First Name: _____ Last Name: _____ Student ID: _____

BBM 471 Database Management Systems

Midterm 2 - May, 11th, 2018

ANSWER KEY

Instructions

This exam contains **8** multi-part problems and you have **75 minutes** to earn 100 points. When the exam begins, please write your name on every page of this exam booklet. Check that the exam booklet contains **6 pages** to the exam, including this one. This exam is a **closed book and notes exam (except for the handwritten A4 sized cheat sheets)**. Show all work, as partial credit will be given since you will be graded not only on the correctness and efficiency of your answers, but also on your clarity that you express it. Be neat. Each question is worth some points. For instance, *2*6+3=15* points means that the question has two parts, each of which is six points. There is additional three points for answering both parts correctly, as well.

Good luck!

Problem	Points	Score
1	15	
2	15	
3	15	
4	15	
5	10	
6	10	
7	5	
8	25	
Total	100	

It is a violation of the Academic Integrity Code to look at any exam paper other than your own, any other reference material (books, lecture notes) except for the handwritten A4 sized cheat sheet, or to give inappropriate help to someone or to receive unauthorized aid by someone. Please also not discuss this exam with the students who are scheduled to take a makeup exam.

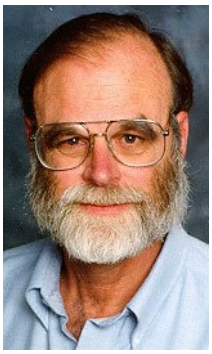
Academic Integrity is expected of all students of Hacettepe University at all times, whether in the presence or absence of members of the faculty. Do NOT sign nor take this exam if you do not agree with the honor code.

Understanding this, I declare I shall not give, use or receive unauthorized aid in this examination.

Signature: _____

First Name: _____ Last Name: _____ Student ID: _____

Question 1 (3*4+3=15 points): **a)** What is the name of the man in the picture below? **b)** How did he die? **c)** What was his major expertise about?
(Note: **Just few words or a single complete statement for each!**)



a) Jim Gray
b) Lost in sea (sailed for scatterring his mom's ashes)
c) Transaction processing

Question 2 (15 points): Consider the code given below. Do you think it is the optimal solution for the purpose it serves? If not, write an optimized version?

```
SET @CT = (SELECT COUNT(*) FROM T);
If @CT > 0
BEGIN
      <Do something>
END
```

```
/* There is no need to select * to count rows. */

If EXISTS (SELECT 1 FROM T)
BEGIN
<Do something>
END
```

Question 3 (2*6+3=15 points): **a)** When you committed a transaction, you can make sure that your changes are permanent in the database. Besides that, what is the other thing you can make sure of? **b)** What does *checkpointing* do within database scope?
(Note: **Just few words or a single complete statement for each!**)

a) Log record of that transaction was written to disk.

b) Checkpoint writes dirty data+log pages to disk.

First Name: _____ Last Name: _____ Student ID: _____

Question 4 (2*6+3=15 points): Consider the SQL statements given below and provide outputs (including headers) for the select statements at the bottom:

a)

```
Create Table myTable
  (c1 int, c2 int not null);
Insert Into myTable
  Values (1, 10), (2, 20), (3, 30);
Create View myView
  As Select c2 From myTable;
Delete From myView Where c2 = 20;


Select * From myTable;
```

```
+------+----+
| c1   | c2 |
+------+----+
|    1 | 10 |
|    3 | 30 |
+------+----+
```

b)

```
Create Table myTable2
  (c3 int, c4 int);
Insert into myTable2
  Values (1, 10), (2, 20), (3, 30);
Create View myView2
  As Select * From myTable, myTable2;
Delete From myView2 Where c2 = 20;
ERROR 1395 (HY000): Can not delete from join
view 'bbm471.myview2'


Select * From myTable2;
```

```
+------+----+
| c3   | c4 |
+------+----+
|    1 | 10 |
|    2 | 20 |
|    3 | 30 |
+------+----+
```

Question 5 (2*4+2=10 points): Consider the content of a table *t* with a column *c* given below.

```
+------+
|   c  |
+------+
|    1 |
|    2 |
|    3 |
+------+
```

What will be the output (including headers) of the following queries?

a) select 2 from t;

```
+---+
| 2 |
+---+
| 2 |
| 2 |
| 2 |
+---+
3 rows in set (0.00 sec)
```

b) select distinct 5 from t;

```
+---+
| 5 |
+---+
| 5 |
+---+
1 row in set (0.00 sec)
```

Question 6 (10 bonus points): Regarding MYSQL index hints, what is the difference between *USE INDEX* and *FORCE INDEX* expressions?

Table scan is assumed to be *very* expensive by force index. In other words, a table scan is used only if there is no way to use one of the named indexes to find rows in the table.

Question 7 (5 points): What is the output of the SQL statements given below? (Assume you are using a MYSQL database)

`SELECT IF(1<2,'yes','no');`
`SELECT IFNULL(1,0);`
`SELECT IFNULL(1/0,10);`
`SELECT NULLIF(1,1);`
`SELECT NULLIF(1,2);`
	Output:
	'yes'
	1
	10
	NULL
	1

Question 8 (2*5+10+5=25 points): Assume you are the DBA of a company which supplies materials to universities. Further assume that your boss is complaining about the slow processing of queries on Hacettepe University's (HU) orders and kindly asks you to device a solution for that.

Your analysis shows that the *Order* table has millions of rows. HU has considerable amount of orders, too. So, you decide to create a materialized view to hold HU's orders. Unfortunately, the RDBMS you are using does not support materialized views. In another words, you do not have the "*Create Materialized View*" command available in your database.

Assume that the Order table is given as below and the customer ID of HU is "635":

Order (orderId, supplyCode, amount, customerId, orderDate)

a) Are you a good DBA? Evaluate yourself based on your so called materialized view solution.
I guess, no. I should have used MV structure to keep data, for example, as already joined.

b) How would you implement a so called materialized view to hold HU's orders without having RDBMS support?
Create Table myMView as Select * From Order Where customerId = "635";

c) Write necessary SQL statements to *incrementally* maintain your so called materialized view in case of INSERT and DELETE operations that would be executed on the *Order* table.
Create Trigger iTrigger On Order After INSERT ();
Create Trigger dTrigger On Order After DELETE ();

Note: If you do not know the correct syntax of the SQL statements that you will use in your answer, do not forget to write some explanation next to your answer.