# Lab report

**Name : ABID ALI**
**Student_no : 2019380141**

# Experiment 2

## Experiment No:2

Manipulate the Data in Table

## Goal

1. Master all kinds of data operation about basic table in GUI.
2. Familiar with SQL statements for data insertion, modification and deletion of basic tables.
3. Master the SQL statement of data query.
4. Master the basic knowledge of SQL query performance analysis.
5. Understand TPC-H benchmark database.

## Content

1. Use SQL statement to insert all the tuples into the database SPJ_MNG and university which have been list in the previous experiment.
2. Modificaion the data of tables with SQL statement.
   (1) Modify one tuple in the table of student
   (2) Delete one tuple from table of student.
3. In the database of SPJ_MNG, use SQL statement to do the following update operations:

(1) Change the color of all red parts to blue.
(2) Part P6 supplied by S5 for J4 is replaced by S3, please make necessary modification.
(3) Delete S2 record from supplier table and delete corresponding record from supply table.
(4) Please insert (S2, J6, p4200) into the supply table SPJ.

**Finish the following queries about the database university with SQL statement. (4-6)**

4. Use three different ways (SQL statement) to find the student ID and name of all students who take "Database System Concept", and then analyze and compare the performance of each query process.

5. For university database, complete the following data query with SQL statement
   (1) Query the total score of credits obtained by each student , and output the student ID, name and credit obtained in the order from high to low.
   (2) Query the name of the student: the student has taken all courses and one of the courses has a grade of better than B .

6. Use at least three different SQL statements to query the university database: query the student ID and name of the course named "database", and then design the experiment by ourselves, compare and analyze the efficiency of the three kinds of query with data, and analyze the reasons.

Hint: in order to compare it more clearly, you'd better create a bigger table.

7. (optional) TPC-H is one of the database benchmarks released by TPC international organization. The database simulates the data of a typical enterprise: parts, customers, parts, suppliers, products, orders and so on- ddl.sql Documents.Based on database TPC-H (database definition statements can be defined referring to the file "tcp-h-ddl.sql"), design the following queries and test the queries with some data:
   (1) Single table query (to realize operations of projection and selection)
   (2) Grouping stastistic query(using "group by", without using "group by")

(3) Single table query with self-join operation
(4) Multple table query with join operation
(5) Nested query with IN clause
(6) Nested query with EXISTS clause
(7) Neseted query with FROM clause
(8) Set query( intersect, union and except)

**The TPC-H database is designed as following:（referring to the file tcp-ds-v2.17.3.docx）:**

# Answer No.1

1. Use SQL statement to insert all the tuples into the database SPJ_MNG and university which have been list in the previous experiment.

SPJ_MNG

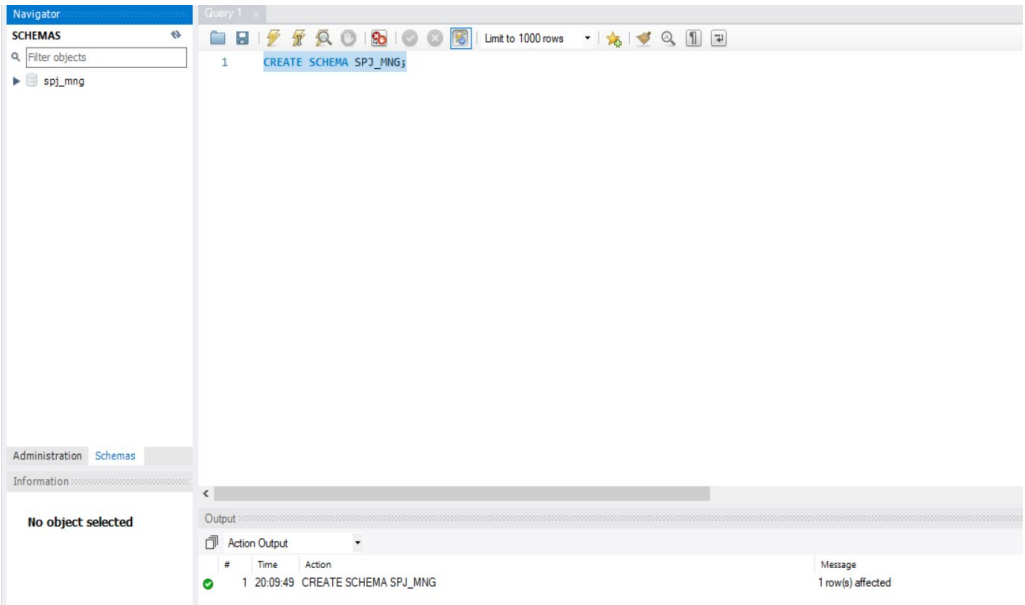



Fig: Creating database SPJ_MNG(By coding)

Code:

```
CREATE SCHEMA SPJ_MNG;
```

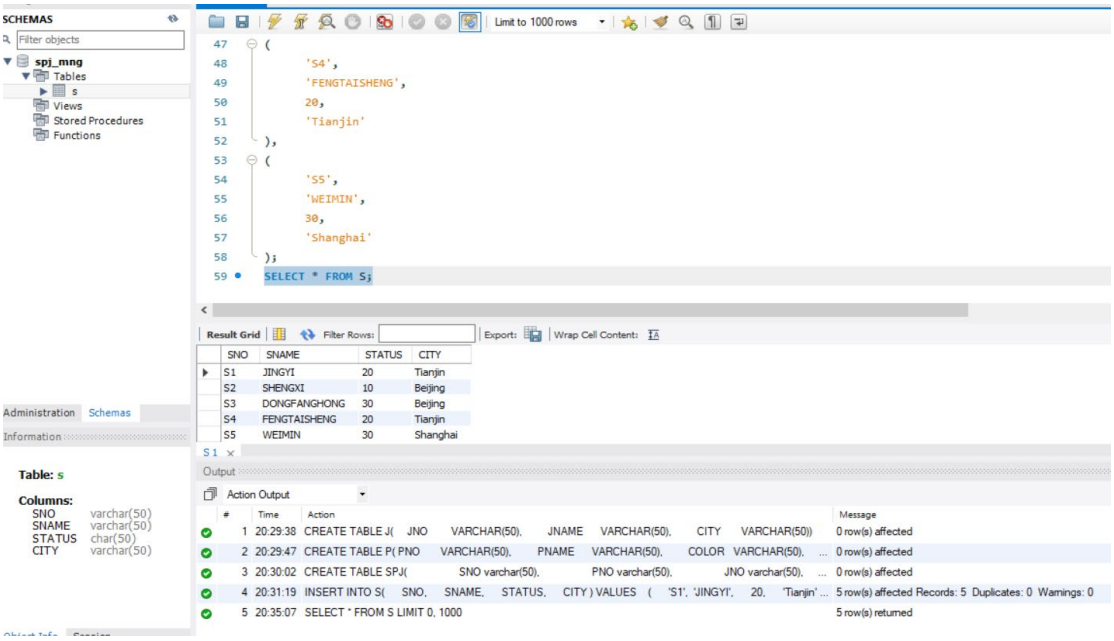



Fig: Creating table S

Code:

```
-- Creating table S
CREATE TABLE S(
        SNO      VARCHAR(50),
      SNAME     VARCHAR(50),
      STATUS    CHAR(50),
      CITY      VARCHAR(50));
```

Code:

```
-- Creating table S
CREATE TABLE S(
        SNO      VARCHAR(50),
      SNAME     VARCHAR(50),
      STATUS    CHAR(50),
      CITY      VARCHAR(50));

-- Inserting value in table S

INSERT INTO S(
    SNO,
```

```
    SNAME,
    STATUS,
    CITY
)
VALUES
  (
    'S1',
      'JINGYI',
    20,
    'Tianjin'
),
(
     'S2',
     'SHENGXI',
     10,
     'Beijing'
),
(
     'S3',
     'DONGFANGHONG',
     30,
     'Beijing'
),
(
     'S4',
     'FENGTAISHENG',
     20,
     'Tianjin'
),
(
     'S5',
     'WEIMIN',
     30,
     'Shanghai'
);
```

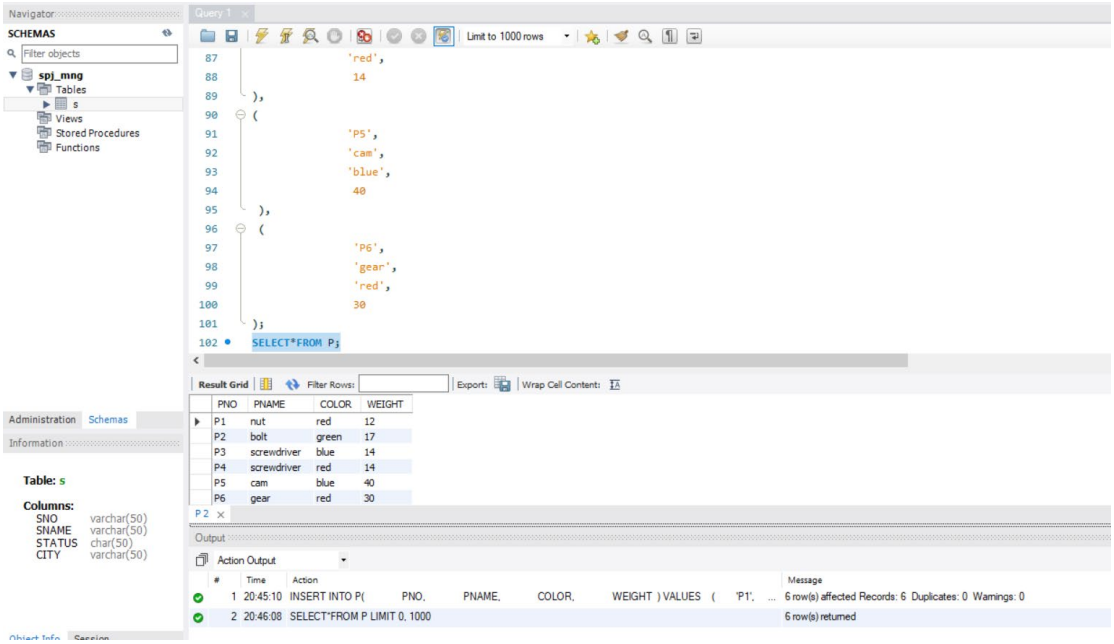



Fig: Creating table P

## Code:

```
-- Creating table P
CREATE TABLE P(
        PNO          VARCHAR(50),
      PNAME      VARCHAR(50),
      COLOR     VARCHAR(50),
      WEIGHT     CHAR(50));

-- Inserting value in table P
INSERT INTO P(
          PNO,
          PNAME,
          COLOR,
          WEIGHT
 )
VALUES
  (
          'P1',
          'nut',
          'red',
           12
```

```
),
(
        'P2',
        'bolt',
        'green',
         17
),
(
        'P3',
        'screwdriver',
        'blue',
         14
),
(
        'P4',
        'screwdriver',
        'red',
         14
),
(
        'P5',
        'cam',
        'blue',
         40
 ),
 (
         'P6',
         'gear',
         'red',
         30
);
```

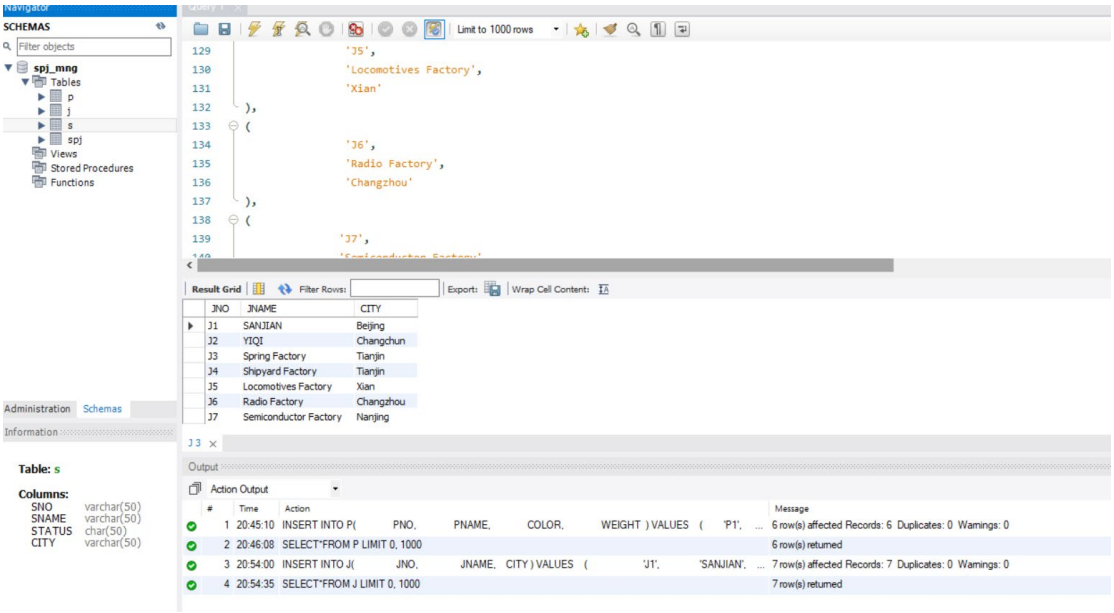



Fig: Creating table J

Code:

```
-- Creating table J

CREATE TABLE J(
        JNO       VARCHAR(50),
      JNAME      VARCHAR(50),
      CITY       VARCHAR(50));
-- Inserting value in table P
INSERT INTO J(
            JNO,
            JNAME,
            CITY
)
VALUES
  (
            'J1',
            'SANJIAN',
            'Beijing'
),
(
            'J2',
            'YIQI',
            'Changchun'
```

```
),
(
        'J3',
    'Spring Factory',
    'Tianjin'
),
(
        'J4',
    'Shipyard Factory',
    'Tianjin'
),
(
    'J5',
    'Locomotives Factory',
    'Xian'
),
(
    'J6',
    'Radio Factory',
    'Changzhou'
),
(
   'J7',
   'Semiconductor Factory',
   'Nanjing'
);
```

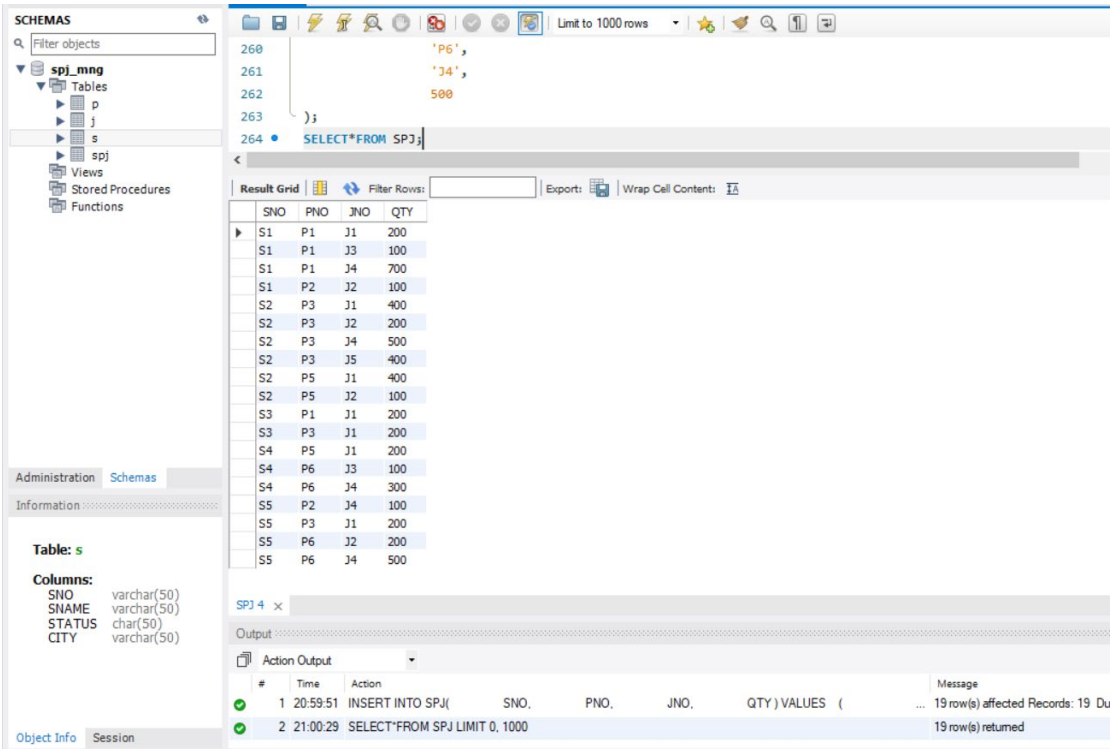



Fig: Creating table SPJ

Code:

```
-- Creating table SPJ

CREATE TABLE SPJ(
        SNO varchar(50),
        PNO varchar(50),
        JNO varchar(50),
        QTY varchar(50));
-- Inserting value in table SPJ

INSERT INTO SPJ(
        SNO,
        PNO,
        JNO,
        QTY
)
VALUES
  (
        'S1',
        'P1',
```

```
        'J1',
        200
),
(
        'S1',
        'P1',
        'J3',
        100
),
(
        'S1',
        'P1',
        'J4',
        700
),
(
        'S1',
        'P2',
        'J2',
        100
),
(
        'S2',
        'P3',
        'J1',
        400
),
(
        'S2',
        'P3',
        'J2',
        200
),
(
        'S2',
        'P3',
        'J4',
        500
),
```

```
(
        'S2',
        'P3',
        'J5',
        400
),
(
        'S2',
        'P5',
        'J1',
        400
),
(
        'S2',
        'P5',
        'J2',
        100
),
(
        'S3',
        'P1',
        'J1',
        200
),
(
        'S3',
        'P3',
        'J1',
        200
),
(
        'S4',
        'P5',
        'J1',
        200
),
(
         'S4',
         'P6',
```

```
        'J3',
        100
),
(
        'S4',
        'P6',
        'J4',
        300
),
(
        'S5',
        'P2',
        'J4',
        100
),
(
        'S5',
        'P3',
        'J1',
        200
),
(
        'S5',
        'P6',
        'J2',
        200
),
(
        'S5',
        'P6',
        'J4',
        500
);
```

# University

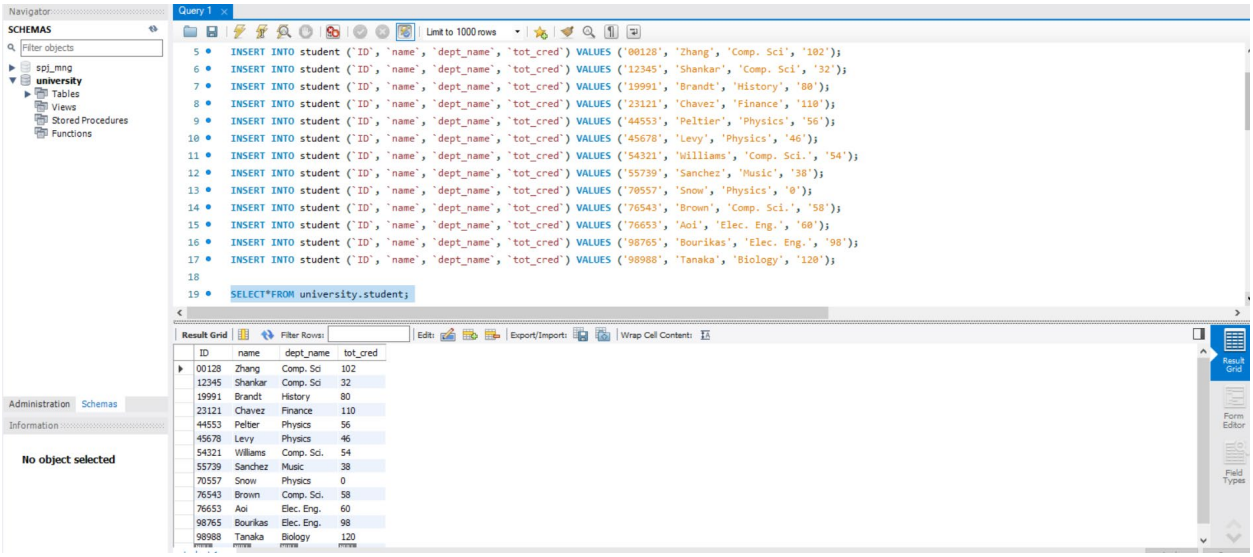



Fig:Table student

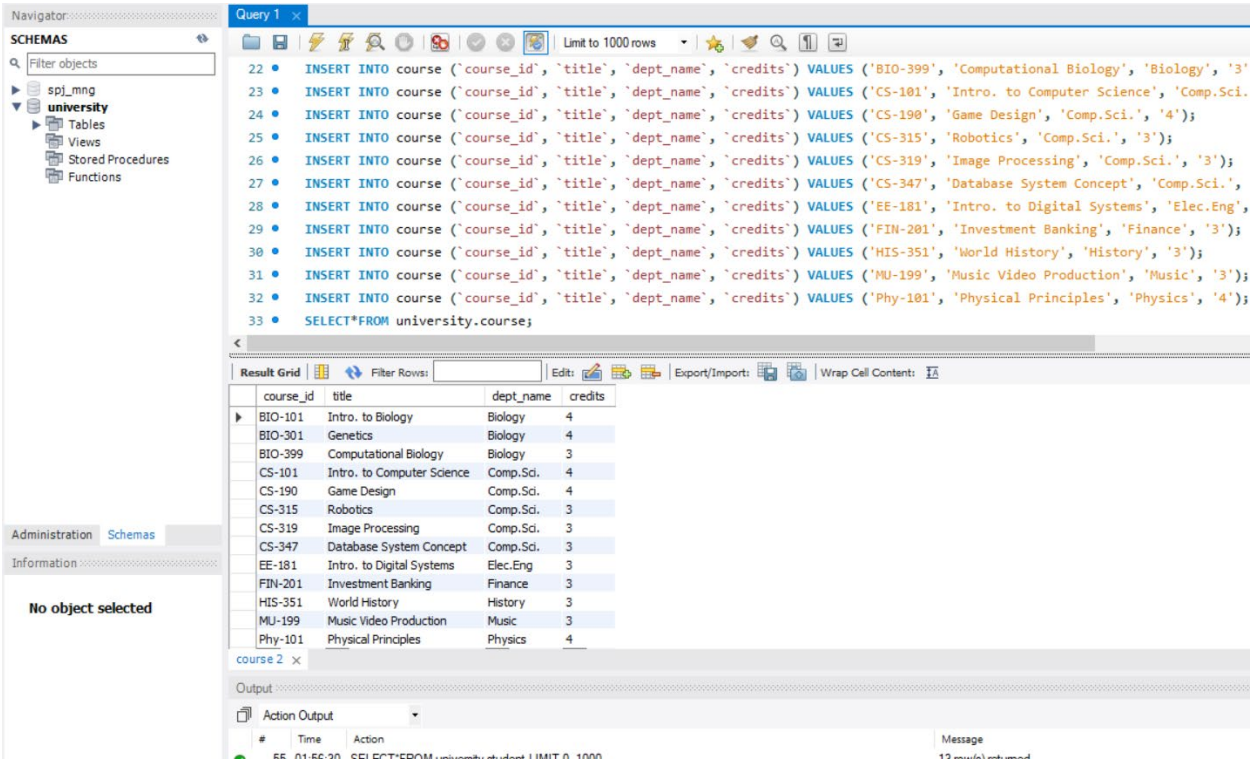



Fig:Table course

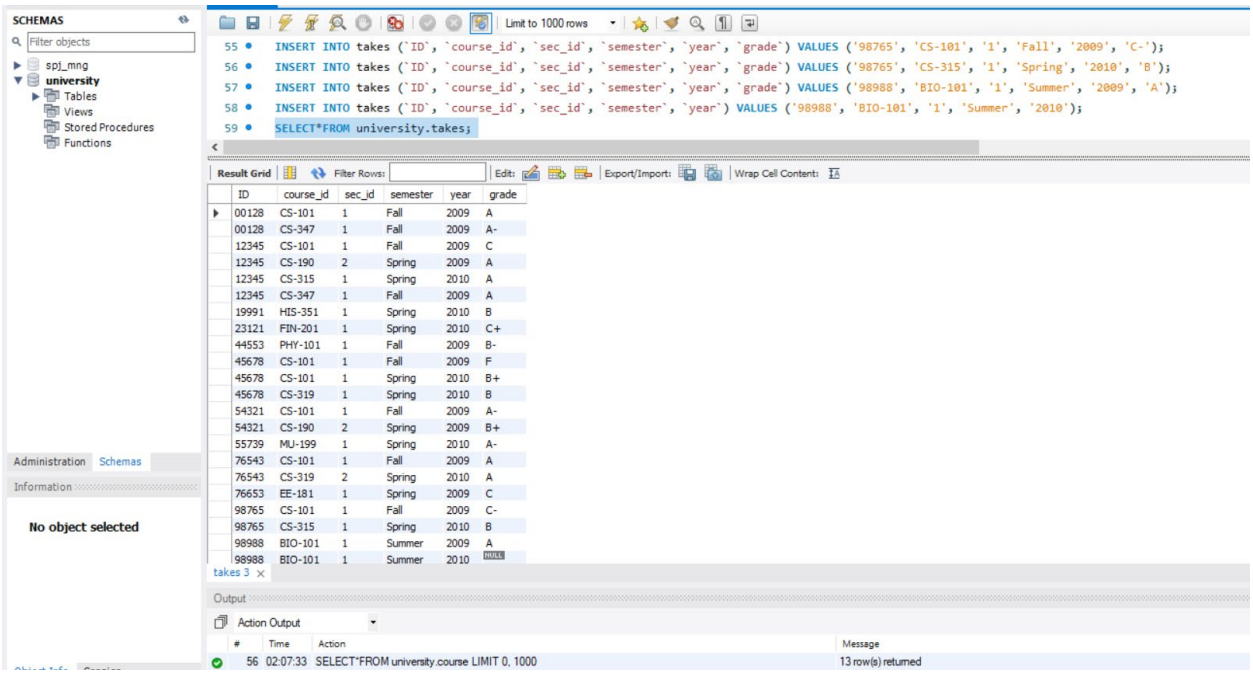
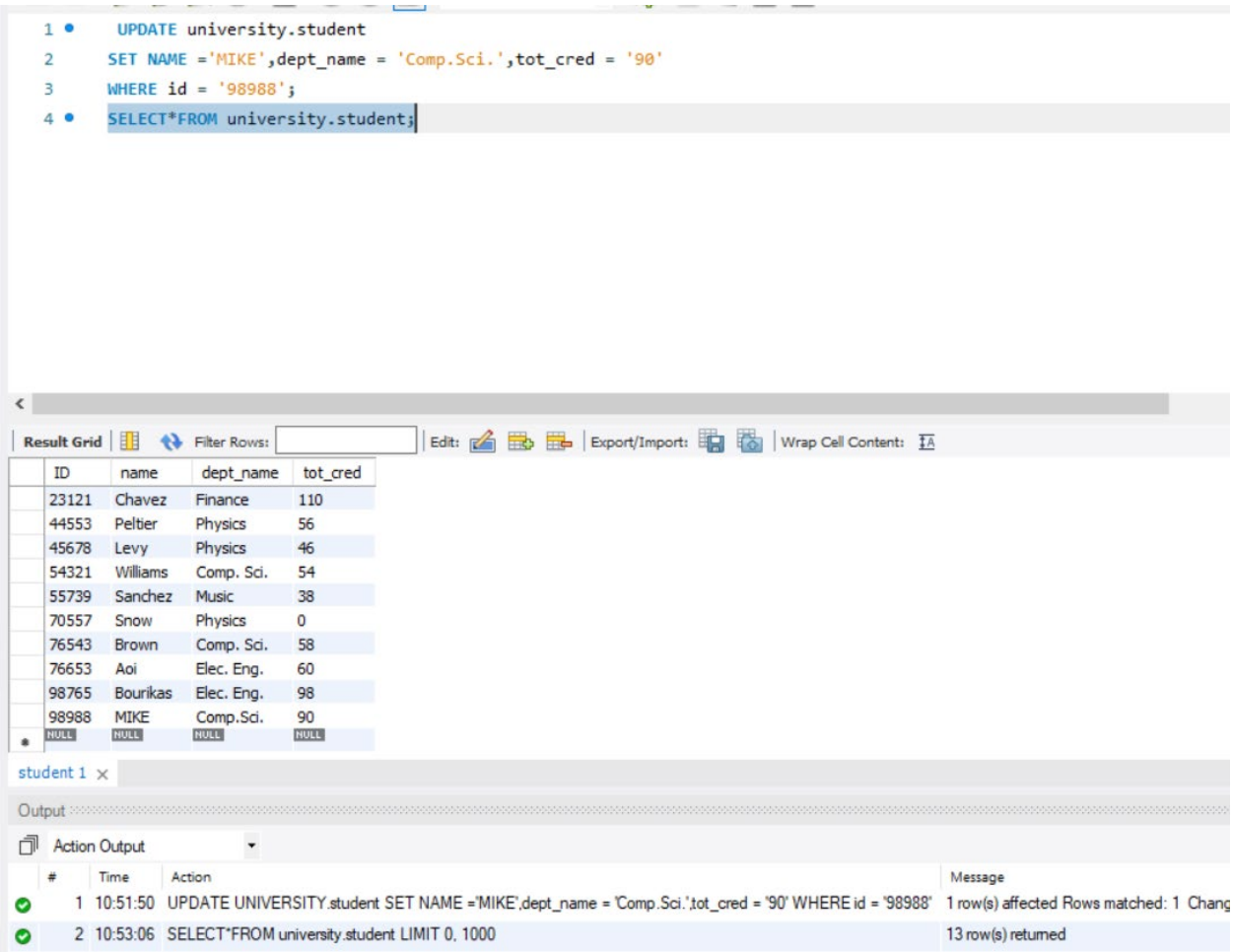



Fig:Table takes

2. Modificaion the data of tables with SQL statement.
   1.Modify one tuple in the table of student
   2.Delete one tuple from table of student.

# Answer No. 2(1)




Fig; Modifying one tuple in the table

Code:

```
UPDATE university.student
SET NAME ='MIKE',dept_name = 'Comp.Sci.',tot_cred = '90'
WHERE id = '98988';
SELECT*FROM university.student;
```

# Answer No. 2(2)

```
1 •  DELETE FROM university.student WHERE id ='98988';
2 •  SELECT * FROM university.student;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

| ID | name | dept_name | tot_cred |
|---|---|---|---|
| 00128 | Zhang | Comp. Sci | 102 |
| 12345 | Shankar | Comp. Sci | 32 |
| 19991 | Brandt | History | 80 |
| 23121 | Chavez | Finance | 110 |
| 44553 | Peltier | Physics | 56 |
| 45678 | Levy | Physics | 46 |
| 54321 | Williams | Comp. Sci. | 54 |
| 55739 | Sanchez | Music | 38 |
| 70557 | Snow | Physics | 0 |
| 76543 | Brown | Comp. Sci. | 58 |
| 76653 | Aoi | Elec. Eng. | 60 |
| 98765 | Bourikas | Elec. Eng. | 98 |
| NULL | NULL | NULL | NULL |

student 2

Output

Action Output

| # | Time | Action | Mes |
|---|---|---|---|
| 3 | 11:01:26 | DELETE FROM university.student WHERE id ='98988' | 1 rov |
| 4 | 11:01:54 | SELECT * FROM university.student LIMIT 0, 1000 | 12 ro |

Fig: Deleting one tuple from table of student.

3. In the database of SPJ_MNG, use SQL statement to do the following update operations

1.Change the color of all red parts to blue.
2.Part P6 supplied by S5 for J4 is replaced by S3, please make necessary modification.
3.Delete S2 record from supplier table and delete corresponding record from supply table.
4.Please insert (S2, J6, p4,200) into the supply table SPJ.

# Answer No.3(1)

```
SET SQL_SAFE_UPDATES = 0;
UPDATE spj_mng.p SET COLOR='blue' WHERE COLOR='red';
SELECT * FROM spj_mng.p;
```

Result Grid

| PNO | PNAME | COLOR | WEIGHT |
|---|---|---|---|
| P1 | nut | blue | 12 |
| P2 | bolt | green | 17 |
| P3 | screwdriver | blue | 14 |
| P4 | screwdriver | blue | 14 |
| P5 | cam | blue | 40 |
| P6 | gear | blue | 30 |

Output

Action Output

| # | Time | Action | Message |
|---|---|---|---|
| 1 | 11:17:09 | SET SQL_SAFE_UPDATES = 0 | 0 row(s) affected |
| 2 | 11:17:16 | UPDATE spj_mng.p SET COLOR='blue' WHERE COLOR='red' | 3 row(s) affected Rows matched: 3 Changed: 3 Warnings: 0 |
| 3 | 11:18:04 | SELECT * FROM spj_mng.p LIMIT 0, 1000 | 6 row(s) returned |

# Answer No.3(2)

```
SET SQL_SAFE_UPDATES = 0;
UPDATE spj_mng.spj SET SNO='S3' WHERE SNO='S5' AND PNO ='P6' AND JNO='J4';
SELECT * FROM spj_mng.spj;
```

Result Grid

| SNO | PNO | JNO | QTY |
|---|---|---|---|
| S1 | P1 | J1 | 200 |
| S1 | P1 | J3 | 100 |
| S1 | P1 | J4 | 700 |
| S1 | P2 | J2 | 100 |
| S2 | P3 | J1 | 400 |
| S2 | P3 | J2 | 200 |
| S2 | P3 | J4 | 500 |
| S2 | P3 | J5 | 400 |
| S2 | P5 | J1 | 400 |
| S2 | P5 | J2 | 100 |
| S3 | P1 | J1 | 200 |
| S3 | P3 | J1 | 200 |
| S4 | P5 | J1 | 200 |
| S4 | P6 | J3 | 100 |
| S4 | P6 | J4 | 300 |
| S5 | P2 | J4 | 100 |
| S5 | P3 | J1 | 200 |
| S5 | P6 | J2 | 200 |

Output

Action Output

| # | Time | Action |
|---|---|---|
| 1 | 11:36:57 | UPDATE spj_mng.spj SET SNO='S3' WHERE SNO='S5' AND PNO ='P6' AND JNO='J4' |
| 2 | 11:37:02 | SELECT * FROM spj_mng.spj LIMIT 0, 1000 |

| SNO | PNO | JNO | QTY |
|---|---|---|---|
| S1 | P1 | J1 | 200 |
| S1 | P1 | J3 | 100 |
| S1 | P1 | J4 | 700 |
| S1 | P2 | J2 | 100 |
| S3 | P1 | J1 | 200 |
| S3 | P3 | J1 | 200 |
| S4 | P5 | J1 | 200 |
| S4 | P6 | J3 | 100 |
| S4 | P6 | J4 | 300 |
| S5 | P2 | J4 | 100 |
| S5 | P3 | J1 | 200 |
| S5 | P6 | J2 | 200 |
| S3 | P6 | J4 | 500 |
| S2 | P4 | J6 | 200 |

## Answer No.3(3)

```
1 •   DELETE FROM spj_mng.spj WHERE  SNO='S2';
2 •   SELECT*FROM spj_mng.spj;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

| SNO | PNO | JNO | QTY |
|---|---|---|---|
| S1 | P1 | J1 | 200 |
| S1 | P1 | J3 | 100 |
| S1 | P1 | J4 | 700 |
| S1 | P2 | J2 | 100 |
| S3 | P1 | J1 | 200 |
| S3 | P3 | J1 | 200 |
| S4 | P5 | J1 | 200 |
| S4 | P6 | J3 | 100 |
| S4 | P6 | J4 | 300 |
| S5 | P2 | J4 | 100 |
| S5 | P3 | J1 | 200 |
| S5 | P6 | J2 | 200 |
| S3 | P6 | J4 | 500 |

spj 5

Output

Action Output

| # | Time | Action | Message |
|---|---|---|---|
| 1 | 11:46:53 | DELETE FROM spj_mng.spj WHERE SNO='S2' | 6 row(s) affected |
| 2 | 11:47:45 | SELECT*FROM spj_mng.spj LIMIT 0, 1000 | 13 row(s) returned |

## Answer No.3(4)

```
INSERT INTO spj_mng.spj (SNO,PNO,JNO,QTY) VALUES ('S2','P4','J6',200);
SELECT*FROM spj_mng.spj;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

| SNO | PNO | JNO | QTY |
|---|---|---|---|
| S1 | P1 | J1 | 200 |
| S1 | P1 | J3 | 100 |
| S1 | P1 | J4 | 700 |
| S1 | P2 | J2 | 100 |
| S3 | P1 | J1 | 200 |
| S3 | P3 | J1 | 200 |
| S4 | P5 | J1 | 200 |
| S4 | P6 | J3 | 100 |
| S4 | P6 | J4 | 300 |
| S5 | P2 | J4 | 100 |
| S5 | P3 | J1 | 200 |
| S5 | P6 | J2 | 200 |
| S3 | P6 | J4 | 500 |
| S2 | P4 | J6 | 200 |

spj 6

Output

Action Output

| # | Time | Action | Message |
|---|---|---|---|
| 1 | 11:46:53 | DELETE FROM spj_mng.spj WHERE SNO='S2' | 6 row(s) affected |
| 2 | 11:47:45 | SELECT*FROM spj_mng.spj LIMIT 0, 1000 | 13 row(s) returned |
| 3 | 11:56:39 | SELECT * FROM spj_mng.spj LIMIT 0, 1000 | 13 row(s) returned |
| 4 | 11:59:25 | INSERT INTO spj_mng.spj (SNO,PNO,JNO,QTY) VALUES ('S2','P4','J6',200) | 1 row(s) affected |
| 5 | 12:00:02 | SELECT*FROM spj_mng.spj LIMIT 0, 1000 | 14 row(s) returned |

(4) Use three different ways (SQL statement) to find the student ID and name of all students who take "Database System Concept", and then analyze and compare the performance of each query process.

## Answer No.4

```
SELECT*FROM university.student as db
natural join university.takes where course_id='CS-347'
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

| ID | name | dept_name | tot_cred | course_id | sec_id | semester | year | grade |
|---|---|---|---|---|---|---|---|---|
| 00128 | Zhang | Comp. Sci | 102 | CS-347 | 1 | Fall | 2009 | A- |
| 12345 | Shankar | Comp. Sci | 32 | CS-347 | 1 | Fall | 2009 | A |

A NATURAL JOIN is a JOIN operation that creates an implicit join clause for

you based on the common columns in the two tables being joined. Common columns are columns that have the same name in both tables. A NATURAL JOIN can be an INNER join, a LEFT OUTER join, or a RIGHT OUTER join. By using this way, we can get the ID and Name .We can make this table has a alias as db .

```
1   SELECT*FROM university.student
2   natural join university.takes where course_id='CS-347'
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

| ID | name | dept_name | tot_cred | course_id | sec_id | semester | year | grade |
|---|---|---|---|---|---|---|---|---|
| 00128 | Zhang | Comp. Sci | 102 | CS-347 | 1 | Fall | 2009 | A- |
| 12345 | Shankar | Comp. Sci | 32 | CS-347 | 1 | Fall | 2009 | A |

Here, it's almost similar like the natural join but we didn't created alias in the statement.

pj spj takes takes student student department SQL File 20* SQL File 21* advisor prereq instructor student student takes course

Limit to 1000 r

```
1   select DISTINCT student.name,student.ID  FROM course
2    JOIN  student  ON course.dept_name =student.dept_name where course.course_id='CS-347' and student.ID='00128'or student.ID='12345' ;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

| name | ID |
|---|---|
| Zhang | 00128 |
| Shankar | 12345 |

Here, we also use join and the student .name and student.ID from the course table that was joined with student table on the course.dept_name but the difference from other statements because of the DISTINCT keyword.

In my opinion, this is the best among all the statements ,I created above. Because, we get the exact answer and not anything else.

We can see that, there are few differences that are evident. However, the output is presented to us. Using 3 different SQL statement act logically differently but the the desire output is not changed. We got the ID and name .

5. For university database, complete the following data query with SQL statement

1. Query the total score of credits obtained by each student , and output the student ID, name and credit obtained in the order from high to low.
2. Query the name of the student: the student has taken all courses and one of the courses has a grade of better than B .

## Answer No. 5(1)

```
1 •  SELECT id,name,tot_cred FROM university.student ORDER by tot_cred DESC
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

| id | name | tot_cred |
|---|---|---|
| 23121 | Chavez | 110 |
| 00128 | Zhang | 102 |
| 98765 | Bourikas | 98 |
| 19991 | Brandt | 80 |
| 76653 | Aoi | 60 |
| 76543 | Brown | 58 |
| 44553 | Peltier | 56 |
| 54321 | Williams | 54 |
| 45678 | Levy | 46 |
| 55739 | Sanchez | 38 |
| 12345 | Shankar | 32 |
| 70557 | Snow | 0 |

student 4

Output

Action Output

| # | Time | Action | Message |
|---|---|---|---|
| 1 | 13:42:49 | SELECT id,name,tot_cred FROM university.student ... | 12 row(s) returned |

## Answer No. 5(2)

```
1 •  SELECT*FROM university.takes WHERE GRADE REGEXP 'A+|A|B+' AND grade !='B'
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell

| ID | course_id | sec_id | semester | year | grade |
|---|---|---|---|---|---|
| 00128 | CS-101 | 1 | Fall | 2009 | A |
| 00128 | CS-347 | 1 | Fall | 2009 | A- |
| 12345 | CS-190 | 2 | Spring | 2009 | A |
| 12345 | CS-315 | 1 | Spring | 2010 | A |
| 12345 | CS-347 | 1 | Fall | 20 2010 | |
| 44553 | PHY-101 | 1 | Fall | 2009 | B- |
| 45678 | CS-101 | 1 | Spring | 2010 | B+ |
| 54321 | CS-101 | 1 | Fall | 2009 | A- |
| 54321 | CS-190 | 2 | Spring | 2009 | B+ |
| 55739 | MU-199 | 1 | Spring | 2010 | A- |
| 76543 | CS-101 | 1 | Fall | 2009 | A |
| 76543 | CS-319 | 2 | Spring | 2010 | A |
| 98988 | BIO-101 | 1 | Summer | 2009 | A |

takes 3

Output

Action Output

| # | Time | Action | Message |
|---|---|---|---|
| 1 | 14:45:33 | SELECT*FROM university.takes WHERE GRADE ... | 13 row(s) returned |

6.Use at least three different SQL statements to query the university database: query the student ID and name of the course named "database", and then design the experiment by ourselves, compare and analyze the efficiency of the three kinds of query with data, and analyze the reasons.

## Answer No. 6

```
SELECT*FROM university.course
natural join university.takes  where title =
(Select title FROM university.course WHERE course_id ='CS-347');
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

| course_id | title | dept_name | credits | ID | sec_id | semester | year | grade |
|---|---|---|---|---|---|---|---|---|
| CS-347 | Database System Concept | Comp.Sci. | 3 | 00128 | 1 | Fall | 2009 | A- |
| CS-347 | Database System Concept | Comp.Sci. | 3 | 12345 | 1 | Fall | 2009 | A |

Output

Action Output

| # | Time | Action |
|---|---|---|
| 1 | 13:27:12 | SELECT * FROM university.course LIMIT 0, 1000 |
| 2 | 13:28:15 | natural join university.takes where title = (Select title FROM university.course WHERE course_id ='CS-347') |
| 3 | 13:29:22 | SELECT*FROM university.course natural join university.takes where title = (Select title FROM university.course ... |

A NATURAL JOIN is a JOIN operation that creates an implicit join clause for you based on the common columns in the two tables being joined. Common columns are columns that have the same name in both tables. A NATURAL JOIN can be an INNER join, a LEFT OUTER join, or a RIGHT OUTER join. In the where, we create a condition using that way, we can get the title and ID of the the desired student.

```
SELECT * FROM university.takes
NATURAL JOIN university.course WHERE course_id='CS-347'
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

| course_id | ID | sec_id | semester | year | grade | title | dept_name | credits |
|---|---|---|---|---|---|---|---|---|
| CS-347 | 00128 | 1 | Fall | 2009 | A- | Database System Concept | Comp.Sci. | 3 |
| CS-347 | 12345 | 1 | Fall | 2009 | A | Database System Concept | Comp.Sci. | 3 |

Output

Action Output

| # | Time | Action | Message |
|---|---|---|---|
| 1 | 14:18:14 | SELECT * FROM university.takes NATURAL JOIN university.course WHERE course_id='CS-347' LIMIT 0, 1000 | 2 row(s) returned |

In the above the statement we can see that, the statements are much more clean and precise .We get our desired answer as per our expectation.

Query 1 | takes | course

Limit to 1000 rows

```
SELECT*FROM university.course as db
natural join university.takes where course_id='CS-347'
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

| course_id | title | dept_name | credits | ID | sec_id | semester | year | grade |
|---|---|---|---|---|---|---|---|---|
| CS-347 | Database System Concept | Comp.Sci. | 3 | 00128 | 1 | Fall | 2009 | A- |
| CS-347 | Database System Concept | Comp.Sci. | 3 | 12345 | 1 | Fall | 2009 | A |

In the above the query statement we can see that,it's almost similar like the natural join we created at the beginning but if we can create it by putting it in alias db.

We can see that, there are few differences that are evident, how the output is presented to us. Using 3 different SQL statement act logically same and we can show that one statement can be modified in many ways but the the desire output is not changed.

**Problem:**

simple syntax error and few logical errors.

**Solution:**

To solve these problems I looked for information in internet. In order to understand some questions and procedure I also asked the teacher to help me understand them. And provided instructions helped to solve some of my errors during the experiment.

**Summary:**

From this experiment I have learned SQL statement to create database and table. I have learned how to update and delete methods of database and table. Have become familiar with SQL statements of data insertion, modification and deletion of basic tables. Learned all kinds of data operation about basic table in GUI. So, I have been able to have basic knowledge of SQL query performances and analysis.

## References


1) https://www.w3schools.com/sql/

2) https://www.w3schools.com/sql/sql_syntax.asp

3) https://www.youtube.com/watch?v=7S_tz1z_5bA

4) https://www.youtube.com/watch?v=ER8oKX5myE0

5)https://www.w3resource.com/mysql/advance-query-in-mysql/mysql-natural-join.php

6) https://docs.oracle.com/javadb/10.8.3.0/ref/rrefsqljnaturaljoin.html


## Attachments:

1) DB2_2019380141_ABID ALI.docx

2) DB2_2019380141_ABID ALI.pdf