**Database Systems, CSCI-GA.2433-011, New York University, Fall 2020**
Instructor: Professor X. Sean Wang (x.sean.wang@nyu.edu)
Course Project Assignment

**Assignment Due time: December 17, 2020 @ 23:00 (CST)**
You need to submit **1 file** to the Classes site. **Read the whole assignment carefully.**

**What to SUBMIT:**
**one Python/Java program source package (you need to include all the libraries you use)**.

**Naming for the file:**
Assuming that your Net ID is abc123 and you are submitting your solution to Homework due 2034-02-15, your files should be named 20340215abc123_proj.zip (of course you need to specify the correct date and the correct Net ID).

**Repeated submission:**
repeated submission on Classes site is allowed as long as it's before due time (and while the submission site remains open), and the **last submission** will be looked at for grading.

**This project is an individual work.** You need to do the work by yourself. Every line of code needs to be done by yourself (except perhaps for the libraries you include). Collaboration is encouraged but only to the extent to asking general questions and getting general answers. No questions about your specific implementation should be asked or answered. No code or code snippets should be shared, and no help in writing the code is allowed.

**A miniature relational database with order**

Given relations whose tuples consist of strings and integers, you are to write a program which will perform the basic operations of relational algebra: selection, projection, join, group by, and sum and average aggregates. The comparators for select and join will be =, <, >, !=, >=, <=.

In addition, you will support importing a vertical bar delimited file into a table, the assignment of the result of a query to a table, sorting a table, and running moving sums and average aggregates on a column of a (probably sorted) table.

Each time you execute a line, you should print the time it took to execute. Outputtofile() command can simply print the table to standardout.

You will support in memory B-trees and hash structures. You are welcome to take those implementations from wherever you can find them, but you must say from where.

You may NOT use any relational algebra or SQL library, but all other libraries are ok. (e.g., pandas_parser: https://github.com/conychou/pandas_parser, HOWEVER: you are only to use the parsing functionality of pandas_parser, not the functions provided by Pandas that the parser uses, which is treated as an SQL library and not allowed (see above). *Please note that this parser is not required for this project.*).

We will run your programs on test cases of our choosing. For ease of parsing there will be one operation per line. Comments begin with //. For example,

**数据库系统，CSCI-GA.2433-011，纽约大学，2020年秋季**

讲师： 王教授 （x.sean.wang@nyu.edu）

课程项目作业

**分配到期时间：2020 年 12 月 17 日 @ 23：00 （CST）**

您需要将 **1 个文件提交** 到"Classes"网站。 **仔细阅读整个作业。**

**提交什么：**

**一个Python/Java程序源包（您需要包括您使用的所有库）**。

**文件的命名：**

Assuming that your Net ID is abc123 and you are submitting your solution to Homework due 2034-02-15, your files should be named 20340215abc123_proj.zip (of course you need to specify the correct date and the correct Net ID).

**重复提交：**

只要在到期时间之前（并且提交网站保持打开状态），允许重复提交 **，并且将查看** 最后一个提交进行评分。

**这个项目是一项单独的工作。** 你需要自己做。每行代码都需要自己完成（可能包括的库除外）。鼓励协作，但只能提出一般问题和获得一般答案。不应询问或回答有关具体实现的问题。不应共享任何代码或代码片段，也不允许在编写代码时提供帮助。

**具有订单的微型关系数据库**

给定其元组由字符串和整数组成的关系，您将编写一个程序，该程序将执行关系代数的基本操作：选择、投影、联接、分组和平均聚合。选择和联接的比较器将为 =，<，>， ! =，>=，<=。

此外，您将支持将垂直条分隔文件导入到表中，将查询结果分配给表，对表进行排序，并在（可能排序的）表的列上运行移动总和和平均聚合。

每次执行行时，都应打印执行时间。输出文件（ ） 命令可以简单地将表打印到标准输出。

您将支持内存 B 树和哈希结构。欢迎您从任何能找到的地方获得这些实现，但您必须从何处进行。

您的程序应该用 Python 或 Java 编写。您不能使用任何关系代数或 SQL 库，但所有其他库都正常（例如，pandas_parser： https://github.com/conychou/pandas_parser，HOWEVER 但是：您只使用 pandas_parser 的解析功能，而不是解析器使用的 Pandas 提供的函数，该函数被视为 SQL 库，不允许使用（见上文）。*请注意，此项目不需要此解析器。*）

我们将在我们选择的测试用例上运行您的程序。为了便于分析，每行将有一个操作。注释以 //开始。例如，

```
R := inputfromfile(foo)     // import vertical bar delimited foo, first line
   // has column headers. Suppose they are A, B, E
R1 := select(R, (A > 5) or (B < 3))
   // select rows of R whose A val is greater than 5 or B < 3
R2 := project(R1, A, B, E)     // R2 gets columns A, B, and E of the rows of R1
R3 := avg(R1, B)     // average value of B
R4 := sumgroup(R1, A, B)     // select B, sum R1.A, group by B
R5 := sumgroup(R1, A, B, E)     // select B, E, sum R1.A, group by B, E
R6 := avggroup(R1, A, B)     // avg R1.A group by B
S := inputfromfile(bar)     // suppose column headers are B, C, D
T := join(R, S, R.B = S.C)     // T has all the columns of R and S but the
   // column headers are prefaced by the table they came from, e.g., R_A, R_B,
   // R_E, S_B, S_C, S_D
T1 := join(R1, S, R.B > S.C)
T2 := sort(T1, S_C)     // sort T1 by S_C
T2prime := sort(T1, R_A, S_C)     // sort T1 by R_A, S_C (in that order)
T3 := movavg(T2, R_B, 3)     // perform the three item moving average of T2
   // on column R_B. This will be as long as R_B with the three way
   // moving average of 4 8 9 7 being 4 6 7 8
T4 := movsum(T2, R_B, 5)     // perform the five item moving sum of T2
   // on column R_B
T4 := movsum(T2, R_B, 3)     // perform the three item moving sum of T2
Q1 := select(R, B = 5)     // there is no index to use
Btree(R, B)     // create an index on R based on column B
   // Equality selections and joins on R should use the index.
Q2 := select(R, B = 5)     // this should use the index
Q3 := select(R, A = 7)     // this should not use an index Hash(R,A)
Q4 := select(R, A = 7)     // this should use the hash index
Q5 := concat(Q4, Q2)     // concatenate the two tables (must have the same schema)
   // Duplicate rows may result (though not with this example).
outputtofile(Q5, bar)     // This should output the table Q4 with vertical bar separators
```

Some constraints to make your life easier:

1. The joins are on single columns
2. All data is in main memory (you may assume there at most 200,000 rows and 20 columns to a table)
3. There is no group by for moving sums and averages

Here is an example of the first few lines of a sales file:

saleid|itemid|customerid|storeid|time|qty|pricerange 45|item133|customer2|store63|49|23|outrageous 658|item75|customer2|store89|46|43|outrageous 149|item103|customer2|store23|67|2|cheap 398|item82|customer2|store41|3|27|outrageous 147|item81|customer2|store4|92|11|outrageous 778|item75|customer160|store72|67|17|supercheap 829|item112|customer2|store70|63|43|supercheap 101|item105|customer2|store9|74|28|expensive 940|item62|customer2|store90|67|39|outrageous 864|item119|customer12|store38|67|49|outrageous 288|item46|customer2|store95|67|26|outrageous 875|item83|customer59|store56|59|20|outrageous 783|item86|customer180|store29|67|46|outrageous 289|item16|customer2|store95|92|2|cheap 814|item101|customer2|store45|49|41|outrageous 572|item92|customer59|store91|63|31|cheap 428|item114|customer51|store29|42|15|outrageous

Full example files can be found in the assignment package: sales1.txt (1,000 rows) and sales2.txt (100,000 rows).

```
R := inputfromfile(foo)     // import vertical bar delimited foo, first line
   // has column headers. Suppose they are A, B, E
R1 := select(R, (A > 5) or (B < 3))
   // select rows of R whose A val is greater than 5 or B < 3
R2 := project(R1, A, B, E)     // R2 gets columns A, B, and E of the rows of R1
R3 := avg(R1, B)     // average value of B
R4 := sumgroup(R1, A, B)     // select B, sum R1.A, group by B
R5 := sumgroup(R1, A, B, E)     // select B, E, sum R1.A, group by B, E
R6 := avggroup(R1, A, B)     // avg R1.A group by B
S := inputfromfile(bar)     // suppose column headers are B, C, D
T := join(R, S, R.B = S.C)     // T has all the columns of R and S but the
   // column headers are prefaced by the table they came from, e.g., R_A, R_B,
   // R_E, S_B, S_C, S_D
T1 := join(R1, S, R.B > S.C)
T2 := sort(T1, S_C)     // sort T1 by S_C
T2prime := sort(T1, R_A, S_C)     // sort T1 by R_A, S_C (in that order)
T3 := movavg(T2, R_B, 3)     // perform the three item moving average of T2
   // on column R_B. This will be as long as R_B with the three way
   // moving average of 4 8 9 7 being 4 6 7 8
T4 := movsum(T2, R_B, 5)     // perform the five item moving sum of T2
   // on column R_B
T4 := movsum(T2, R_B, 3)     // perform the three item moving sum of T2
Q1 := select(R, B = 5)     // there is no index to use
Btree(R, B)     // create an index on R based on column B
   // Equality selections and joins on R should use the index.
Q2 := select(R, B = 5)     // this should use the index
Q3 := select(R, A = 7)     // this should not use an index Hash(R,A)
Q4 := select(R, A = 7)     // this should use the hash index
Q5 := concat(Q4, Q2)     // concatenate the two tables (must have the same schema)
   // Duplicate rows may result (though not with this example).
outputtofile(Q5, bar)     // This should output the table Q4 with vertical bar separators
```

一些限制因素，让你的生活更轻松：

1. 联接在单个列上
2. 所有数据都在主内存中（您可能假设表中最多有 200，000 行和 20 列）
3. 移动总和和平均值没有按组

下面是销售文件的前几行示例：

saleid|itemid|customerid|storeid|time|qty|pricerange 45|item133|customer2|store63|49|23|outrageous 658|item75|customer2|store89|46|43|outrageous 149|item103|customer2|store23|67|2|cheap 398|item82|customer2|store41|3|27|outrageous 147|item81|customer2|store4|92|11|outrageous 778|item75|customer160|store72|67|17|supercheap 829|item112|customer2|store70|63|43|supercheap 101|item105|customer2|store9|74|28|expensive 940|item62|customer2|store90|67|39|outrageous 864|item119|customer12|store38|67|49|outrageous 288|item46|customer2|store95|67|26|outrageous 875|item83|customer59|store56|59|20|outrageous 783|item86|customer180|store29|67|46|outrageous 289|item16|customer2|store95|92|2|cheap 814|item101|customer2|store45|49|41|outrageous 572|item92|customer59|store91|63|31|cheap 428|item114|customer51|store29|42|15|outrageous

可以在分配包中找到完整的示例文件：销售 1.txt（1，000 行）和销售 2.txt（100，000 行）。