Exercise session 2 : Query Execution

Exercise 1: Query Processing Models Comparison
Explain the key differences between the following query processing models:
- Iterator (Tuple-at-a-time)
- Block-oriented (Column-at-a-time)
- Vectorization

Given the SQL query below, give one possible plan for execution and describe how each of these models would execute your plan.

```
SELECT A.id, B.value
FROM A, B
WHERE A.id = B.id
  AND B.c = B.d
  AND B.value > 100;
```

Which model would have the best performance if A and B both have billions of elements? Why?

Exercise 2: Implementing Query Operators in the Iterator Model
Consider the following class

```
class Operator:
    def next(self):
        pass #todo implement this in subclasses
```

Provide pseudo-code that defines the Project, Select and Join operators in the Iterator model.
Hint: Use constructors to store state that should be maintained across different next() calls, like the Select predicate or the Join key.
Hint 2: You may use dictionaries for the Join operator. Assume that all joins are equi-joins on two relations using a single attribute from each relation as the join attribute.

Write a short explanation of the inefficiencies of this approach and how they could be mitigated using a different processing model.

In one sentence and at a very high level, what would you change to make it Block-oriented (with a column granularity)?

Exercise 3: First look at performance problems
Consider the following SQL query:

```
SELECT E.Name, D.Budget
FROM Employees E, Departments D
WHERE E.department_id = D.id
AND D.Budget > 1000000
```

Employees has 10,000 rows and Departments has 20.

Give two ways to execute this query using the Select, Join and Project operators.

Which is likely to be more resource-efficient? Why?