LAB 2

Emmanuel J Lopez 1005407

Nicholas Gandhi Peradidjaya 1005295

1. Filter and Join

Design Decision: The join was implemented using a nested loop via the two child iterators. The second child is fully iterated through before rewinding for the next iteration of the first child.

2. Aggregates

Design Decision: Aggregate without grouping was implemented using a count and/or a value variable to store the totals for the respective operator. Aggregate with grouping was implemented via a Hashmap to store the counts and/or values for each group calculated according to their respective operator.

3. HeapFile

HeapFile.java
In the `insertTuple` method, we handled the scenario where no existing page has empty slots. This method involves creating a new page, assigning it a page ID, inserting the tuple into it, and updating the file by writing the new page data.

HeapPage.java
The `markSlotUsed` method involves bitwise operations to set or clear a specific bit in the header byte array.

4. Insertion and Deletion

In the `BufferPool.java`, after inserting or deleting a tuple, we check if the buffer pool contains the affected page. If not and the buffer pool size exceeds the maximum number of pages, it evicts a page using the evictPage method. Then, it updates the buffer pool with the modified page.

5. Page Eviction (BufferPool.java)

LRU eviction policy was implemented. The `bufferPool` hash map was iterated through in order, to find the first clean page. When tuples are inserted, the affected pages are removed and re-inserted into the `bufferPool` hashmap causing them to move to the end of the hashmap as the 'most recently used'.