Steps for Project

Step 1: Get the 6 arguments for the Phi Operator

```
if user wants to get the argument from a file:
    read the file line by line
else:
    manually input the 6 arguments
```

Step 2: Connect to the database

```
import psycopg2
import tabulate

# connect to the database
conn = psycopg2.connect(host = 'localhost', dbname = 'postgres', user = 'postgres', password = 'password', port = 5432)

# test that the connection works
cursor = conn.cursor()
cursor.execute('SELECT * FROM sales')

result = []
for row in cursor:
    result.append(row)

output = tabulate.tabulate(result, headers = "keys", tablefmt = "psql")
print(output)
```

- To import psycopg2 and tabulate, you need a package manager
- Modify the parameters of psycopg2.connect() so that it works for your database

Step 3: Generate the code for creating data structure of the H table

```
class H:
    groupingAttribute1 = Null
    groupingAttribute2 = Null
    ...
    groupingAttributeM = Null
    aggregate1 = Null
    aggregate2 = Null
    ...
    aggregateN = Null
```

- the values of the grouping attributes and aggregates will depend on the arguments for the Phi operator
- write the data structure of the H table to a file like _generated.py

Step 4: Generate the code that implements a modified version of Algorithm 3.1 of the second research paper

```
# first scan of the sales table to initialize the H table
for each ROW in sales:
    if ROW contains a new combination of grouping attributes:
        - Add the grouping attributes to the H table
        - Initialize the 0th grouping variable
    else
        - update the 0th grouping variable
```

- The aggregates of the 0th grouping variable are aggregates like avg(quant) or max(length) as opposed to aggregates like avg(X.quant) or max(Y.length)

```
# scan the sales table N times to compute the aggregates of the N grouping variables
N = number of grouping variables
for i = 1, ... N:
    for each ROW in sales:
        if the row satisfies the defining condition of the ith grouping variable:
            - get the row in the H table that matches the grouping attributes of ROW
            - update the row's ith grouping variable aggregates
```

- "This algorithm performs N + 1 scans on the base relation (sales). On scan i, it computes the aggregates of [the ith grouping variable]."
- Note that the algorithm here is not the same as Algorithm 3.1. This algorithm implements MF queries while Algorithm 3.1 implements EMF queries.