Milestone N
CS165, Fall 2017
John Q. Student

1. Introduction
Summarise in one paragraph what you plan to develop in this milestone. In terms of technical detail this section should be easily understandable to someone with familiarity with data systems. It shouldn't make assumptions about what they know about the milestone but it can make assumptions about what they know about CS. For example, don't assume that the reader knows we are building a column oriented data system, but you can assume that they know what that is.

2. Problems Tackled
List the different problems and challenges that you address in this milestone with one bullet per each problem.

- **Example: [Database Catalog]** We need a fast way to find to keep track of tables and columns and identify the tables and columns referred to by a query.

3. Technical Description
This should be a detailed overview of the problems and solutions tackled in this milestone, in the form of a list. Your descriptions should be in a detail that allows the reader to implement the design in code. For each problem/solution pair, include the following three bullets:

- **Problem framing:** what is the problem you are tackling, why does it arise, and why is it important to solve? For example, if you are adding core functionality, why is it needed? If you are trying to improve performance, what is the bottleneck you are trying to alleviate?
- **High-level solution:** what is the solution that you will attempt, what is the intuition for why this solution should work well, and what performance characteristics do you expect? This should be described at the level of data structures and their performance characteristics.
- **Deeper details:** elaborate on the technical details of your design that you think are most interesting, describe how these design decisions impact performance. Depending on the milestone, you may include architecture diagrams, flowcharts, and/or pseudocode for clarification. The choice between the three depends on the nature of the problem. More technical/detailed/algorithmic problems should use pseudo-code, higher level problems should use flow charts and architecture plots. You can also use drawings to visually describe ideas. For example, if you had a drawing that shows the data layout of your index; a flow chart that shows how your system makes some decision; or a concurrency algorithm for avoiding race conditions in a multi-threaded result set merge, that information/chart/drawing/figure would go here.
Example: Database Catalog

- **Problem framing:** A database may consist of numerous tables (and columns within each table), and so we need a fast way to find which tables and columns a particular query is targeting.
- **High-level solution:** we use a hash table to map from table and column names to handles on the corresponding data structures. A hash table allows finding key-value pairs in constant time, and so this allows finding which columns a query is targeting quickly regardless of the number of tables and columns in the database.
- **Deeper details:** we use fat nodes for the hash table in order to achieve better cache locality. The size of each node is set to be the size of a cache line. We perform deletes in the fat nodes by...

4. Challenges

List possible challenges that you expect to face during the implementation of this milestone.

---

*The following (non-exhaustive) list gives examples of problems that you will most likely encounter at each of the milestones:*

**Milestone 1:** Why do we need a variable pool, and how do you design yours? How did you design the database catalog?

**Milestone 2:** Why do we need indexes, and which data structure do you use to support them and why? What are the problems that arise having multiple indexes per table and how do you address them? How do you keep track of existing indexes? What is the layout of each B-tree node that you used?

**Milestone 3:** Why are scans problematic in terms of performance, and what can we do to speed them up (e.g., SIMD, multiple threads, shared scans)? If you use multiple threads, how do you divide the work among them, and how do you prevent race conditions, starvation and deadlocks? If you use SIMD, which commands did you use and to what effect?

**Milestone 4:** Why do we need joins in a database, what join algorithm did you use, and why did you choose it?

**Milestone 5:** What are the difficulties associated with updating data, and how do you deal with them?