Assessment was due by Wed, 2022-10-26 10:55:00

Midterm 2 COMP 421 Fall 2022

Note: There are a total of 100 points on this exams.

Don't panic!

You have 75 minutes to finish the exam.

- You *should* stay in full screen mode.
    - A <ctrl-f> find will give you a warning, which is OK
- You must hand this midterm in on time.
    - Points removed for late submissions.
- Absolutely no exams will be accepted after the end of class.
    - Only your **first** submission will be accepted/graded.
    - Avoid accidental submissions. Fill in your name when you are ready to submit.

I recommend that you have several pieces of scrap paper to doodle notes on during the exam. I *strongly* recommend you read the whole exam and begin with questions you know how to solve quickly. Some questions will be harder or take longer than others; don't spend all your time on one question worth only a few points! Also, don't worry if you don't have time to solve every question - that's expected.

Consider this midterm **closed book**.
You can **NOT** reference other online homeworks, worksheets, etc.
You can use your notes or other things printed out. They should be on paper as you may not switch screens after starting the exam.

You **MAY NOT** Google for anything, You **MAY NOT** leave this website, you **MAY NOT** visit any websites, and you **MAY NOT** copy from a friend. Do not paste information into your midterm unless you know it came from your midterm. You **MUST NOT** not receive help from anyone.

If you do not know the origin of material you should not paste it into this exam. All material pasted into this exam must originate from this exam. This implies, but is not limited to, copying from previous assignments, copying from text messages, or copying from **any** website.

You **MUST** use the Google Chrome browser.

The instruction team will **not** answer questions about course content, SQL syntax, etc. We will only deal with issues related to exam implementation.

If your browser hangs, for example because of a bad SQL query, simply kill the page and refresh. It *should* restore all your work even if it doesn't re-evaluate all answers, color-highlight boxes, etc.

You may **NOT** leave before you submit your exam. When you submit your exam you must enter the code displayed on the screen at the front of the class or given to you by ARS.`

You **must not** use your computer or phone in the classroom after you submit your exam.

The browser will change input box color green to indicate correctness. A black or red box indicates an incorrect answer.

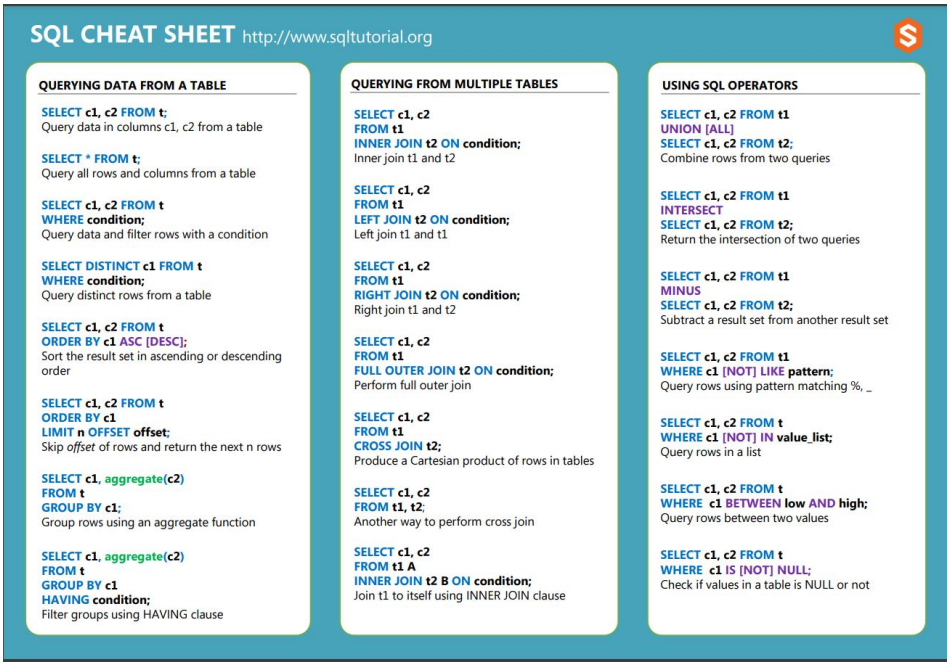
Note that HTML select statements with drop-downs are simple multiple choice questions. No highlighting of correct answers are done for select questions.

Green highlight should just assist you. If you believe your answer is correct and the input box did not turn green, continue on. Per the syllabus, highlighting is simply an aide not a guarantee.

Note: For database queries that are applied to two databases, **two** green lights are required to get any credit for the question.

SQL Tutorial Cheat Sheet

Following are three SQL tutorial cheat sheets available from http://www.sqltutorial.org



SQL CHEAT SHEET http://www.sqltutorial.org

QUERYING DATA FROM A TABLE

```
SELECT c1, c2 FROM t;
```
Query data in columns c1, c2 from a table

```
SELECT * FROM t;
```
Query all rows and columns from a table

```
SELECT c1, c2 FROM t
WHERE condition;
```
Query data and filter rows with a condition

```
SELECT DISTINCT c1 FROM t
WHERE condition;
```
Query distinct rows from a table

```
SELECT c1, c2 FROM t
ORDER BY c1 ASC [DESC];
```
Sort the result set in ascending or descending order

```
SELECT c1, c2 FROM t
ORDER BY c1
LIMIT n OFFSET offset;
```
Skip *offset* of rows and return the next n rows

```
SELECT c1, aggregate(c2)
FROM t
GROUP BY c1;
```
Group rows using an aggregate function

```
SELECT c1, aggregate(c2)
FROM t
GROUP BY c1
HAVING condition;
```
Filter groups using HAVING clause

QUERYING FROM MULTIPLE TABLES

```
SELECT c1, c2
FROM t1
INNER JOIN t2 ON condition;
```
Inner join t1 and t2

```
SELECT c1, c2
FROM t1
LEFT JOIN t2 ON condition;
```
Left join t1 and t1

```
SELECT c1, c2
FROM t1
RIGHT JOIN t2 ON condition;
```
Right join t1 and t2

```
SELECT c1, c2
FROM t1
FULL OUTER JOIN t2 ON condition;
```
Perform full outer join

```
SELECT c1, c2
FROM t1
CROSS JOIN t2;
```
Produce a Cartesian product of rows in tables

```
SELECT c1, c2
FROM t1, t2;
```
Another way to perform cross join

```
SELECT c1, c2
FROM t1 A
INNER JOIN t2 B ON condition;
```
Join t1 to itself using INNER JOIN clause

USING SQL OPERATORS

```
SELECT c1, c2 FROM t1
UNION [ALL]
SELECT c1, c2 FROM t2;
```
Combine rows from two queries

```
SELECT c1, c2 FROM t1
INTERSECT
SELECT c1, c2 FROM t2;
```
Return the intersection of two queries

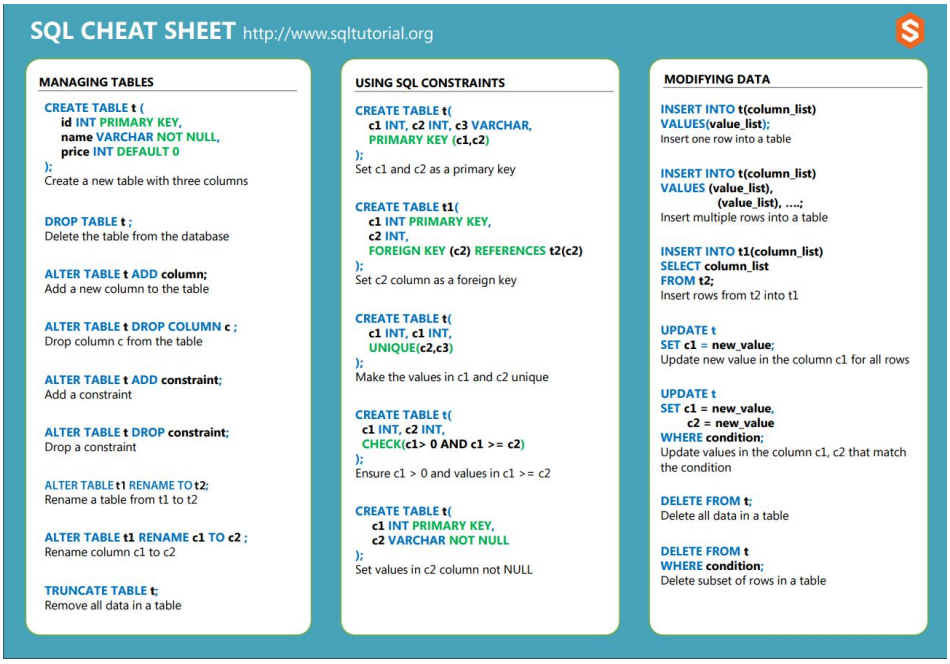
```
SELECT c1, c2 FROM t1
MINUS
SELECT c1, c2 FROM t2;
```
Subtract a result set from another result set

```
SELECT c1, c2 FROM t1
WHERE c1 [NOT] LIKE pattern;
```
Query rows using pattern matching %, _

```
SELECT c1, c2 FROM t
WHERE c1 [NOT] IN value_list;
```
Query rows in a list

```
SELECT c1, c2 FROM t
WHERE  c1 BETWEEN low AND high;
```
Query rows between two values

```
SELECT c1, c2 FROM t
WHERE  c1 IS [NOT] NULL;
```
Check if values in a table is NULL or not



SQL CHEAT SHEET http://www.sqltutorial.org

MANAGING TABLES

```
CREATE TABLE t (
    id INT PRIMARY KEY,
    name VARCHAR NOT NULL,
    price INT DEFAULT 0
);
```
Create a new table with three columns

```
DROP TABLE t ;
```
Delete the table from the database

```
ALTER TABLE t ADD column;
```
Add a new column to the table

```
ALTER TABLE t DROP COLUMN c ;
```
Drop column c from the table

```
ALTER TABLE t ADD constraint;
```
Add a constraint

```
ALTER TABLE t DROP constraint;
```
Drop a constraint

```
ALTER TABLE t1 RENAME TO t2;
```
Rename a table from t1 to t2

```
ALTER TABLE t1 RENAME c1 TO c2 ;
```
Rename column c1 to c2

```
TRUNCATE TABLE t;
```
Remove all data in a table

USING SQL CONSTRAINTS

```
CREATE TABLE t(
    c1 INT, c2 INT, c3 VARCHAR,
    PRIMARY KEY (c1,c2)
);
```
Set c1 and c2 as a primary key

```
CREATE TABLE t1(
    c1 INT PRIMARY KEY,
    c2 INT,
    FOREIGN KEY (c2) REFERENCES t2(c2)
);
```
Set c2 column as a foreign key

```
CREATE TABLE t(
    c1 INT, c1 INT,
    UNIQUE(c2,c3)
);
```
Make the values in c1 and c2 unique

```
CREATE TABLE t(
  c1 INT, c2 INT,
  CHECK(c1> 0 AND c1 >= c2)
);
```
Ensure c1 > 0 and values in c1 >= c2

```
CREATE TABLE t(
    c1 INT PRIMARY KEY,
    c2 VARCHAR NOT NULL
);
```
Set values in c2 column not NULL

MODIFYING DATA

```
INSERT INTO t(column_list)
VALUES(value_list);
```
Insert one row into a table

```
INSERT INTO t(column_list)
VALUES (value_list),
       (value_list), ....;
```
Insert multiple rows into a table

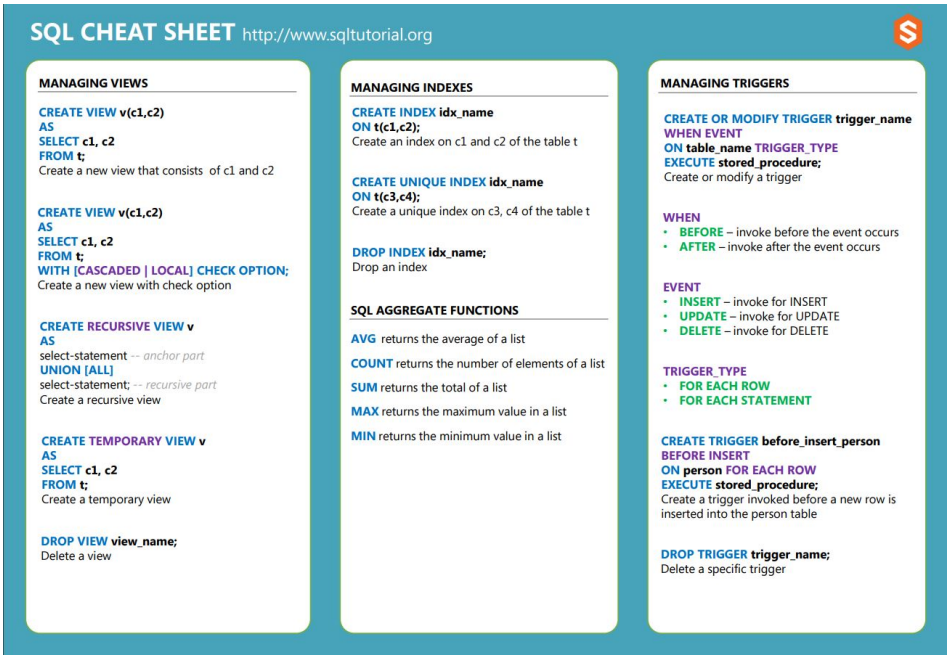
```
INSERT INTO t1(column_list)
SELECT column_list
FROM t2;
```
Insert rows from t2 into t1

```
UPDATE t
SET c1 = new_value;
```
Update new value in the column c1 for all rows

```
UPDATE t
SET c1 = new_value,
    c2 = new_value
WHERE condition;
```
Update values in the column c1, c2 that match the condition

```
DELETE FROM t;
```
Delete all data in a table

```
DELETE FROM t
WHERE condition;
```
Delete subset of rows in a table



Database Schema

Here are the tables you'll find for the database used in the midterm. Your queries will be run against two versions of the database. One of the databases will be much smaller and only contain a subset of the information.

```
CREATE TABLE Titles (
     t_id INTEGER PRIMARY KEY,
     primaryTitle text
     )
CREATE TABLE Ratings (
     r_id INTEGER PRIMARY KEY,
     t_id INTEGER,
     averageRating FLOAT,
     numVotes INTEGER,
     FOREIGN KEY(t_id) REFERENCES Titles(t_id)
     )
CREATE TABLE Names (
     n_id INTEGER PRIMARY KEY,
     primaryName TEXT,
     birthYear INTEGER,
     deathYear INTEGER
     )
CREATE TABLE WorkedOn (
     t_id INTEGER,
     n_id INTEGER,
     category TEXT,
     FOREIGN KEY(t_id) REFERENCES Titles(t_id),
     FOREIGN KEY(n_id) REFERENCES Names(n_id)
     )
```

Scratch area

The following scratch space can be used to help develop and test queries against a database described above. The database used by the exam grader will be different.

```
select primaryTitle, primaryName, (deathYear - birthYear) as age
  FROM Titles T, Names N, WorkedOn W
 WHERE N.n_id = W.n_id
```

```
  AND W.t_id = T.t_id
   AND age > 40
   And primaryTitle like 'A%'
ORDER BY deathYear DESC
```

Execute

Questions For a total of 100 points

SQL Queries 60 points

In this section, you will write SQL queries for the movies schema at the beginning of the exam. Your queries will be tested immediately against two different databases. If your queries output matches the expected output, the displayed answers will be outlined in green. You need green, highlighted output from the query of both databases to get any credit. Your actual score will be determined when your query is tested against a different database but **two** green feedbacks should mean that you are on track to receive full credit.

Titles.1: List two things about all the movies that start with the letter A:
1. List the movie titles reverse alphabetically.
2. For the movies that have a rating greater than or equal to 3, list the number of votes. If the movies has a rating less than 3, list `null`.

Execute 15 points Minimize Output

Names.1: List three things about the names of people whose first name is exactly Alex:
1. Their names alphabetically.
2. Their year of birth.
3. Their year of death. If they are still alive, list their year of death as `null`.

Execute 15 points Minimize Output

Age.1: List three things about all the movies that start with the letter A:
1. List the title of the movie in reversed alphabetically order.
2. List the names of the people who worked on the movie in reversed alphabetically.
3. If the people were at least 40 years old when they died, list their age. But if they are still alive or less than 40 years old when they died, list `null`.

Execute 15 points Minimize Output

Jobs.1: List three things about all the movies:
1. The title of the movie.
2. The average rating of the movie.
3. The name of the movie's producer(s). If there are no producers list `null`.

Execute 15 points Minimize Output

B+ Trees 25 points

Figure 1 is a B+ Tree with index pages that can contain at most two keys and leaf pages that can contain at most two data entries. The root has two keys, 6 and 9, and three pointers.

After inserting the data with key 5, the resulting B+ tree will have the shape of Figure 2. Use the table at the right to fill in the key and data values for the B+ tree after the insertion. You must follow the DBMS book Chapter 10 algorithm for insertion into a B+ tree. (This means, do not do redistribution of nodes which was not covered in class.)

For example, if you believe that after inserting 5 into the B+ tree in Figure 1, then the resulting root′ would be have a key value of 5, then in the table for **Variable name in tree** row **A**, put the value 5. For any values that are empty, enter `null`.

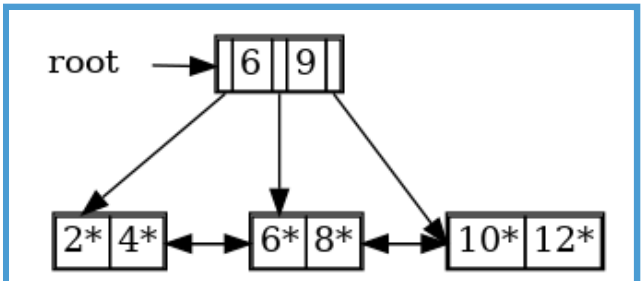


Figure 1. B+ tree before insertion of 5

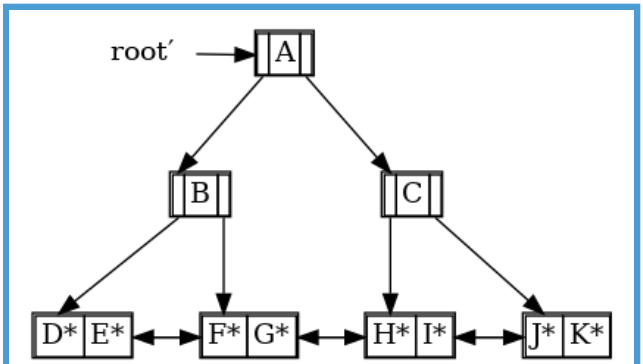


Figure 2. B+ tree after insertion of 5 with variables for key and data values

Btree.After.Insertion.1: Fill in the key and data values for the resulting B+ tree.
Note: Even though, alternate 2 leaf pages include an asterisk in the notation, the pull downs do not include the asterisk.

Variable name in tree	Value
A	
B	
C	
D	
E	
F	
G	
H	
I	
J	
K	

15 points

Btree.Max.Min.1: Fill out the table below with the characteristics of Figure 3 B+ tree which has a maximum of two entries per page.

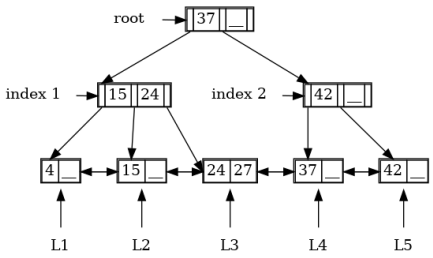


Figure 3. B+ tree

Characteristic of the tree	Value
What is ceiling of the average fanout of the B+ Tree?	
What is the minimum number of unique entries that can be added to increase the tree's height?	
What is the maximum number of unique entries that can be added without increasing the tree's height?	

10 points

I/O Evaluations 15 points

Calculator: You may use this box as a calculator. Just type in any expression that can be evaluated by a JavaScript eval. For example, `2 ** 0.5` will show you the result of the square root of 2 in the red box to the right. Leaving the box empty or filled in will **NOT** affect your grade.

5*11 55

For the following table, assume that each tuple of Sailors is 50 bytes long, that a page can hold 80 Sailors tuples, and there are 250 pages with 67% occupancy. Sailor integer ratings are evenly distributed from 1 to 10, inclusive.

There is a B+ tree and hash index on relation Sailors. There are 2 index links from root to leaf on the search key ⟨ Sailors.rating ⟩. The leaf nodes use alternative 2 to store the data entries. The hash index is on Sailors.id and

requires 1.2 I/Os on average to get to the primary hash bucket.

Also assume that **no** buffering is used as we haven't covered this in class yet.

Evaluation.1: You must correctly answer both questions in the table about the I/Os required for $\sigma_{rating \geq 6} Sailors$ to receive any credit for **Evaluation.1**. There is no partial credit for this question.

How many I/Os are required for $\sigma_{rating \geq 6} Sailors...$	**Answer**
if the B+ tree is used?	
if the B+ tree is not used?	

15 points