663565 NJERU OLIVER NJIRU



APT1050B: Lab#4 – Structured Query Language (SQL)

SUMMER 2022

Given on: 8th June 2022 Due: 22nd June 2022

Instructor – Gerald Chege, PhD

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. **INTRODUCTION**

Structured Query Language (SQL) has emerged as the standard data language for accessing relational databases. There are well over 100 products in the market that support SQL running on machines ranging from microcomputers to large mainframes. Implemented by IBM in their main database product (DB2) in 1982, SQL was adopted by ANSI as the standard access language in 1986. In 1987 the ANSI standard was accepted by ISO. Later standards are SQL1999 and SQL2016 standards.

SQL can be used in four ways:

1. Interactively

2. Through 4GL applications .e.g. RDB, ORACLE, INGRES

3. Embedded SQL in a high-level programming language

4. Report specification in a Report-Writer

The standard SQL set of commands fall into three categories:

1. Data Definition Language (DDL) commands for creating and altering the structure of the database.

2. Data Manipulation Language (DML) commands for extracting and modifying data.

3. Data Control Language (DCL) for controlling access to the database.

DDL commands are used to create new objects, alter the structure of existing objects, or completely remove objects from the system.

Once a database is established, DML commands are the most frequently used of the three kinds of SQL commands.

DCL commands control access to the database transactions. They are used frequently by the database administrator.

SQL commands can not be abbreviated and individual commands must be separated by at least one space or tab. They can be typed in upper or lower case. These notes use upper case consistently to differentiate SQL commands from the text.

SQL syntax includes a few common punctuation marks. Commas separate items in a list, parentheses set off discrete elements (such as column specifications and subqueries), periods specify objects precisely by concatenating additional information to the object names, quotation marks enclose a character string, and a semi-colon indicates the end of a SQL statement. A few more punctuation marks are reserved for special situations.

These notes use the following notation:

[ ] optional

{ } group or sequence of elements

I choice (either/or)

… repetition

The examples used throughout these notes are based on the suppliers parts database from C. J. Date – “An introduction to Database Systems,” Volume I,

4th edition. Briefly, the database contains information concerning suppliers (SUPPLIER) and parts (PART). Suppliers and parts are uniquely identified by supplier number (SNO) and part number (PNO), respectively. A specified supplier supplies specified parts in a specified quantity, and hence a shipment (SHIPMENT), uniquely identified by a combination of SNO and PNO.

1. **DATA DEFINTION**

The Data Definition Language (DDL) is used to create, save and delete tables; create and delete views and indexes. The DDL is also used to modify the storage structure of tables. The DDL commands are:

CREATE - create table/view/index

ALTER - modify table

DROP - remove table/view/index

The general syntax of the CREATE TABLE command is:

CREATE TABLE table\_name

(field\_name field\_type [ NOT NULL ]

[ {, field\_name field\_type [ NOT NULL ] } ] …)

Example – To create the three tables of the example database

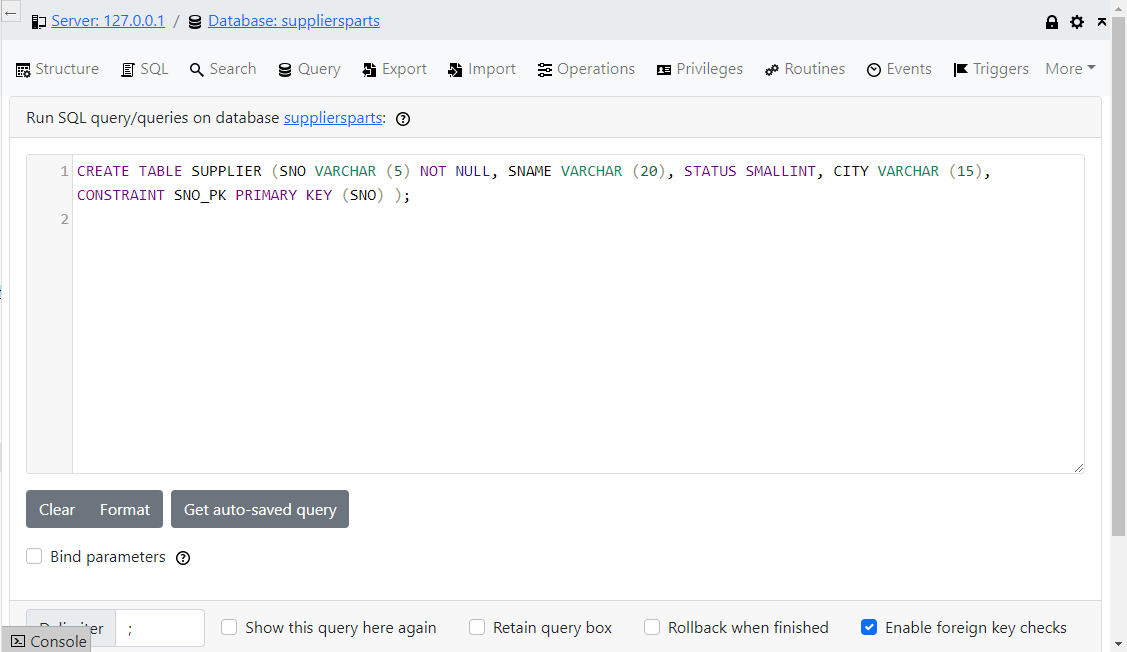
CREATE TABLE SUPPLIER (SNO VARCHAR2 (5) NOT NULL ,

SNAME VARCHAR2 (20) ,

STATUS SMALLINT ,

CITY VARCHAR2 (15),

CONSTRAINT SNO\_PK PRIMARY KEY (SNO) );



CREATE TABLE PART (PNO VARCHAR2 (6) NOT NULL,

PNAME VARCHAR2 (20),

COLOUR VARCHAR2 (6),

WEIGHT SMALLINT,

CITY VARCHAR2 (15),

CONSTRAINT PNO\_PK PRIMARY KEY (PNO)) ;

Graphical user interface, text, application, email

Description automatically generated

CREATE TABLE SHIPMENT (SNO VARCHAR2 (5) NOT NULL ,

PNO VARCHAR2 (5) NOT NULL ,

QTY INTEGER,

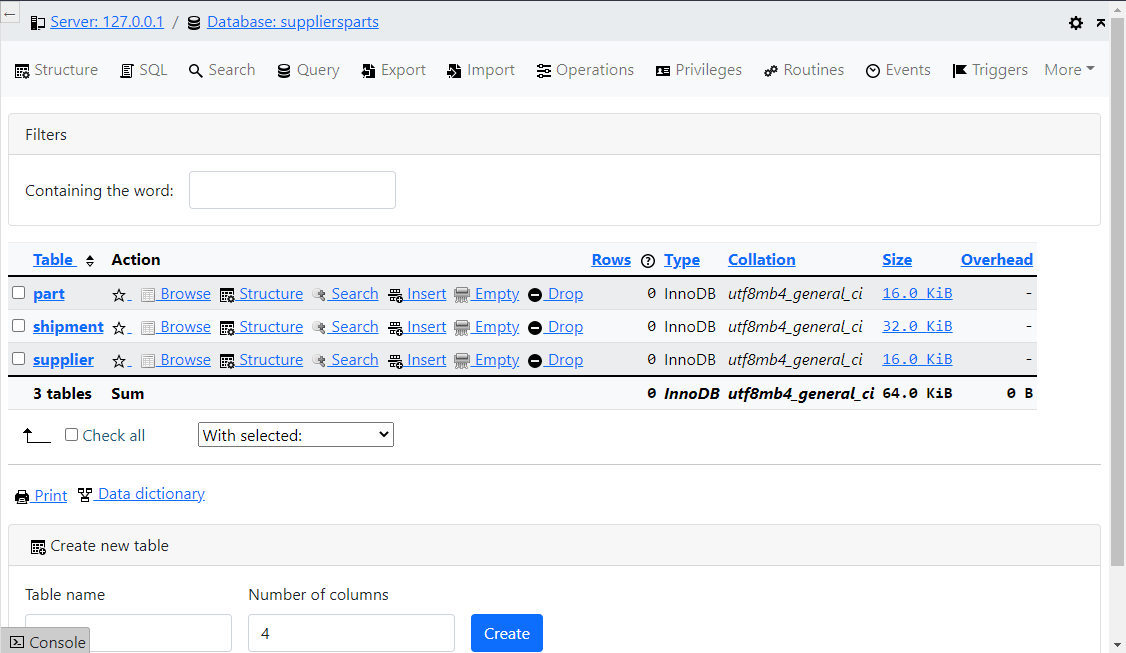
CONSTRAINT SHIP\_PK PRIMARY KEY (PNO, SNO),

CONSTRAINT SHIP\_FK1 FOREIGN KEY (SNO) REFERENCES SUPPLIER (SNO),

CONSTRAINT SHIP\_FK2 FOREIGN KEY (PNO) REFERENCES PART (PNO) ) ;

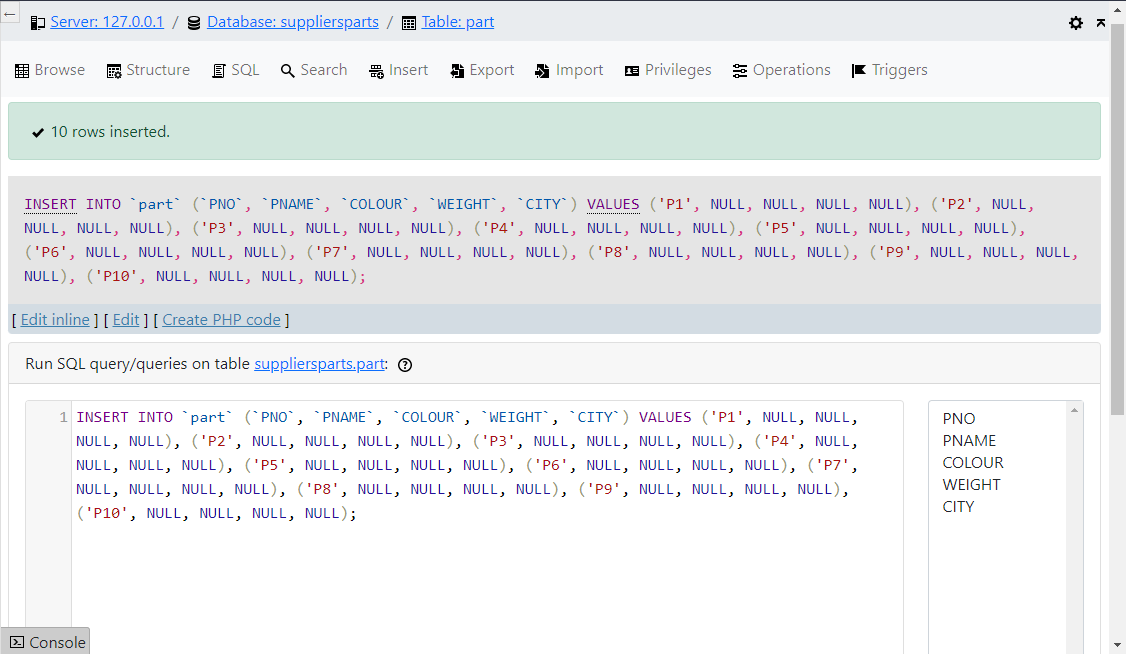
Graphical user interface, text, application, email

Description automatically generated

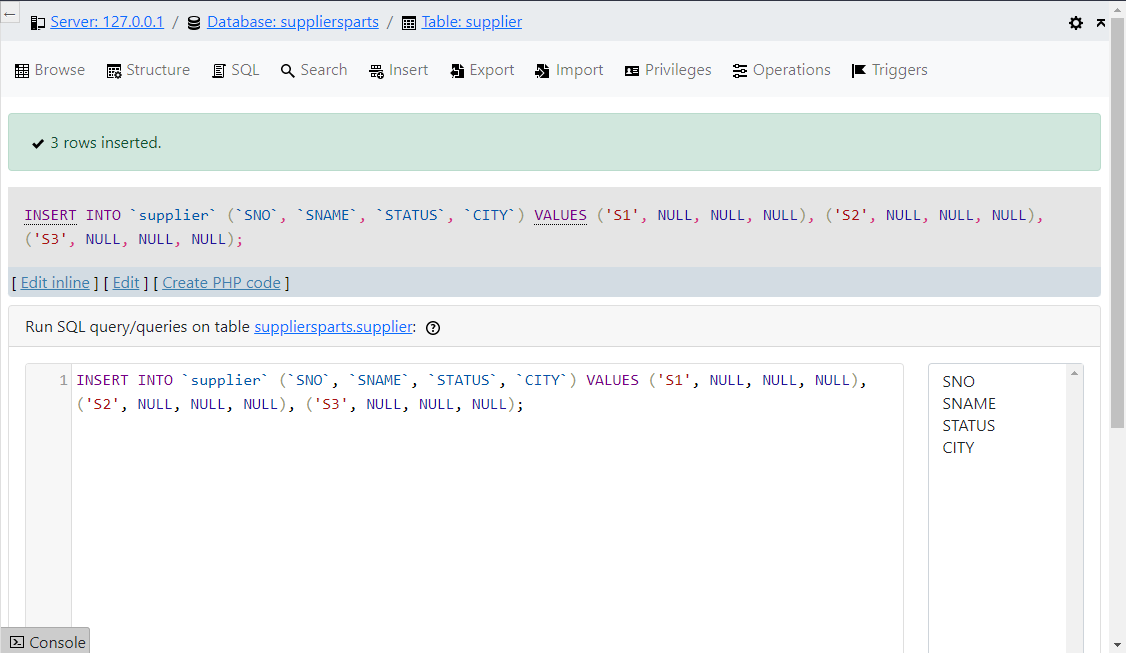


SAMPLE DATA:

PART Nos: P1, P2, P3, P4, P5, P6, P7, P8, P9, P10



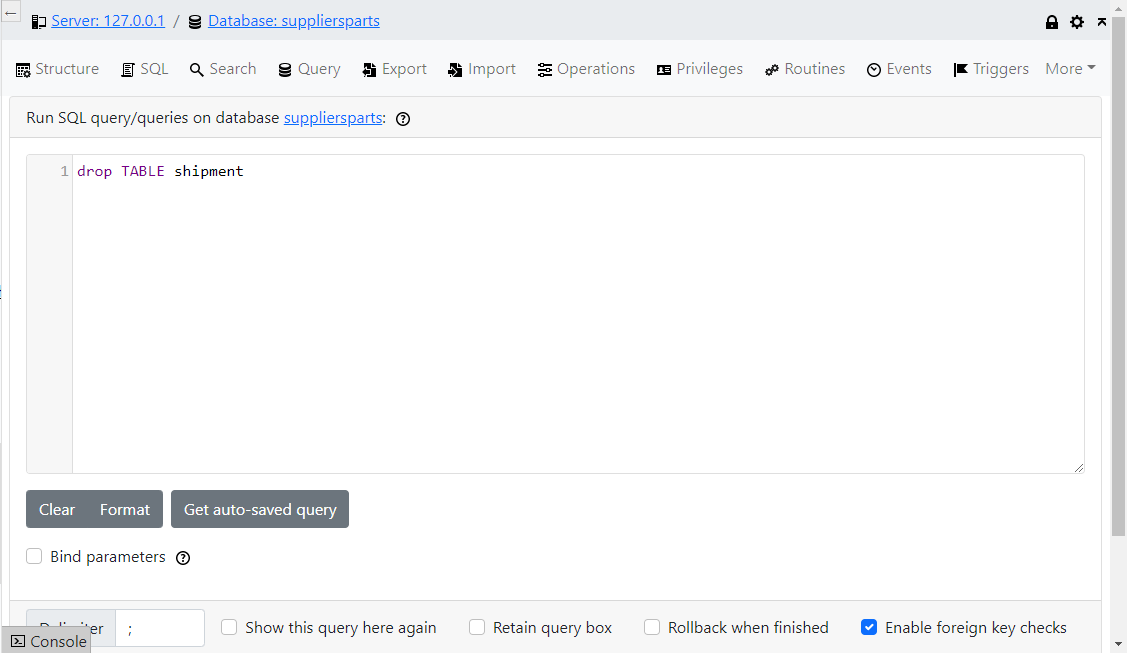
SUPPLIER Nos: S1, S2, S3;



STATUS: 10, 20, 30

To delete a table use the following command:

DROP TABLE table\_name



SQL supports the following data types:

CHAR - Character REAL - Real

INTEGER - Integer (fullword) NUMERIC - Real

SMALLINT - Integer (halfword) DECIMAL - Real

FLOAT - Floating Point DOUBLE PRECISION - Real

1. **DATA MANIPULATION**

The Data Manipulation Language (DML) commands are:

SELECT - Extracts records from a table

INSERT - Enters records into a table

UPDATE - Changes the values of fields in a record

DELETE - Removes records from a table

**3.1 SELECT**

The select command extracts data from a table. The result can be displayed, printed or saved. The general syntax is:

SELECT [DISTINCT] [ \*: field\_name {field\_name} …]

FROM table\_name [ { , table\_name} …]

[WHERE condition]

[ [ GROUP BY grouping\_criteria] [HAVING predicate] ]

[ORDER BY ordering\_criteria]

The WHERE clause is used to introduce conditions. The following are the relational operators:

= Equal to

> Greater than

< Less than

<> Not equal to

>= Greater than or equal to

<= Less than or equal to

IN Equal to an element of

BETWEEN…AND True if left value lies in inclusive interval specified by BETWEEN and AND.

EXISTS Used only with subqueries. True if subquery returns at least one record.

ALL/ANY/SOME Used only with suybqueries.

LIKE Used in pattern matching

IS NULL Used in searches to determine whether a field contains a value. Returns true if field empty.

The following are the logical operators:

AND True if both expressions are true

NOT Negation

OR True if one or both expressions are true

XOR True if one expresses is true, but not both.

The ORDER BY clause is used to sort the records and may be followed by the field name whose values indicate the order of sorting. These are ASC (ASCENDING) or DESC (DESCENDING). ASC is the default.

The GROUP BY clause allows you to group the results of a SELECT command.

The HAVING clause has the same effect with the GROUP by as WHERE has on individual records.

SQL includes only simple statistics as standard functions. These are:

AVG - Average

COUNT - Number

MAX - Maximum

MIN - Minimum

SUM - Summation

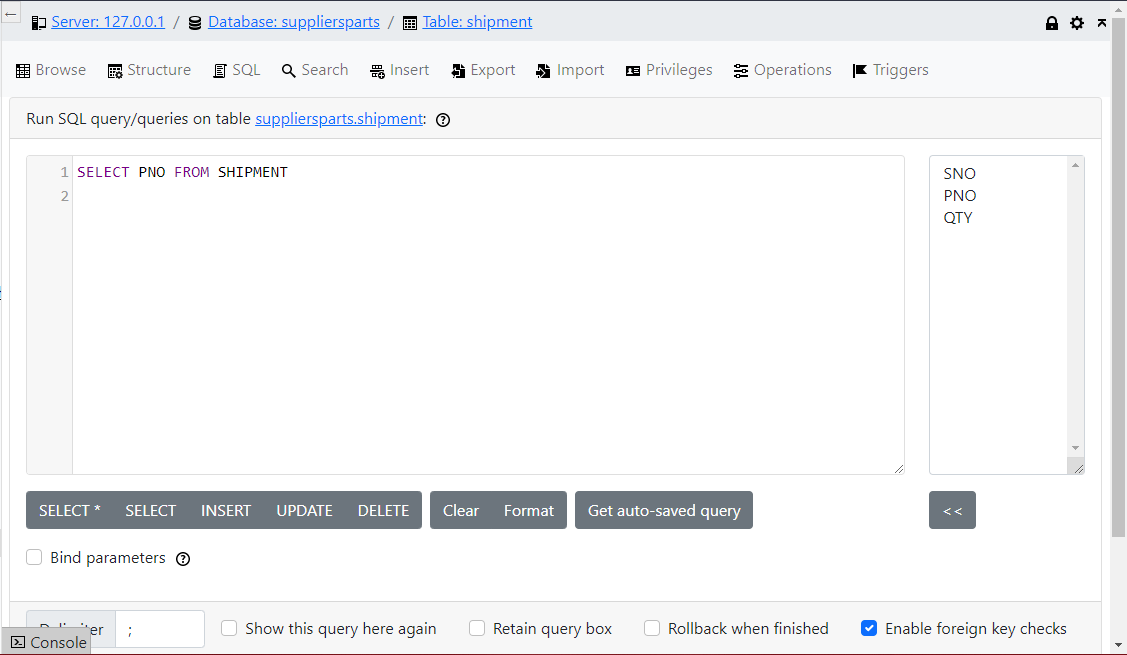
The SELECT command may be nested into subqueries using parentheses. The innermost subquery is executed first.

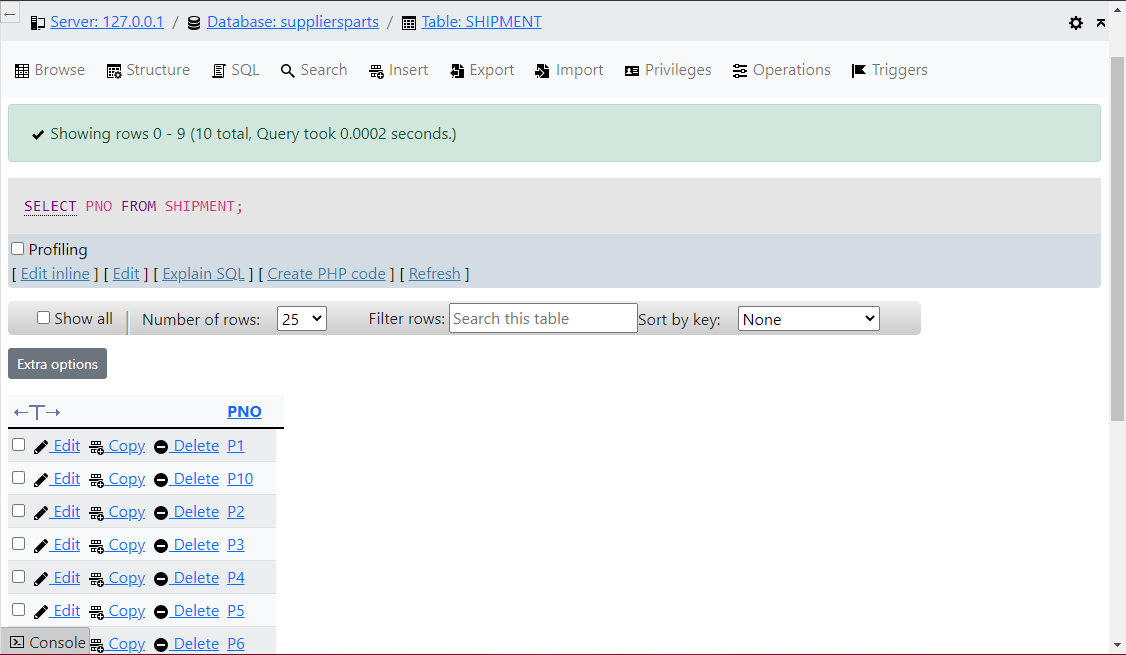
# Examples

1. Simple retrieval - Get part numbers for all parts supplied.

SELECT PNO

FROM SHIPMENT





2. Simple retrieval - Get full details of all suppliers.

SELECT \*

FROM SUPPLIER

Graphical user interface, text, email

Description automatically generated

Graphical user interface, text

Description automatically generated

3. Qualified retrieval - Get supplier numbers for suppliers in Paris with

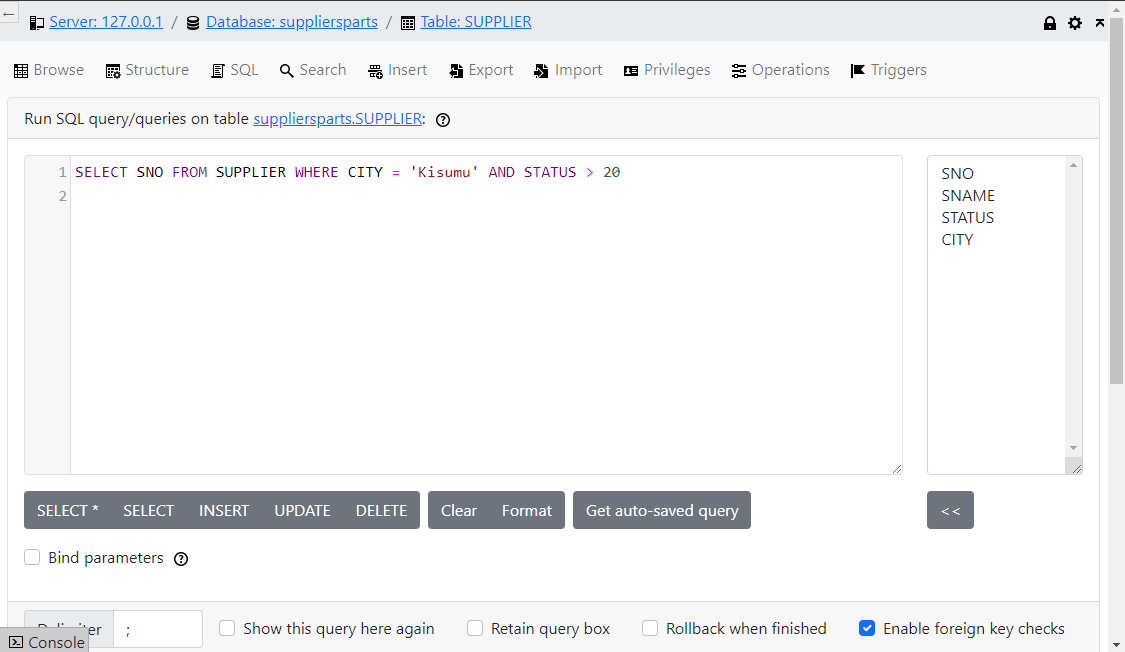
status > 20

SELECT SNO

FROM SUPPLIER

WHERE CITY = ‘Kisumu’

AND STATUS > 20



Graphical user interface, text, email

Description automatically generated

4. Retrieval with ordering - Get supplier numbers and status for suppliers in Paris in descending order of status.

SELECT DISTINCT SNO, STATUS

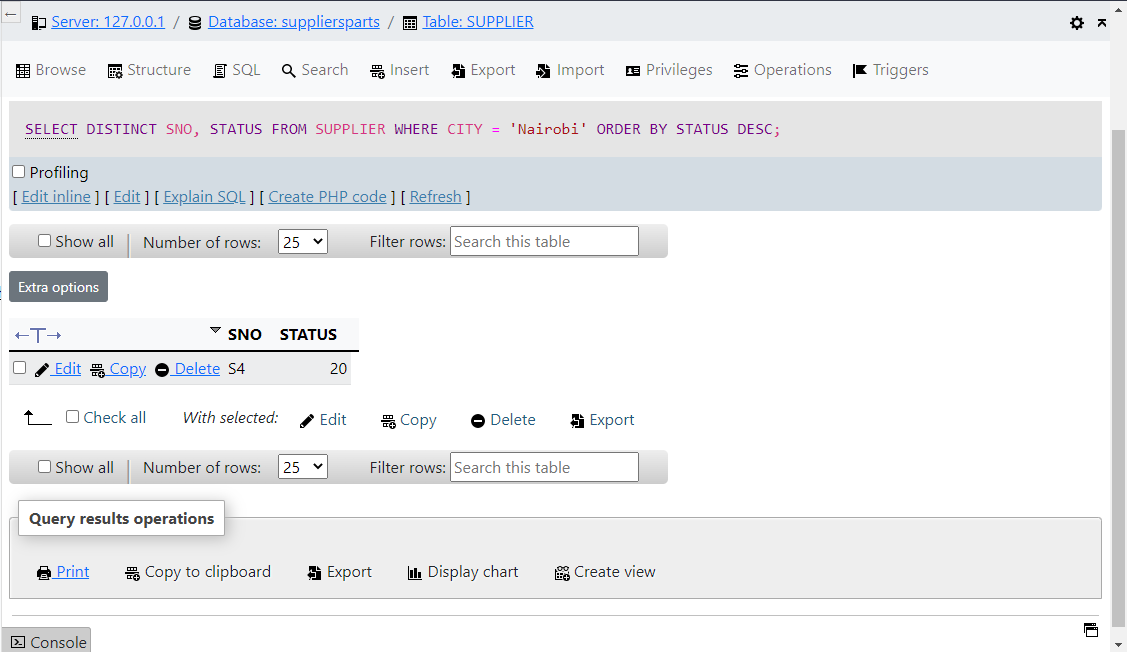
FROM SUPPLIER

WHERE CITY = ‘Nairobi’

ORDER BY STATUS DESC

Graphical user interface, text, email

Description automatically generated

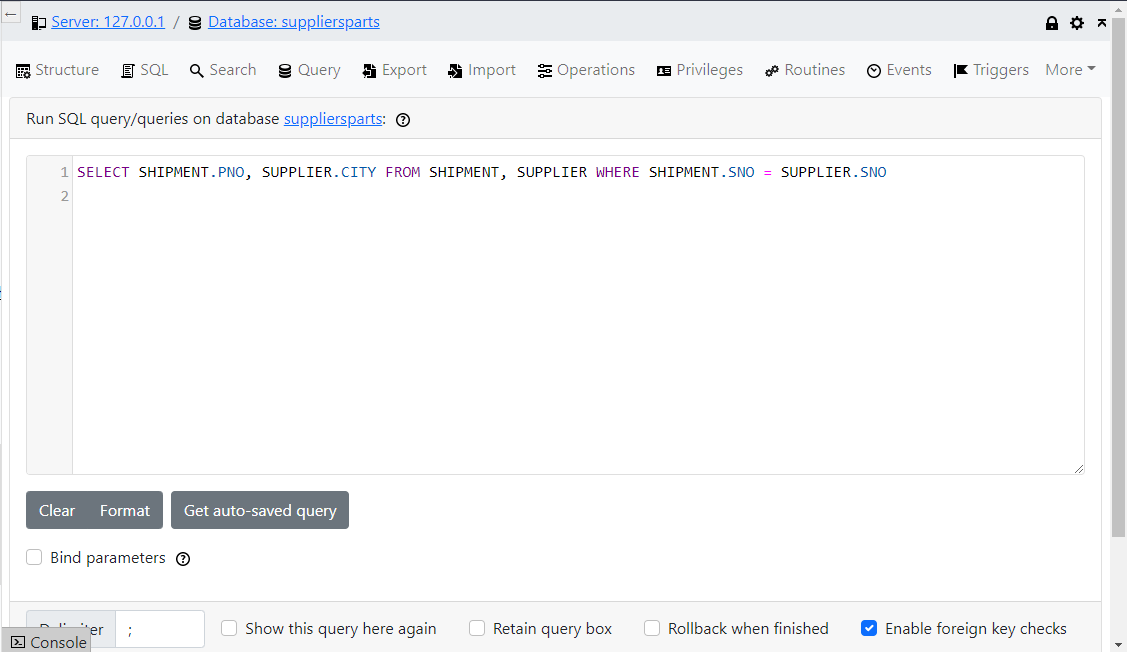


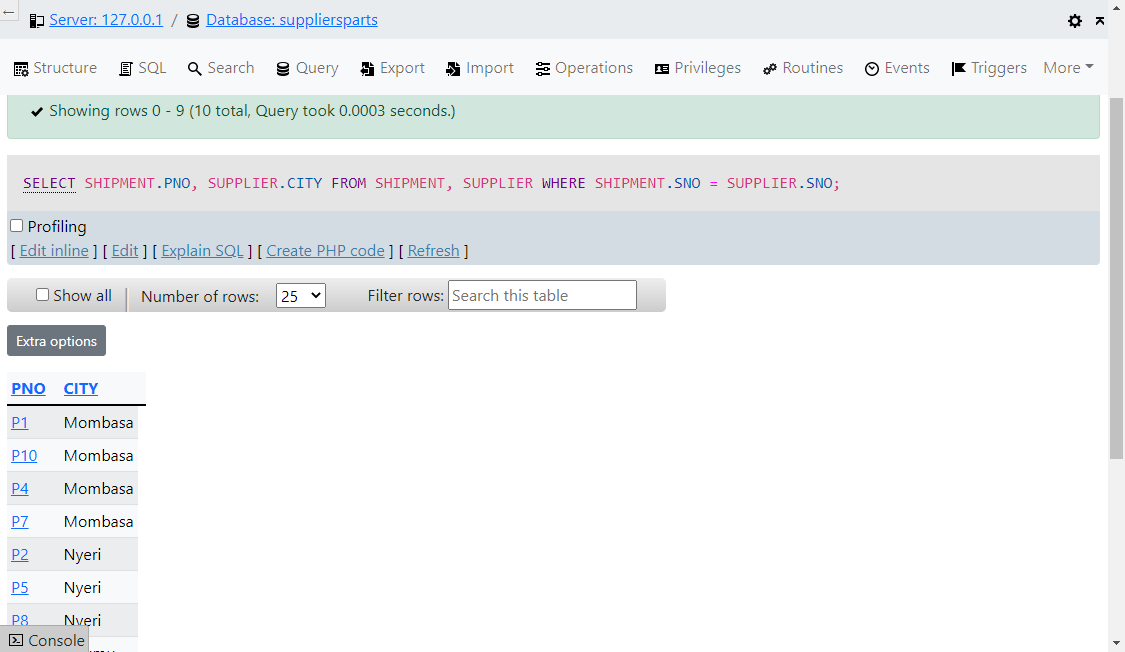
1. Retrieval from more than one table - For each part supplied, get part number and names of all cities supplying the part.

SELECT SHIPMENT. PNO, SUPPLIER. CITY

FROM SHIPMENT, SUPPLIER

WHERE SHIPMENT. SNO = SUPPLIER. SNO





6. Retrieval involving a join of a table with itself - Get all pairs of supplier numbers such that two suppliers are locate in the same city.

SELECT FIRST. SNO , SECOND. SNO

FROM SUPPLIER FIRST, SUPPLIER SECOND

WHERE FIRST. CITY = SECOND. CITY

AND FIRST. SNO < SECOND. SNO

Graphical user interface, text, application

Description automatically generated

7. Retrieval using IN - Get supplier names for suppliers who supply part P2.

SELECT SNAME

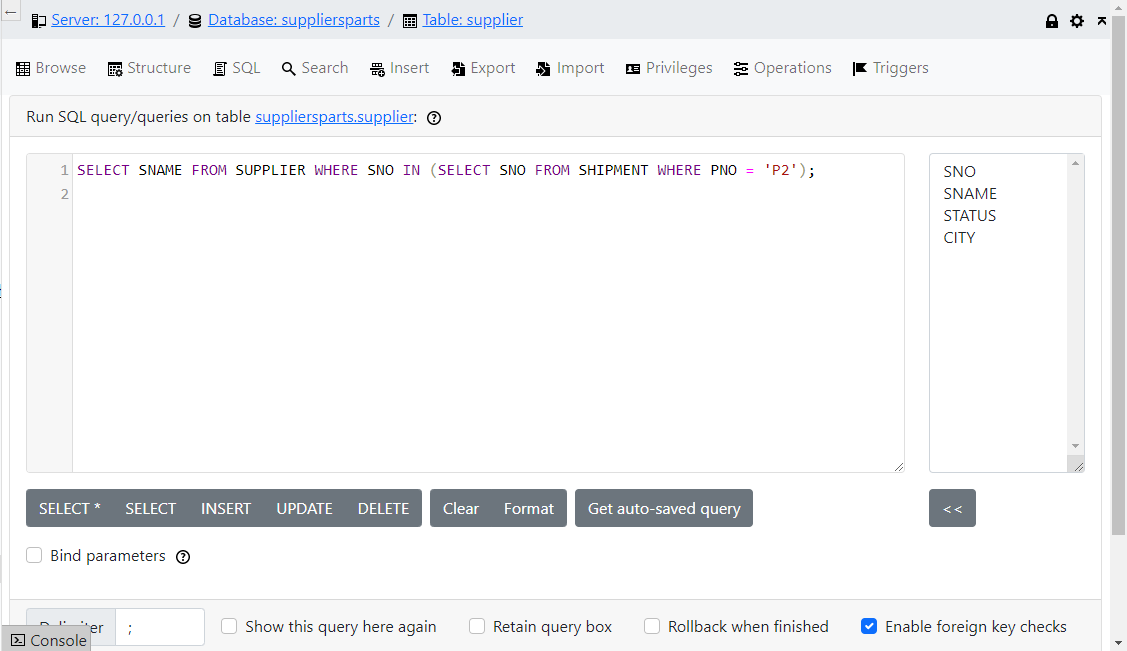
FROM SUPPLIER

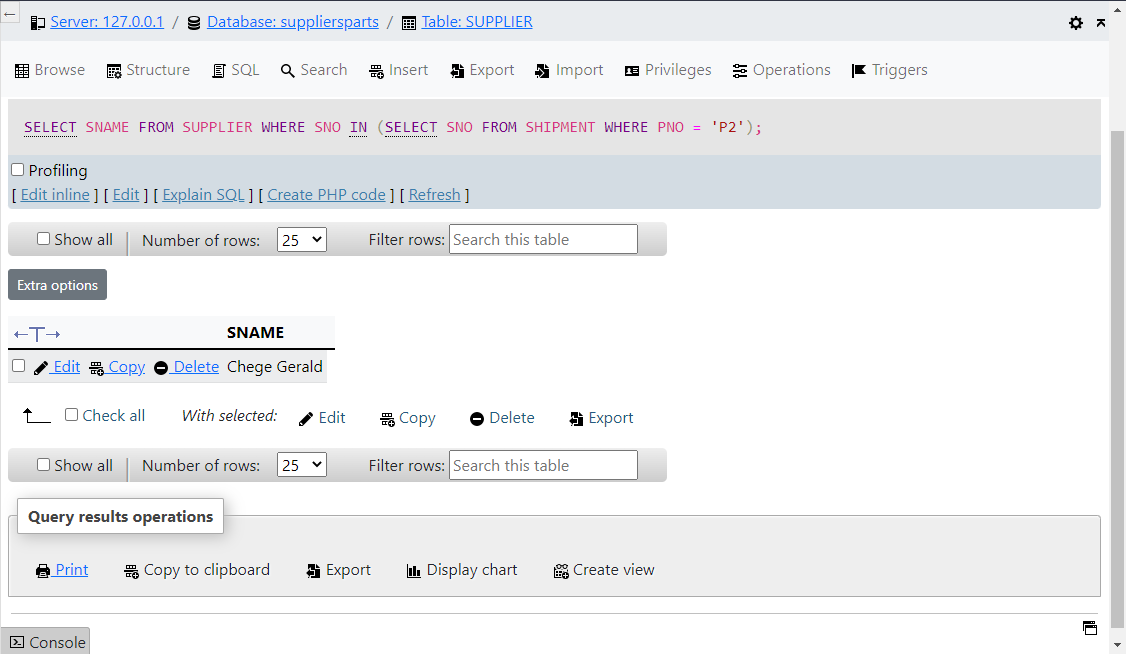
WHERE SNO IN

(SELECT SNO

FROM SHIPMENT

WHERE PNO = ‘P2’);





8. Retrieval with multiple levels of nesting - Get supplier names for suppliers who supply at least one red part.

SELECT SNAME

FROM SUPPLIER

WHERE SNO IN

(SELECT SNO

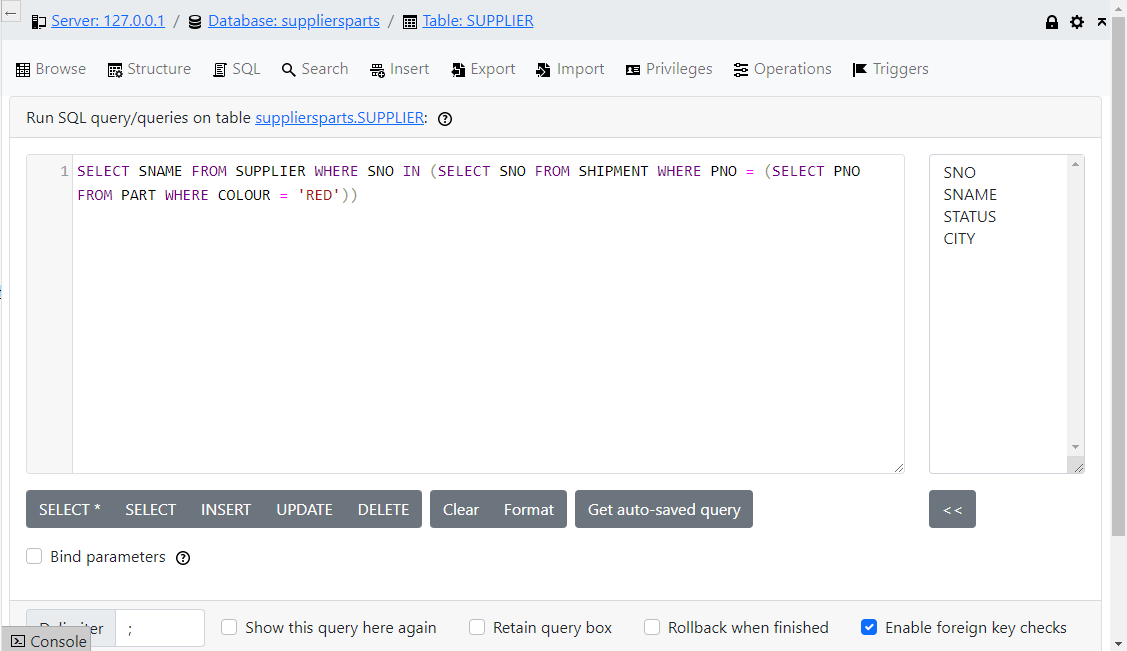
FROM SHIPMENT

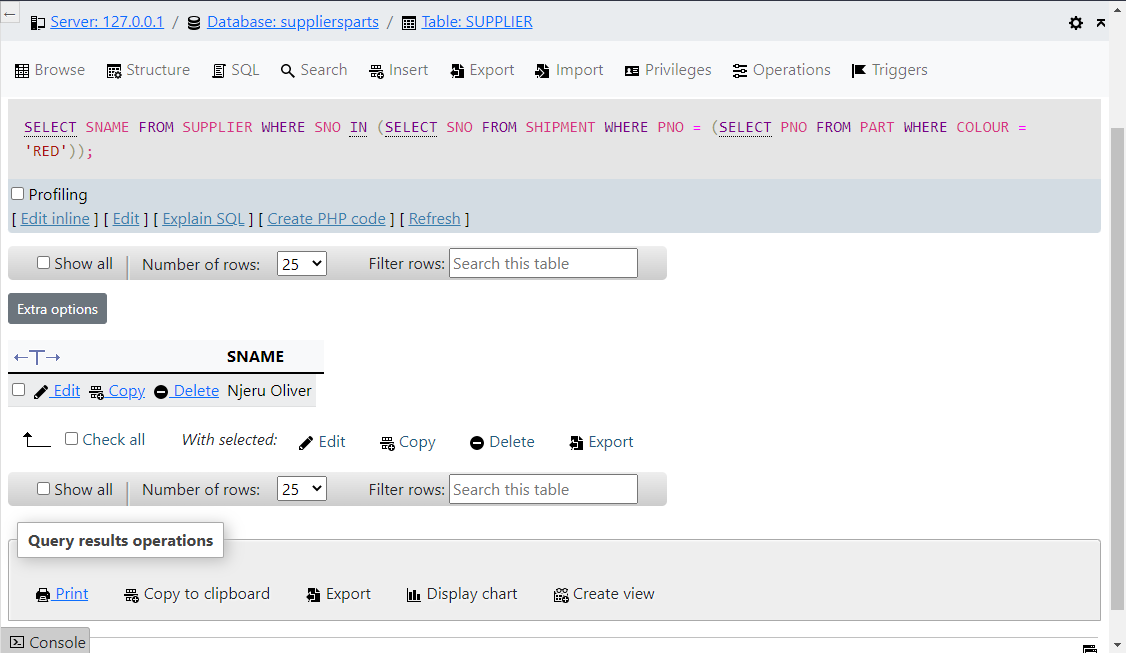
WHERE PNO =

(SELECT PNO

FROM PART

WHERE COLOUR = ‘RED’ ) )





9. Retrieval using ALL - Get supplier names for suppliers who do not supply

part P2.

SELECT SNAME

FROM SUPPLIER

WHERE ‘P2’ <> ALL

(SELECT PNO

FROM SHIPMENT

WHERE SNO = SUPPLIER. SNO )

Graphical user interface, text, email

Description automatically generated

Graphical user interface, text

Description automatically generated

10. Retrieval using EXISTS - Get supplier names for suppliers who supply part P2.

SELECT SNAME

FROM SUPPLIER

WHERE EXISTS

(SELECT \*

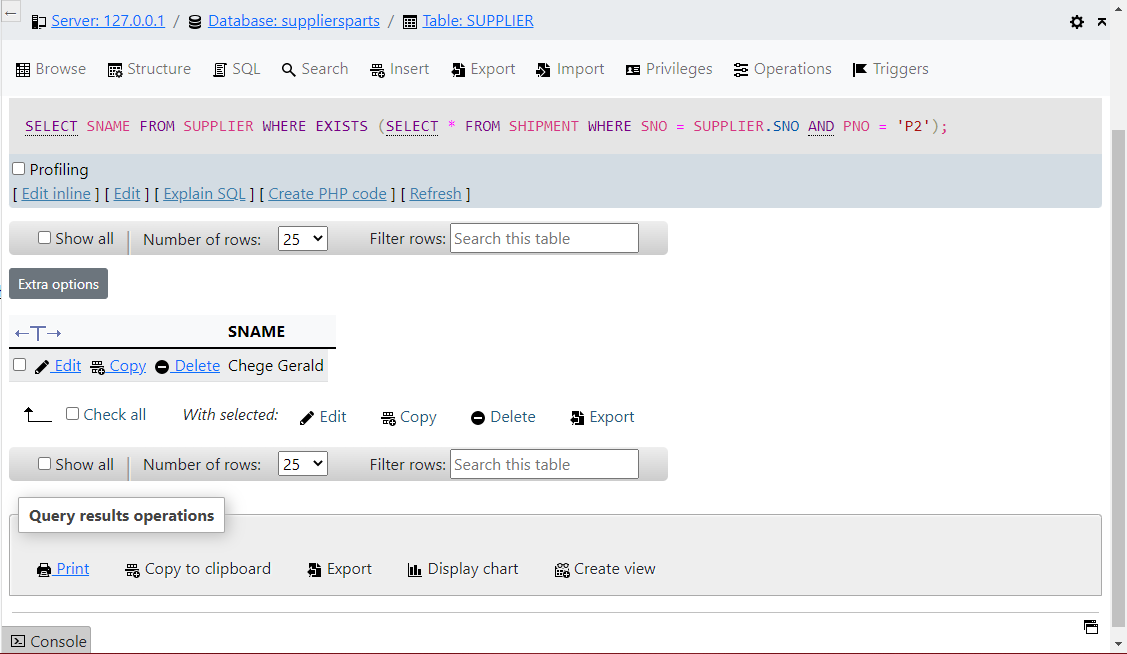
FROM SHIPMENT

WHERE SNO = SUPPLIER. SNO

AND PNO = ‘P2’)

Graphical user interface, text, email

Description automatically generated



11. Retrieval using NOT EXISTS - Get supplier names for suppliers who do not supply part P2.

SELECT NAME

FROM SUPPLIER

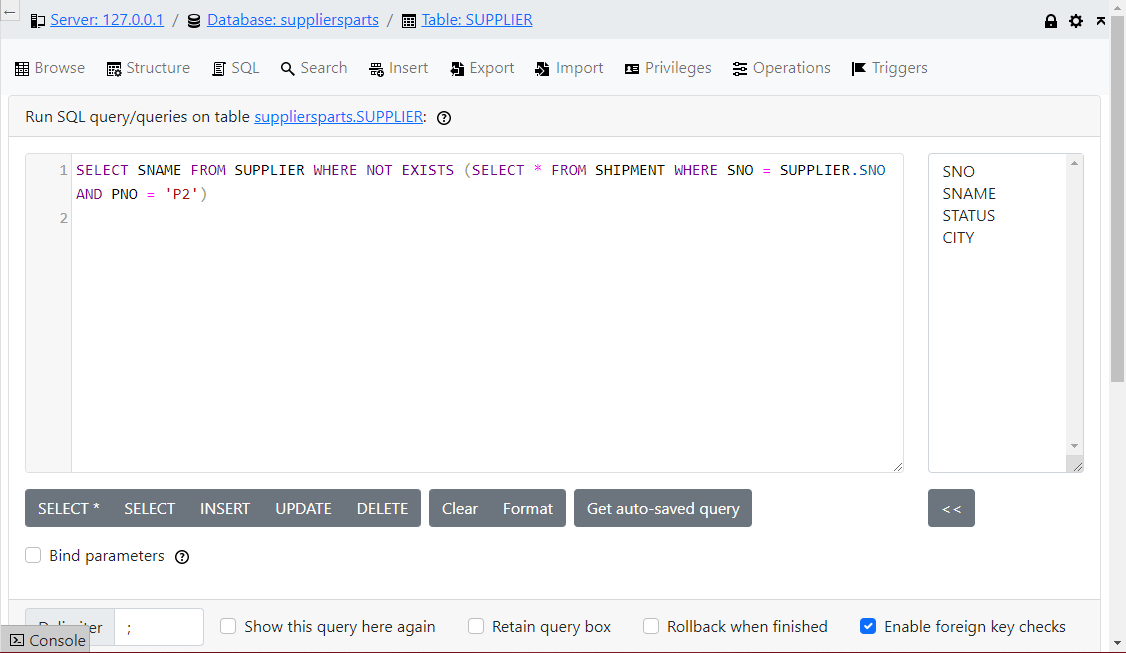
WHERE NOT EXISTS

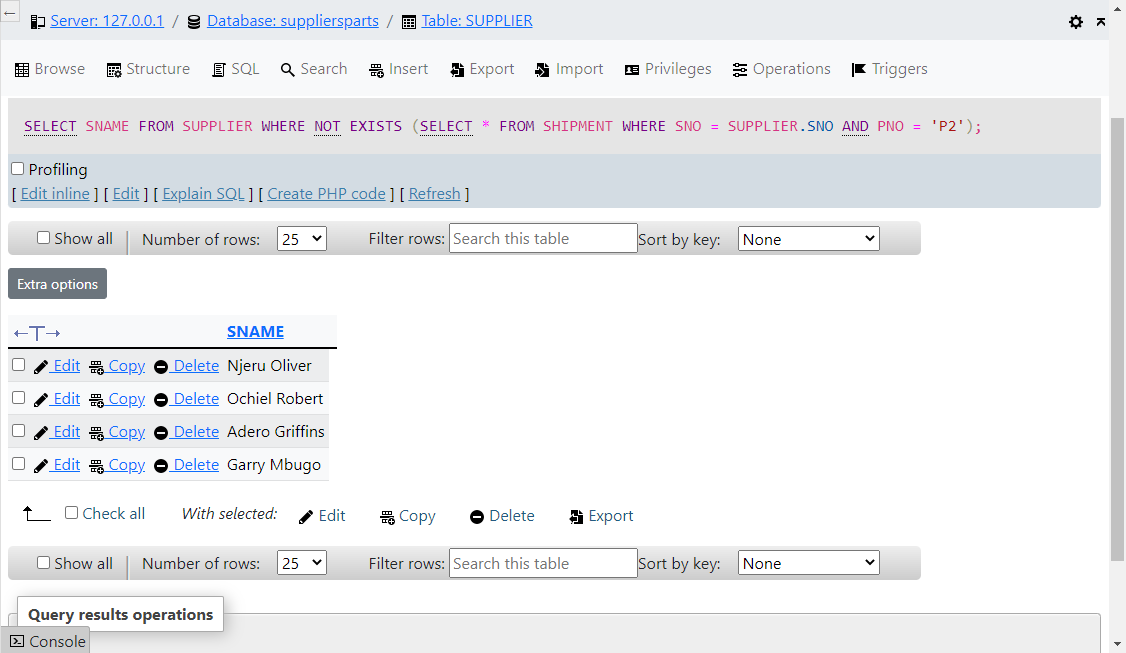
(SELECT \*

FROM SHIPMENT

WHERE SNO = SUPPLIER. SNO

AND PNO = ‘P2’)

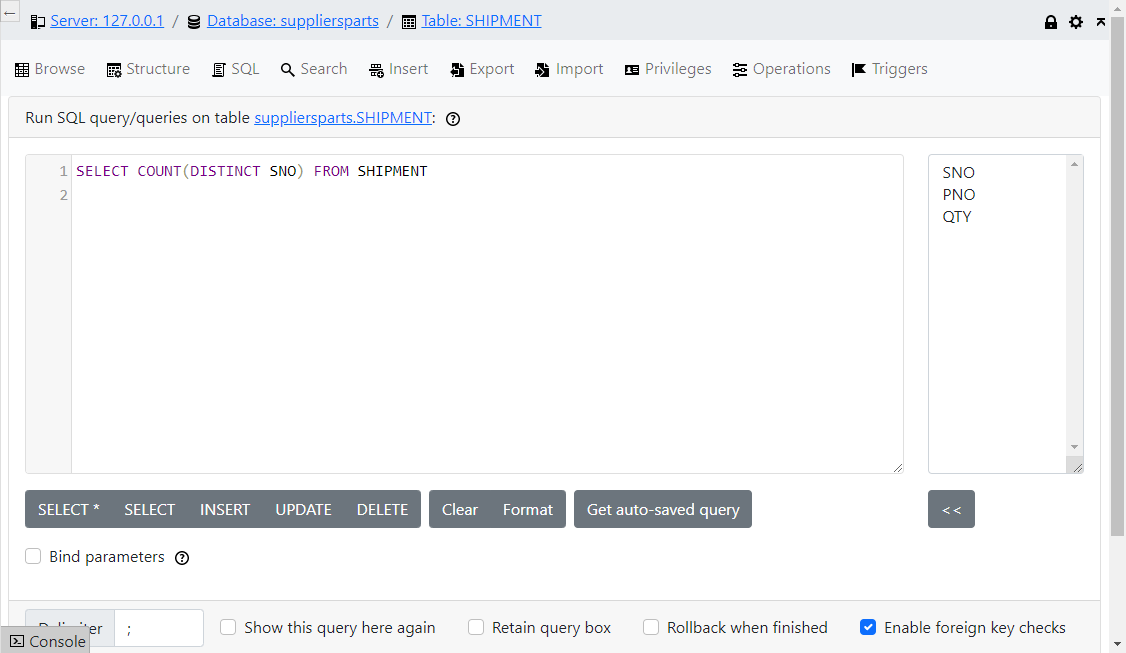


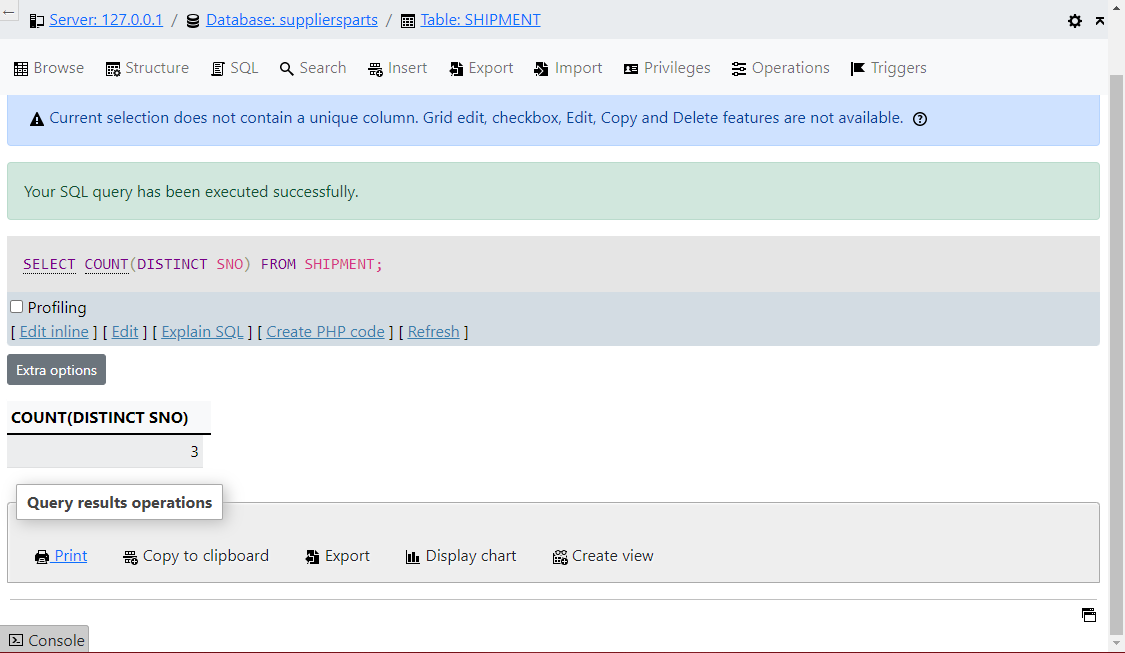


12. Functions in the SELECT clause - Get the total number of suppliers currently supplying parts.

SELECT COUNT (DISTINCT SNO)

FROM SHIPMENT





13. Function in the SELECT clause, with a predicate - Get the total quantity of part P2 supplied.

SELECT SUM (QTY)

FROM SHIPEMENT

WHERE PNO = ‘P2’

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

14. Function in a subquery – Get supplier names for suppliers with status value less than current maximum status value in the SUPPLIER table.

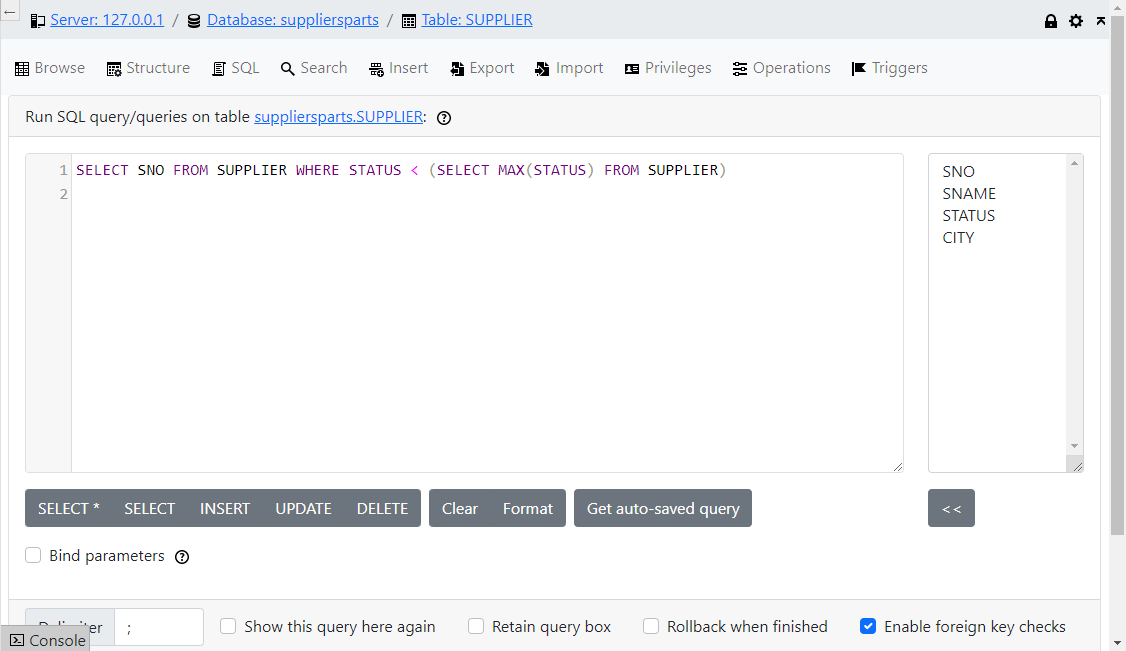
SELECT SNO

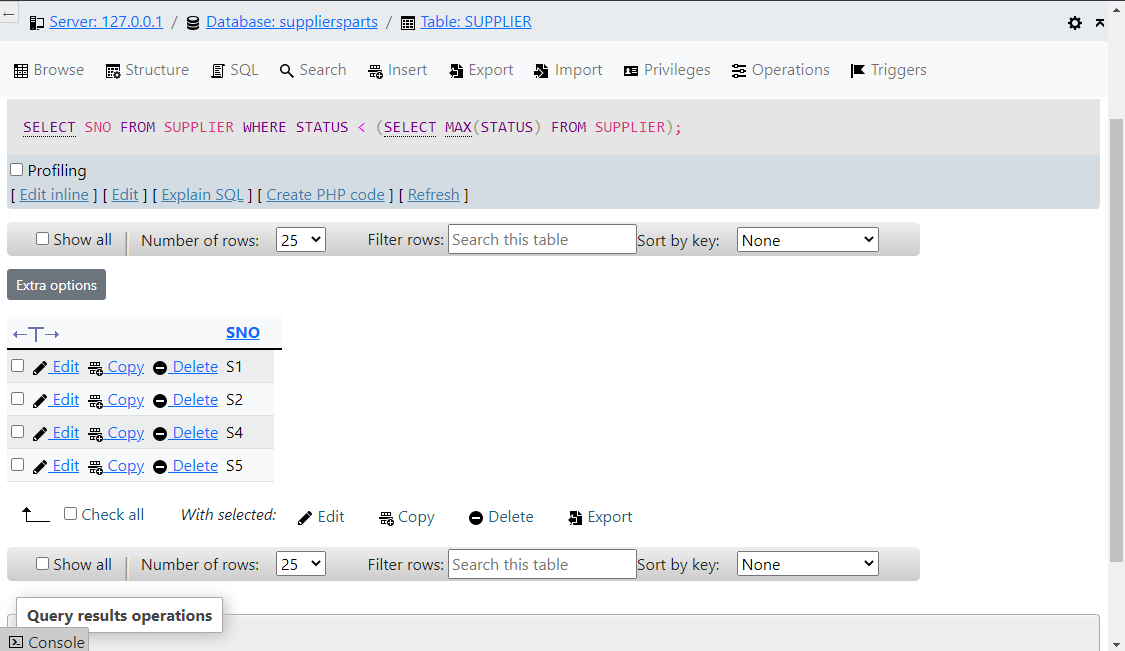
FROM SUPPLIER

WHERE STATUS<

(SELECT MAX (STATUS)

FROM SUPPLIER





15. Use of GROUP BY – For each part supplied, get the part number and the total quantity supplied of that part.

SELECT PNO, SUM (QTY)

FROM SHIPMENT

GROUP BY PNO

Graphical user interface, text, email

Description automatically generated

Graphical user interface, text

Description automatically generated

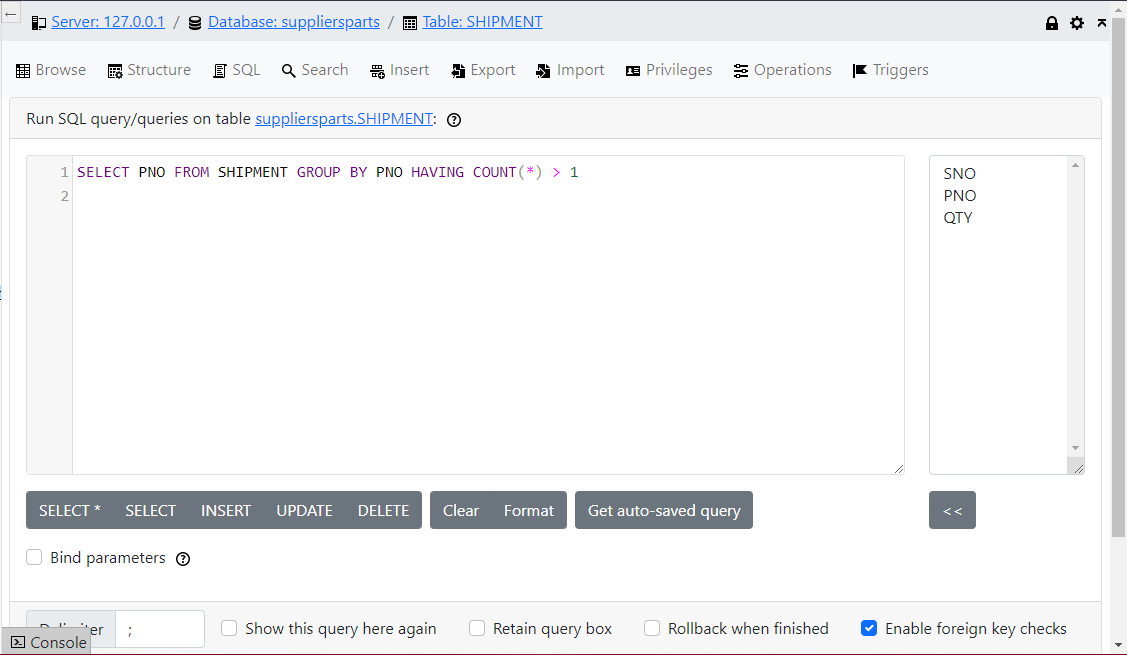
16. Use of GROUP BY with HAVING - Get part numbers for all parts supplied by more than one supplier.

SELECT PNO

FROM SHIPMENT

GROUP BY PNO

HAVING COUNT (\*) > 1



Graphical user interface, text, email

Description automatically generated

17. Comprehensive example – For all parts such that the total quantity supplied is greater than 300 (excluding from the total all shipments for which the quantity is less than or equal to 200), get the part number and the maximum quantity of the part supplied; and order the result by descending part number within those maximum quantity values.

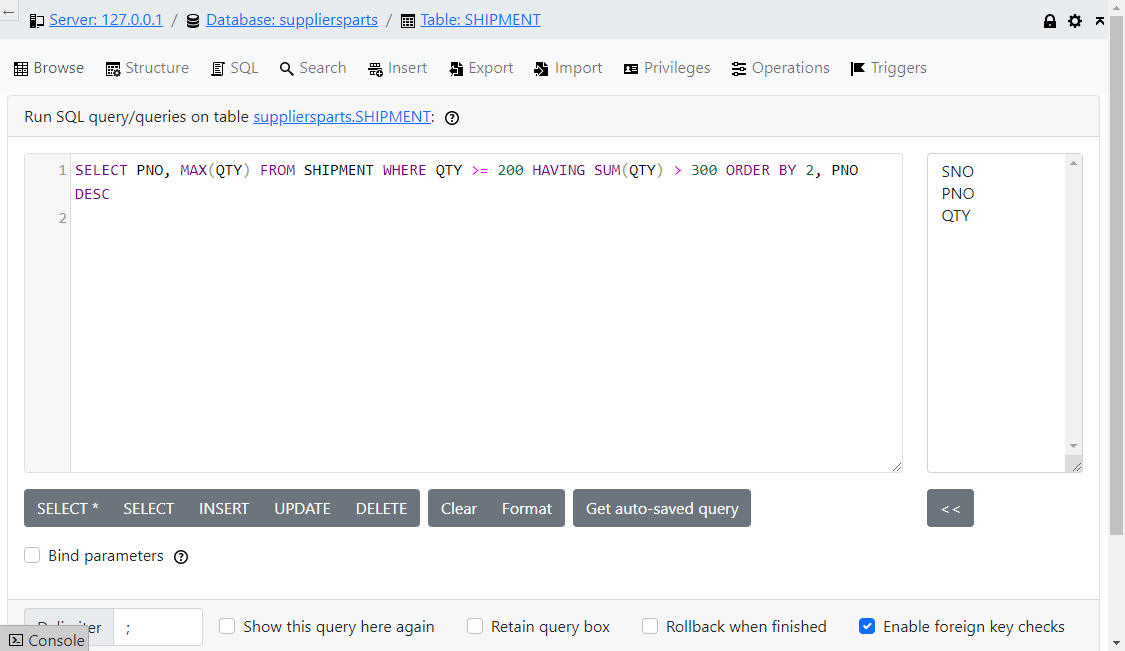
SELECT PNO, MAX (QTY)

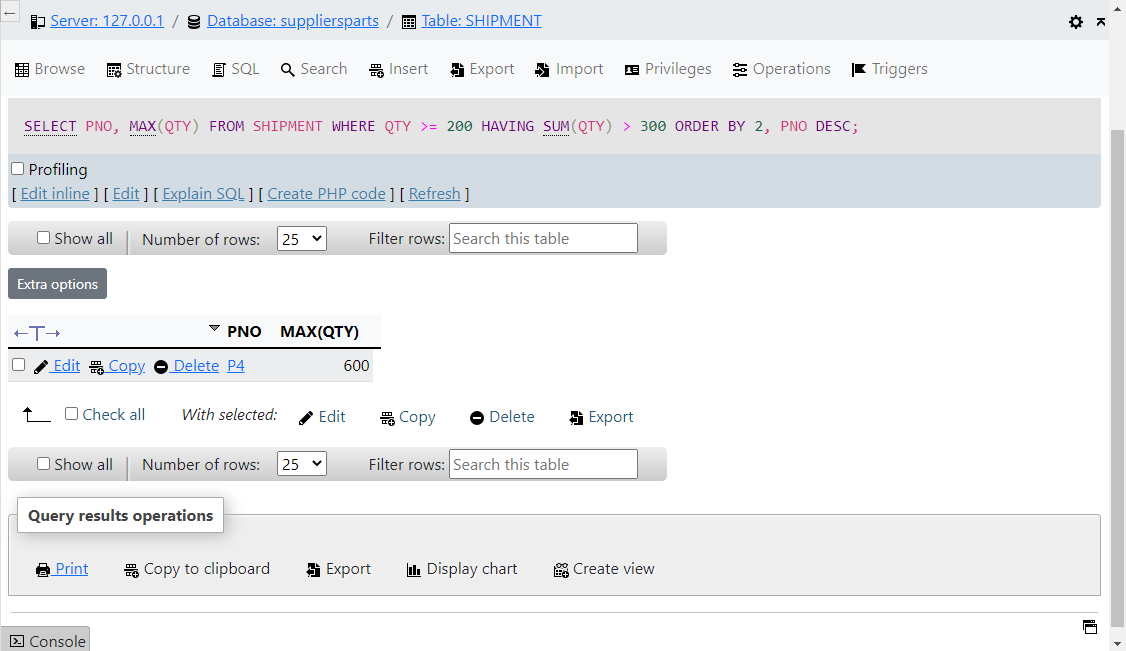
FROM SHIPMENT

WHERE QTY => 200

HAVING SUM (QTY) > 300

ORDER BY 2, PNO DESC





* 1. **INSERT**

INSERT allows us to enter data into a table. We can enter the data directly, transfer it via parameters, or select it from another table. The general syntax is:

INSERT INTO table\_name

[ ( field\_name {, field\_name} ]

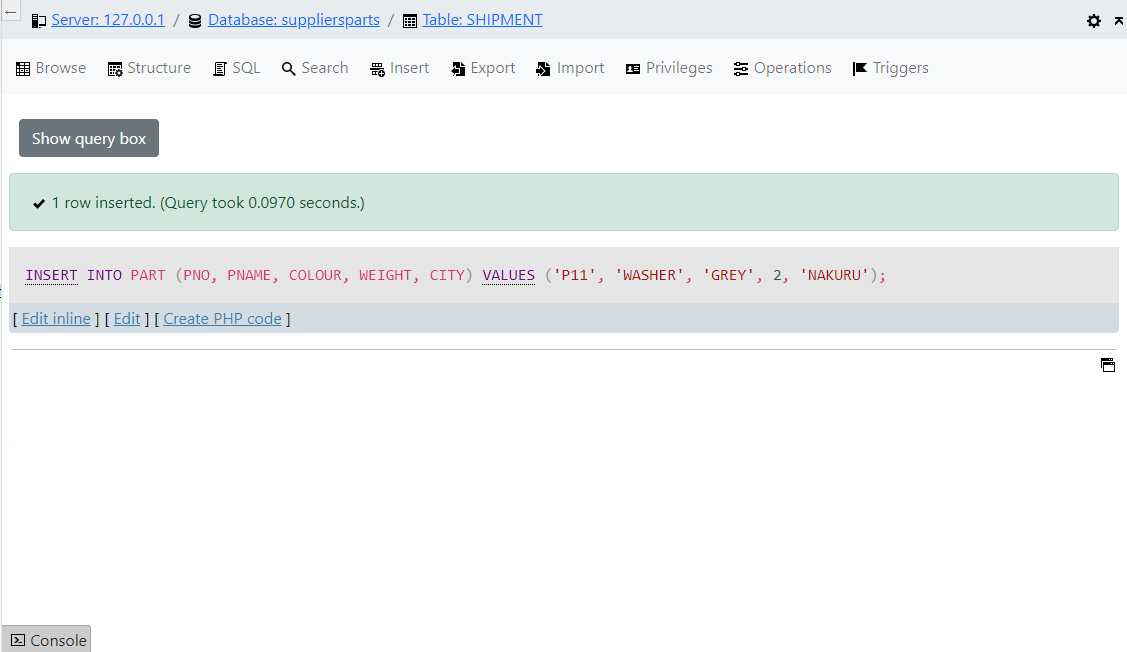
[ VALUES (expression {,expression} ]

# Examples

1. Single-record insertion – Add part P7 (name ‘washer’, colour ‘GREY’ , weight 2, city ‘ATHENS’) to table PART.

INSERT INTO PART (PNO, PNAME, COLOUR, WEIGHT, CITY)

VALUES (‘P7’ , ‘WASHER’ , ‘GREY’ , 2, ‘NAKURU’)



2. Multiple-record insertion – Table TEMP has one column, called PNO. Enter into TEMP part numbers for all parts supplied by supplier S2.

INSERT INTO TEMP:

SELECT PNO

FROM SHIPMENT

WHERE SNO = ‘S2’

Graphical user interface, text, application, website

Description automatically generated

**3.3 UPDATE**

The UPDATE command is used to change filed values. General syntax is:

UPDATE table\_name

SET field\_name = new\_value [{,field\_name = new\_value}]…

[ Where condition ]

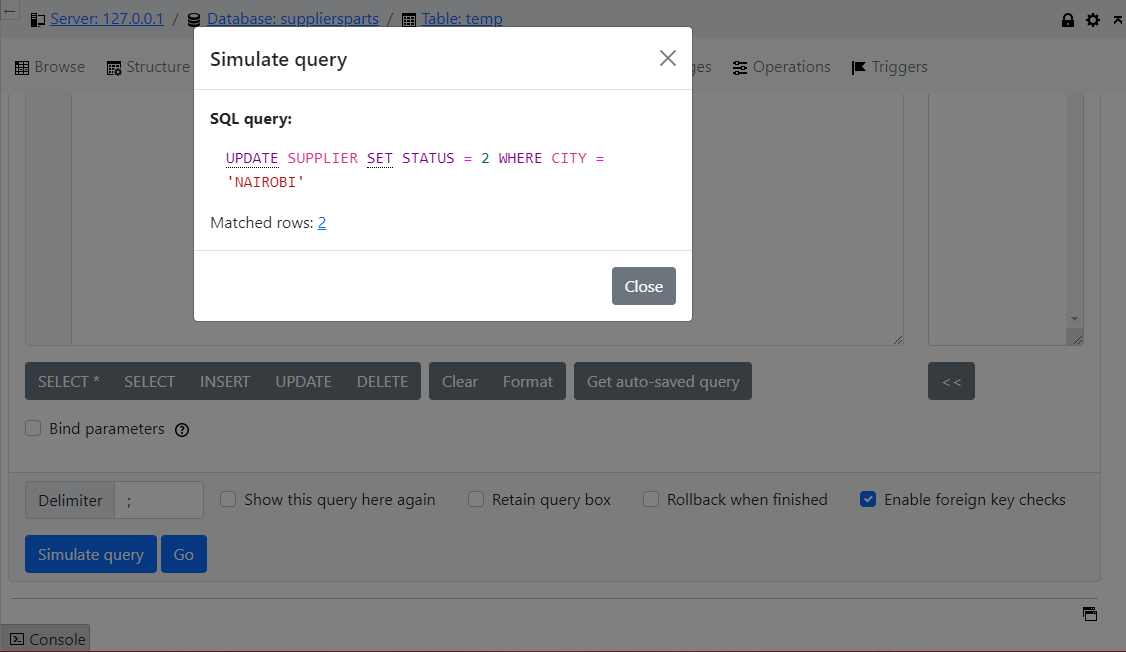
# Examples

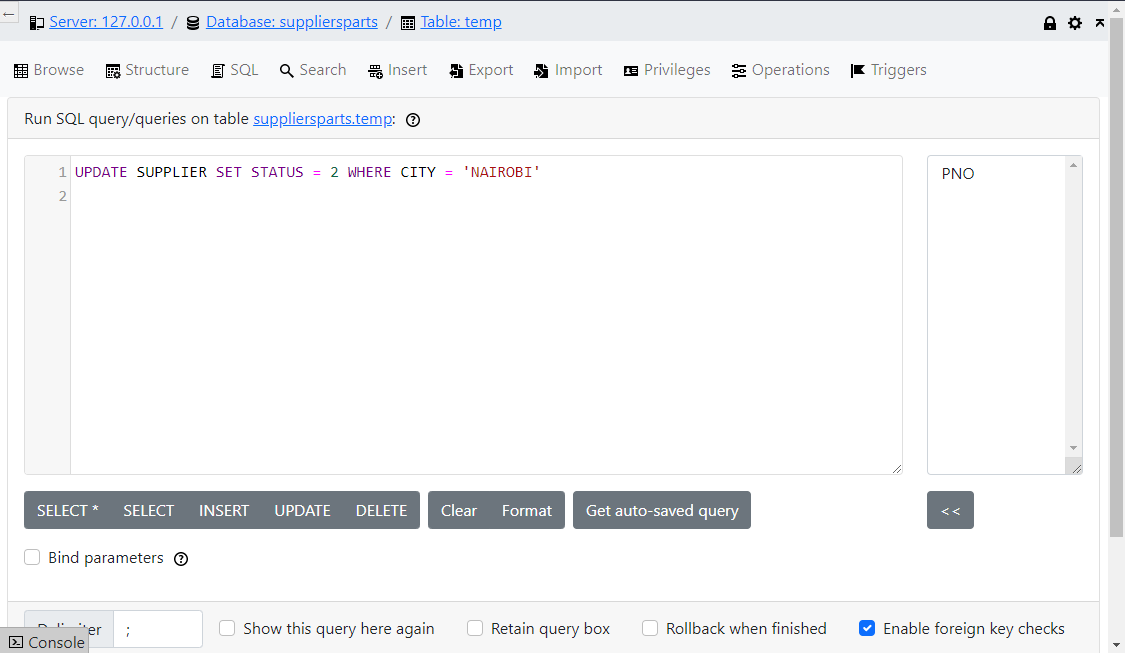
1. Multiple record update - Double the status of all suppliers in London.

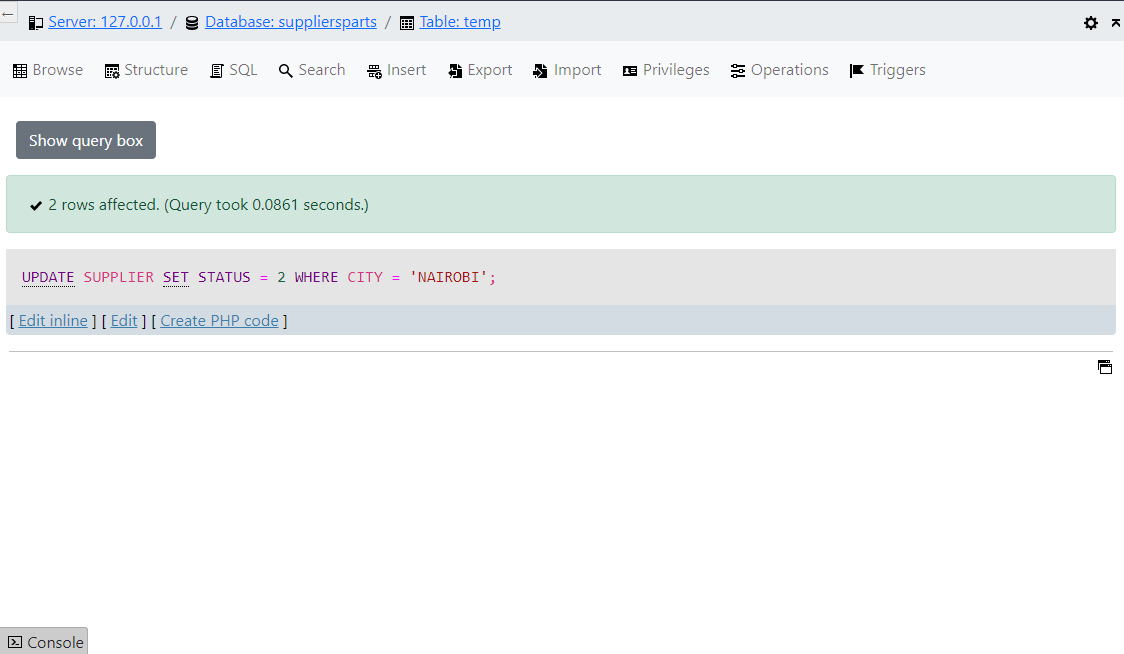
UPDATE SUPPLIER

SET STATUS = 2 \* STATUS

WHERE CITY = ‘NAIROBI’







2. Update with a subquery – set the quantity to zero for all suppliers in London.

UPDATE SHIPMENT

SET QTY = 0

WHERE ‘LONDON’ =

(SELECT CITY

FROM SUPPLIER

WHERE SNO = SHIPMENT. SNO

Graphical user interface, text, email

Description automatically generated

Graphical user interface, text, website

Description automatically generated

* 1. **DELETE**

The DELETE command removes records from a table. General syntax is:

DELETE FROM table\_name

[ WHERE condition ]

If there is no WHERE condition all records are removed but the structure remains

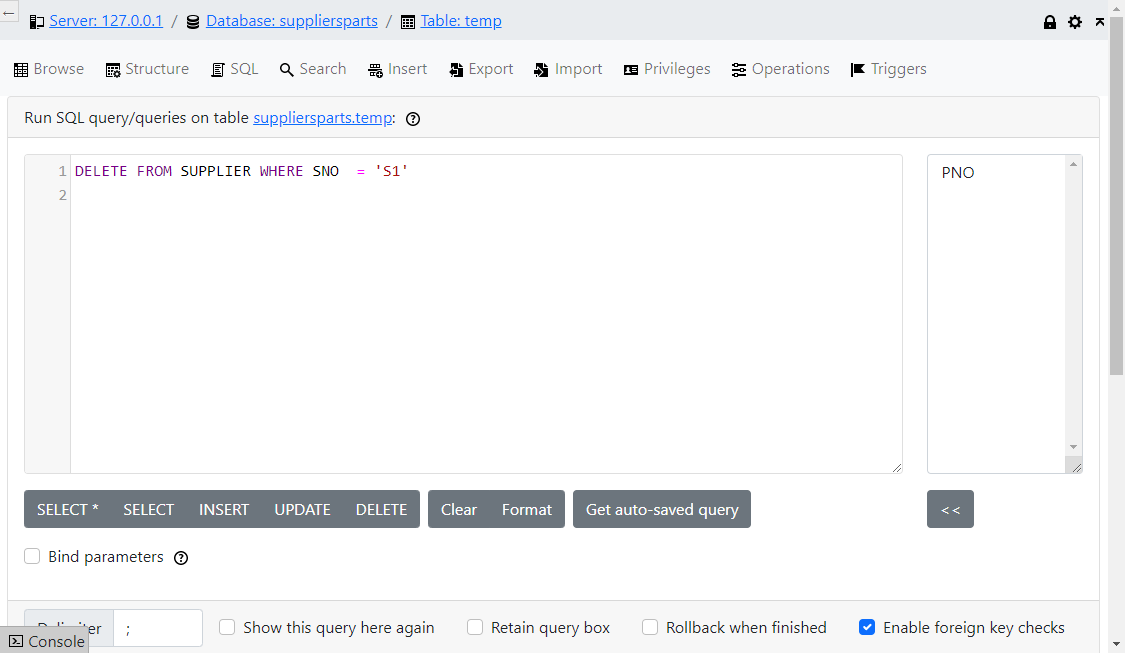
the same.

# Examples

1. Single-record deletion - Delete supplier S1.

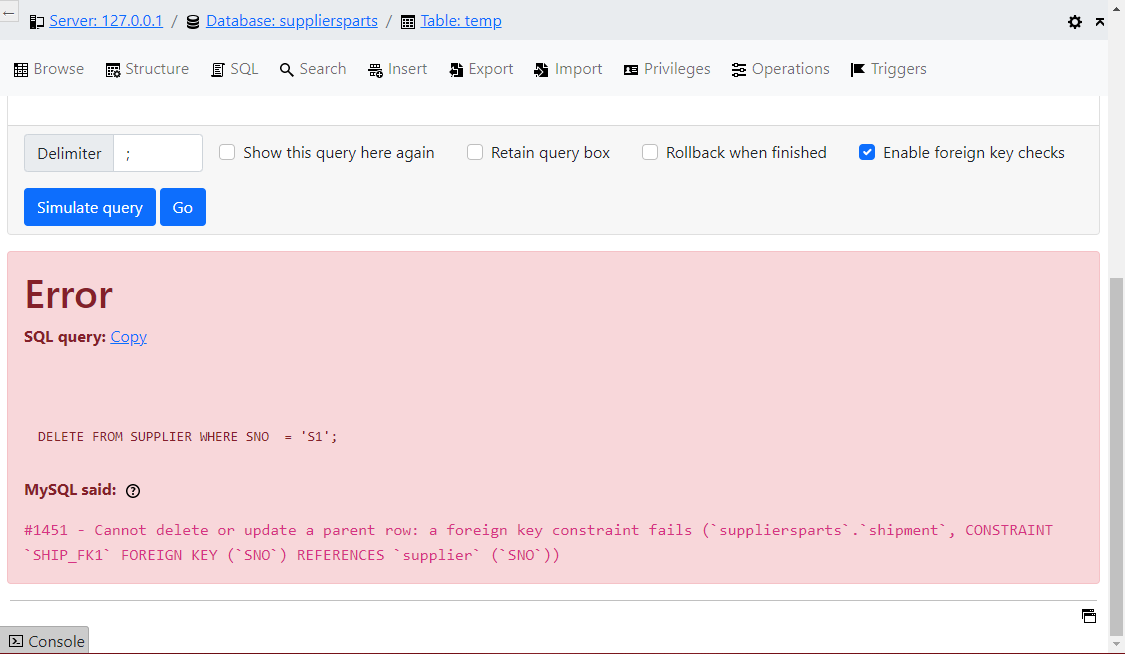
DELETE SUPPLIER

WHERE SNO = ‘S1’



Graphical user interface, text, application, email

Description automatically generated



2. Multiple – record, multiple table deletion - Delete all shipments from suppliers in London and also the suppliers concerned.

DELETE SHIPMENT

WHERE ‘MOMBASA’ =

(SELECT CITY

FROM SUPPLIER

WHERE SNO = SHIPMENT. SNO)

DELETE SUPPLIER

WHERE CITY = ‘MOMBASA’

Graphical user interface, text, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

* 1. **VIEWS**

A view can be defined as a virtual table – a table that does not exist on its won rights. Views are not directly supported by their own physically stored data.

THE CREATE VIEW has the following general syntax:

CREATE VIEW view\_name [[ field ] [{, field}]…]

AS SELECT [DISTINCT] [\* I field1 [{, field2}]…]

FROM table\_name I VIEW\_name

[ WHERE condition ]

[ GROUP BY grouping\_criteria ]

[ HAVING predicate ]

To delete a view use the following command:

DROP VIEW view\_name

There are restrictions on manipulating views, particularly when using UPDATE, INSERT OR DELETE.

1. **DATA CONTROL**

The Data Control Language (DCL) commands are used to control user access to the database. The DCL is concerned with privacy and security issues. The DCL commands are:

GRANT - assign the status of

REVOKE - remove the status of

There are three user privileges or user types:

1. CONNECT: lowest privileges with the following facilities:

* Access to the database
* Right to manipulate publicly accessible data
* Permission to access other people’s data only if allowed by owners.

2. RESOURCE: Intermediate level with the following facilities:

* All rights that accrue to a CONNECT
* The right to create tables, indexes and clusters
* Permission to grant the rights to create tables, indexes and clusters to other users
* Permission to GRANT or REVOKE rights such as SELECT or UPDATE to own objects.
* Perform auditing with own database objects

3. DBA: Highest authority with the following facilities:

* All rights that accrue to CONNECT and RESOURCE
* GRANT and REVOKE access privileges to the database
* Perform all DDL, DML and DCL commands
* Establish and change parameters of the database program
* Establish and change database logical and physical structure
* Backup database and regenerate damaged databases
* Audit how users have used their rights and generally oversee database operations
* See everything in the system data dictionary and make limited changes to it

1. **INDEXES AND CLUSTERS**

The general syntax of the CREATE INDEX command is:

CREATE [ UNIQUE ] INDEX index\_name

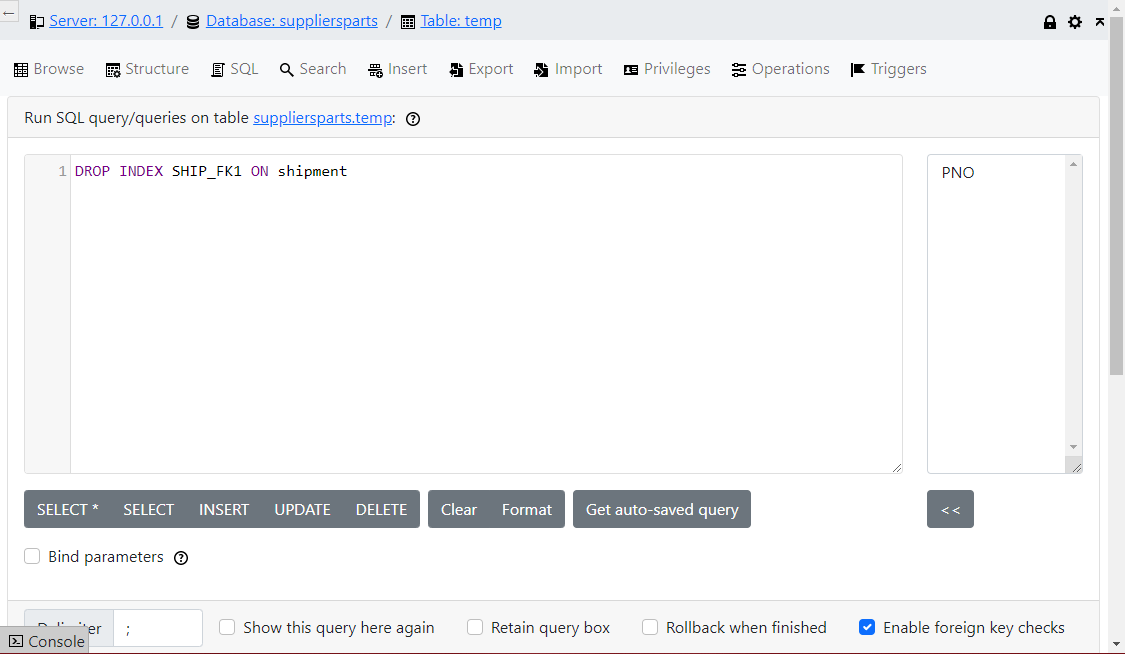
ON table\_name

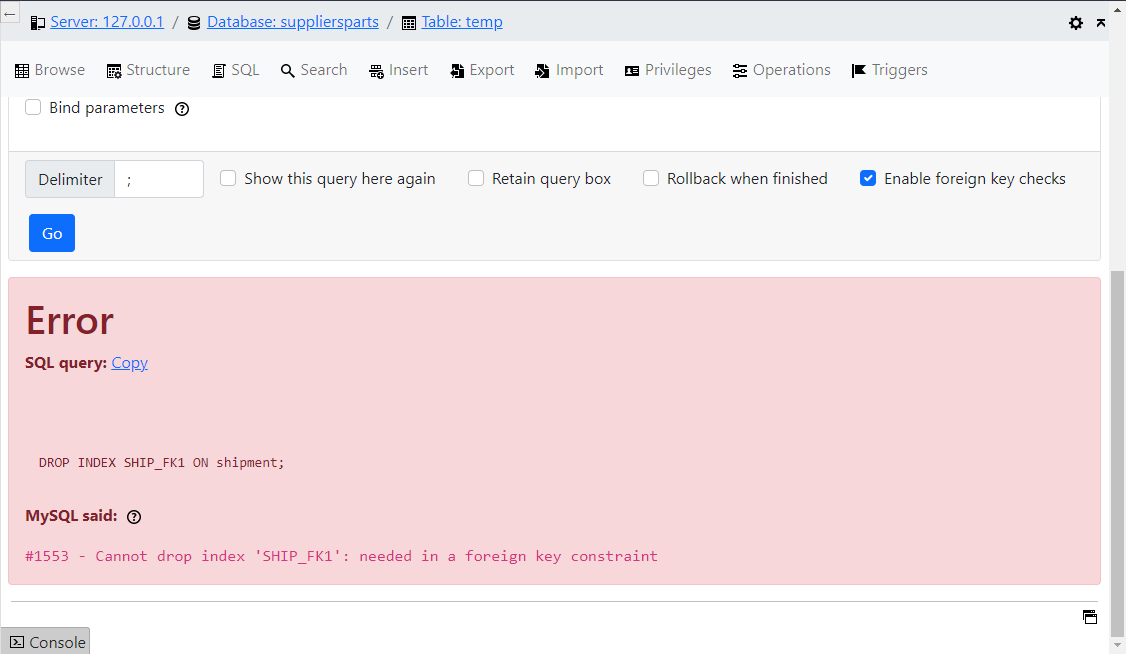
( field\_name [ ASC I DESC ] [, field\_name ] [ASC I DESC ]…])

# Example

To remove an index use the command

DROP INDEX index\_name





You can create a cluster by issuing a command such as:

CREATE CLUSTER sale\_employees

(customer\_no INTEGER)

1. **DATA DICTIONARY**

The SQL system creates and maintains the data dictionary. No user (not even the DBA) can create or delete data dictionary tables, but the DBA can only alter them to a very limited extent. There are thus no CREATE, UPDATE DELETE or INSERT commands for the data dictionary.

Most data dictionary tables are accessible to everyone. Users can extract data from them with the SELECT command. By examining the data dictionary tables users can determine which indexes, tables or views they have created. They can also determine the GRANTS they have issued. The DBA can use the data dictionary most to check the daily operations of the database.

The data dictionary tables can be divided into three groups:

1. DATABASE OPERATION: They contain information about what happens in the database as it operates. They are called audit tables. Primarily intended for the DBA. For example the following command records all unsuccessful attempts to delete, select, alter or update records in the PART table.

AUDIT DELETE, SELECT, ALTER, UPDATE

ON part BY ACCESS WHENEVER NOT SUCCESSFUL

2. DATABASE TABLES: They contain information about the physical and logical organization of the databases. This may include lists of tables and where they are physically stored.

1. **EMBEDDED SQL**

SQL is a non-procedural language. It is not intended for end-users but for application programmers and database administrators. SQL’s weaknesses include:

* Inability to handle user interaction (user interface)
* Limited provisions for maintaining integrity

One way out of these weaknesses is to embed SQL statements in a high-level programming language (host language). SQL statements can only be embedded in a language for which a pre-compiler is available. Such tools exist only for certain combinations of SQL implementations, DBMS, compilers and operating systems. Languages for which pre-compilers are available include Java, PhP, C++, C, Python, Pascal, FORTRAN, COBOL, C, Ada and PL/1. Embedded SQL is also available on some procedural database languages such as Oracle, MySQL, PostGre, MS SQL Server.

An embedded SQL program consists of three parts:

* declaration part
* include part
* program part

For example to delete a record from a CUSTOMER table:

EXEC SQL BEGIN DECLARE SECTION

INTEGER cno

EXEC SQL END DECLARE SECTION

EXEC SQL INCLUDE SQLCA

begin

print ( ‘ Enter number of customer to delete:’ )

read (cno)

EXEC SQL DELETE FROM customers

WHERE customer\_no = : cno

print ( ‘customer’ , cno, ‘ removed from database’)

end

This is done in Lab 5 with Java via the Netbeans IDE

Embedded SQL can use all interactive SQL commands, as well as other commands for example:

(a) DECLARE – used to declare variables to host language (host variables). The host variables are declared in a segment between the statements.

EXEC SQL BEGIN DECLARE SECTION

and

EXEC SQL END DECLARE SECTION

(b) SQLCA – SQL Communications Area is a set of variables used by the DBMS and host language to exchange information about the status of operations, for example:

EXEC SQL INCLUDE SQLCA

### APPENDIX A – SQL COMMANDS

**I. SQL DDL COMMANDS**

ALTER PARTITION Adds a file to database partition

ALTER SPACE Alters a space definition

ALTER TABLE Adds a column to, or redefines a column in, an existing

table

COMMENT Inserts a comment about a table or column into data

dictionary

CREATE CLUSTER Creates a cluster which may contain two or more tables

CREATE DATABASE Creates a link to user name in a remote database

LINK

CREATE INDEX Creates an index for a table

CREATE PARTITION Creates a new partition in the database

CREATE SPACE Creates a space definition which then may be used to

define the space allocation properties of a table

CREATE SYNONYM Creates a synonym for a table or view name

CREATE TABLE Creates a table and defines its columns and other properties

CREATE VIEW Defines a view of one or more tables or other views

DROP Deletes a table, view, index or cluster from the

database

RENAME Changes the name of a table, view or synonym

## II. SQL DML COMMANDS

DELETE Deletes one or more rows from a table or view

INSERT Adds new rows to a table or view

SELECT Performs a query and selects rows and columns from

one or more tables or views

UPDATE Changes the value of fields in a table or view

## III. SQL DCL COMMANDS

AUDIT Makes the DBMS audit use of a table, view, synonym,

or system facility

COMMIT Makes a database transaction irreversible

GRANT Grants access to objects stored in database

LOCK TABLE Locks a table and thus permits shared access to the

table by multiple users while simultaneously preserving

the ‘tables integrity.’

NOAUDIT Partially or completely reverses the effect of prior

AUDIT command.

REVOKE Revokes database privileges or table access privileges

from a user

ROLLBACK Cancels database transactions

VALIDATE INDEX Checks the integrity of a table index.