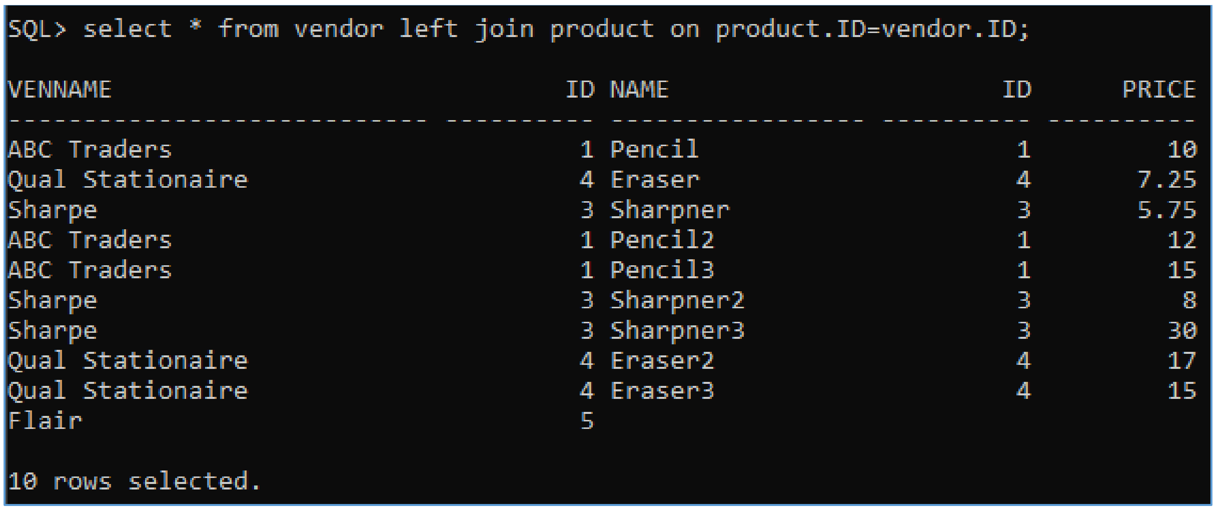
b) Left Outer Join

i. To select all the records from the left outer join of ‘product’ on ‘vendor’ where they are joined on the attribute ‘ID’



```
SQL> select * from product left outer join vendor on product.ID=vendor.ID;

NAME                                   ID      PRICE VENNAME                          ID
------------------------------ ---------- ---------- ------------------------- ----------
Pencil3                                 1         15 ABC Traders                       1
Pencil2                                 1         12 ABC Traders                       1
Pencil                                  1         10 ABC Traders                       1
Eraser3                                 4         15 Qual Stationaire                  4
Eraser2                                 4         17 Qual Stationaire                  4
Eraser                                  4       7.25 Qual Stationaire                  4
Sharpner3                               3         30 Sharpe                            3
Sharpner2                               3          8 Sharpe                            3
Sharpner                                3       5.75 Sharpe                            3
Pen3                                    2         80
Pen2                                    2         50
Pen                                     2         25

12 rows selected.
```

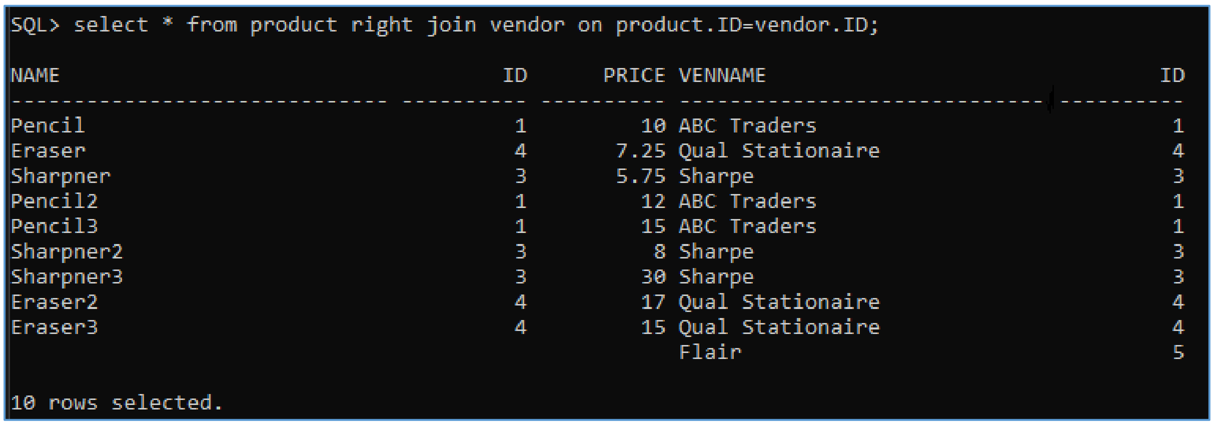
ii. To select all the records from the left outer join of ‘vendor’ on ‘product’ where they are joined on the attribute ‘ID’



```
SQL> select * from vendor left join product on product.ID=vendor.ID;

VENNAME                                ID NAME                      ID      PRICE
---------------------------- ---------- ----------------- ---------- ----------
ABC Traders                             1 Pencil                     1         10
Qual Stationaire                        4 Eraser                     4       7.25
Sharpe                                  3 Sharpner                   3       5.75
ABC Traders                             1 Pencil2                    1         12
ABC Traders                             1 Pencil3                    1         15
Sharpe                                  3 Sharpner2                  3          8
Sharpe                                  3 Sharpner3                  3         30
Qual Stationaire                        4 Eraser2                    4         17
Qual Stationaire                        4 Eraser3                    4         15
Flair                                   5

10 rows selected.
```

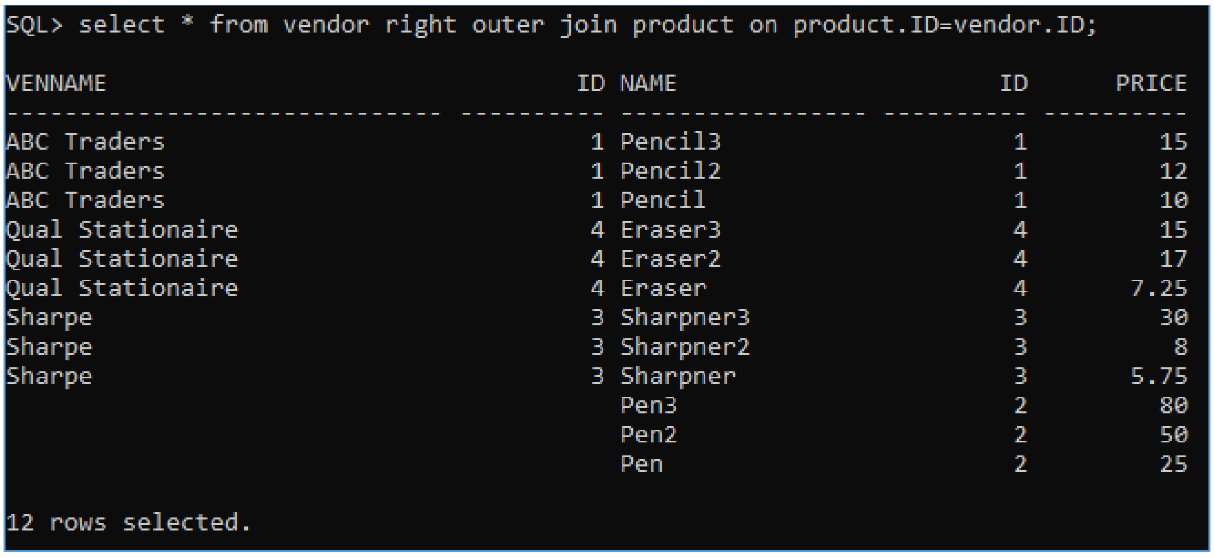
<u>c) Right Outer Join</u>

i. To select all the records from the right outer join of ‘product’ on ‘vendor’ where they are joined on the attribute ‘ID’



```
SQL> select * from product right join vendor on product.ID=vendor.ID;

NAME                                   ID      PRICE VENNAME                              ID
---------------------------- ---------- ---------- ---------------------------- ----------
Pencil                                  1         10 ABC Traders                           1
Eraser                                  4       7.25 Qual Stationaire                      4
Sharpner                                3       5.75 Sharpe                                3
Pencil2                                 1         12 ABC Traders                           1
Pencil3                                 1         15 ABC Traders                           1
Sharpner2                               3          8 Sharpe                                3
Sharpner3                               3         30 Sharpe                                3
Eraser2                                 4         17 Qual Stationaire                      4
Eraser3                                 4         15 Qual Stationaire                      4
                                                     Flair                                 5

10 rows selected.
```

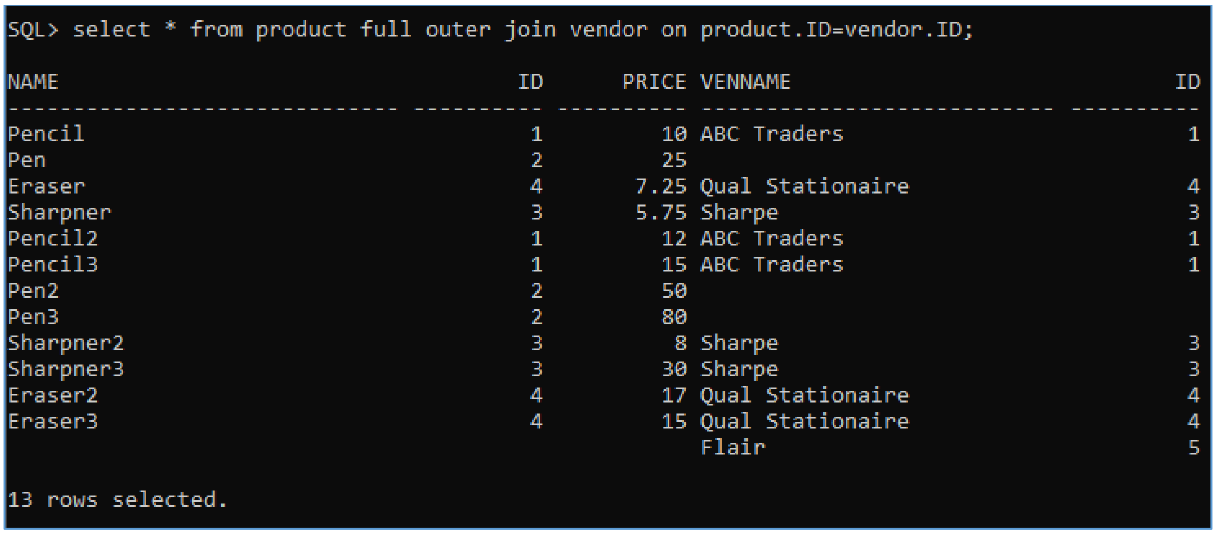
ii. To select all the records from the right outer join of ‘vendor’ on ‘product’ where they are joined on the attribute ‘ID’



```
SQL> select * from vendor right outer join product on product.ID=vendor.ID;

VENNAME                                ID NAME                      ID      PRICE
---------------------------- ---------- ----------------- ---------- ----------
ABC Traders                             1 Pencil3                    1         15
ABC Traders                             1 Pencil2                    1         12
ABC Traders                             1 Pencil                     1         10
Qual Stationaire                        4 Eraser3                    4         15
Qual Stationaire                        4 Eraser2                    4         17
Qual Stationaire                        4 Eraser                     4       7.25
Sharpe                                  3 Sharpner3                  3         30
Sharpe                                  3 Sharpner2                  3          8
Sharpe                                  3 Sharpner                   3       5.75
                                          Pen3                       2         80
                                          Pen2                       2         50
                                          Pen                        2         25

12 rows selected.
```

d) Full Outer Join

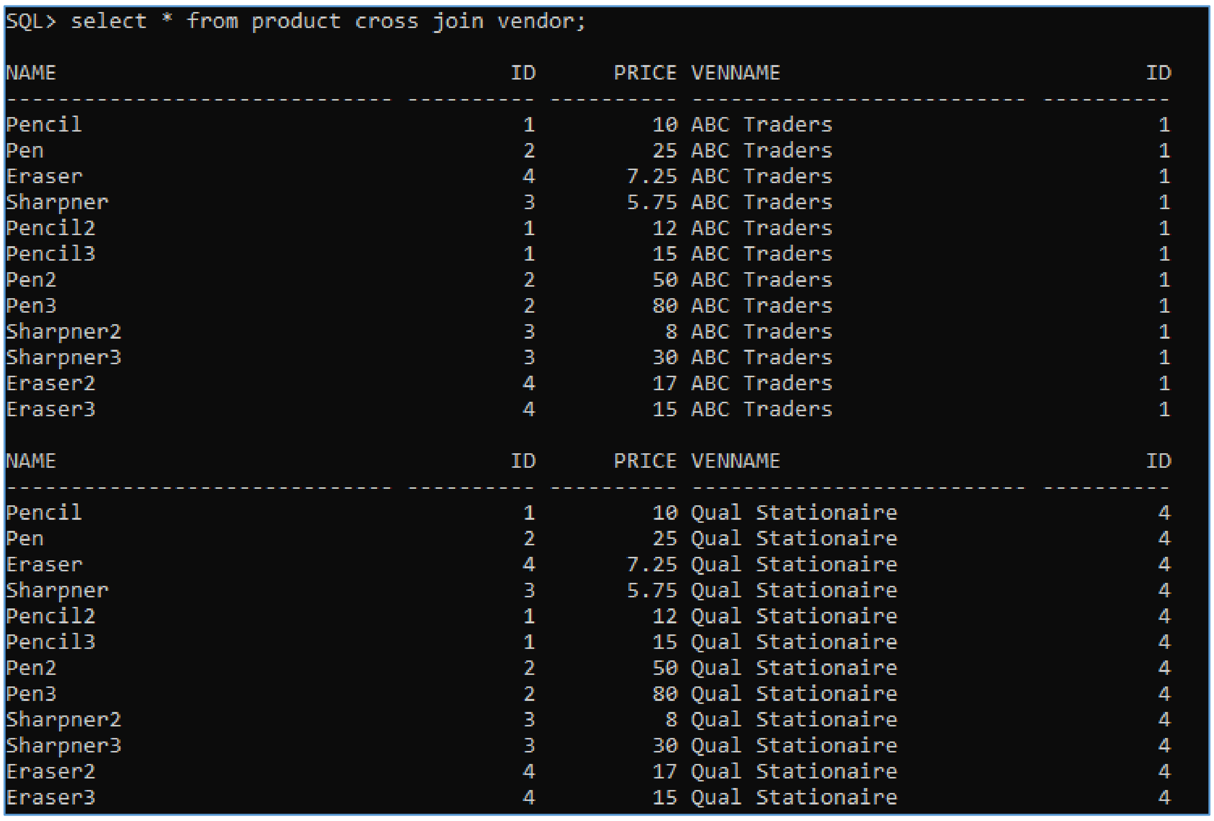
To select all the records from the full outer join of ‘product’ and ‘vendor’ where they are joined on the attribute ‘ID’



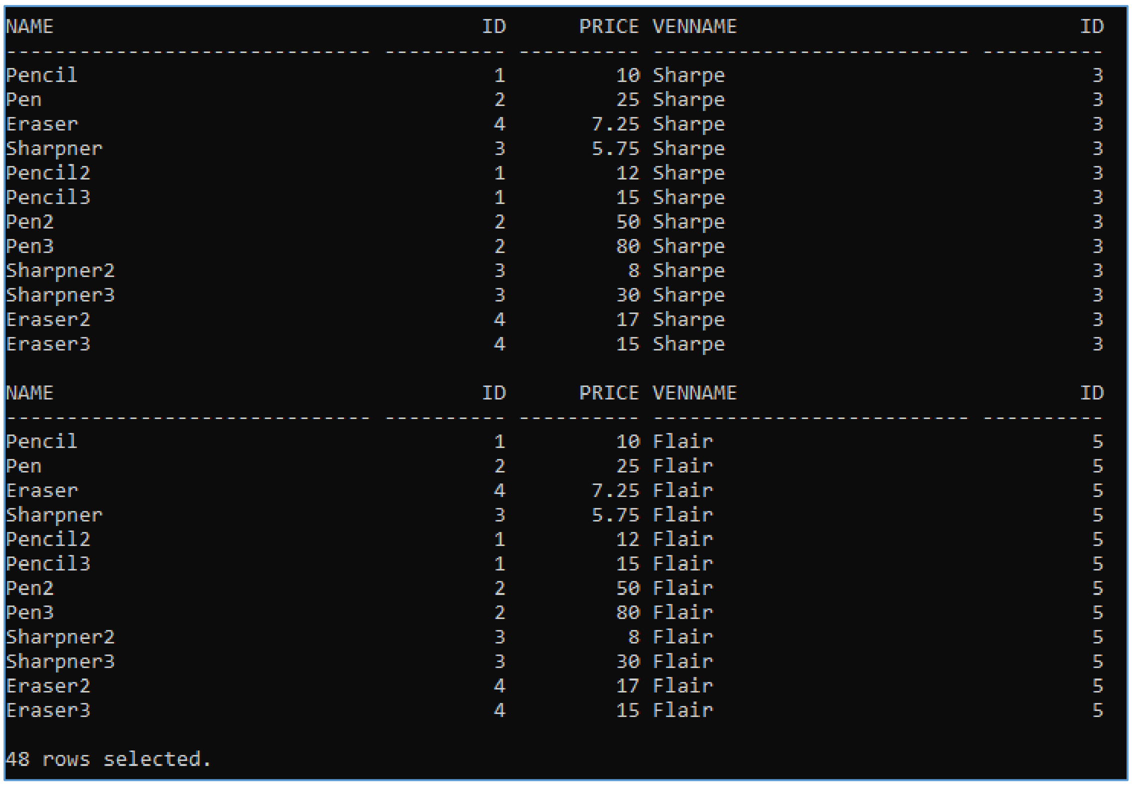
```
SQL> select * from product full outer join vendor on product.ID=vendor.ID;

NAME                                   ID      PRICE VENNAME                                ID
------------------------------ ---------- ---------- -------------------------- ----------
Pencil                                  1         10 ABC Traders                             1
Pen                                     2         25
Eraser                                  4       7.25 Qual Stationaire                        4
Sharpner                                3       5.75 Sharpe                                  3
Pencil2                                 1         12 ABC Traders                             1
Pencil3                                 1         15 ABC Traders                             1
Pen2                                    2         50
Pen3                                    2         80
Sharpner2                               3          8 Sharpe                                  3
Sharpner3                               3         30 Sharpe                                  3
Eraser2                                 4         17 Qual Stationaire                        4
Eraser3                                 4         15 Qual Stationaire                        4
                                                     Flair                                   5

13 rows selected.
```

e) Cross Join

To select all the records from the cross join of ‘product’ and ‘vendor’



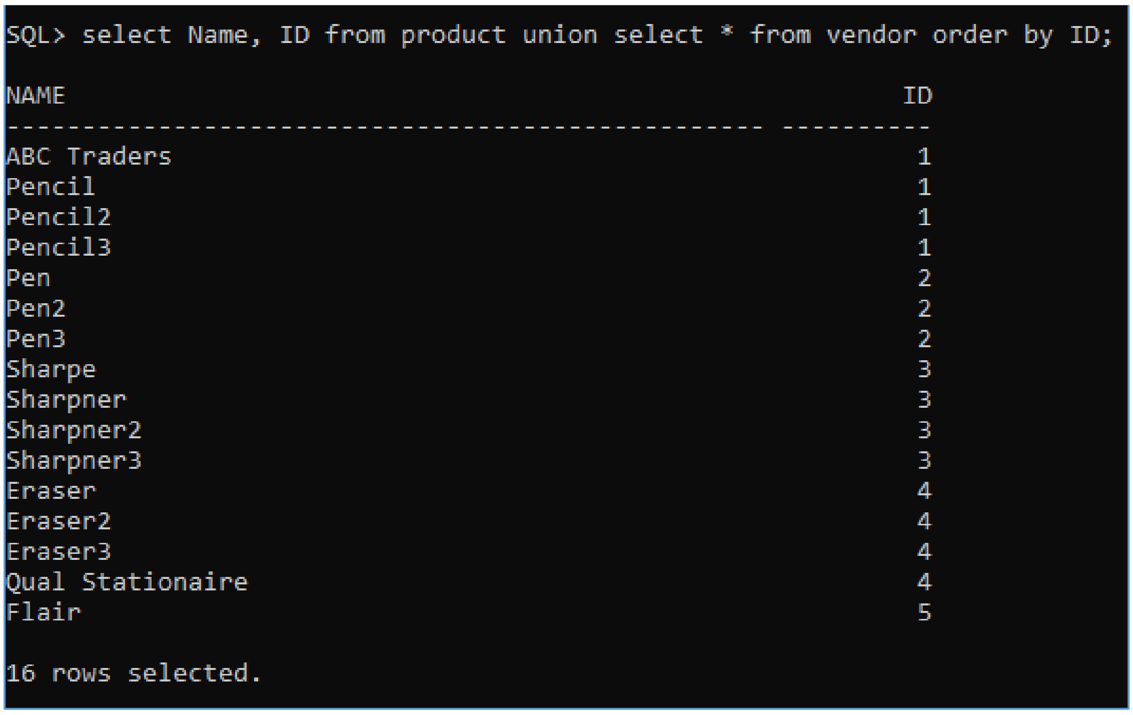
```
SQL> select * from product cross join vendor;

NAME                                   ID      PRICE VENNAME                                ID
------------------------------ ---------- ---------- -------------------------- ----------
Pencil                                  1         10 ABC Traders                             1
Pen                                     2         25 ABC Traders                             1
Eraser                                  4       7.25 ABC Traders                             1
Sharpner                                3       5.75 ABC Traders                             1
Pencil2                                 1         12 ABC Traders                             1
Pencil3                                 1         15 ABC Traders                             1
Pen2                                    2         50 ABC Traders                             1
Pen3                                    2         80 ABC Traders                             1
Sharpner2                               3          8 ABC Traders                             1
Sharpner3                               3         30 ABC Traders                             1
Eraser2                                 4         17 ABC Traders                             1
Eraser3                                 4         15 ABC Traders                             1

NAME                                   ID      PRICE VENNAME                                ID
------------------------------ ---------- ---------- -------------------------- ----------
Pencil                                  1         10 Qual Stationaire                        4
Pen                                     2         25 Qual Stationaire                        4
Eraser                                  4       7.25 Qual Stationaire                        4
Sharpner                                3       5.75 Qual Stationaire                        4
Pencil2                                 1         12 Qual Stationaire                        4
Pencil3                                 1         15 Qual Stationaire                        4
Pen2                                    2         50 Qual Stationaire                        4
Pen3                                    2         80 Qual Stationaire                        4
Sharpner2                               3          8 Qual Stationaire                        4
Sharpner3                               3         30 Qual Stationaire                        4
Eraser2                                 4         17 Qual Stationaire                        4
Eraser3                                 4         15 Qual Stationaire                        4
```



```
NAME                                   ID      PRICE VENNAME                           ID
------------------------------ ---------- ---------- ------------------------- ----------
Pencil                                  1         10 Sharpe                             3
Pen                                     2         25 Sharpe                             3
Eraser                                  4       7.25 Sharpe                             3
Sharpner                                3       5.75 Sharpe                             3
Pencil2                                 1         12 Sharpe                             3
Pencil3                                 1         15 Sharpe                             3
Pen2                                    2         50 Sharpe                             3
Pen3                                    2         80 Sharpe                             3
Sharpner2                               3          8 Sharpe                             3
Sharpner3                               3         30 Sharpe                             3
Eraser2                                 4         17 Sharpe                             3
Eraser3                                 4         15 Sharpe                             3

NAME                                   ID      PRICE VENNAME                           ID
------------------------------ ---------- ---------- ------------------------- ----------
Pencil                                  1         10 Flair                              5
Pen                                     2         25 Flair                              5
Eraser                                  4       7.25 Flair                              5
Sharpner                                3       5.75 Flair                              5
Pencil2                                 1         12 Flair                              5
Pencil3                                 1         15 Flair                              5
Pen2                                    2         50 Flair                              5
Pen3                                    2         80 Flair                              5
Sharpner2                               3          8 Flair                              5
Sharpner3                               3         30 Flair                              5
Eraser2                                 4         17 Flair                              5
Eraser3                                 4         15 Flair                              5

48 rows selected.
```

f) Union

To return the union of the selection all records of 'product' with attributes 'Name' and 'ID' and the selection of all records of 'vendor'



```
SQL> select Name, ID from product union select * from vendor order by ID;

NAME                                                     ID
-------------------------------------------------- ----------
ABC Traders                                               1
Pencil                                                    1
Pencil2                                                   1
Pencil3                                                   1
Pen                                                       2
Pen2                                                      2
Pen3                                                      2
Sharpe                                                    3
Sharpner                                                  3
Sharpner2                                                 3
Sharpner3                                                 3
Eraser                                                    4
Eraser2                                                   4
Eraser3                                                   4
Qual Stationaire                                          4
Flair                                                     5

16 rows selected.
```

RESULT: Thus the operation of database joins was successfully implemented using SQL query.

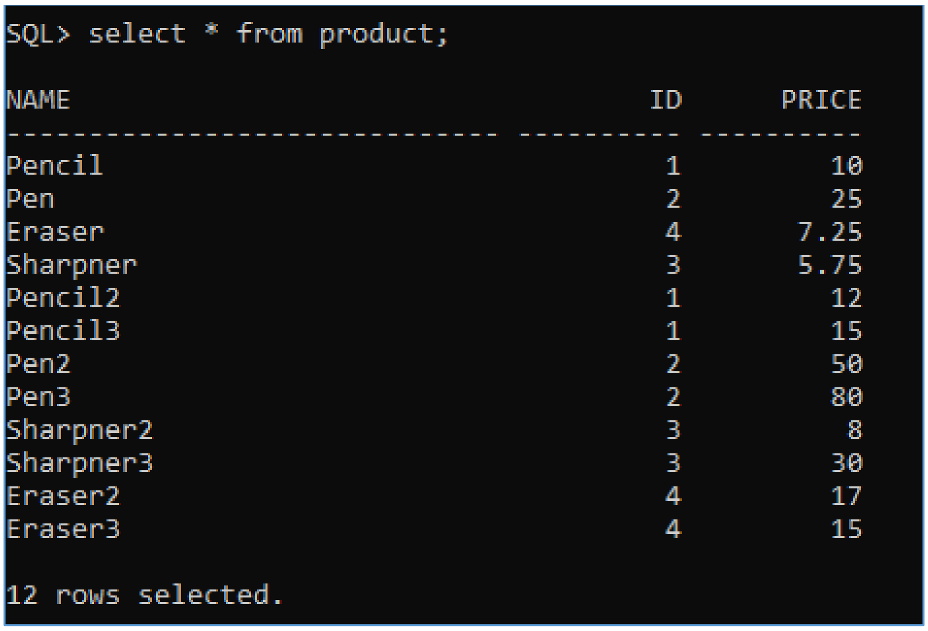
------ X ------

Exp. No.: 07	PL/SQL Anonymous and Named Block

Aim: To demonstrate the working of PL/SQL Blocks – Anonymous and Named

Queries and Corresponding Outputs:

➔ Existing table 'product'

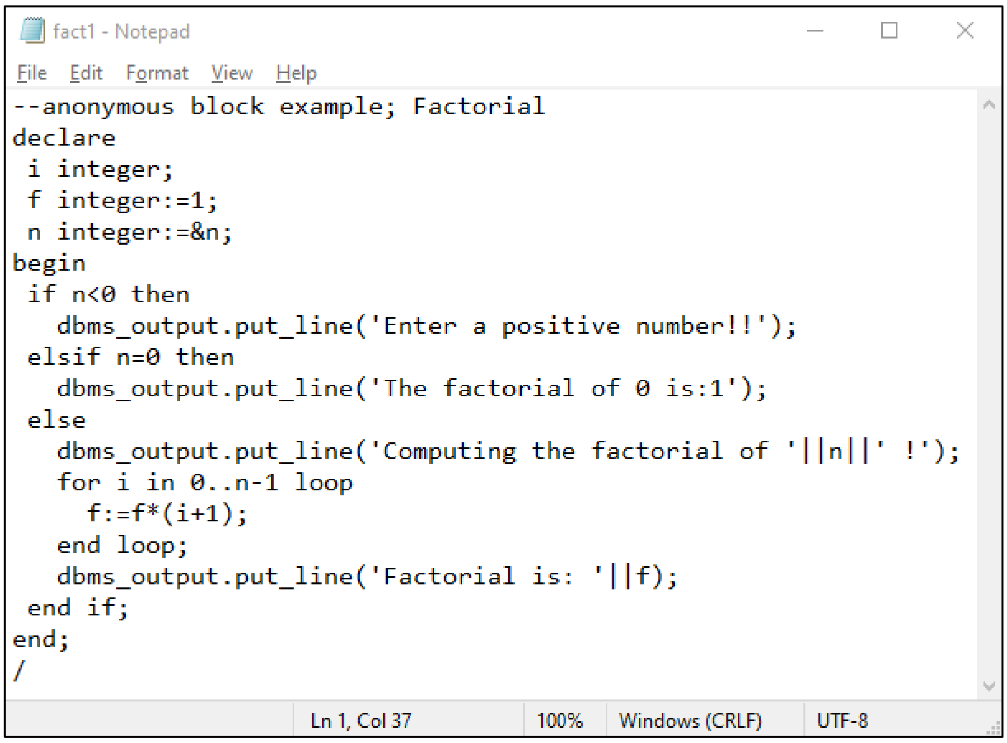


```
SQL> select * from product;

NAME                                   ID      PRICE
------------------------------ ---------- ----------
Pencil                                  1         10
Pen                                     2         25
Eraser                                  4       7.25
Sharpner                                3       5.75
Pencil2                                 1         12
Pencil3                                 1         15
Pen2                                    2         50
Pen3                                    2         80
Sharpner2                               3          8
Sharpner3                               3         30
Eraser2                                 4         17
Eraser3                                 4         15

12 rows selected.
```

a) Anonymous Block

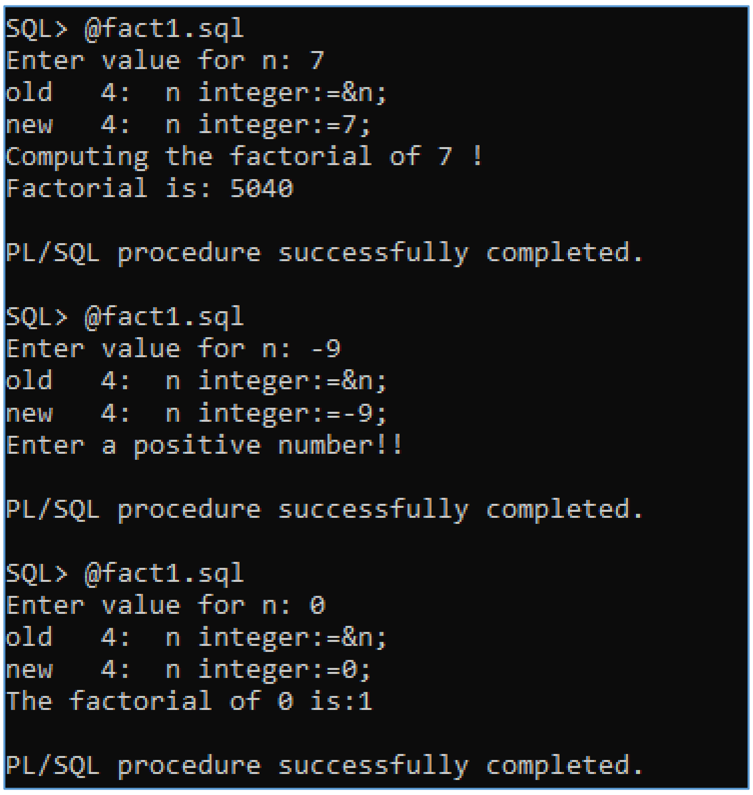
To write a PL/SQL Program that computes the factorial of a number, taken as input, using a for-loop and if-else ladder

➔ PL/SQL Code ['fact1.sql']



```
--anonymous block example; Factorial
declare
 i integer;
 f integer:=1;
 n integer:=&n;
begin
 if n<0 then
   dbms_output.put_line('Enter a positive number!!');
 elsif n=0 then
   dbms_output.put_line('The factorial of 0 is:1');
 else
   dbms_output.put_line('Computing the factorial of '||n||' !');
   for i in 0..n-1 loop
     f:=f*(i+1);
   end loop;
   dbms_output.put_line('Factorial is: '||f);
 end if;
end;
/
```

➔ Output

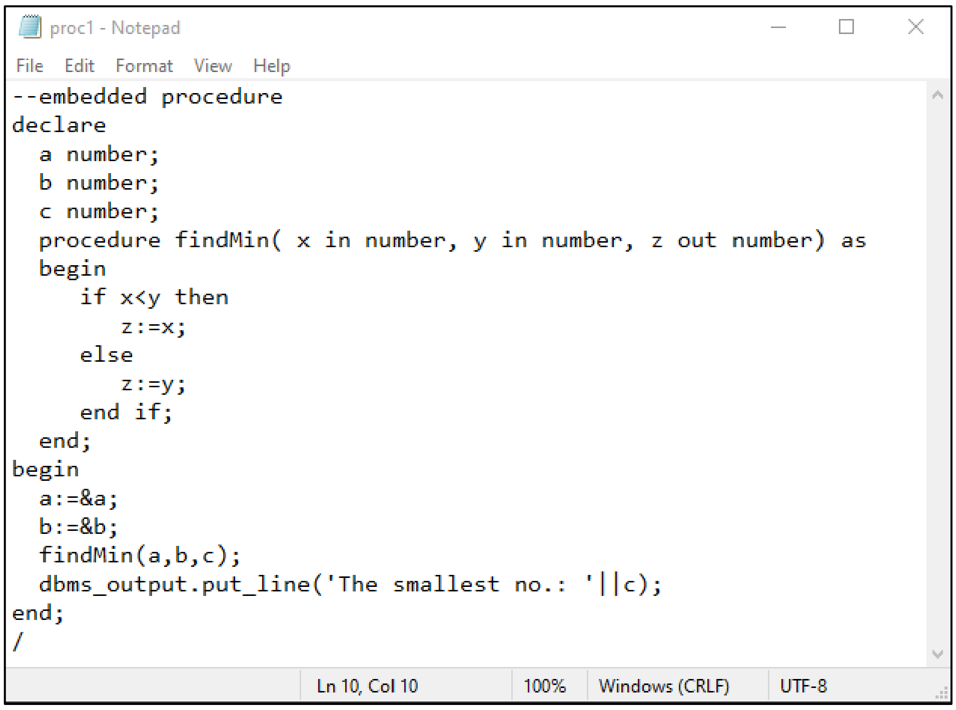


```
SQL> @fact1.sql
Enter value for n: 7
old   4:  n integer:=&n;
new   4:  n integer:=7;
Computing the factorial of 7 !
Factorial is: 5040

PL/SQL procedure successfully completed.

SQL> @fact1.sql
Enter value for n: -9
old   4:  n integer:=&n;
new   4:  n integer:=-9;
Enter a positive number!!

PL/SQL procedure successfully completed.

SQL> @fact1.sql
Enter value for n: 0
old   4:  n integer:=&n;
new   4:  n integer:=0;
The factorial of 0 is:1

PL/SQL procedure successfully completed.
```

b) PL/SQL Procedure

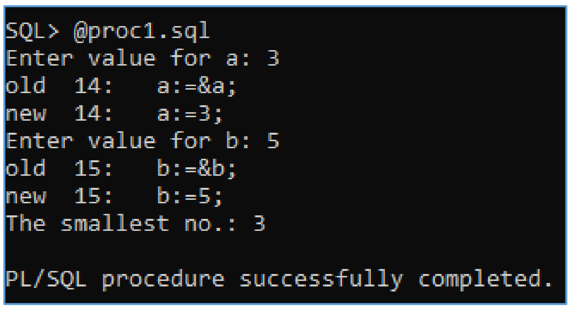
i. To write a PL/SQL Program with an embedded procedure block to find the smallest of two numbers taken as input

➔ PL/SQL Code ['proc1.sql']

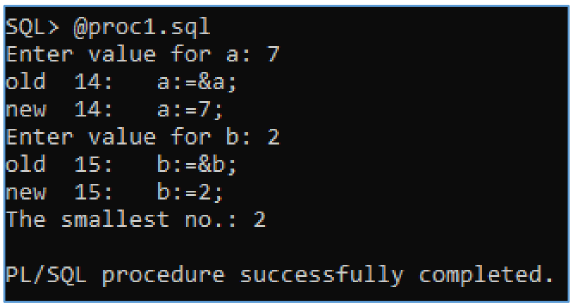


```
--embedded procedure
declare
  a number;
  b number;
  c number;
  procedure findMin( x in number, y in number, z out number) as
  begin
     if x<y then
        z:=x;
     else
        z:=y;
     end if;
  end;
begin
  a:=&a;
  b:=&b;
  findMin(a,b,c);
  dbms_output.put_line('The smallest no.: '||c);
end;
/
```

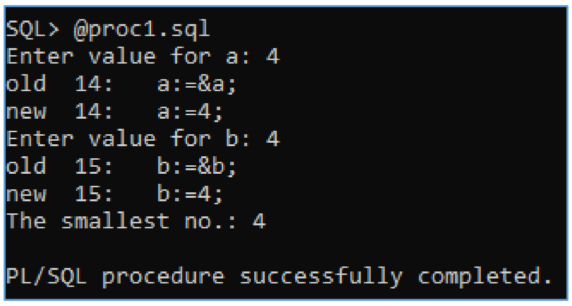
➔ Output



```
SQL> @proc1.sql
Enter value for a: 3
old  14:    a:=&a;
new  14:    a:=3;
Enter value for b: 5
old  15:    b:=&b;
new  15:    b:=5;
The smallest no.: 3

PL/SQL procedure successfully completed.
```



```
SQL> @proc1.sql
Enter value for a: 7
old  14:    a:=&a;
new  14:    a:=7;
Enter value for b: 2
old  15:    b:=&b;
new  15:    b:=2;
The smallest no.: 2

PL/SQL procedure successfully completed.
```

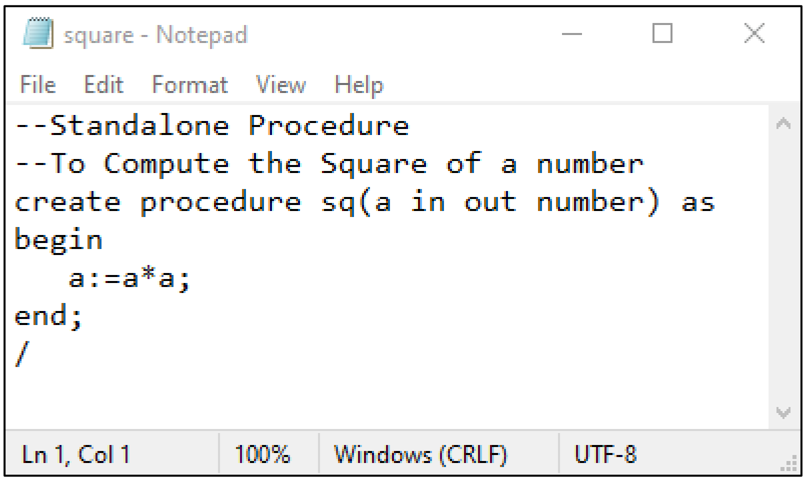


```
SQL> @proc1.sql
Enter value for a: 4
old  14:    a:=&a;
new  14:    a:=4;
Enter value for b: 4
old  15:    b:=&b;
new  15:    b:=4;
The smallest no.: 4

PL/SQL procedure successfully completed.
```

ii. To write a PL/SQL Program that uses a standalone procedure block to find the square of a positive number that is taken as input

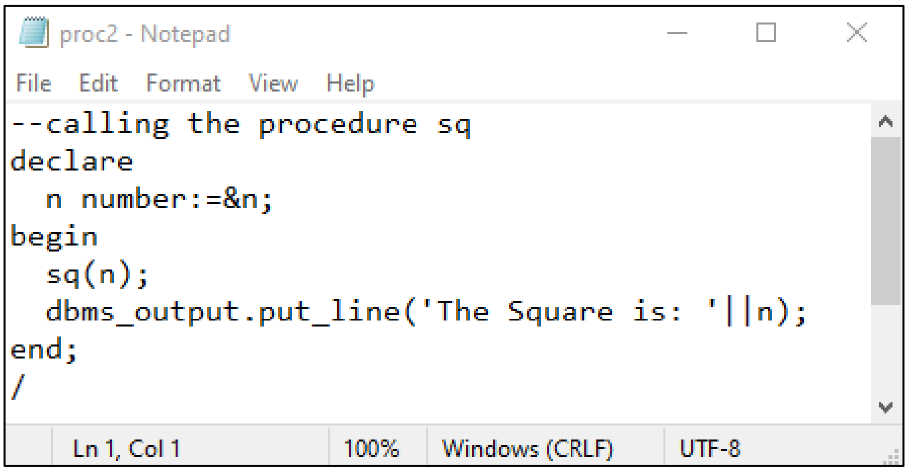
➔ PL/SQL Code

Standalone Procedure, 'sq()' ['square.sql']



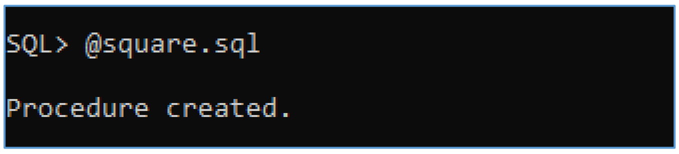
```
--Standalone Procedure
--To Compute the Square of a number
create procedure sq(a in out number) as
begin
    a:=a*a;
end;
/
```

Anonymous block that invokes 'sq()' ['proc2.sql']



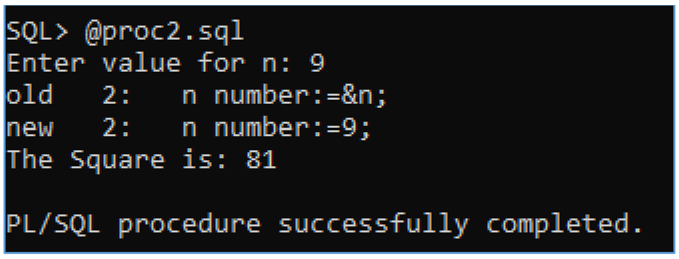
```
--calling the procedure sq
declare
  n number:=&n;
begin
  sq(n);
  dbms_output.put_line('The Square is: '||n);
end;
/
```

➔ Creating Procedure

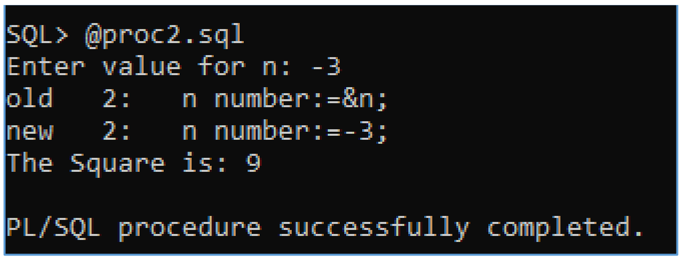


```
SQL> @square.sql

Procedure created.
```

➔ Output



```
SQL> @proc2.sql
Enter value for n: 9
old   2:    n number:=&n;
new   2:    n number:=9;
The Square is: 81

PL/SQL procedure successfully completed.
```



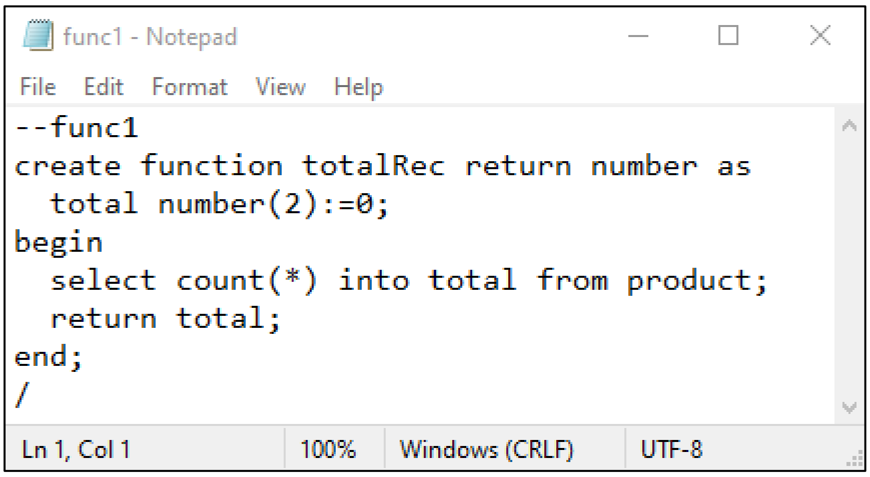
```
SQL> @proc2.sql
Enter value for n: -3
old   2:    n number:=&n;
new   2:    n number:=-3;
The Square is: 9

PL/SQL procedure successfully completed.
```

c) PL/SQL Function

i. To write a PL/SQL Program with an standalone function block to find the total number of records present in the table 'product'

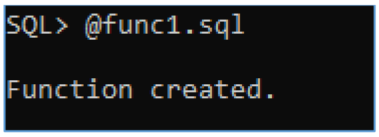
➔ PL/SQL Code

Standalone Function, 'totalRec()' ['func1.sql']



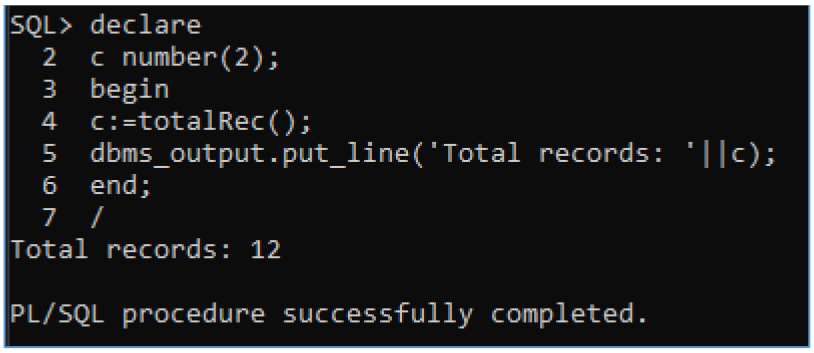
```
--func1
create function totalRec return number as
  total number(2):=0;
begin
  select count(*) into total from product;
  return total;
end;
/
```

➔ Creating Function



```
SQL> @func1.sql

Function created.
```

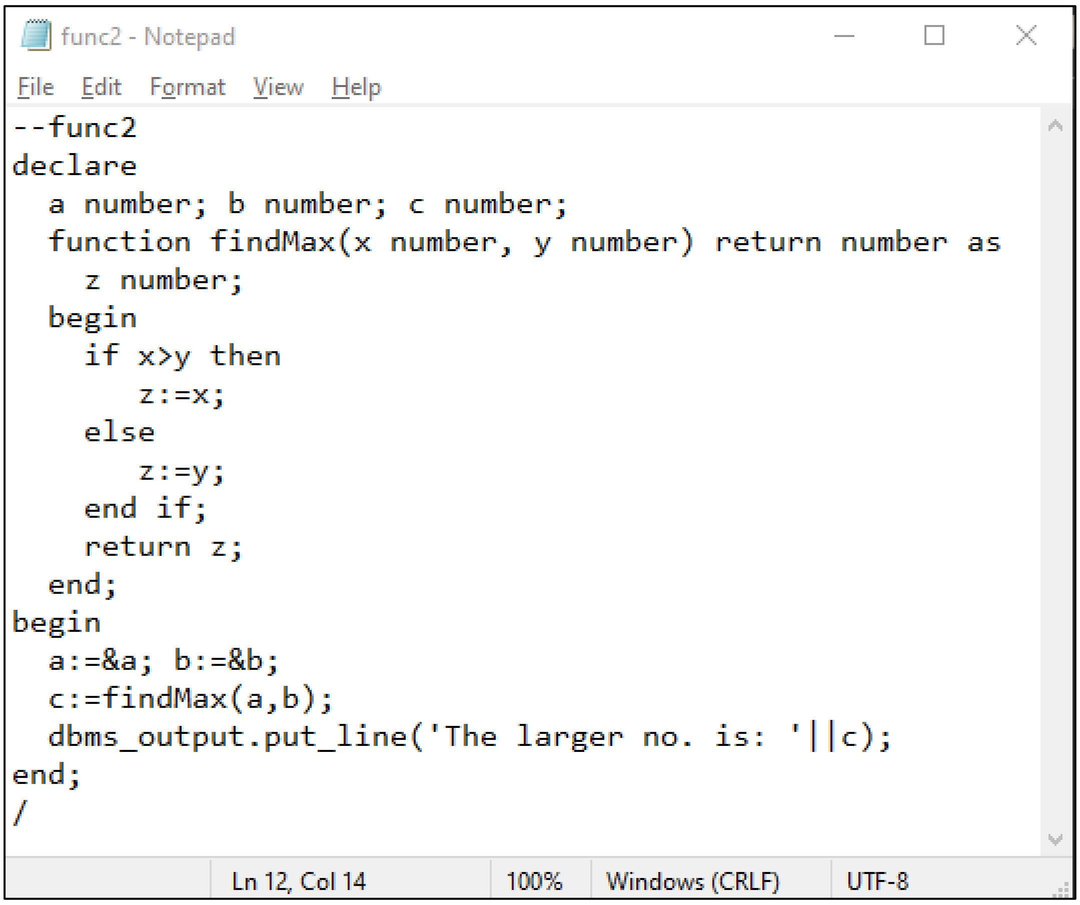
➔ PL/SQL Code that invokes 'totalRec()' and the corresponding output



```
SQL> declare
  2  c number(2);
  3  begin
  4  c:=totalRec();
  5  dbms_output.put_line('Total records: '||c);
  6  end;
  7  /
Total records: 12

PL/SQL procedure successfully completed.
```

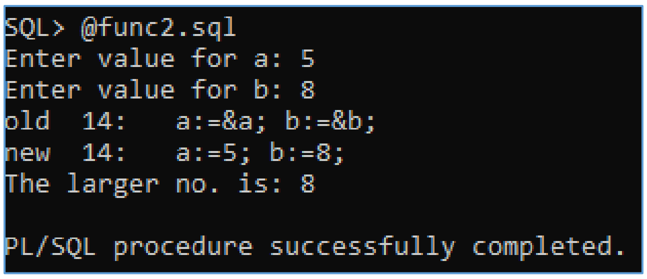
ii. To write a PL/SQL Program with an embedded function block to find the largest of two numbers taken as input

➔ PL/SQL Code [func2.sql']

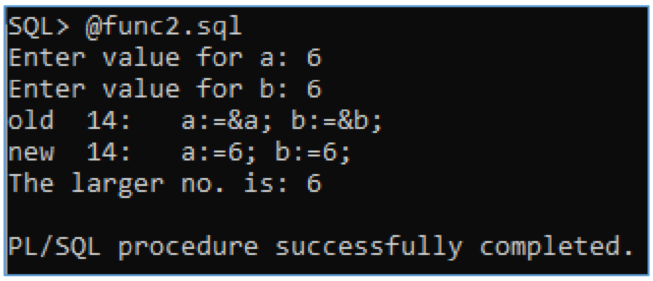


```
--func2
declare
  a number; b number; c number;
  function findMax(x number, y number) return number as
    z number;
  begin
    if x>y then
       z:=x;
    else
       z:=y;
    end if;
    return z;
  end;
begin
  a:=&a; b:=&b;
  c:=findMax(a,b);
  dbms_output.put_line('The larger no. is: '||c);
end;
/
```

➔ Output



```
SQL> @func2.sql
Enter value for a: 5
Enter value for b: 8
old  14:   a:=&a; b:=&b;
new  14:   a:=5; b:=8;
The larger no. is: 8

PL/SQL procedure successfully completed.
```



```
SQL> @func2.sql
Enter value for a: 6
Enter value for b: 6
old  14:   a:=&a; b:=&b;
new  14:   a:=6; b:=6;
The larger no. is: 6

PL/SQL procedure successfully completed.
```

RESULT: Thus the operations of PL/SQL anonymous block, procedures and functions were successfully implemented using SQL query.

------ X ------

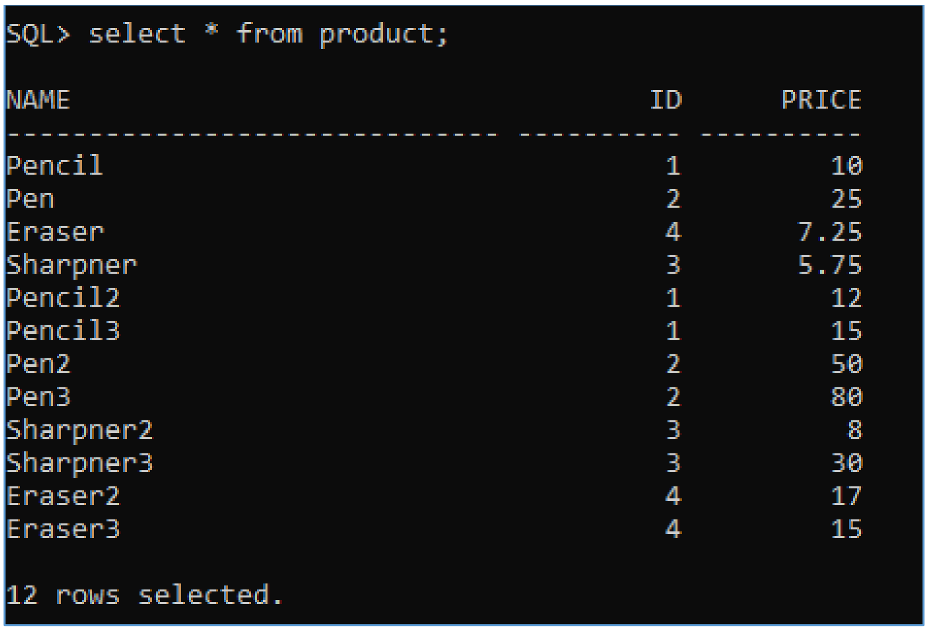
Exp. No.: 08

PL/SQL Cursors

Aim: To demonstrate the working of PL/SQL Cursors – Implicit and Explicit

Queries and Corresponding Outputs:

➔ Existing table 'product'

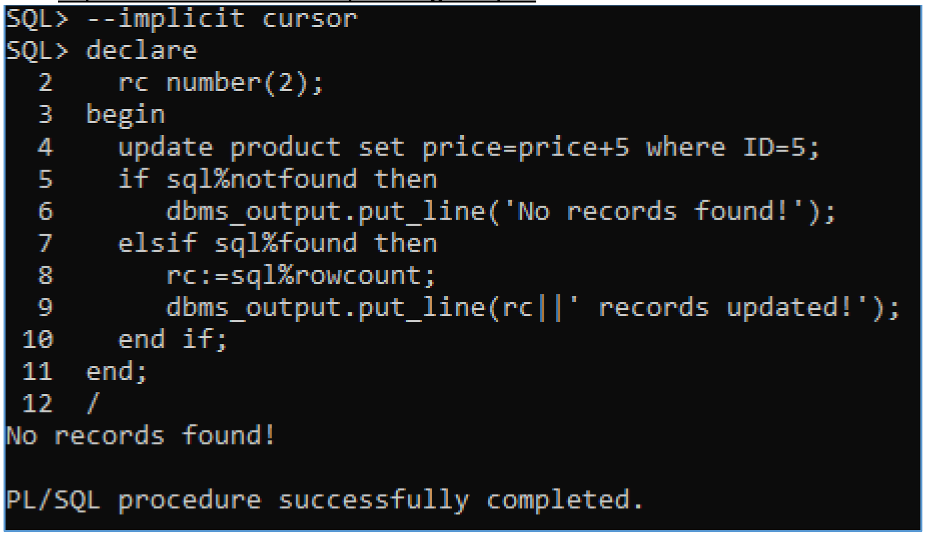


```
SQL> select * from product;

NAME                                   ID      PRICE
------------------------------ ---------- ----------
Pencil                                  1         10
Pen                                     2         25
Eraser                                  4       7.25
Sharpner                                3       5.75
Pencil2                                 1         12
Pencil3                                 1         15
Pen2                                    2         50
Pen3                                    2         80
Sharpner2                               3          8
Sharpner3                               3         30
Eraser2                                 4         17
Eraser3                                 4         15

12 rows selected.
```

a) Implicit Cursor

i. To write a PL/SQL Program that updates the 'Price' of records in 'Product' that have the 'ID' 5 and display whether such records are present and, if so, how many

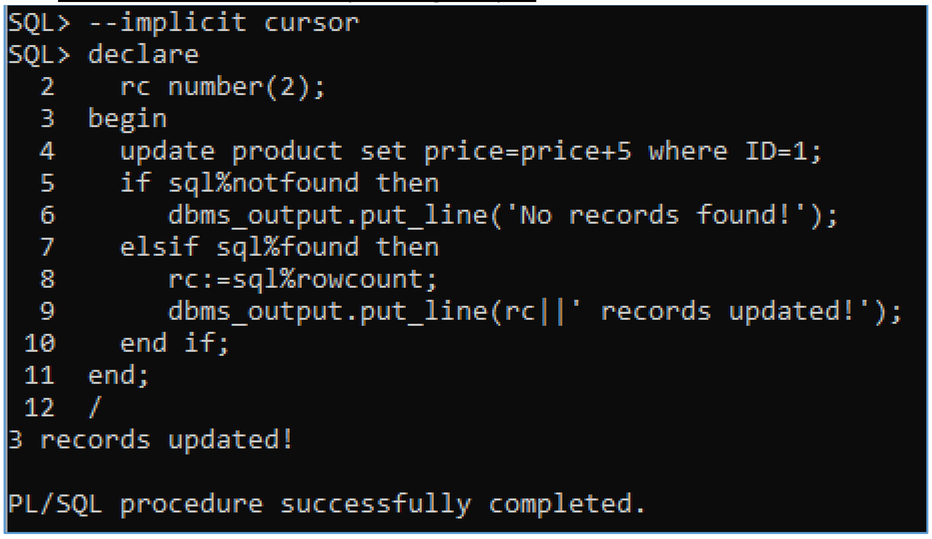
➔ PL/SQL Code and Corresponding Output



```
SQL> --implicit cursor
SQL> declare
       rc number(2);
     begin
       update product set price=price+5 where ID=5;
       if sql%notfound then
          dbms_output.put_line('No records found!');
       elsif sql%found then
          rc:=sql%rowcount;
          dbms_output.put_line(rc||' records updated!');
       end if;
     end;
     /
No records found!

PL/SQL procedure successfully completed.
```

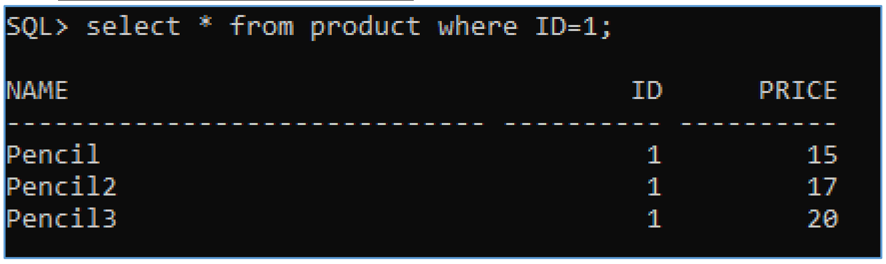
ii. To write a PL/SQL Program that updates the ‘Price’ of records in ‘Product’ that have the ‘ID’ 1 and display whether such records are present and, if so, how many

➔ PL/SQL Code and Corresponding Output



```
SQL> --implicit cursor
SQL> declare
       rc number(2);
     begin
       update product set price=price+5 where ID=1;
       if sql%notfound then
          dbms_output.put_line('No records found!');
       elsif sql%found then
          rc:=sql%rowcount;
          dbms_output.put_line(rc||' records updated!');
       end if;
     end;
     /
3 records updated!

PL/SQL procedure successfully completed.
```

➔ Updated Records of the Table

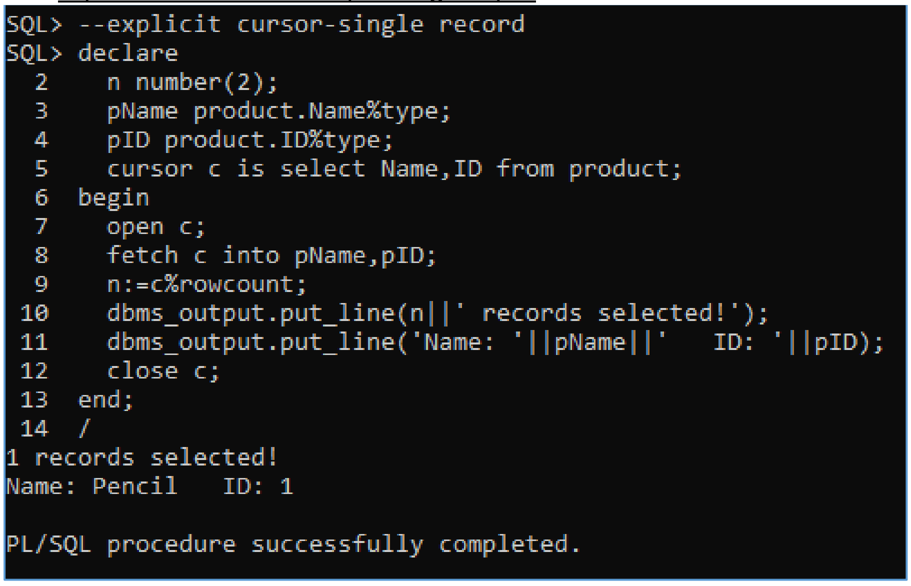


```
SQL> select * from product where ID=1;

NAME                                   ID      PRICE
------------------------------ ---------- ----------
Pencil                                  1         15
Pencil2                                 1         17
Pencil3                                 1         20
```

b) Explicit Cursor

i. To write a PL/SQL Program that uses a cursor to read the first record of the table ‘Product’ if it is present

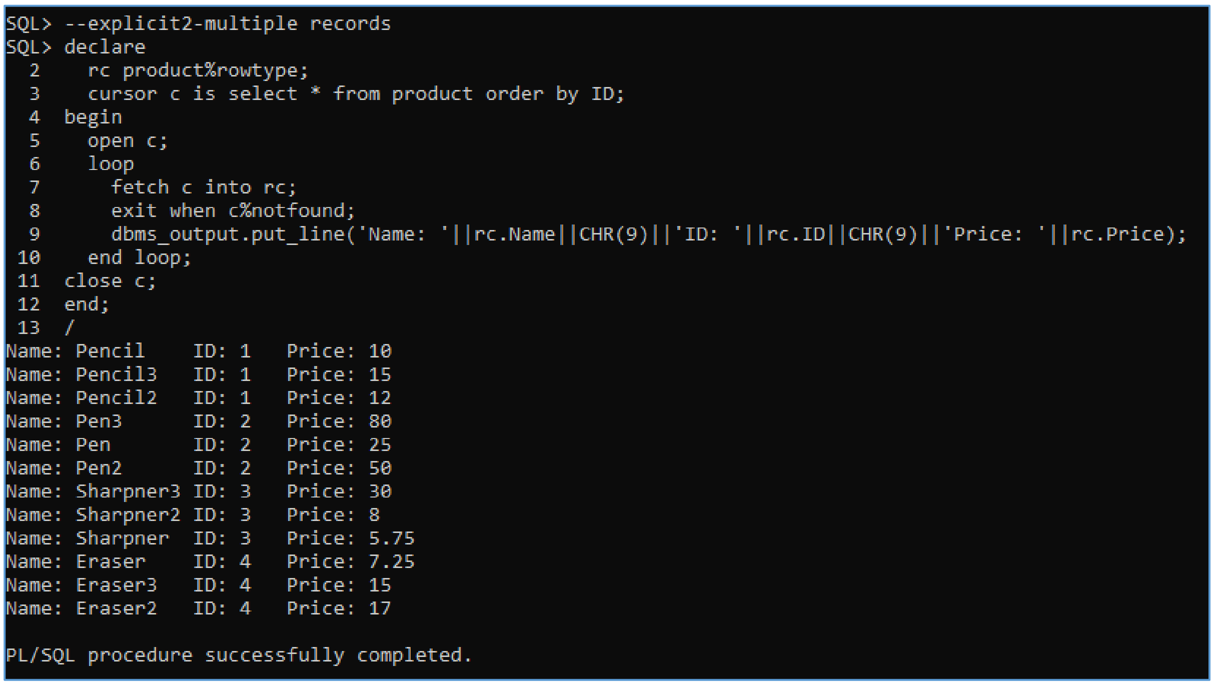
➔ PL/SQL Code and Corresponding Output



```
SQL> --explicit cursor-single record
SQL> declare
       n number(2);
       pName product.Name%type;
       pID product.ID%type;
       cursor c is select Name,ID from product;
     begin
       open c;
       fetch c into pName,pID;
       n:=c%rowcount;
       dbms_output.put_line(n||' records selected!');
       dbms_output.put_line('Name: '||pName||'   ID: '||pID);
       close c;
     end;
     /
1 records selected!
Name: Pencil   ID: 1

PL/SQL procedure successfully completed.
```

ii. To write a PL/SQL Program that uses a cursor to read all the records of the table ‘Product’

➔ <u>PL/SQL Code and Corresponding Output</u>



```
SQL> --explicit2-multiple records
SQL> declare
       rc product%rowtype;
       cursor c is select * from product order by ID;
     begin
       open c;
       loop
         fetch c into rc;
         exit when c%notfound;
         dbms_output.put_line('Name: '||rc.Name||CHR(9)||'ID: '||rc.ID||CHR(9)||'Price: '||rc.Price);
       end loop;
     close c;
     end;
     /
Name: Pencil    ID: 1   Price: 10
Name: Pencil3   ID: 1   Price: 15
Name: Pencil2   ID: 1   Price: 12
Name: Pen3      ID: 2   Price: 80
Name: Pen       ID: 2   Price: 25
Name: Pen2      ID: 2   Price: 50
Name: Sharpner3 ID: 3   Price: 30
Name: Sharpner2 ID: 3   Price: 8
Name: Sharpner  ID: 3   Price: 5.75
Name: Eraser    ID: 4   Price: 7.25
Name: Eraser3   ID: 4   Price: 15
Name: Eraser2   ID: 4   Price: 17

PL/SQL procedure successfully completed.
```

RESULT: Thus the operation of PL/SQL cursors was successfully implemented using SQL query.

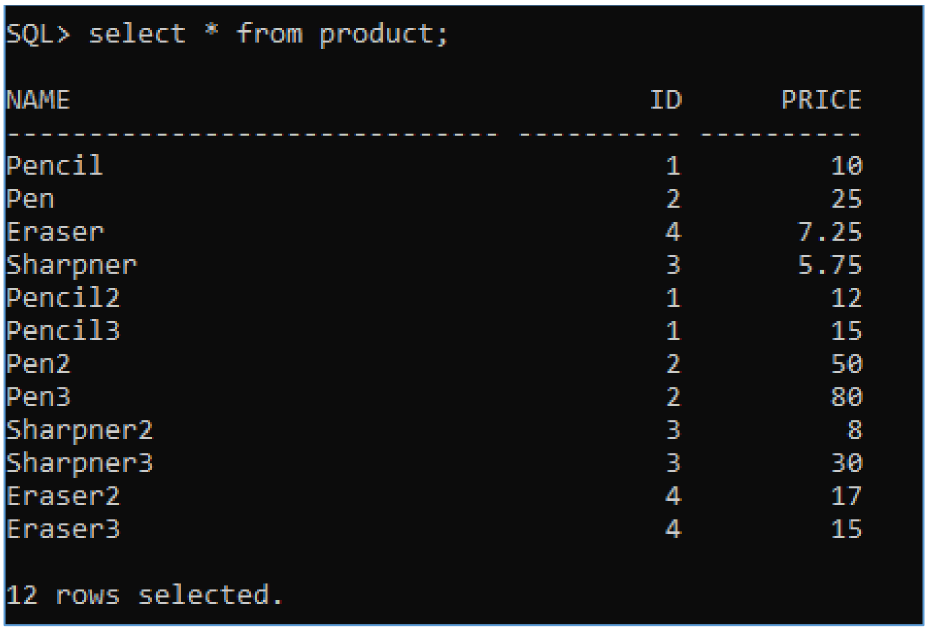
------ X ------

Exp. No.: 09	**PL/SQL Triggers**

Aim: To demonstrate the working of PL/SQL Triggers

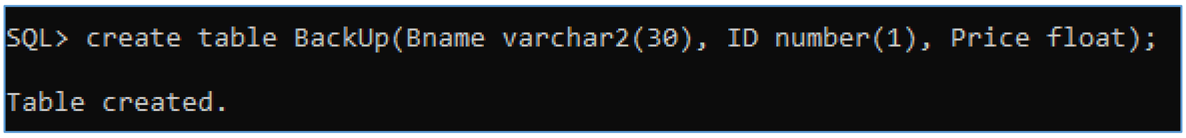
Queries and Corresponding Outputs:

➔ Existing table ‘product’



```
SQL> select * from product;

NAME                                   ID      PRICE
------------------------------ ---------- ----------
Pencil                                  1         10
Pen                                     2         25
Eraser                                  4       7.25
Sharpner                                3       5.75
Pencil2                                 1         12
Pencil3                                 1         15
Pen2                                    2         50
Pen3                                    2         80
Sharpner2                               3          8
Sharpner3                               3         30
Eraser2                                 4         17
Eraser3                                 4         15

12 rows selected.
```

➔ Creating table ‘BackUp’

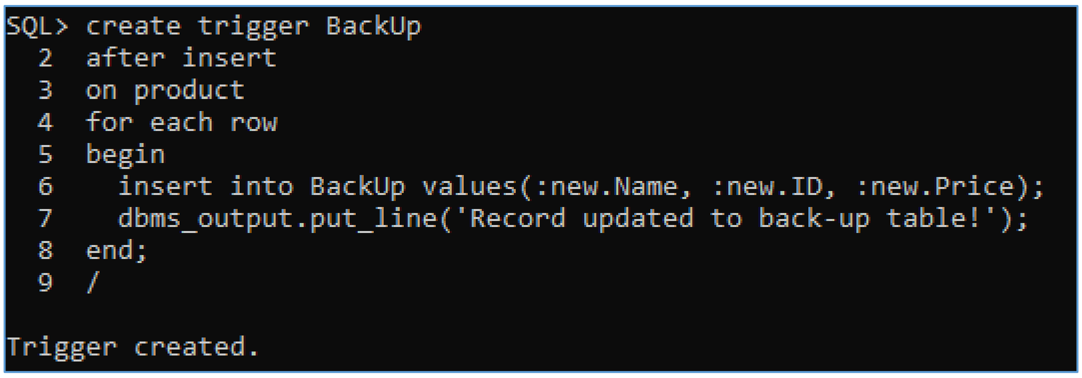


```
SQL> create table BackUp(Bname varchar2(30), ID number(1), Price float);

Table created.
```

a) After Trigger with ‘:new’ Clause

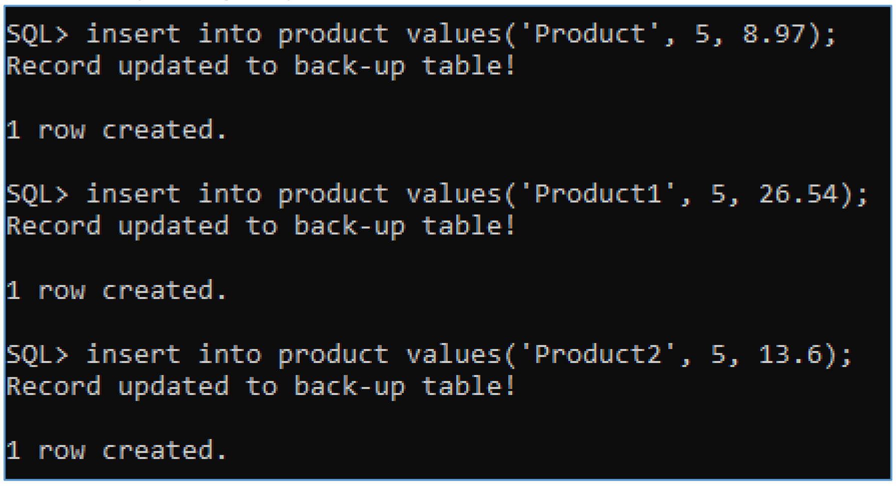
To create a PL/SQL Trigger that inserts the newly added records of ‘Product’ to ‘BackUp’

➔ PL/SQL Code



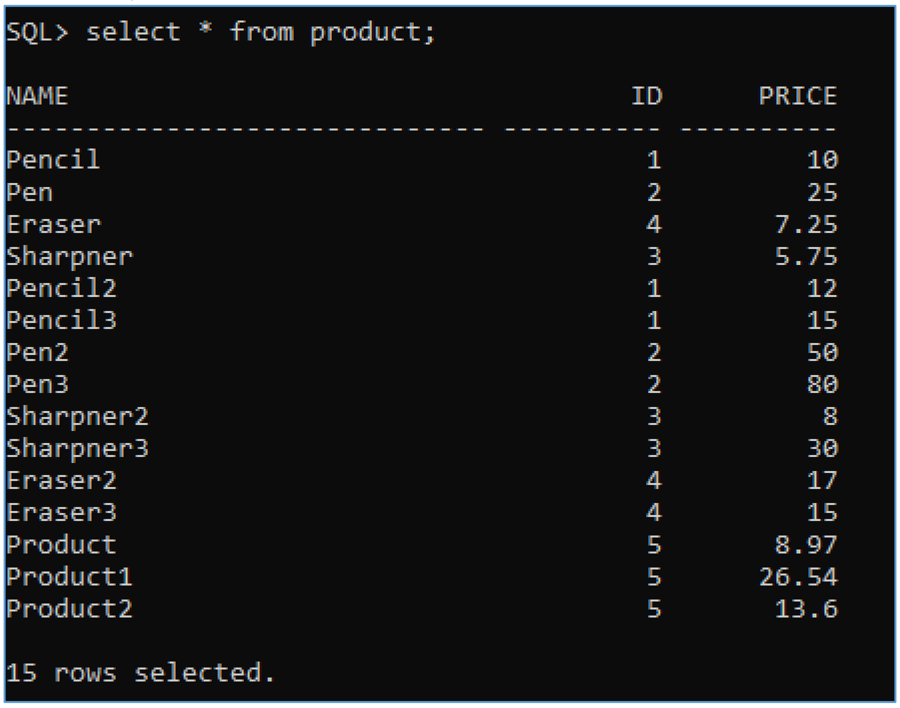
```
SQL> create trigger BackUp
     after insert
     on product
     for each row
     begin
       insert into BackUp values(:new.Name, :new.ID, :new.Price);
       dbms_output.put_line('Record updated to back-up table!');
     end;
     /

Trigger created.
```

➔ Corresponding Output



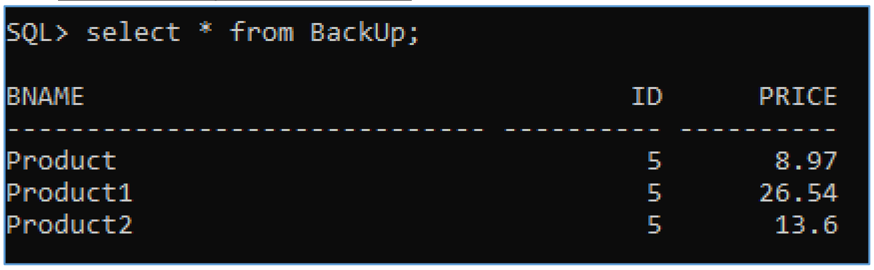
```
SQL> insert into product values('Product', 5, 8.97);
Record updated to back-up table!

1 row created.

SQL> insert into product values('Product1', 5, 26.54);
Record updated to back-up table!

1 row created.

SQL> insert into product values('Product2', 5, 13.6);
Record updated to back-up table!

1 row created.
```

➔ Table ‘product’ after insertion



```
SQL> select * from product;

NAME                                   ID      PRICE
------------------------------ ---------- ----------
Pencil                                  1         10
Pen                                     2         25
Eraser                                  4       7.25
Sharpner                                3       5.75
Pencil2                                 1         12
Pencil3                                 1         15
Pen2                                    2         50
Pen3                                    2         80
Sharpner2                               3          8
Sharpner3                               3         30
Eraser2                                 4         17
Eraser3                                 4         15
Product                                 5       8.97
Product1                                5      26.54
Product2                                5       13.6

15 rows selected.
```

➔ Table ‘BackUp’ after insertion

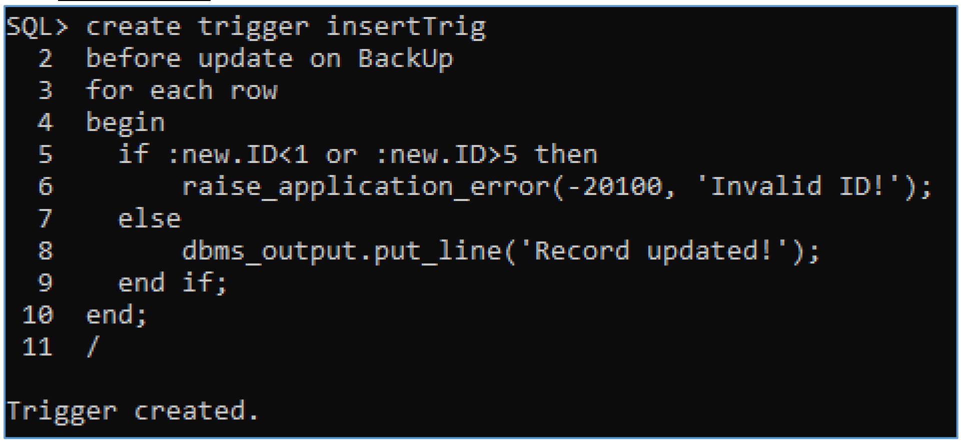


```
SQL> select * from BackUp;

BNAME                                  ID      PRICE
------------------------------ ---------- ----------
Product                                 5       8.97
Product1                                5      26.54
Product2                                5       13.6
```

b) Before Trigger with ':new' Clause

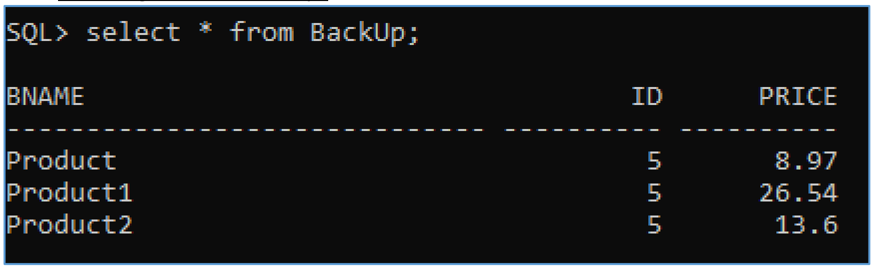
To create a PL/SQL Trigger that allows the newly updated value of 'ID' of a record of 'BackUp' to be between 1 and 5 only

➔ PL/SQL Code



```
SQL> create trigger insertTrig
     before update on BackUp
     for each row
     begin
       if :new.ID<1 or :new.ID>5 then
           raise_application_error(-20100, 'Invalid ID!');
       else
           dbms_output.put_line('Record updated!');
       end if;
     end;
     /

Trigger created.
```

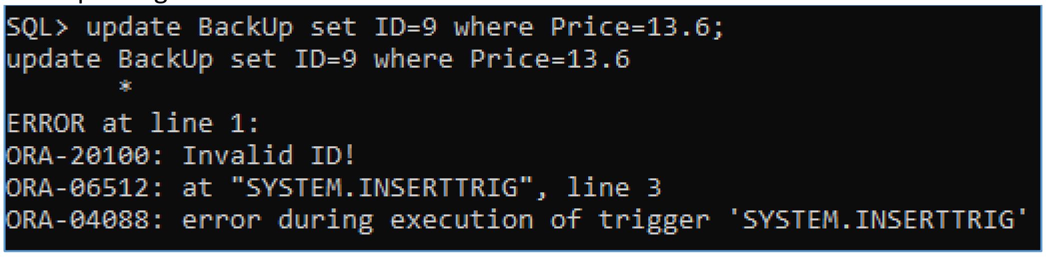
➔ Existing table 'BackUp'



```
SQL> select * from BackUp;

BNAME                                  ID      PRICE
------------------------------ ---------- ----------
Product                                 5       8.97
Product1                                5      26.54
Product2                                5       13.6
```

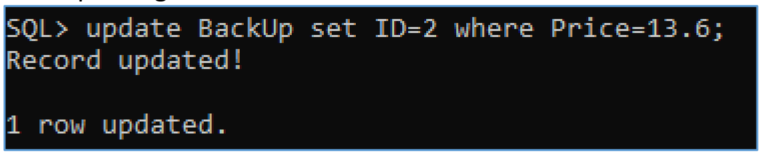
➔ Corresponding Output

i. Updating the value of 'ID' of the record that has 'Price' as 13.6 to 9



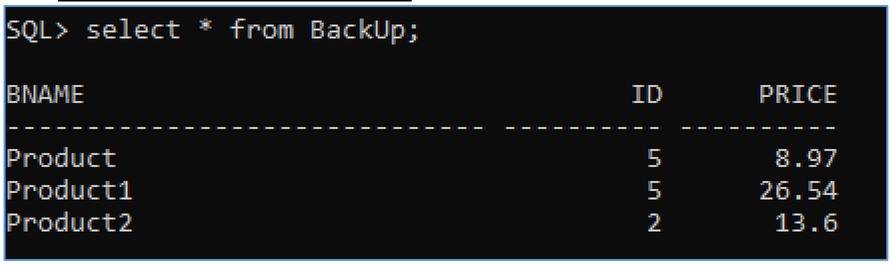
```
SQL> update BackUp set ID=9 where Price=13.6;
update BackUp set ID=9 where Price=13.6
       *
ERROR at line 1:
ORA-20100: Invalid ID!
ORA-06512: at "SYSTEM.INSERTTRIG", line 3
ORA-04088: error during execution of trigger 'SYSTEM.INSERTTRIG'
```

ii. Updating the value of 'ID' of the record that has 'Price' as 13.6 to 2



```
SQL> update BackUp set ID=2 where Price=13.6;
Record updated!

1 row updated.
```

➔ Table 'BackUp' after updating

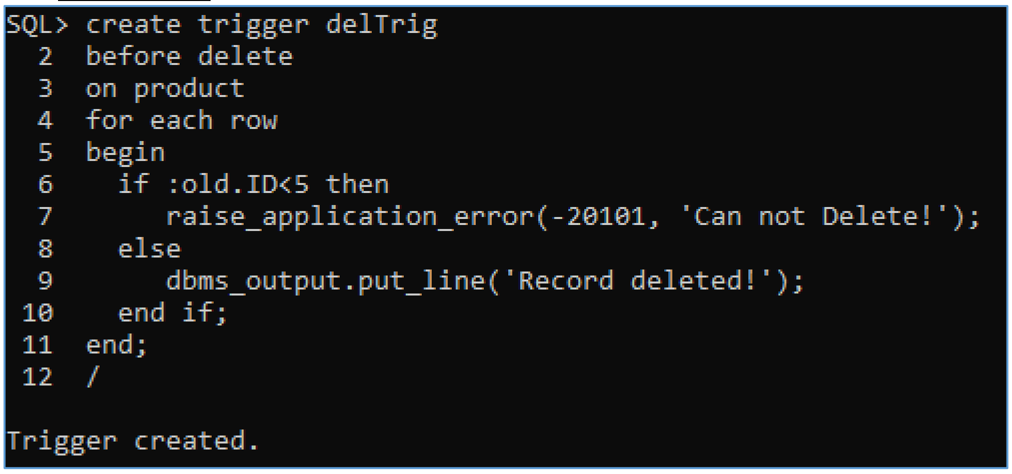


```
SQL> select * from BackUp;

BNAME                                  ID      PRICE
------------------------------ ---------- ----------
Product                                 5       8.97
Product1                                5      26.54
Product2                                2       13.6
```

c) Before Trigger with ':old' Clause

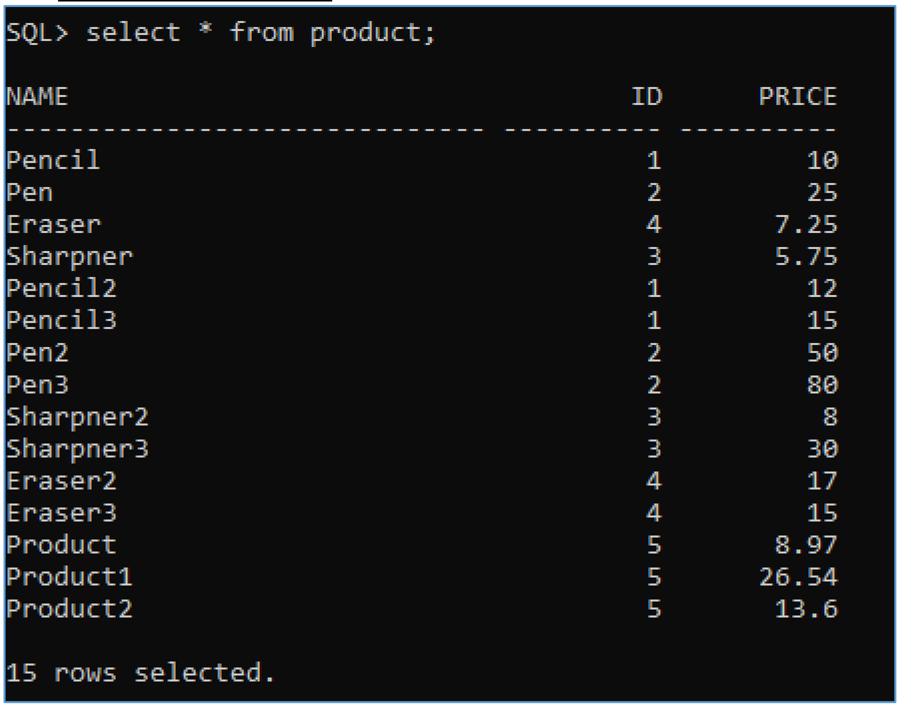
To create a PL/SQL Trigger that doesn't allow the deletion of those records of 'product' that have 'ID' less than 5

➔ PL/SQL Code



```
SQL> create trigger delTrig
     before delete
     on product
     for each row
     begin
       if :old.ID<5 then
          raise_application_error(-20101, 'Can not Delete!');
       else
          dbms_output.put_line('Record deleted!');
       end if;
     end;
     /

Trigger created.
```

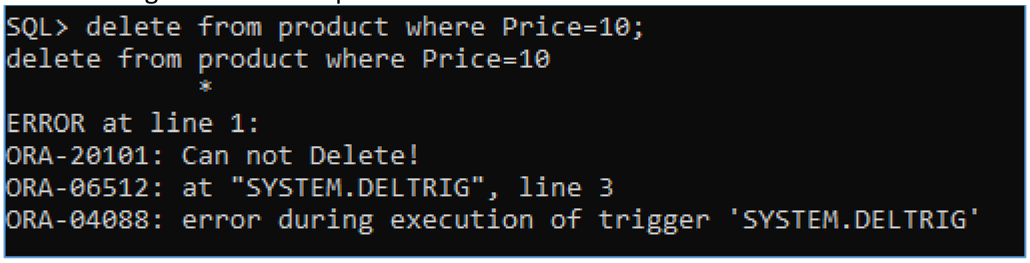
➔ Existing table 'product'



```
SQL> select * from product;

NAME                                   ID      PRICE
------------------------------ ---------- ----------
Pencil                                  1         10
Pen                                     2         25
Eraser                                  4       7.25
Sharpner                                3       5.75
Pencil2                                 1         12
Pencil3                                 1         15
Pen2                                    2         50
Pen3                                    2         80
Sharpner2                               3          8
Sharpner3                               3         30
Eraser2                                 4         17
Eraser3                                 4         15
Product                                 5       8.97
Product1                                5      26.54
Product2                                5       13.6

15 rows selected.
```

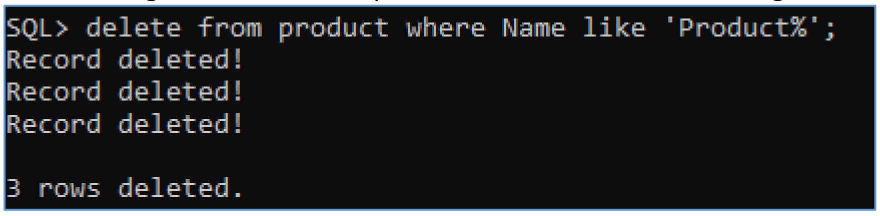
➔ Corresponding Output

i. Deleting the record of 'product' that has the 'Price' 10



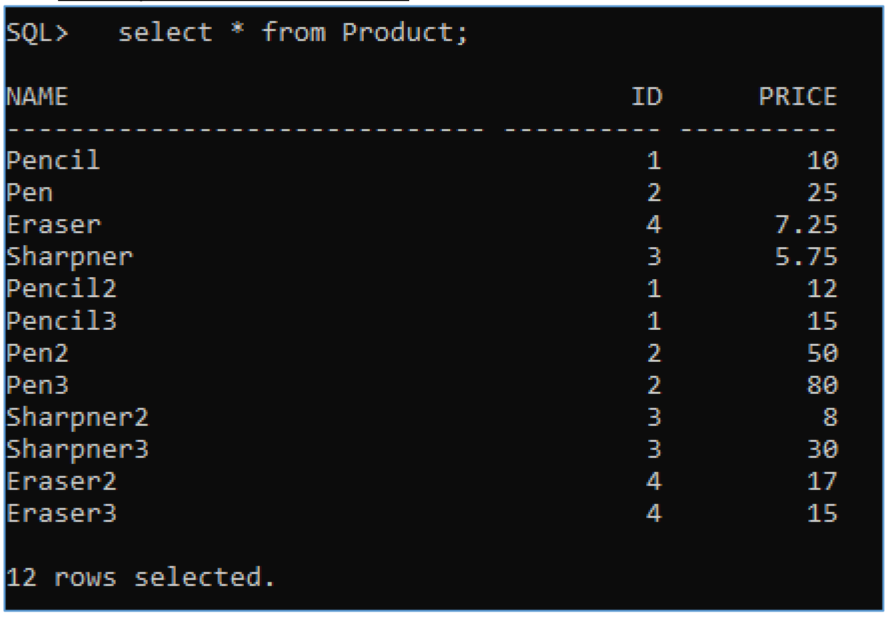
```
SQL> delete from product where Price=10;
delete from product where Price=10
            *
ERROR at line 1:
ORA-20101: Can not Delete!
ORA-06512: at "SYSTEM.DELTRIG", line 3
ORA-04088: error during execution of trigger 'SYSTEM.DELTRIG'
```

ii. Deleting those records of ‘product’ that have ‘Name’ starting with the string ‘Product’



```
SQL> delete from product where Name like 'Product%';
Record deleted!
Record deleted!
Record deleted!

3 rows deleted.
```

➔ Table ‘product’ after deletion



```
SQL>   select * from Product;

NAME                                   ID      PRICE
------------------------------ ---------- ----------
Pencil                                  1         10
Pen                                     2         25
Eraser                                  4       7.25
Sharpner                                3       5.75
Pencil2                                 1         12
Pencil3                                 1         15
Pen2                                    2         50
Pen3                                    2         80
Sharpner2                               3          8
Sharpner3                               3         30
Eraser2                                 4         17
Eraser3                                 4         15

12 rows selected.
```

RESULT: Thus the operation of PL/SQL trigger was successfully implemented using SQL query.

------ X ------

Exp. No.: 10	**<u>GUI Based Database Application</u>**

Aim: To demonstrate the working of a GUI based database application that uses Python

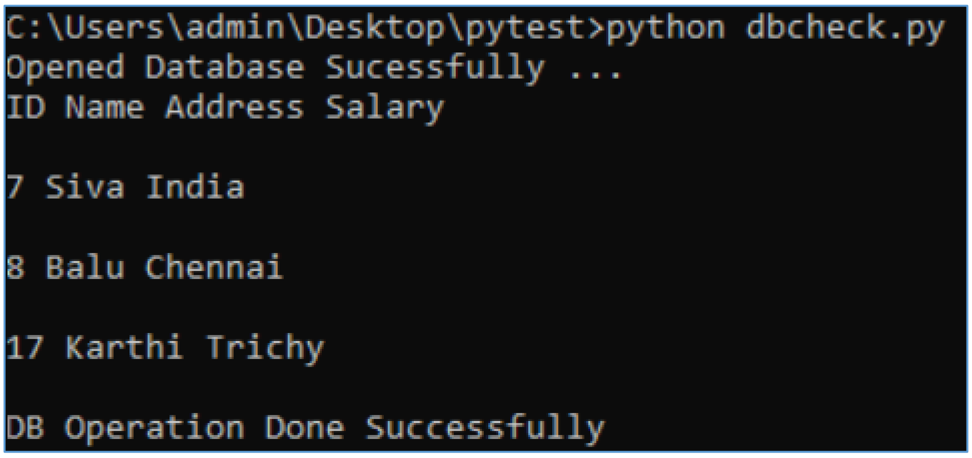
Code and Corresponding Outputs:

<u>a) To display the records of a table from the database</u>

Source Code:

```
import sqlite3
sqlquery="select * from info"
con=sqlite3.connect('ganesh.db')
print ("Opened Database Successfully ...")
# create cursor
cursor = con.cursor()
# execute sql query using cursor
cursor.execute(sqlquery)
# read all data from table using cursor and store them to variable rows
rows = cursor.fetchall()
# display the records
print("ID Name Address Salary\n")
for row in rows:
      print (row[0], row[1], row[2],"\n")
print ("DB Operation Done Successfully")
con.close()
```

Output:



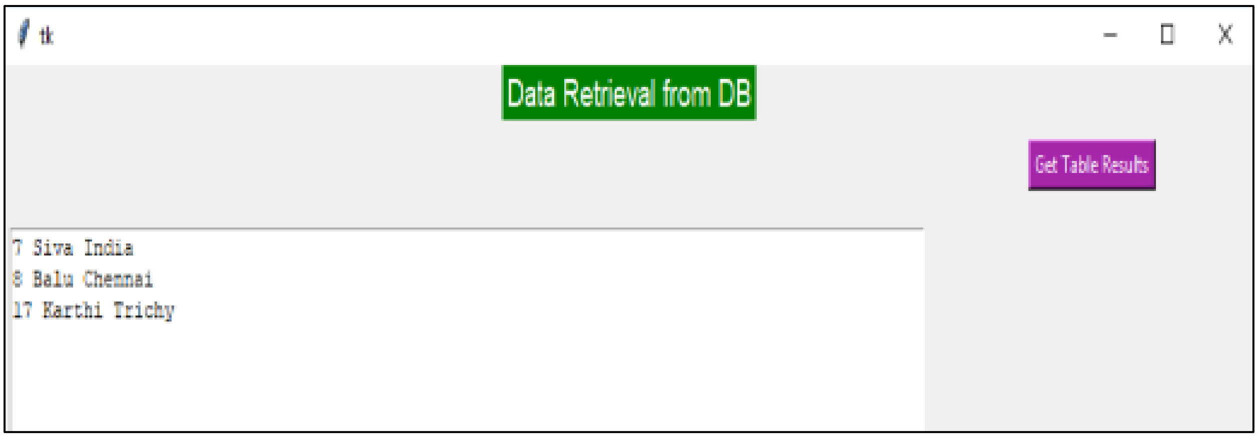
```
C:\Users\admin\Desktop\pytest>python dbcheck.py
Opened Database Sucessfully ...
ID Name Address Salary

7 Siva India

8 Balu Chennai

17 Karthi Trichy

DB Operation Done Successfully
```

b) Display the results from the table using PYTH TKINTER GUI

Source Code:

```
from tkinter import *
import sqlite3
# create an empty list to store the table results
db=[]
# create root window
r=Tk()
# create label widget
msg=Label(r,bg="green", fg="white", text="Data Retrieval from DB")
# set the font to label
msg.config(font =("Arial", 14))
msg.pack()
# create the text widget
tb=Text(r,width=90,height=25)
tb.pack(side="left",padx=5, pady=5)
# BUTTON CLICK EVENT HANDLER
def disp():
# clear the values of text widget to avoid the duplicate values
      tb.delete("1.0",END)
# clear the values of list to avoid the duplicate values
      db.clear()
# set the sql query
      sqlquery="select * from info"
# connect the newly dynamic database in current folder
      con=sqlite3.connect('ganesh.db')
      print ("Opened Database Sucessfully ...")
# create cursor object
      cursor = con.cursor()
# execute sql query using cursor
      cursor.execute(sqlquery)
# read all data from table using cursor and store them to variable rows
      rows = cursor.fetchall()
# display the records through for-loop
      print("ID Name Address Salary\n")
      for row in rows:
            db.append(str(row[0])+" ")
            db.append(row[1]+" ")
            db.append(row[2]+" ")
            db.append("\n")
            #print(row[0],row[1],row[2],"\n")
      print ("DB Operation Done Successfully")
      print(db)
      #tb.insert(END,str(db))
# iterate the list using for-loop
      for k in db:
            #tb.insert('1.0',k)
            tb.insert(END,k)
      con.close()
# create the button widget
bt=Button(r,text="Get Table Results",bg="#a626aa",fg="white",
command=disp)
bt.pack(side="top", padx=10, pady=10)
# set the dimension of form
r.geometry("990x543")
# show the GUI Form
r.mainloop()
```

Output:



```
7 Siva India
8 Balu Chennai
17 Karthi Trichy
```

RESULT: Thus the operation of GUI based database application was successfully implemented using Python and PYTH TKINTER GUI.

------ X ------