Assignmen3 Report

Implement Explain operantion:

Outline:

STEP 1: 新增變數 is_explain 到 QueryData， 用來確認 EXPLAIN 是否有出現在 Query 裡面
STEP 2: 在 Lexer 中新增 keyword: "explain"
STEP 3: 在 Parser 裡面的 queryCommand 看看能不能吃到 explain 這個 keyword 如果可以就把 is_explain 設為 true
STEP 4: 將 Explain plan 新增到 BasicQueryPlanner的最底部
STEP 5: 新增一個新的 class : ExplainPlan
STEP 6: 新增一個新的 class : ExplainScan
STEP 7: 分別於各個 plan 中以遞迴方式回傳 result 的文字

STEP 1: 新增變數 is_explain 到 QueryData， 用來確認 EXPLAIN 是否有出現在 Query 裡面

```
// TODO: add is_explain
private boolean is_explain;


// TODO: check if explain had been called
this.is_explain = is_explain;
```

STEP 2: 在 Lexer 中新增 keyword: "explain"

```
private void initKeywords() {
    keywords = Arrays.asList("select", "from", "where", "and", "insert",
            "into", "values", "delete", "drop", "update", "set", "create", "table",
            "int", "double", "varchar", "view", "as", "index", "on",
            "long", "order", "by", "asc", "desc", "sum", "count", "avg",
            "min", "max", "distinct", "group", "add", "sub", "mul", "div",
            "using", "hash", "btree", "explain");
}
```

STEP 3: 在 Parser 裡面的 queryCommand 看看能不能吃到 explain 這個 keyword 如果可以就把 is_explain 設為 true

```
// TODO
boolean is_explain = false;
if (lex.matchKeyword("explain")) {
    lex.eatKeyword("explain");
    is_explain = true;
}
```

STEP 4: 將 Explain plan 新增到 BasicQueryPlanner 的最底部

```
// Step 7: Add a Explain Plan if specified
if (data.is_explainFn())
    p = new ExplainPlan(p);
```

STEP 5: 新增一個新的 class : ExplainPlan

5-1 宣告一個 explain_query_plan (這是結果) 呼叫 explained 遞迴下去執行，最後在把 ExplainScan 回傳回去

```
/
@Override
public Scan open() {
    Scan s1 = p1.open(); // s1 is explain's scan
    String explain_query_plan = "";
    String a = explained(explain_query_plan, 0, false);
    //System.out.println("1: " + a);
    return new ExplainScan(s1, a, schema());
}
```

5-2 實作 ExplainPlan 裡面的 explained 他會去遞迴呼叫下一層 p1 的 explained 其中 whiteSpace 是用來看看下面那一層前面要幾個空白的，is_product 是在 productplan 那裏才會用到

```
@Override
public String explained(String explain_query_plan, int whiteSpace, boolean is_product
    /*String ans = "";
    for (int i=0; i<explain_query_plan.size(); i++) {
        String whiteSpace = "";
        String temp = "";
        temp = explain_query_plan.get(explain_query_plan.size() - 1 - i);
        for (int j=0; j<i*4; j++) whiteSpace += " ";

        ans += whiteSpace;
        ans += temp;
        ans += '\n';
    }
    System.out.println(ans);*/
    //System.out.println(explain_query_plan);
    return p1.explained(explain_query_plan, whiteSpace, false);
}
```

5-3 用一個 Schema 來存我們的答案 其中只有一個 field 名字是 query-plan, type 是 varchar(500)

```
@Override
public Schema schema() {
    Schema schema = new Schema();
    schema.addField("query-plan", Type.VARCHAR(500));
    return schema;
}
```

5-4 因為他只有一個 record 所以回傳 1

```
@Override
public long recordsOutput() {
    return 1;
}
```

STEP 6: 新增一個新的 class : ExplainScan

6-1 這裡去呼叫 s.beforeFirst 這樣才會呼叫到 s.getvalue 接著我們看看實際上現在的 records 是多少，把他加到答案裡面

```
public ExplainScan(Scan s, String a, Schema schema) {
    this.a = a;
    this.schema = schema;
    s.beforeFirst(); // explainscan.beforefirst
    while (s.next()) {
        recs++;
    }
    s.close();
    //System.out.println("3: " + this.a);
    this.a = a + "\nActual #recs: " + recs;
    flag = true;
}
```

6-2 在 beforeFirst 的時候要把 flag 設為 true 這樣才可以跑 next

```
@Override
public void beforeFirst() {
    flag = true;
}
```

6-3 如果她還沒跑過 flag 是 true 所以可以跑，要把 flag 設為 false 並且回傳 true，否則就回傳 false

```
@Override
public boolean next() {
    if (flag) {
        flag = false;
        return true;
    } else
        return false;
}
```

6-4 看看 field name 是不是 query-plan 如果是的話，那就回傳我們的答案

```
@Override
public Constant getVal(String fldName) {
    //System.out.println("2: " + a);
    if (fldName.equals("query-plan")) {
        return new VarcharConstant(a);
    } else
        throw new RuntimeException("field " + fldName + " not found.");
}
```

STEP 7: 分別於各個 plan 中以遞迴方式回傳 result 的文字；其中我們考慮到如果有多個 table plan 的情況下，product plan 回傳說明文字有些許變動

```
public String explained(String explain_query_plan, int whiteSpace, boolean is_productplan) {
    String temp_string = "";
    if (!is_productplan) {
        for(int i=0; i<whiteSpace*4; i++) temp_string += " ";

        temp_string += "->ProductPlan: (#blks=";
        temp_string += this.blocksAccessed();
        temp_string += ", #recs=";
        temp_string += this.recordsOutput();
        temp_string += ")";
        temp_string += "\n";

        explain_query_plan += temp_string;
    }
    int temp = 0;

    if (!is_productplan) temp = whiteSpace + 1;
    else temp = whiteSpace;

    String string_p1 = p1.explained("", temp, true);
    String string_p2 = p2.explained("", temp, true);

    if (!is_productplan)
        return explain_query_plan + string_p1 + string_p2;
    else
        return string_p1 + string_p2;
}
```

7-1 建立 is_productplan 作為判斷是否該 plan 為最上層之 plan 的 flag

7-2-1 如為最上層之 product plan，則將 temp_string 加上 "ProductPlan... " 的文字敘述

7-2-2 如為另外兩個 table 所組成的product plan，則呼叫下面兩個 plan，重複直到抵達 TablePlan.explained( )，其中為考慮回傳格式之問題，在對 whiteSpace做了處理

```
if (!is_productplan) temp = whiteSpace + 1;
else temp = whiteSpace;
```

7-3 TablePlan.explained( ) 即會回傳各自之資訊回 ProductPlan.explained( )，於此組裝所有資訊以回傳至上一層，以供 console 提供 EXPLAIN 之資訊

Explain Result:

1. A query accessing single table with WHERE
   EXPLAIN SELECT w_tax FROM warehouse WHERE w_tax > 0.05

```
query-plan
---------------------------------------------------------------
->ProjectPlan: (#blks=2, #recs=1)
    ->SelectPlan pred: (w_tax>0.05) (#blks=2, #recs=1)
        ->TablePlan on (warehouse) (#blks=2, #recs=1)

Actual #recs: 1
```

2. A query accessing multiple tables with WHERE
   EXPLAIN SELECT w_city, w_tax FROM district, warehouse WHERE w_tax > d_tax

```
query-plan
-------------------------------------------------------------
->ProjectPlan: (#blks=22, #recs=10)
    ->SelectPlan pred: (w_tax>d_tax) (#blks=22, #recs=10)
        ->ProductPlan: (#blks=22, #recs=10)
            ->TablePlan on (district) (#blks=2, #recs=10)
            ->TablePlan on (warehouse) (#blks=2, #recs=1)

Actual #recs: 6
```

3. A query with ORDER BY
   EXPLAIN SELECT d_id, w_id FROM district, warehouse WHERE d_w_id = w_id ORDER BY w_id

```
query-plan
---------------------------------------------------------------

->ProjectPlan: (#blks=22, #recs=10)
        ->SelectPlan pred: (d_w_id=w_id) (#blks=22, #recs=10)
            ->ProductPlan: (#blks=22, #recs=10)
                ->TablePlan on (district) (#blks=2, #recs=10)
                ->TablePlan on (warehouse) (#blks=2, #recs=1)

Actual #recs: 10
```

4. A query with GROUP BY and at least one aggregation function (MIN, MAX, COUNT, AVG... etc.)
   EXPLAIN SELECT COUNT(d_id) FROM district, warehouse WHERE d_w_id = w_id GROUP BY w_id

```
query-plan
-----------------------------------------------------------------
->ProjectPlan: (#blks=2, #recs=1)
    ->GroupbyPlan: (#blks=2, #recs=1)
        ->SortPlan: (#blks=2, #recs=10)
            ->SelectPlan pred: (d_w_id=w_id) (#blks=22, #recs=10)
                ->ProductPlan: (#blks=22, #recs=10)
                    ->TablePlan on (district) (#blks=2, #recs=10)
                    ->TablePlan on (warehouse) (#blks=2, #recs=1)

Actual #recs: 1
```

5. A query of single table with MAX( )
   EXPLAIN SELECT MAX(i_price) FROM item

```
query-plan
-------------------------------------------------------------
->ProjectPlan: (#blks=6251, #recs=1)
    ->GroupbyPlan: (#blks=6251, #recs=1)
        ->SelectPlan pred: () (#blks=6251, #recs=100000)
            ->TablePlan on (item) (#blks=6251, #recs=100000)

Actual #recs: 1
```