

CSC343 Worksheet 7 Solution

June 23, 2020

1. a)

```
1  void askUserForPrice() {
2      EXEC SQL BEGIN DECLARE SECTION;
3          int model;
4          float speed;
5          int ram;
6          int hd;
7          float price;
8          char maker;
9          float targetPrice;
10
11         float minDiff;
12         int modelSol;
13         float speedSol;
14         char makerSol;
15     EXEC SQL END DECLARE SECTION;
16
17     EXEC SQL DECLARE execCursor CURSOR FOR
18         SELECT * FROM Product NATURAL JOIN PC
19
20     EXEC SQL OPEN execCursor;
21
22     printf("Enter target price:");
23     scanf("%f", &targetPrice);
24
25     while(1) {
26         EXEC SQL FETCH FROM execCursor INTO :model,
27             :speed, :ram, :hd, :price, :maker;
28
29         if (NO_MORE_TUPLES) break;
30
31         if (abs(price - targetPrice) >= minDiff) {
32             continue;
33         }
34
35         minDiff = abs(price - targetPrice);
36         modelSol = model;
37         speedSol = speed;
38         makerSol = maker;
39     }
```

```
40
41     EXEC SQL CLOSE execCursor;
42
43     printf("maker=%c, model=%d, speed=%.2f\n", makerSol, modelSol
44           , speedSol);
45 }
46
```

Notes:

- EXEC SQL
 - Allows to use SQL statements within a host-language program
- The DECLARE Section
 - is used to declare shared variables
 - **Syntax:**
EXEC SQL BEGIN DECLARE SECTION;
... // Variable declarations in any language
EXEC SQL END DECLARE SECTION;

Example:

```
1     void getStudio() {
2         EXEC SQL BEGIN DECLARE SECTION;
3         char studioName[50], studioAddr[256]; // <- c
4         variables
5         char SQLSTATE[6];
6         EXEC SQL END DECLARE SECTION;
7
8         EXEC SQL INSERT INTO Studio(name, address)
9             VALUES (:studioName, :studioAddr);
10    }
```

- Cursors
 - Is the most versatile way to connect SQL queries
 - **Syntax:**
EXEC SQL DECLARE < cursor name > CURSOR FOR < query >

EXEC SQL OPEN < cursor name >;
...
EXEC SQL CLOSE < cursor name >;

Example:

```

1      void getStudio() {
2          EXEC SQL BEGIN DECLARE SECTION;
3              char studioName[50], studioAddr[256]; // <- c
variables
4              char SQLSTATE[6];
5          EXEC SQL END DECLARE SECTION;
6
7          EXEC SQL INSERT INTO Studio(name, address)
8              VALUES (:studioName, :studioAddr);
9      }
10

```

Example in Python:

```

1      import sqlite3
2      connection = sqlite3.connect("company.db")
3
4      cursor = connection.cursor()
5
6      staff_data = [ ("William", "Shakespeare", "m", "
1961-10-25"),
7                      ("Frank", "Schiller", "m", "1955-08-17"
8                      ),
9                      ("Jane", "Wall", "f", "1989-03-14") ]
10
11      for p in staff_data:
12          format_str = """INSERT INTO employee (staff_number,
13              fname, lname, gender, birth_date)
14              VALUES (NULL, "{first}", "{last}", "{gender}", "{
15              birthdate}");"""
16
17          sql_command = format_str.format(first=p[0], last=p
18          [1], gender=p[2], birthdate = p[3])
19          cursor.execute(sql_command)
20

```

• Fetch Statement

– fetch data from the result table one row at a time

– Syntax:

EXEC SQL FETCH FROM < cursor name > INTO < list of variables >

Example:

```

1      void worthRanges() {
2          int i, digits, counts[15];
3          EXEC SQL BEGIN DECLARE SECTION;
4              int worth;
5              char SQLSTATE[6];
6          EXEC SQL END DECLARE SECTION;
7          EXEC SQL DECLARE execCursor CURSOR FOR
8              SELECT netWorth FROM MovieExec;
9

```

```

9
10         EXEC SQL OPEN execCursor;
11         for (i=1; i < 15; i++) counts[i] = 0;
12         while(1) {
13             EXEC SQL FETCH FROM execCursor INTO :worth; //
fetches a row of value from movieExec and stores in worth
14             if (NO_MORE_TUPLES) break;
15
16             ...
17         }
18     }
19

```

b)

```

2     void findLaptops() {
3         EXEC SQL BEGIN DECLARE SECTION;
4         int model;
5         float speed;
6         int ram;
7         int hd;
8         int screen;
9         float price;
10
11         float minSpeed;
12         int minRam;
13         int minHd;
14         float minPrice;
15     EXEC SQL END DECLARE SECTION;
16
17     EXEC SQL DECLARE execCursor CURSOR FOR
18         SELECT model, speed, ram, hd, screen, price, maker
19         FROM Product NATURAL JOIN Laptop;
20
21     EXEC SQL OPEN execCursor;
22
23     printf("Enter minimum speed:");
24     scanf("%f", &minSpeed);
25
26     printf("Enter minimum ram:");
27     scanf("%f", &minRam);
28
29     printf("Enter minimum hard-drive space:");
30     scanf("%f", &minHd);
31
32     printf("Enter minimum price:");
33     scanf("%f", &minPrice);
34
35     while(1) {
36         EXEC SQL FETCH FROM execCursor INTO :model,
37             :speed, :ram, :hd, :screen, :price, :maker;
38
39         if (NO_MORE_TUPLES) break;
40
41         if (
            speed >= minSpeed &&

```

```

42         ram >= minRam &&
43         hd >= minHd &&
44         screen >= minScreen
45     ) {
46         printf("model=%d, speed=%.2f, ram=%d, hd=%d, screen=%d, price=%.2f, maker=%c",
47             model, speed, ram, hd, screen, price, maker);
48     }
49 }
50
51 EXEC SQL CLOSE execCursor;
52 }
53

```

```

c) void findLaptops() {
2     EXEC SQL BEGIN DECLARE SECTION;
3         int model;
4         float speed;
5         int ram;
6         int hd;
7         int screen;
8         float price;
9
10        float minSpeed;
11        int minRam;
12        int minHd;
13        float minPrice;
14    EXEC SQL END DECLARE SECTION;
15
16    EXEC SQL DECLARE execCursor CURSOR FOR
17        SELECT model, speed, ram, hd, screen, price, maker
18        FROM Product NATURAL JOIN Laptop;
19
20    EXEC SQL OPEN execCursor;
21
22    printf("Enter minimum speed:");
23    scanf("%f", &minSpeed);
24
25    printf("Enter minimum ram:");
26    scanf("%f", &minRam);
27
28    printf("Enter minimum hard-drive space:");
29    scanf("%f", &minHd);
30
31    printf("Enter minimum price:");
32    scanf("%f", &minPrice);
33
34    while(1) {
35        EXEC SQL FETCH FROM execCursor INTO :model,
36            :speed, :ram, :hd, :screen, :price, :maker;
37
38        if (NO_MORE_TUPLES) break;
39
40        if (

```

```

41         speed >= minSpeed &&
42         ram >= minRam &&
43         hd >= minHd &&
44         screen >= minScreen
45     ) {
46         printf("model=%d, speed=%.2f, ram=%d, hd=%d, screen=%d, price=%.2f, maker=%c",
47             model, speed, ram, hd, screen, price, maker);
48     }
49 }
50
51 EXEC SQL CLOSE execCursor;
52 }
53

```

d)

```

1  #include <stdbool.h>
2  #include <string.h>
3  ...
4  void printSpecifications() {
5      EXEC SQL BEGIN DECLARE SECTION;
6          int model;
7          bool color;
8          char printType[50];
9          float price;
10
11         float speed;
12         int ram;
13         int hd;
14         int screen;
15
16         char maker;
17         int productModel;
18         char productType[50];
19
20         char targetMaker;
21     EXEC SQL END DECLARE SECTION;
22
23     EXEC SQL DECLARE execCursor CURSOR FOR
24         SELECT DISTINCT maker, DISTINCT productType FROM Product;
25
26     printf("Enter manufacturer:");
27     scanf("%c", &targetMaker);
28
29     EXEC SQL OPEN execCursor;
30     while (1) {
31         EXEC SQL FETCH FROM execCursor INTO :maker, :productType;
32
33         if (NO_MORE_TUPLES) break;
34
35         if (tolower(maker) != tolower(targetMaker)) continue;
36
37         if (strcmp(productType, 'pc')) {
38             EXEC SQL DECLARE pcCursor CURSOR FOR
39                 SELECT speed, ram, hd, price FROM PC

```

```

40         NATURAL JOIN Product
41         WHERE type=productType;
42
43     EXEC SQL OPEN pcCursor;
44     while(1) {
45         EXEC SQL FETCH FROM pcCursor INTO :speed,
46             :ram, :hd, :price;
47
48         if (NO_MORE_TUPLES) break;
49
50         printf("model=%d, speed=%.2f, ram=%d, hd=%d,
51 price=%.2f, maker=%c, type=%s",
52             model, speed, ram, hd, screen, price, maker,
53 productType);
54     }
55     EXEC SQL CLOSE pcCursor;
56
57 } else if (strcmp(productType, 'laptop')) {
58
59     EXEC SQL DECLARE laptopCursor CURSOR FOR
60     SELECT speed, ram, hd, screen, price FROM Laptop
61     NATURAL JOIN Product
62     WHERE type=productType;
63
64     EXEC SQL OPEN laptopCursor;
65     while(1) {
66         EXEC SQL FETCH FROM laptopCursor INTO :speed,
67             :ram, :hd, :screen, :price;
68
69         if (NO_MORE_TUPLES) break;
70
71         printf("model=%d, speed=%.2f, ram=%d, hd=%d,
72 screen=%d, price=%.2f, maker=%c, type=%s",
73             model, speed, ram, hd, screen, screen, price,
74 maker, productType);
75     }
76     EXEC SQL CLOSE laptopCursor;
77
78 } else if (strcmp(productType, 'printer')) {
79
80     EXEC SQL DECLARE printerCursor CURSOR FOR
81     SELECT color, printType, price FROM Printer
82     NATURAL JOIN Product
83     WHERE type=productType;
84
85     EXEC SQL OPEN printerCursor;
86     while(1) {
87         EXEC SQL FETCH FROM printerCursor INTO :color,
88             :printType, :price;
89
90         if (NO_MORE_TUPLES) break;
91
92         printf("model=%d, color=%s, price=%.2f, maker=%c,
93 type=%s",

```

```

89         model, color ? "true" : "false", price, maker,
type);
90     }
91     EXEC SQL CLOSE printerCursor;
92 }
93 }
94 EXEC SQL CLOSE execCursor;
95 }
96

```

e)

```

f) #include <stdbool.h>
2  #include <string.h>
3  ...
4  void insertNewPC() {
5      EXEC SQL BEGIN DECLARE SECTION;
6          int model;
7          float speed;
8          int ram;
9          int hd;
10         float price;
11         char maker;
12
13         int modelCount;
14     EXEC SQL END DECLARE SECTION;
15
16     printf("Enter manufacturer:\n");
17     scanf("%c", &maker);
18
19     printf("Enter model:\n");
20     scanf("%d", &model);
21
22     printf("Enter speed:\n");
23     scanf("%f", &speed);
24
25     printf("Enter ram:\n");
26     scanf("%d", &ram);
27
28     printf("Enter hd:\n");
29     scanf("%d", &hd);
30
31     printf("Enter price:\n");
32     scanf("%f", &price);
33
34     printf("Enter maker:\n");
35     scanf("%c", &maker);
36
37     EXEC SQL DECLARE execCursor CURSOR FOR
38         SELECT COUNT(model) FROM (
39             (SELECT model FROM Product WHERE model=:model)
40             UNION
41             (SELECT model FROM PC WHERE model=:model)
42         );
43

```



```

44     EXEC SQL OPEN execCursor;
45     EXEC SQL FETCH FROM execCursor INTO :modelCount;
46
47     if (modelCount != 0) {
48         printf("Error. Model already exists in database.");
49     } else {
50         EXEC SQL INSERT INTO PC(model, speed, ram, hd, price)
51             VALUES(:model, :speed, :ram, :hd, :
price);
52
53         EXEC SQL INSERT INTO Product(model, maker, type)
54             VALUES(:model, :maker, "pc")
55     }
56
57
58     EXEC SQL CLOSE execCursor;
59 }
60

```

2. a)

```

void classWithLargestPower() {
2     EXEC SQL BEGIN DECLARE SECTION;
3     int class;
4     EXEC SQL END DECLARE SECTION;
5
6     EXEC SQL SELECT class FROM FROM Classes
7         INTO :class
8         WHERE numGuns * POWER(bore, 3) >= ALL (
9             SELECT numGuns * POWER(bore, 3) FROM Classes
10        );
11
12    printf("Class = %s\n", class);
13 }
14

```

b)

```

void countryWithMostShipsSunk() {
2     EXEC SQL BEGIN DECLARE SECTION;
3     char targetBattle[255];
4     EXEC SQL END DECLARE SECTION;
5
6     printf("Enter name of battle:\n");
7     scanf("%s", &targetBattle);
8
9     EXEC SQL DECLARE execCursor CURSOR FOR
10        SELECT country, Ships.name, Outcomes.result FROM Classes
11        INNER JOIN Ships ON Classes.class = Ships.class
12        INNER JOIN Outcomes ON Ships.name = Outcomes.ship
13        INNER JOIN Battles ON Battles.name = Outcome.battle;
14        WHERE Battles.name=:targetBattle;
15
16    EXEC SQL OPEN execCursor;
17
18    EXEC SQL CLOSE execCursor;
19

```

```
20     }  
21
```