Simple DBMS

Description, features and design decisions

Description

We designed and implemented a Database Management System that uses SQL queries to give instructions to the program to; create or drop a database or table, add rows to the table, delete rows or certain columns from the tables and update values in the table based on conditional statements.

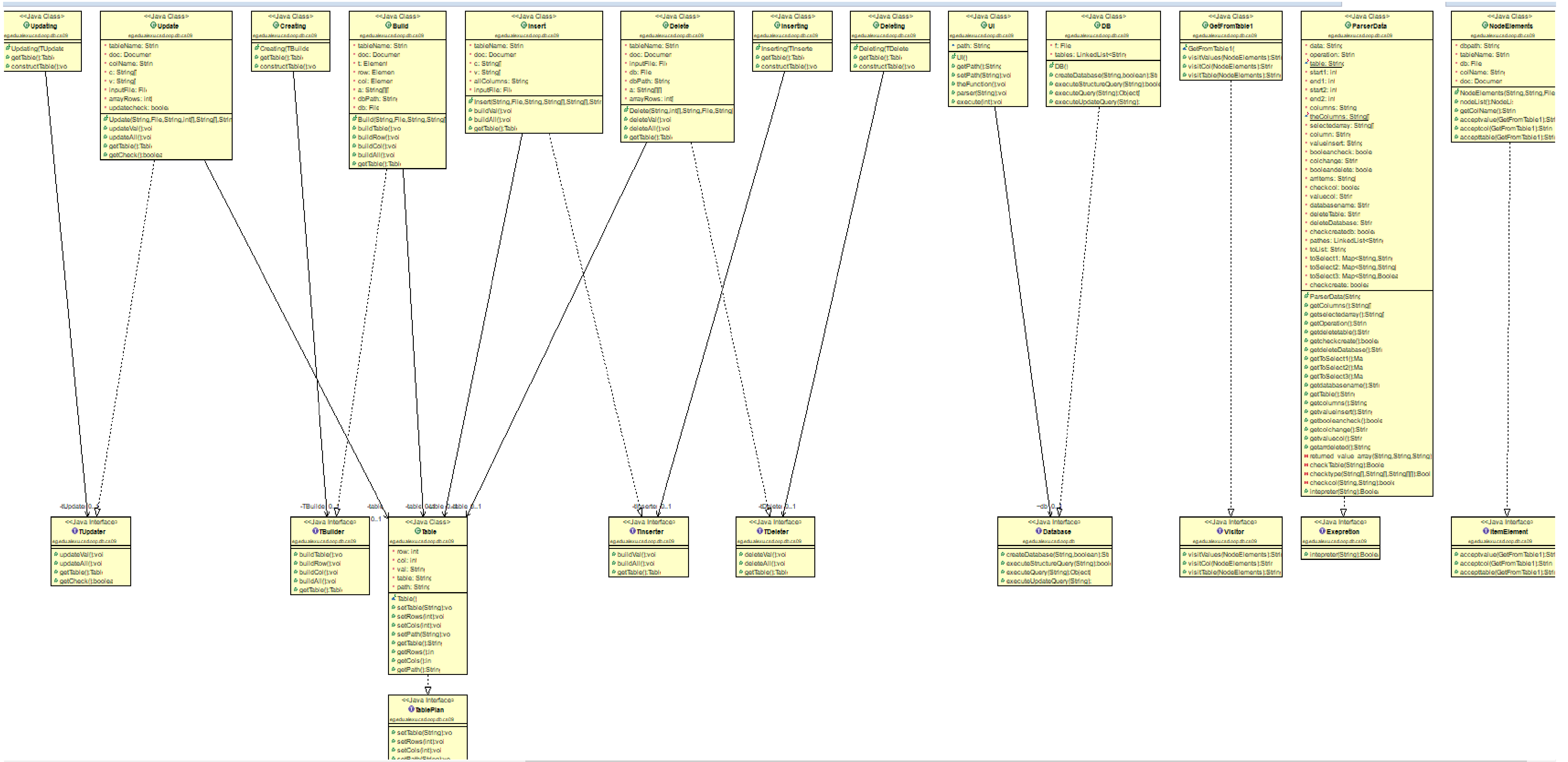
Features and design decisions

- We use the console or CMD in the User Interface.
- We use SQL queries to give instructions to the program.
- You must enter a correct query otherwise the program will not carry out the instruction that you need.
- You can create a new database either by the name of the database or its path.
- We assume that the paths of all databases start at a directory named “Tables”.
- When a table is created, new XML and XSD files with the table’s name are added to the directory of the database.
- When a new database is created a directory with its name is created.
- Any editing in a table is automatically added to its corresponding XML file.

Features and design decisions

- When a table or a database is dropped, its corresponding file or directory is deleted.
- If you create a database with an already existing name, it will delete the old database and create a new one.
- You cannot create a table with a previously used name in the same database.
- We used Builder design pattern to create and edit tables.
- We used Singleton design pattern to check if a database exists or not.
- We used Interpreter design patter to parse the SQL query entered by the user.
- We used Visitor design pattern to get objects from the table.
- We used Façade design pattern to build the UI of the program.

UML diagram



<<Java Class>>
Updating
eg.edu.alexu.csd.oop.db.cs09
Updating(TUpdate
getTable():Tabl
constructTable():vo

<<Java Class>>
Update
eg.edu.alexu.csd.oop.db.cs09
tableName: Strin
doc: Documer
colName: Strin
c: String[
v: String[
inputFile: Fil
arrayRows: int[
updatecheck: boole
Update(String,File,String,int[],String[],Strin
updateVal():voi
updateAll():voi
getTable():Tabl
getCheck():boolea

<<Java Class>>
Creating
eg.edu.alexu.csd.oop.db.cs09
Creating(TBuilde
getTable():Tabl
constructTable():vo

<<Java Class>>
Build
eg.edu.alexu.csd.oop.db.cs09
tableName: Strin
doc: Documer
t: Element
row: Elemen
col: Elemer
a: String[][]
dbPath: Strin
db: File
Build(String,File,String,String[
buildTable():vo
buildRow():voi
buildCol():voi
buildAll():voi
getTable():Tabl

<<Java Class>>
Insert
eg.edu.alexu.csd.oop.db.cs09
tableName: Strin
doc: Documer
c: String[
v: String[
allColumns: String
inputFile: Fil
Insert(String,File,String,String[],String[],Strir
buildVal():voi
buildAll():voi
getTable():Tabl

<<Java Class>>
Delete
eg.edu.alexu.csd.oop.db.cs09
tableName: Strin
doc: Documer
inputFile: Fil
db: File
dbPath: String
a: String[][]
arrayRows: int[]
Delete(String,int[],String,File,String[
deleteVal():voi
deleteAll():voi
getTable():Tabl

<<Java Class>>
Inserting
eg.edu.alexu.csd.oop.db.cs09
Inserting(TInserte
getTable():Tabl
constructTable():vo

<<Java Class>>
Deleting
eg.edu.alexu.csd.oop.db.cs09
Deleting(TDelete
getTable():Tabl
constructTable():vo

<<Java Class>>
UI
eg.edu.alexu.csd.oop.db.cs09
path: String
UI()
getPath():String
setPath(String):voi
theFunction():voi
parser(String):voi
execute(int):voi

<<Java Class>>
DB
eg.edu.alexu.csd.oop.db.cs09
f: File
tables: LinkedList<Strin
DB()
createDatabase(String,boolean):St
executeStructureQuery(String):boole
executeQuery(String):Object[
executeUpdateQuery(String):

<<Java Class>>
GetFromTable1
eg.edu.alexu.csd.oop.db.cs09
GetFromTable1(
visitValues(NodeElements):Stri
visitCol(NodeElements):Strir
visitTable(NodeElements):String

<<Java Class>>
ParserData
eg.edu.alexu.csd.oop.db.cs09
data: String
operation: Strin
table: String
start1: int
end1: int
start2: int
end2: int
columns: String
theColumns: String[
selectedarray: String[
column: Strin
valueinsert: String
booleancheck: boole
colchange: Strir
booleandelete: boole
arritems: String[
checkcol: boolea
valuecol: Strir
databasename: Strir
deleteTable: Strin
deleteDatabase: Strir
checkcreatedb: boole
pathes: LinkedList<Strin
toList: String
toSelect1: Map<String,Strin
toSelect2: Map<String,String[
toSelect3: Map<String,Boolea
checkcreate: boolea
ParserData(String
getColumns():String[
getselectedarray():String[
getOperation():Strin
getdeletetable():Strir
getcheckcreate():boole
getdeleteDatabase():Stri
getToSelect1():Ma
getToSelect2():Ma
getToSelect3():Ma
getdatabasename():Stri
getTable():Strin
getcolumns():String
getvalueinsert():Strin
getbooleancheck():boole
getcolchange():Strir
getvaluecol():Strir
getarrdeleted():String
returned_value_array(String,String,String[
checkTable(String):Boole
checktype(String[],String[],String[][]):Bool
checkcol(String,String):boole
intepreter(String):Boole

<<Java Class>>
NodeElements
eg.edu.alexu.csd.oop.db.cs09
dbpath: String
tableName: Strin
db: File
colName: Strin
doc: Documer
NodeElements(String,String,File
nodeList():NodeLi
getColName():Strin
acceptvalue(GetFromTable1):Str
acceptcol(GetFromTable1):Strin
accepttable(GetFromTable1):Stri

-tUpdater 0..1

<<Java Interface>>
TUpdater
eg.edu.alexu.csd.oop.db.cs09
updateVal():voi
updateAll():voi
getTable():Tabl
getCheck():boolea

-TBuilde 0..1

<<Java Interface>>
TBuilder
eg.edu.alexu.csd.oop.db.cs09
buildTable():vo
buildRow():voi
buildCol():voi
buildAll():voi
getTable():Tabl

-table 0..1 -table 0..1 -table 0..1 -table 0..1

<<Java Class>>
Table
eg.edu.alexu.csd.oop.db.cs09
row: int
col: int
val: Strin
table: String
path: String
Table()
setTable(String):vo
setRows(int):voi
setCols(int):voi
setPath(String):vo
getTable():Strin
getRows():in
getCols():in
getPath():Strin

<<Java Interface>>
TablePlan
eg.edu.alexu.csd.oop.db.cs09
setTable(String):vo
setRows(int):voi
setCols(int):voi
setPath(String):vo

-tInserter 0..1

<<Java Interface>>
TInserter
eg.edu.alexu.csd.oop.db.cs09
buildVal():voi
buildAll():voi
getTable():Tabl

-tDeleter 0..1

<<Java Interface>>
TDeleter
eg.edu.alexu.csd.oop.db.cs09
deleteVal():voi
deleteAll():voi
getTable():Tabl

~db 0..1

<<Java Interface>>
Database
eg.edu.alexu.csd.oop.db
createDatabase(String,boolean):St
executeStructureQuery(String):boole
executeQuery(String):Object[
executeUpdateQuery(String):

<<Java Interface>>
Visitor
eg.edu.alexu.csd.oop.db.cs09
visitValues(NodeElements):Stri
visitCol(NodeElements):Strir
visitTable(NodeElements):String

<<Java Interface>>
Exepretion
eg.edu.alexu.csd.oop.db.cs09
intepreter(String):Boole

<<Java Interface>>
ItemElement
eg.edu.alexu.csd.oop.db.cs09
acceptvalue(GetFromTable1):Str
acceptcol(GetFromTable1):Strin
accepttable(GetFromTable1):Stri

User manual

1) Any instructions are given to the program in the form of SQL queries.

2) You must enter a query in a correct syntax in order for the program to perform the instruction you require.

2) You cannot create a database with an already existing name.

3) You cannot create a table with a previously used name in the same database.

4) When you create or drop a database the program will tell you if it is a success or not.

5) When you create a table, you must set the type of values that each column will take (either integer or varchar).

6) When you insert a new row into a table, the values you enter must be of the same type as that of the column in which you will insert them.

User manual

7) If the user doesn’t determine the order of the columns he wishes to insert the values in, the program inserts them by default in the order of the columns in the table.

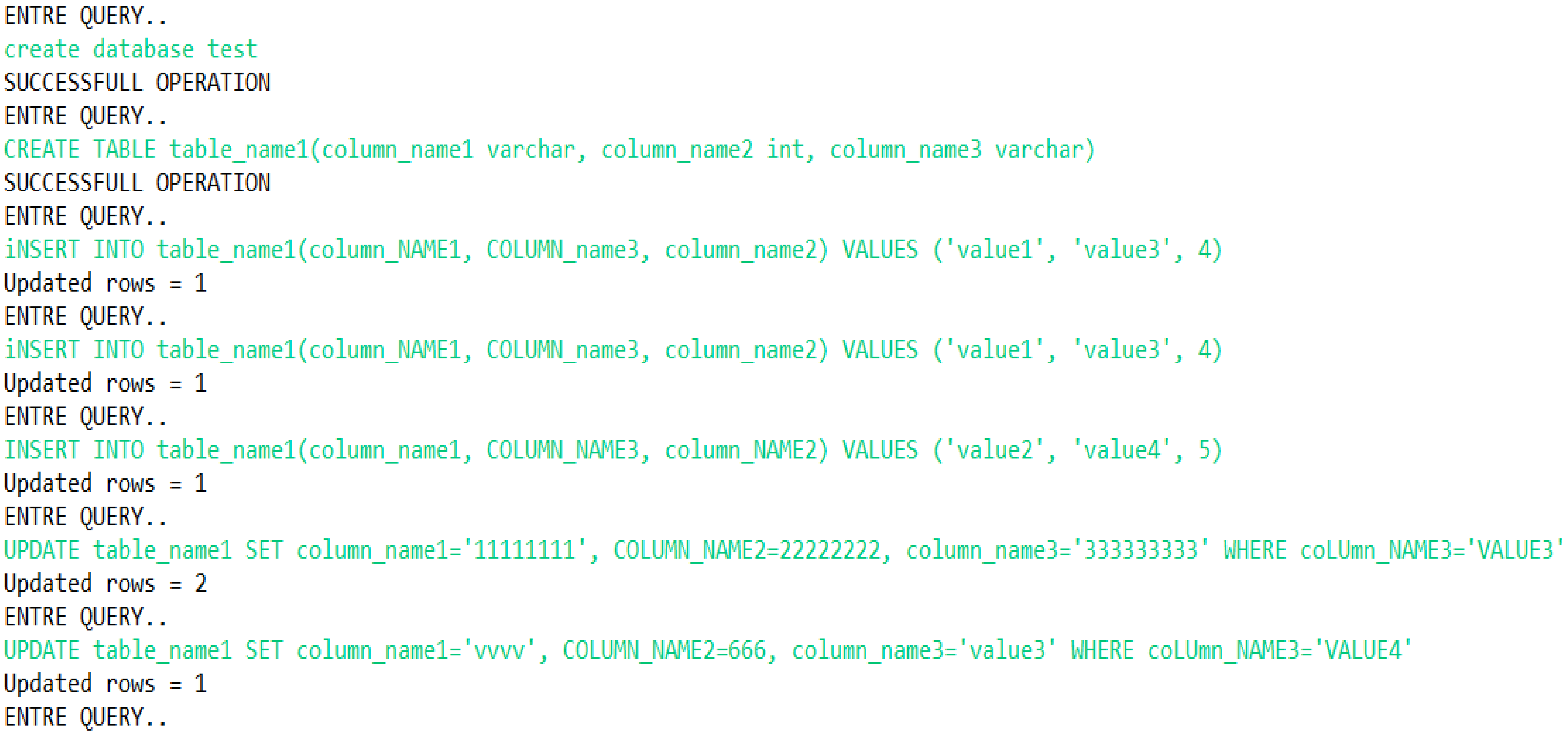
8) When you update, insert into or delete from a table, the program will return to you the number of rows on which the instruction was carried out.

9) You cannot update a table without inserting elements into it first.

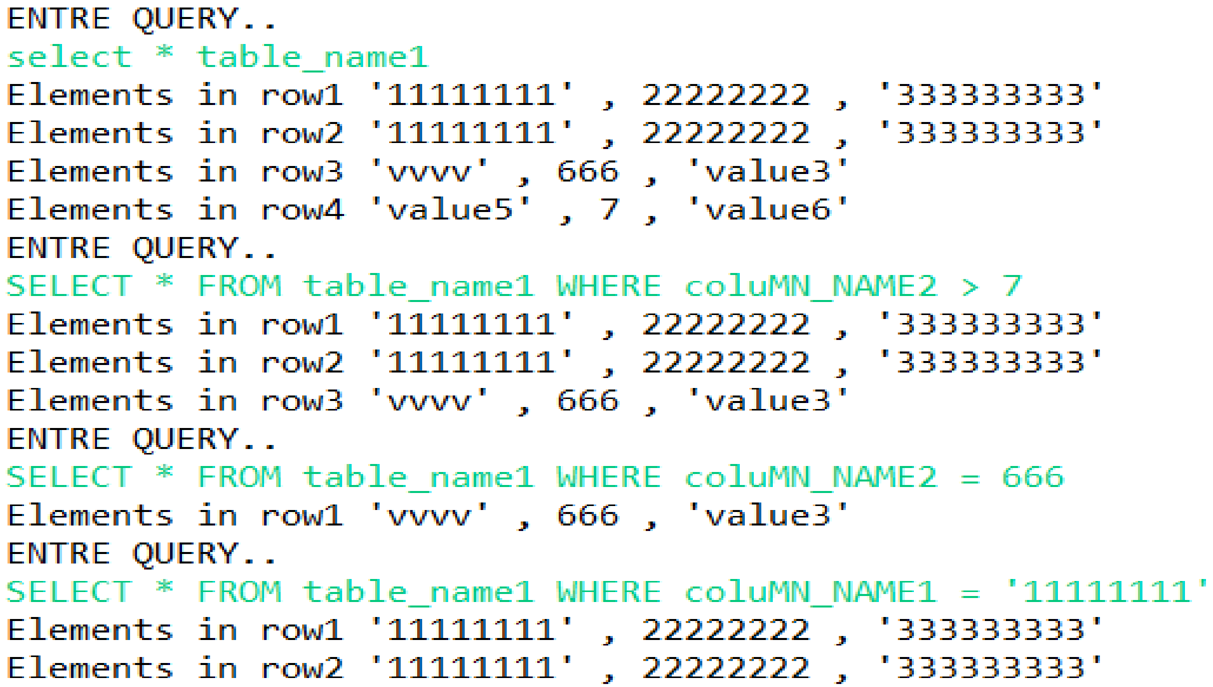
10) The program supports simple conditions as =, >, and < in all conditional queries.

11) The program supports multiple conditions as AND, OR, or NOT in all conditional queries except for the SELECT query (it supports SELECT *)

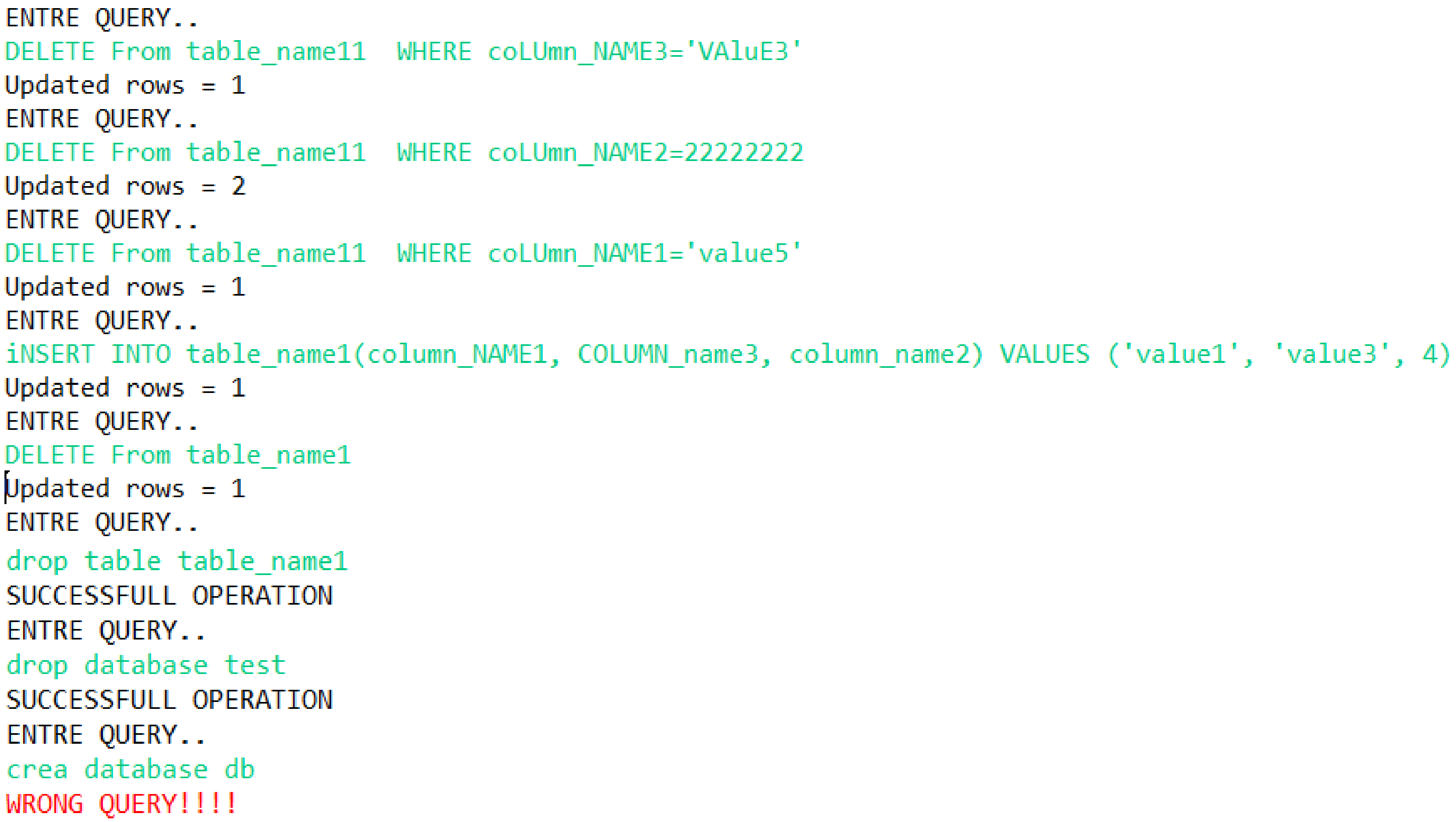
Sample runs



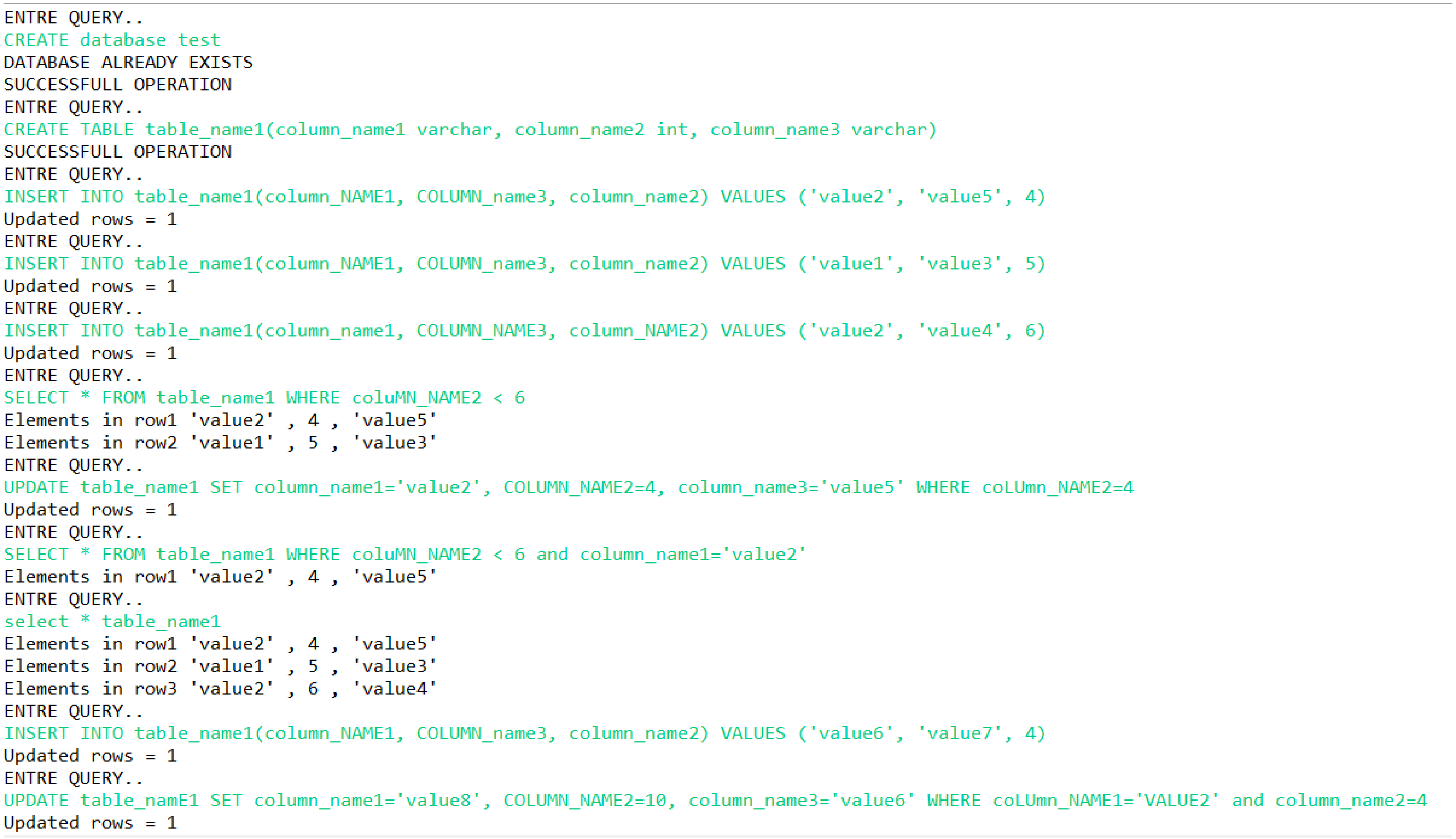
```
ENTRE QUERY..
create database test
SUCCESSFULL OPERATION
ENTRE QUERY..
CREATE TABLE table_name1(column_name1 varchar, column_name2 int, column_name3 varchar)
SUCCESSFULL OPERATION
ENTRE QUERY..
iNSERT INTO table_name1(column_NAME1, COLUMN_name3, column_name2) VALUES ('value1', 'value3', 4)
Updated rows = 1
ENTRE QUERY..
iNSERT INTO table_name1(column_NAME1, COLUMN_name3, column_name2) VALUES ('value1', 'value3', 4)
Updated rows = 1
ENTRE QUERY..
INSERT INTO table_name1(column_name1, COLUMN_NAME3, column_NAME2) VALUES ('value2', 'value4', 5)
Updated rows = 1
ENTRE QUERY..
UPDATE table_name1 SET column_name1='11111111', COLUMN_NAME2=22222222, column_name3='333333333' WHERE coLUmn_NAME3='VALUE3'
Updated rows = 2
ENTRE QUERY..
UPDATE table_name1 SET column_name1='vvvv', COLUMN_NAME2=666, column_name3='value3' WHERE coLUmn_NAME3='VALUE4'
Updated rows = 1
ENTRE QUERY..
```



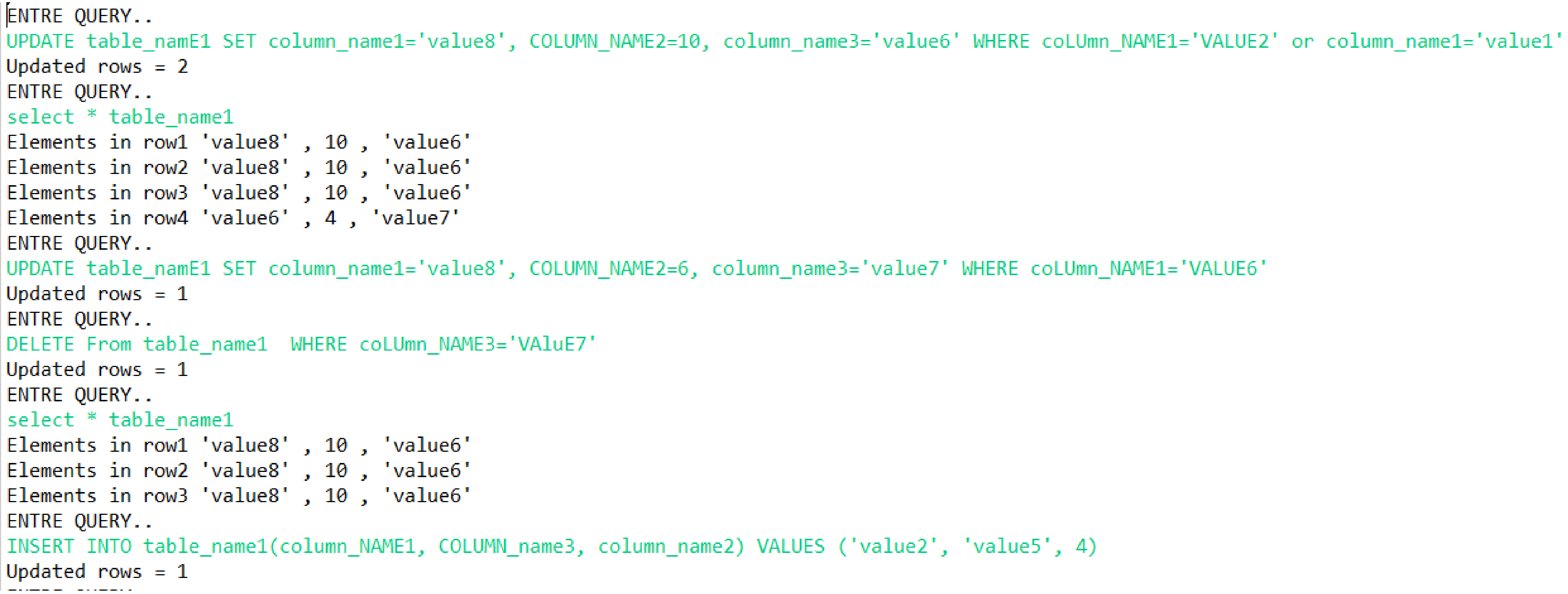
```
ENTRE QUERY..
select * table_name1
Elements in row1 '11111111' , 22222222 , '333333333'
Elements in row2 '11111111' , 22222222 , '333333333'
Elements in row3 'vvvv' , 666 , 'value3'
Elements in row4 'value5' , 7 , 'value6'
ENTRE QUERY..
SELECT * FROM table_name1 WHERE coluMN_NAME2 > 7
Elements in row1 '11111111' , 22222222 , '333333333'
Elements in row2 '11111111' , 22222222 , '333333333'
Elements in row3 'vvvv' , 666 , 'value3'
ENTRE QUERY..
SELECT * FROM table_name1 WHERE coluMN_NAME2 = 666
Elements in row1 'vvvv' , 666 , 'value3'
ENTRE QUERY..
SELECT * FROM table_name1 WHERE coluMN_NAME1 = '11111111'
Elements in row1 '11111111' , 22222222 , '333333333'
Elements in row2 '11111111' , 22222222 , '333333333'
```



```
ENTRE QUERY..
DELETE From table_name11  WHERE coLUmn_NAME3='VAluE3'
Updated rows = 1
ENTRE QUERY..
DELETE From table_name11  WHERE coLUmn_NAME2=22222222
Updated rows = 2
ENTRE QUERY..
DELETE From table_name11  WHERE coLUmn_NAME1='value5'
Updated rows = 1
ENTRE QUERY..
iNSERT INTO table_name1(column_NAME1, COLUMN_name3, column_name2) VALUES ('value1', 'value3', 4)
Updated rows = 1
ENTRE QUERY..
DELETE From table_name1
Updated rows = 1
ENTRE QUERY..
drop table table_name1
SUCCESSFULL OPERATION
ENTRE QUERY..
drop database test
SUCCESSFULL OPERATION
ENTRE QUERY..
crea database db
WRONG QUERY!!!!
```



```
ENTRE QUERY..
CREATE database test
DATABASE ALREADY EXISTS
SUCCESSFULL OPERATION
ENTRE QUERY..
CREATE TABLE table_name1(column_name1 varchar, column_name2 int, column_name3 varchar)
SUCCESSFULL OPERATION
ENTRE QUERY..
INSERT INTO table_name1(column_NAME1, COLUMN_name3, column_name2) VALUES ('value2', 'value5', 4)
Updated rows = 1
ENTRE QUERY..
INSERT INTO table_name1(column_NAME1, COLUMN_name3, column_name2) VALUES ('value1', 'value3', 5)
Updated rows = 1
ENTRE QUERY..
INSERT INTO table_name1(column_name1, COLUMN_NAME3, column_NAME2) VALUES ('value2', 'value4', 6)
Updated rows = 1
ENTRE QUERY..
SELECT * FROM table_name1 WHERE coluMN_NAME2 < 6
Elements in row1 'value2' , 4 , 'value5'
Elements in row2 'value1' , 5 , 'value3'
ENTRE QUERY..
UPDATE table_name1 SET column_name1='value2', COLUMN_NAME2=4, column_name3='value5' WHERE coLUmn_NAME2=4
Updated rows = 1
ENTRE QUERY..
SELECT * FROM table_name1 WHERE coluMN_NAME2 < 6 and column_name1='value2'
Elements in row1 'value2' , 4 , 'value5'
ENTRE QUERY..
select * table_name1
Elements in row1 'value2' , 4 , 'value5'
Elements in row2 'value1' , 5 , 'value3'
Elements in row3 'value2' , 6 , 'value4'
ENTRE QUERY..
INSERT INTO table_name1(column_NAME1, COLUMN_name3, column_name2) VALUES ('value6', 'value7', 4)
Updated rows = 1
ENTRE QUERY..
UPDATE table_namE1 SET column_name1='value8', COLUMN_NAME2=10, column_name3='value6' WHERE coLUmn_NAME1='VALUE2' and column_name2=4
Updated rows = 1
```



```
ENTRE QUERY..
UPDATE table_namE1 SET column_name1='value8', COLUMN_NAME2=10, column_name3='value6' WHERE coLUmn_NAME1='VALUE2' or column_name1='value1'
Updated rows = 2
ENTRE QUERY..
select * table_name1
Elements in row1 'value8' , 10 , 'value6'
Elements in row2 'value8' , 10 , 'value6'
Elements in row3 'value8' , 10 , 'value6'
Elements in row4 'value6' , 4 , 'value7'
ENTRE QUERY..
UPDATE table_namE1 SET column_name1='value8', COLUMN_NAME2=6, column_name3='value7' WHERE coLUmn_NAME1='VALUE6'
Updated rows = 1
ENTRE QUERY..
DELETE From table_name1  WHERE coLUmn_NAME3='VAluE7'
Updated rows = 1
ENTRE QUERY..
select * table_name1
Elements in row1 'value8' , 10 , 'value6'
Elements in row2 'value8' , 10 , 'value6'
Elements in row3 'value8' , 10 , 'value6'
ENTRE QUERY..
INSERT INTO table_name1(column_NAME1, COLUMN_name3, column_name2) VALUES ('value2', 'value5', 4)
Updated rows = 1
ENTRE QUERY..
```