Designing Data-Intensive Applications

LAB1 - OLAP

Name	ID
Mohamed Ali Riyad	20011457
Fareeda Mohamed Ragab	19016154
Mariam Hossam Ali	20011882
Mostafa Mohamed Galal	20011945

TPC Data Generation

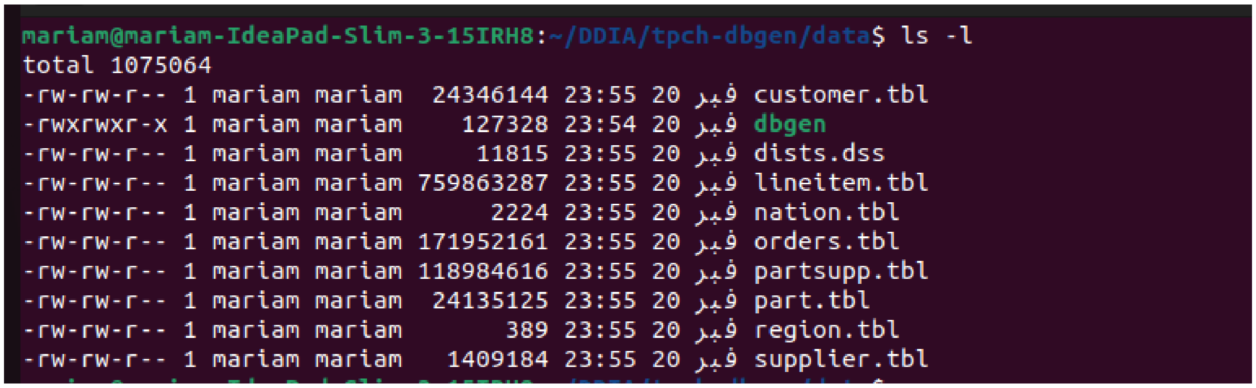
1. Download and compile TPC-H tools:

```
$ git clone https://github.com/electrum/tpch-dbgen.git
$ cd tpch-dbgen
$ make -f makefile.suite
```

2. Run dbgen to generate data for a scale factor of 1 (1 GB):

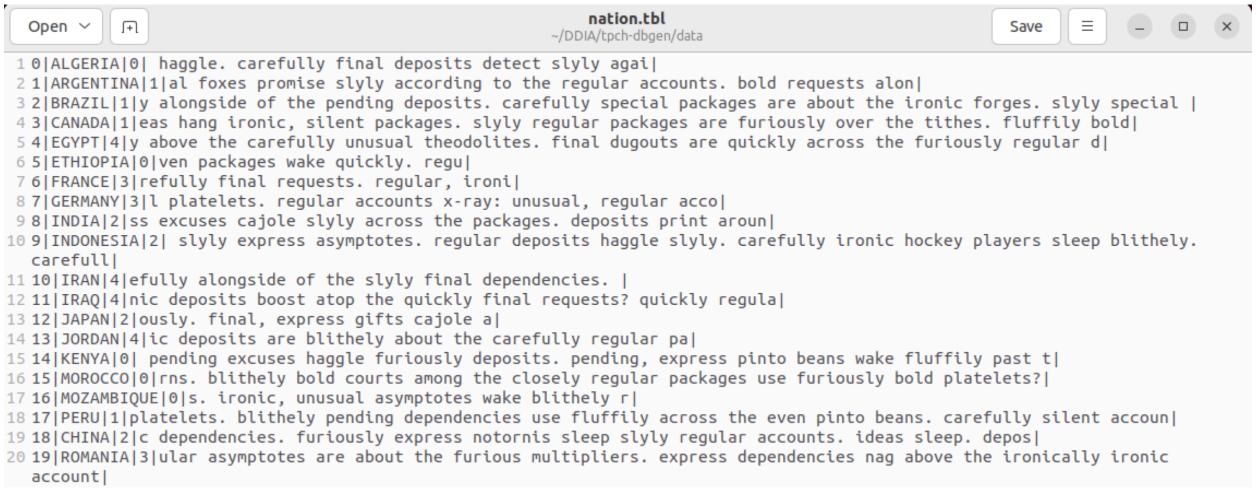
```
$ mkdir data
$ cd data
$ cp ../dbgen .
$ cp ../dists.dss .
$ ./dbgen -s 1
```

3. Output files:



```
mariam@mariam-IdeaPad-Slim-3-15IRH8:~/DDIA/tpch-dbgen/data$ ls -l
total 1075064
-rw-rw-r-- 1 mariam mariam  24346144 23:55 20 فبر customer.tbl
-rwxrwxr-x 1 mariam mariam    127328 23:54 20 فبر dbgen
-rw-rw-r-- 1 mariam mariam     11815 23:55 20 فبر dists.dss
-rw-rw-r-- 1 mariam mariam 759863287 23:55 20 فبر lineitem.tbl
-rw-rw-r-- 1 mariam mariam      2224 23:55 20 فبر nation.tbl
-rw-rw-r-- 1 mariam mariam 171952161 23:55 20 فبر orders.tbl
-rw-rw-r-- 1 mariam mariam 118984616 23:55 20 فبر partsupp.tbl
-rw-rw-r-- 1 mariam mariam  24135125 23:55 20 فبر part.tbl
-rw-rw-r-- 1 mariam mariam       389 23:55 20 فبر region.tbl
-rw-rw-r-- 1 mariam mariam   1409184 23:55 20 فبر supplier.tbl
```

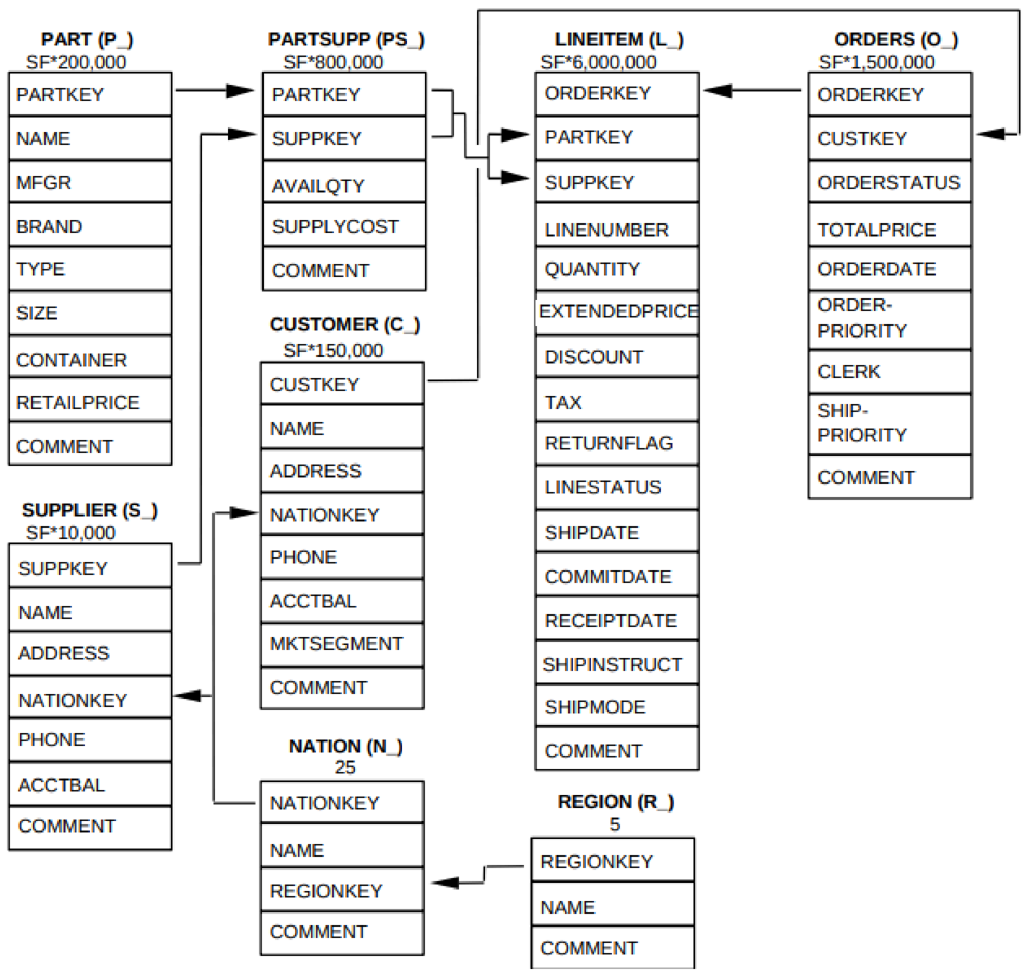
4. Sample file “nation.tbl”:



```
0|ALGERIA|0| haggle. carefully final deposits detect slyly agai|
1|ARGENTINA|1|al foxes promise slyly according to the regular accounts. bold requests alon|
2|BRAZIL|1|y alongside of the pending deposits. carefully special packages are about the ironic forges. slyly special |
3|CANADA|1|eas hang ironic, silent packages. slyly regular packages are furiously over the tithes. fluffily bold|
4|EGYPT|4|y above the carefully unusual theodolites. final dugouts are quickly across the furiously regular d|
5|ETHIOPIA|0|ven packages wake quickly. regu|
6|FRANCE|3|refully final requests. regular, ironi|
7|GERMANY|3|l platelets. regular accounts x-ray: unusual, regular acco|
8|INDIA|2|ss excuses cajole slyly across the packages. deposits print aroun|
9|INDONESIA|2| slyly express asymptotes. regular deposits haggle slyly. carefully ironic hockey players sleep blithely. carefull|
10|IRAN|4|efully alongside of the slyly final dependencies. |
11|IRAQ|4|nic deposits boost atop the quickly final requests? quickly regula|
12|JAPAN|2|ously. final, express gifts cajole a|
13|JORDAN|4|ic deposits are blithely about the carefully regular pa|
14|KENYA|0| pending excuses haggle furiously deposits. pending, express pinto beans wake fluffily past t|
15|MOROCCO|0|rns. blithely bold courts among the closely regular packages use furiously bold platelets?|
16|MOZAMBIQUE|0|s. ironic, unusual asymptotes wake blithely r|
17|PERU|1|platelets. blithely pending dependencies use fluffily across the even pinto beans. carefully silent accoun|
18|CHINA|2|c dependencies. furiously express notornis sleep slyly regular accounts. ideas sleep. depos|
19|ROMANIA|3|ular asymptotes are about the furious multipliers. express dependencies nag above the ironically ironic account|
```

OLTP Schema

Figure 2: The TPC-H Schema



DDL

```
DROP TABLE IF EXISTS lineitem, orders, partsupp, supplier, part, customer,
nation, region;

CREATE TABLE region (
    regionkey INT PRIMARY KEY,
    name VARCHAR(50),
    comment TEXT
);

CREATE TABLE nation (
```

```sql
    nationkey INT PRIMARY KEY,
    name VARCHAR(50),
    regionkey INT,
    comment TEXT,
    FOREIGN KEY (regionkey) REFERENCES region(regionkey)
);

CREATE TABLE customer (
    custkey INT PRIMARY KEY,
    name VARCHAR(100),
    address VARCHAR(255),
    nationkey INT,
    phone VARCHAR(20),
    acctbal DECIMAL(10,2),
    mktsegment VARCHAR(50),
    comment TEXT,
    FOREIGN KEY (nationkey) REFERENCES nation(nationkey)
);

CREATE TABLE orders (
    orderkey INT PRIMARY KEY,
    custkey INT,
    orderstatus CHAR(1),
    totalprice DECIMAL(12,2),
    orderdate DATE,
    orderpriority VARCHAR(20),
    clerk VARCHAR(50),
    shippriority INT,
    comment TEXT,
    FOREIGN KEY (custkey) REFERENCES customer(custkey)
);

CREATE TABLE supplier (
    suppkey INT PRIMARY KEY,
    name VARCHAR(100),
    address VARCHAR(255),
    nationkey INT,
    phone VARCHAR(20),
    acctbal DECIMAL(10,2),
    comment TEXT,
    FOREIGN KEY (nationkey) REFERENCES nation(nationkey)
);
```

```sql
CREATE TABLE part (
    partkey INT PRIMARY KEY,
    name VARCHAR(100),
    mfgr VARCHAR(50),
    brand VARCHAR(50),
    type VARCHAR(50),
    size INT,
    container VARCHAR(20),
    retailprice DECIMAL(10,2),
    comment TEXT
);

CREATE TABLE partsupp (
    partkey INT,
    suppkey INT,
    availqty INT,
    supplycost DECIMAL(10,2),
    comment TEXT,
    PRIMARY KEY (partkey, suppkey),
    FOREIGN KEY (partkey) REFERENCES part(partkey),
    FOREIGN KEY (suppkey) REFERENCES supplier(suppkey)
);
```

```sql
CREATE TABLE lineitem (
    orderkey INT,
    partkey INT,
    suppkey INT,
    linenumber INT,
    quantity INT,
    extendedprice DECIMAL(12,2),
    discount DECIMAL(5,2),
    tax DECIMAL(5,2),
    returnflag CHAR(1),
    linestatus CHAR(1),
    shipdate DATE,
    commitdate DATE,
    receiptdate DATE,
    shipinstruct VARCHAR(50),
    shipmode VARCHAR(20),
    comment TEXT,
    PRIMARY KEY (orderkey, linenumber),
    FOREIGN KEY (orderkey) REFERENCES orders(orderkey),
    FOREIGN KEY (partkey) REFERENCES part(partkey),
    FOREIGN KEY (suppkey) REFERENCES supplier(suppkey)
);
```

Data Insertion into MySQL

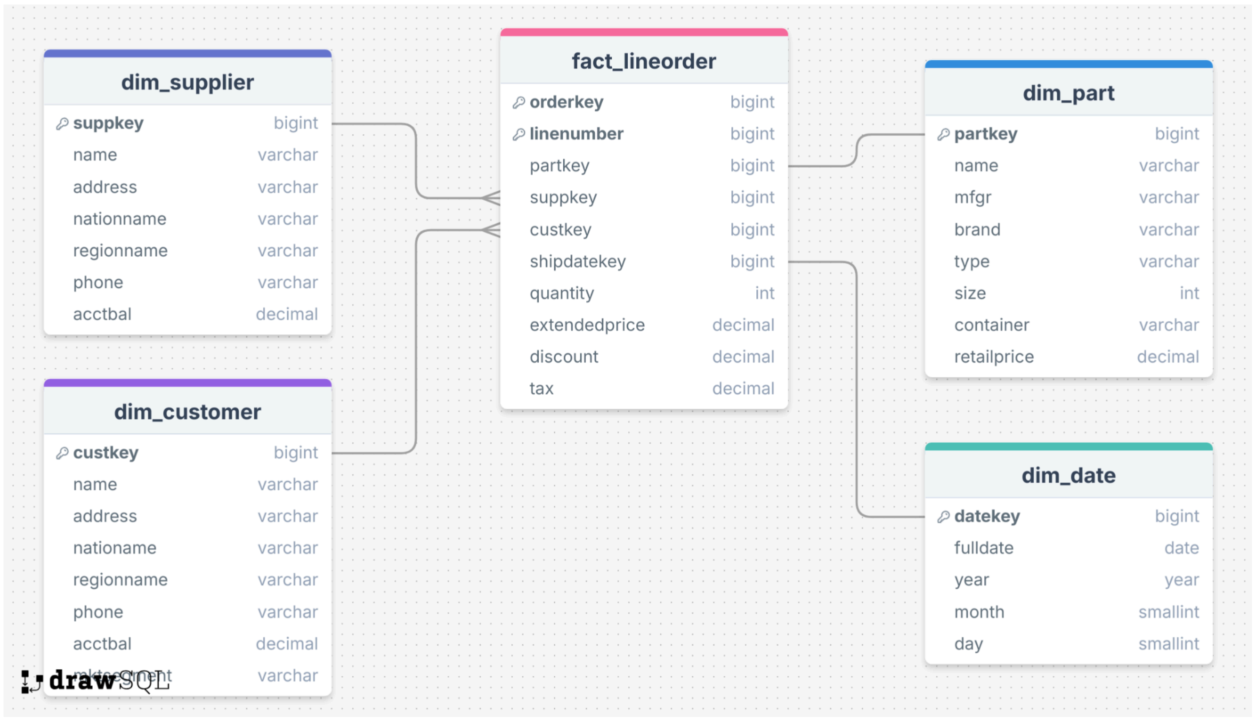
1. Create MySQL database “tpch” with the above tables.
2. Load data from files into tables using `LOAD DATA INFILE`:

```
LOAD DATA LOCAL INFILE '/home/user/DDIA/tpch-dbgen/data/region.tbl'
INTO TABLE region
FIELDS TERMINATED BY '|'
LINES TERMINATED BY '\n';

LOAD DATA LOCAL INFILE '/home/user/DDIA/tpch-dbgen/data/nation.tbl'
INTO TABLE nation
FIELDS TERMINATED BY '|'
LINES TERMINATED BY '\n';

LOAD DATA LOCAL INFILE '/home/user/DDIA/tpch-dbgen/data/customer.tbl'
INTO TABLE customer
FIELDS TERMINATED BY '|'
LINES TERMINATED BY '\n';

LOAD DATA LOCAL INFILE '/home/user/DDIA/tpch-dbgen/data/orders.tbl'
INTO TABLE orders
FIELDS TERMINATED BY '|'
LINES TERMINATED BY '\n';

LOAD DATA LOCAL INFILE '/home/user/DDIA/tpch-dbgen/data/supplier.tbl'
INTO TABLE supplier
FIELDS TERMINATED BY '|'
LINES TERMINATED BY '\n';

LOAD DATA LOCAL INFILE '/home/user/DDIA/tpch-dbgen/data/part.tbl'
INTO TABLE part
FIELDS TERMINATED BY '|'
LINES TERMINATED BY '\n';

LOAD DATA LOCAL INFILE '/home/user/DDIA/tpch-dbgen/data/partsupp.tbl'
INTO TABLE partsupp
FIELDS TERMINATED BY '|'
```

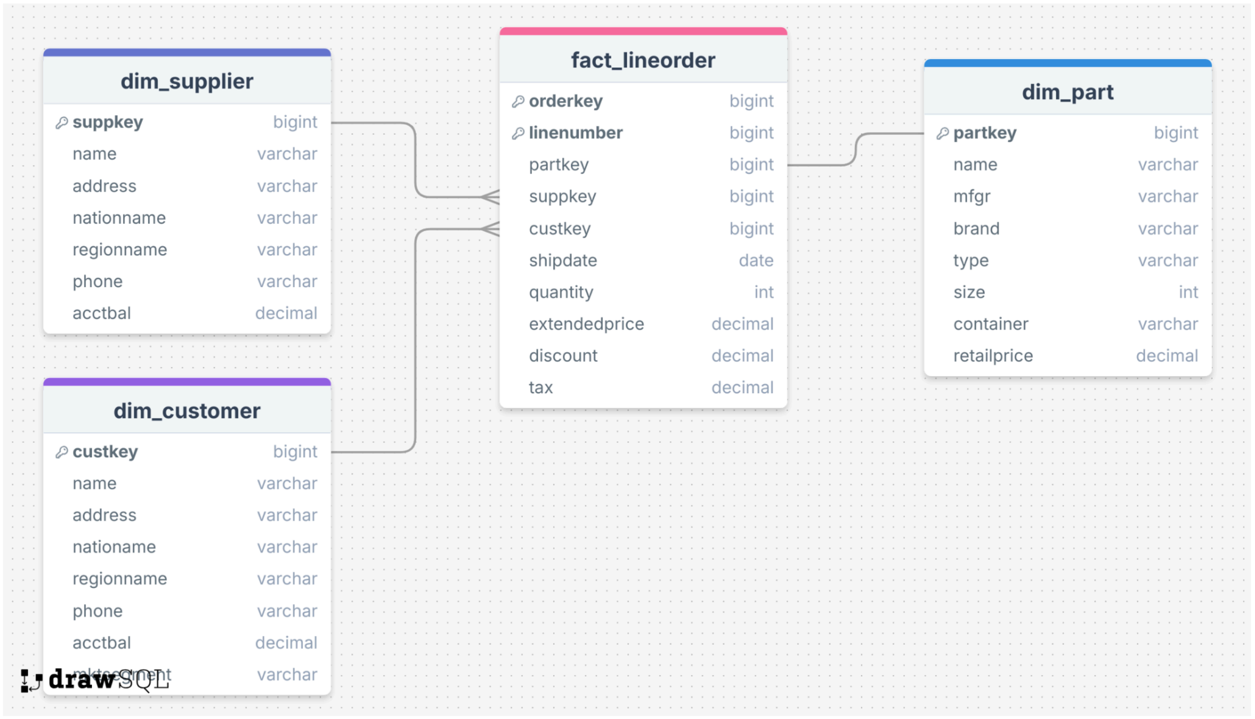
```
LINES TERMINATED BY '\n';

LOAD DATA LOCAL INFILE '/home/user/DDIA/tpch-dbgen/data/lineitem.tbl'
INTO TABLE lineitem
FIELDS TERMINATED BY '|'
LINES TERMINATED BY '\n';
```

Star Schema

Initial (more generic) schema:

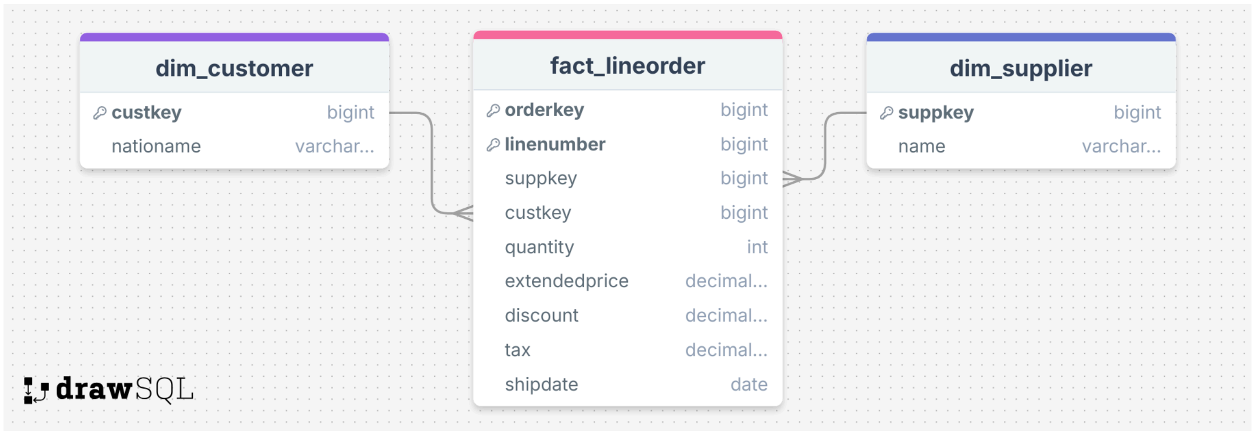


After removing date dimension (which is complex to construct):



Final (more specific) schema:

- Removed unused tables (e.g., dim_part) and attributes not needed by the given query.



DDL

```
CREATE TABLE `fact_lineorder`(
    `orderkey` INT,
    `linenumber` INT,
    `suppkey` INT,
    `custkey` INT,
    `quantity` INT,
    `extendedprice` DECIMAL(12, 2),
    `discount` DECIMAL(5, 2),
    `tax` DECIMAL(5, 2) NOT NULL,
    `shipdate` DATE,
    PRIMARY KEY(`orderkey`, `linenumber`),
    FOREIGN KEY(`suppkey`) REFERENCES `dim_supplier`(`suppkey`),
    FOREIGN KEY(`custkey`) REFERENCES `dim_customer`(`custkey`)
);

CREATE TABLE `dim_customer`(
    `custkey` INT PRIMARY KEY,
    `nationame` VARCHAR(50)
);

CREATE TABLE `dim_supplier`(
    `suppkey` INT PRIMARY KEY,
    `name` VARCHAR(100)
);
```

NiFi

First and foremost, it's important to clarify that NiFi is not designed to be a transformation tool in the strict sense of ETL. It is primarily a dataflow automation tool. Therefore, any attempts to use it for transformations come with inherent limitations, as we are trying to accomplish something beyond its intended purpose.

During our work, we explored three approaches using:

1. ExecuteSQL processor
2. LookupRecord processor
3. JoinEnrichment processor

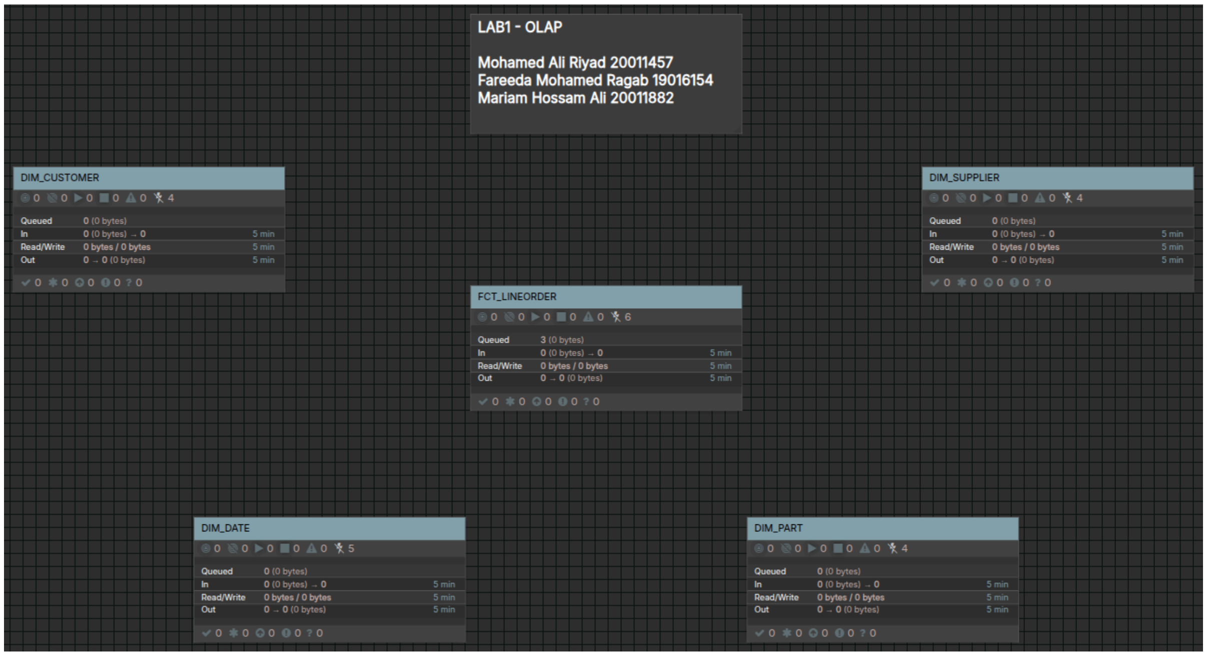
1. ExecuteSQL processor:

Our initial approach was the simplest: using the ExecuteSQL processor to fetch joined tables directly from the MySQL database. However, performing the join operation on the MySQL server led to exhausted database connections. The ExecuteSQL processor produced flow files in Avro format, which we then converted to Parquet using the ConvertRecord processor. Finally, we used the PutFile processor to write the file to disk.

A key aspect of our approach was adhering to Star Schema construction principles, particularly focusing on the dimensions of Who, Where, When, and What. This led us to create a date dimension, which was a complex process that raised several questions. For instance, when creating indices, should this process be executed periodically, potentially overwriting the existing table? And where should such a table be maintained?

To address this, we extracted all date-related columns from our database and combined them into a unified dataset. We then sorted the dates in ascending order, assigned each a unique index, and replaced actual dates in the LineOrder table with reference keys pointing to this dynamically created date table. After constructing the LineOrder table, we dropped the temporary date table. However, a more efficient approach could be to maintain the date table in the database for future bulk ETL processes, ensuring consistency and eliminating concerns about misaligned date indices.

- Data flow

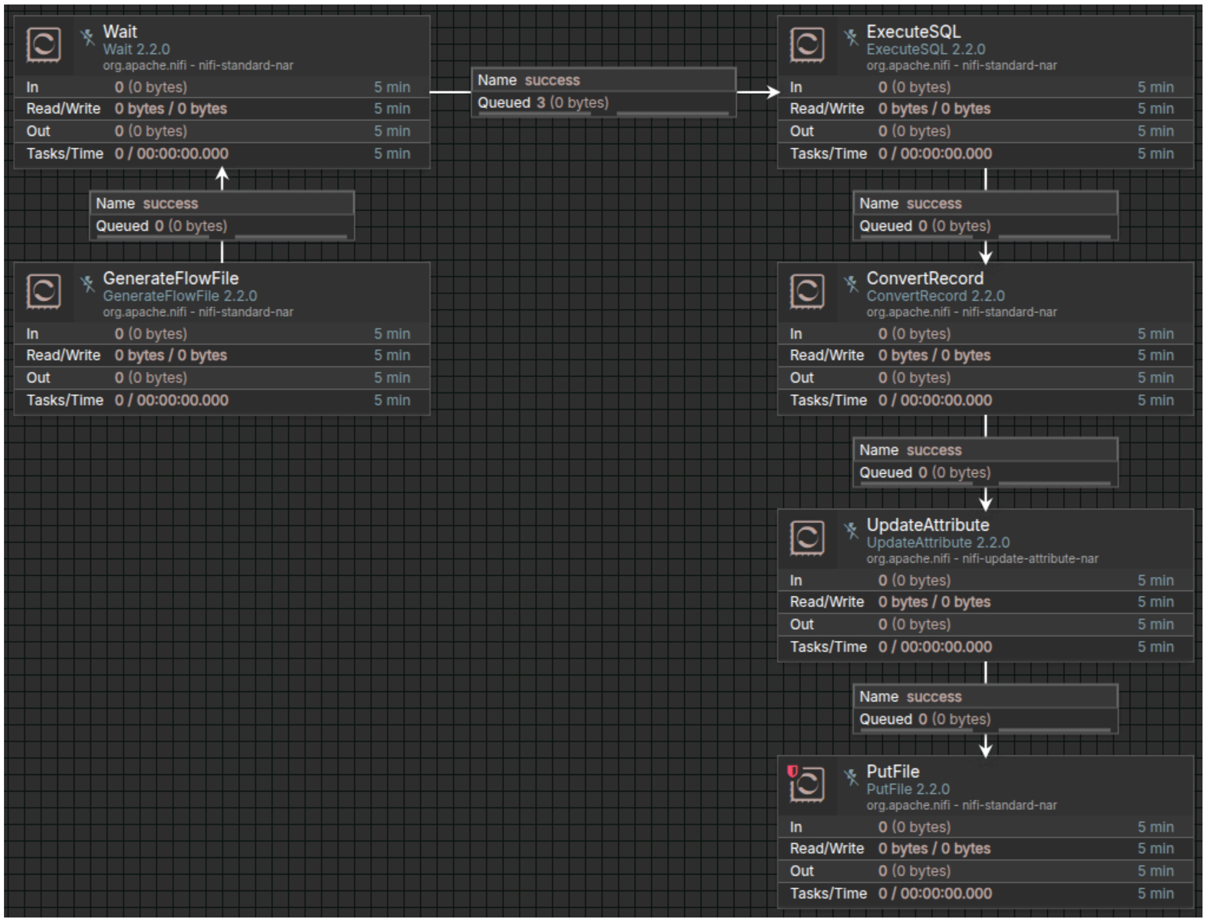


LAB1 - OLAP

Mohamed Ali Riyad 20011457
Fareeda Mohamed Ragab 19016154
Mariam Hossam Ali 20011882

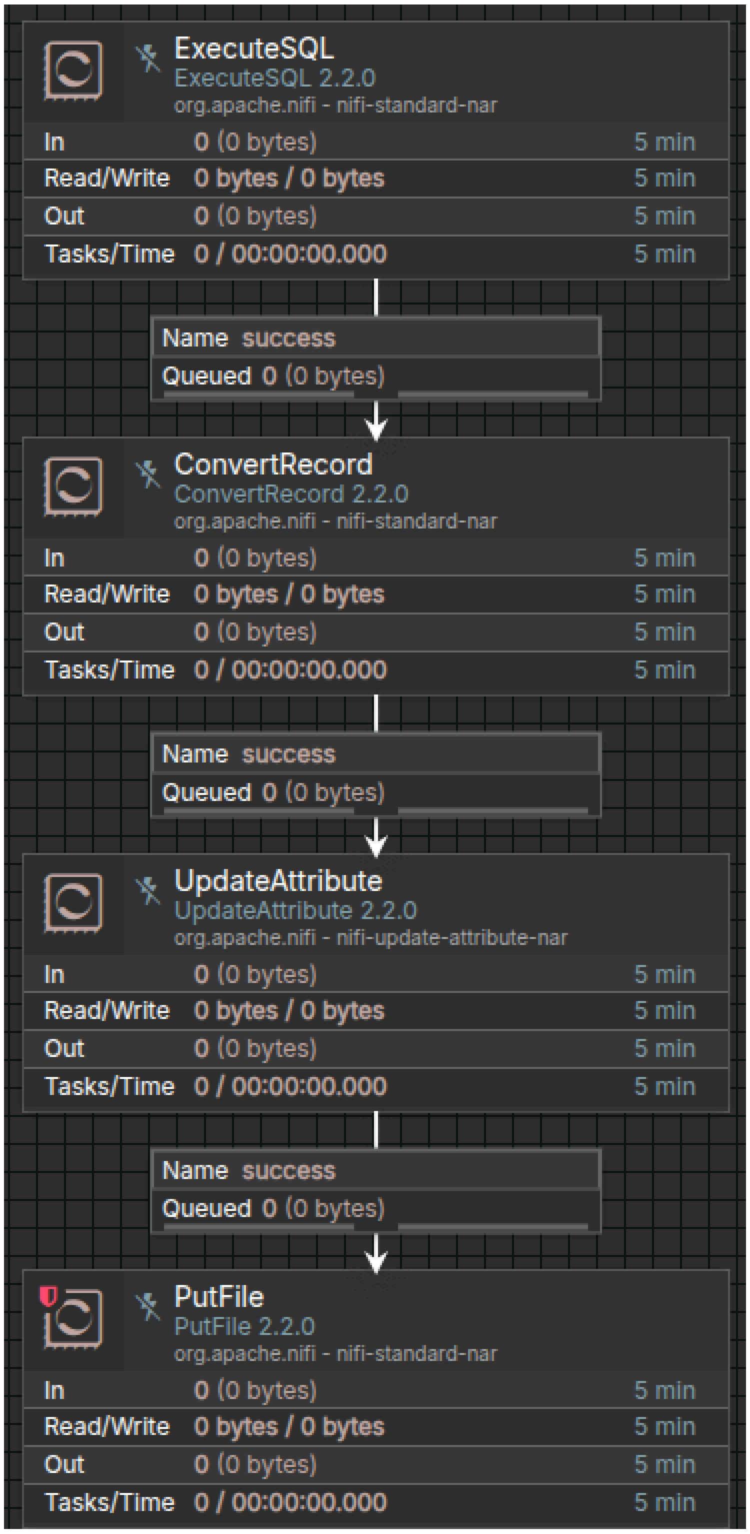
```
DIM_CUSTOMER
0  0  ▶ 0  ■ 0  ⚠ 0  4
Queued       0 (0 bytes)
In           0 (0 bytes) → 0          5 min
Read/Write   0 bytes / 0 bytes        5 min
Out          0 → 0 (0 bytes)          5 min
✓ 0  ✱ 0  0  0  ? 0

DIM_SUPPLIER
0  0  ▶ 0  ■ 0  ⚠ 0  4
Queued       0 (0 bytes)
In           0 (0 bytes) → 0          5 min
Read/Write   0 bytes / 0 bytes        5 min
Out          0 → 0 (0 bytes)          5 min
✓ 0  ✱ 0  0  0  ? 0

FCT_LINEORDER
0  0  ▶ 0  ■ 0  ⚠ 0  6
Queued       3 (0 bytes)
In           0 (0 bytes) → 0          5 min
Read/Write   0 bytes / 0 bytes        5 min
Out          0 → 0 (0 bytes)          5 min
✓ 0  ✱ 0  0  0  ? 0

DIM_DATE
0  0  ▶ 0  ■ 0  ⚠ 0  5
Queued       0 (0 bytes)
In           0 (0 bytes) → 0          5 min
Read/Write   0 bytes / 0 bytes        5 min
Out          0 → 0 (0 bytes)          5 min
✓ 0  ✱ 0  0  0  ? 0

DIM_PART
0  0  ▶ 0  ■ 0  ⚠ 0  4
Queued       0 (0 bytes)
In           0 (0 bytes) → 0          5 min
Read/Write   0 bytes / 0 bytes        5 min
Out          0 → 0 (0 bytes)          5 min
✓ 0  ✱ 0  0  0  ? 0
```

- FCT_LINEORDER



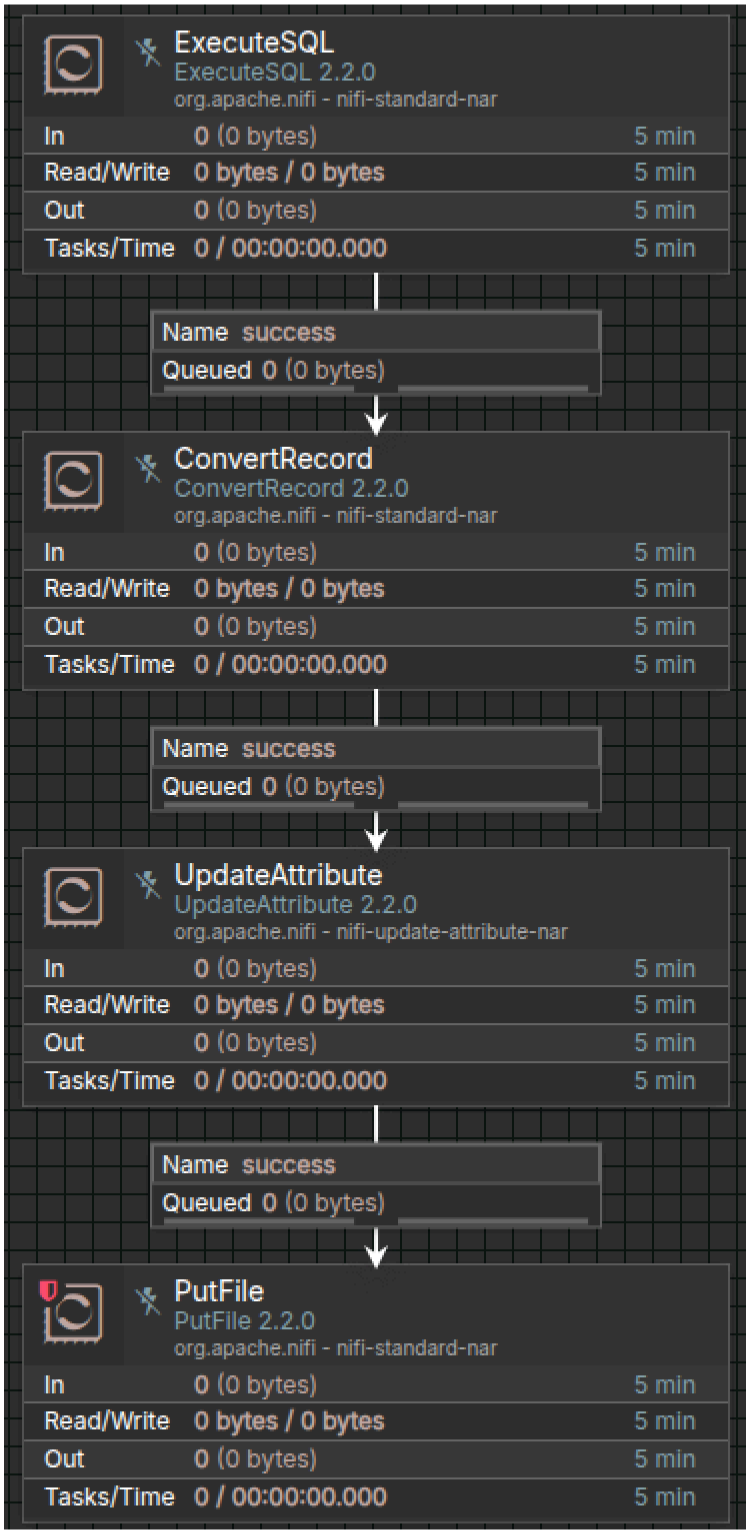
```
GenerateFlowFile
GenerateFlowFile 2.2.0
org.apache.nifi - nifi-standard-nar
In           0 (0 bytes)              5 min
Read/Write   0 bytes / 0 bytes        5 min
Out          0 (0 bytes)              5 min
Tasks/Time   0 / 00:00:00.000         5 min

Name success
Queued 0 (0 bytes)

Wait
Wait 2.2.0
org.apache.nifi - nifi-standard-nar
In           0 (0 bytes)              5 min
Read/Write   0 bytes / 0 bytes        5 min
Out          0 (0 bytes)              5 min
Tasks/Time   0 / 00:00:00.000         5 min

Name success
Queued 3 (0 bytes)

ExecuteSQL
ExecuteSQL 2.2.0
org.apache.nifi - nifi-standard-nar
In           0 (0 bytes)              5 min
Read/Write   0 bytes / 0 bytes        5 min
Out          0 (0 bytes)              5 min
Tasks/Time   0 / 00:00:00.000         5 min

Name success
Queued 0 (0 bytes)

ConvertRecord
ConvertRecord 2.2.0
org.apache.nifi - nifi-standard-nar
In           0 (0 bytes)              5 min
Read/Write   0 bytes / 0 bytes        5 min
Out          0 (0 bytes)              5 min
Tasks/Time   0 / 00:00:00.000         5 min

Name success
Queued 0 (0 bytes)

UpdateAttribute
UpdateAttribute 2.2.0
org.apache.nifi - nifi-update-attribute-nar
In           0 (0 bytes)              5 min
Read/Write   0 bytes / 0 bytes        5 min
Out          0 (0 bytes)              5 min
Tasks/Time   0 / 00:00:00.000         5 min

Name success
Queued 0 (0 bytes)

PutFile
PutFile 2.2.0
org.apache.nifi - nifi-standard-nar
In           0 (0 bytes)              5 min
Read/Write   0 bytes / 0 bytes        5 min
Out          0 (0 bytes)              5 min
Tasks/Time   0 / 00:00:00.000         5 min
```

- DIM_SUPPLIER



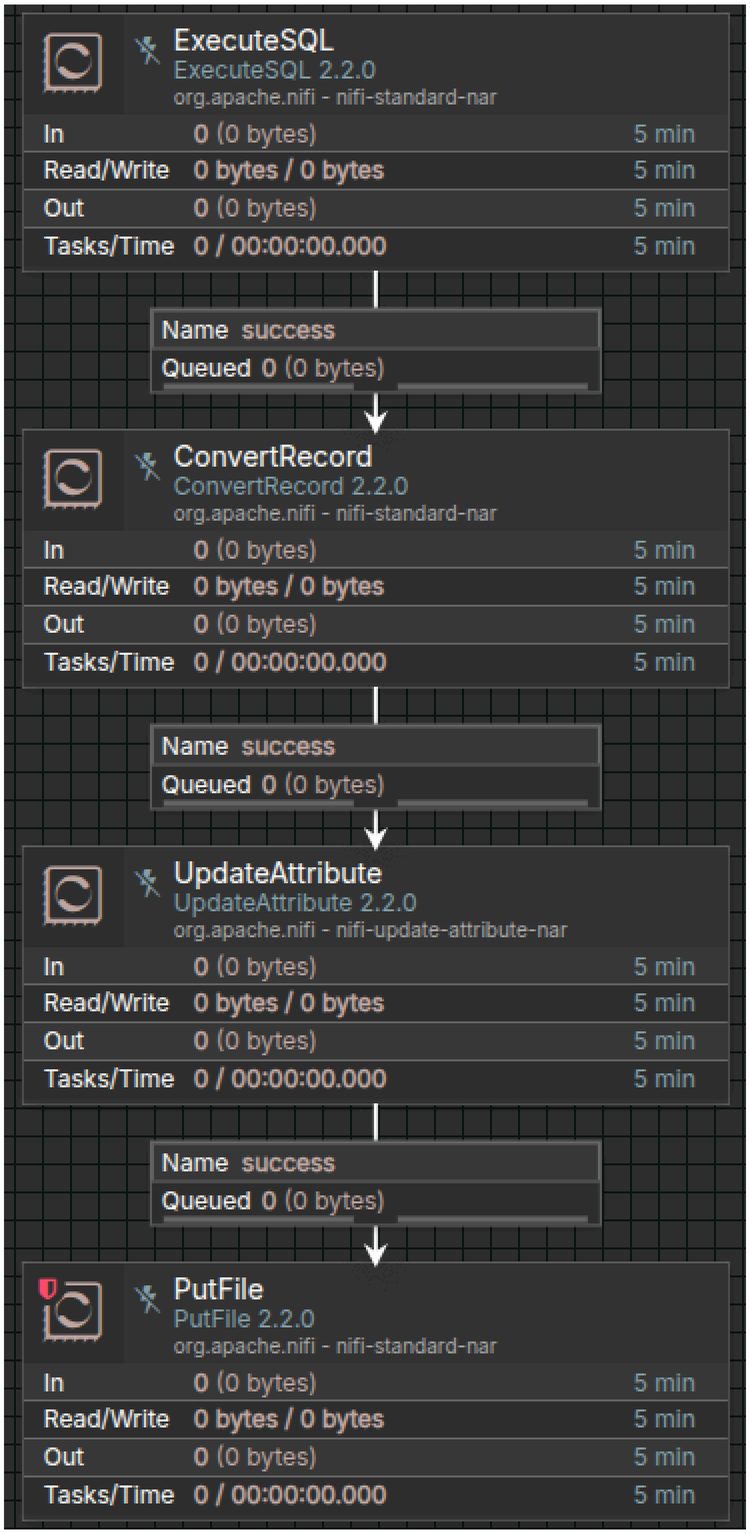
```
ExecuteSQL
ExecuteSQL 2.2.0
org.apache.nifi - nifi-standard-nar
In           0 (0 bytes)            5 min
Read/Write   0 bytes / 0 bytes      5 min
Out          0 (0 bytes)            5 min
Tasks/Time   0 / 00:00:00.000       5 min

Name    success
Queued  0 (0 bytes)

ConvertRecord
ConvertRecord 2.2.0
org.apache.nifi - nifi-standard-nar
In           0 (0 bytes)            5 min
Read/Write   0 bytes / 0 bytes      5 min
Out          0 (0 bytes)            5 min
Tasks/Time   0 / 00:00:00.000       5 min

Name    success
Queued  0 (0 bytes)

UpdateAttribute
UpdateAttribute 2.2.0
org.apache.nifi - nifi-update-attribute-nar
In           0 (0 bytes)            5 min
Read/Write   0 bytes / 0 bytes      5 min
Out          0 (0 bytes)            5 min
Tasks/Time   0 / 00:00:00.000       5 min

Name    success
Queued  0 (0 bytes)

PutFile
PutFile 2.2.0
org.apache.nifi - nifi-standard-nar
In           0 (0 bytes)            5 min
Read/Write   0 bytes / 0 bytes      5 min
Out          0 (0 bytes)            5 min
Tasks/Time   0 / 00:00:00.000       5 min
```

- DIM_CUSTOMER



```
ExecuteSQL
ExecuteSQL 2.2.0
org.apache.nifi - nifi-standard-nar
In            0 (0 bytes)            5 min
Read/Write    0 bytes / 0 bytes      5 min
Out           0 (0 bytes)            5 min
Tasks/Time    0 / 00:00:00.000       5 min

Name  success
Queued  0 (0 bytes)

ConvertRecord
ConvertRecord 2.2.0
org.apache.nifi - nifi-standard-nar
In            0 (0 bytes)            5 min
Read/Write    0 bytes / 0 bytes      5 min
Out           0 (0 bytes)            5 min
Tasks/Time    0 / 00:00:00.000       5 min

Name  success
Queued  0 (0 bytes)

UpdateAttribute
UpdateAttribute 2.2.0
org.apache.nifi - nifi-update-attribute-nar
In            0 (0 bytes)            5 min
Read/Write    0 bytes / 0 bytes      5 min
Out           0 (0 bytes)            5 min
Tasks/Time    0 / 00:00:00.000       5 min

Name  success
Queued  0 (0 bytes)

PutFile
PutFile 2.2.0
org.apache.nifi - nifi-standard-nar
In            0 (0 bytes)            5 min
Read/Write    0 bytes / 0 bytes      5 min
Out           0 (0 bytes)            5 min
Tasks/Time    0 / 00:00:00.000       5 min
```

- DIM_PART


ExecuteSQL
ExecuteSQL 2.2.0
org.apache.nifi - nifi-standard-nar

In	0 (0 bytes)	5 min
Read/Write	0 bytes / 0 bytes	5 min
Out	0 (0 bytes)	5 min
Tasks/Time	0 / 00:00:00.000	5 min

Name success
Queued 0 (0 bytes)

ConvertRecord
ConvertRecord 2.2.0
org.apache.nifi - nifi-standard-nar

In	0 (0 bytes)	5 min
Read/Write	0 bytes / 0 bytes	5 min
Out	0 (0 bytes)	5 min
Tasks/Time	0 / 00:00:00.000	5 min

Name success
Queued 0 (0 bytes)

UpdateAttribute
UpdateAttribute 2.2.0
org.apache.nifi - nifi-update-attribute-nar

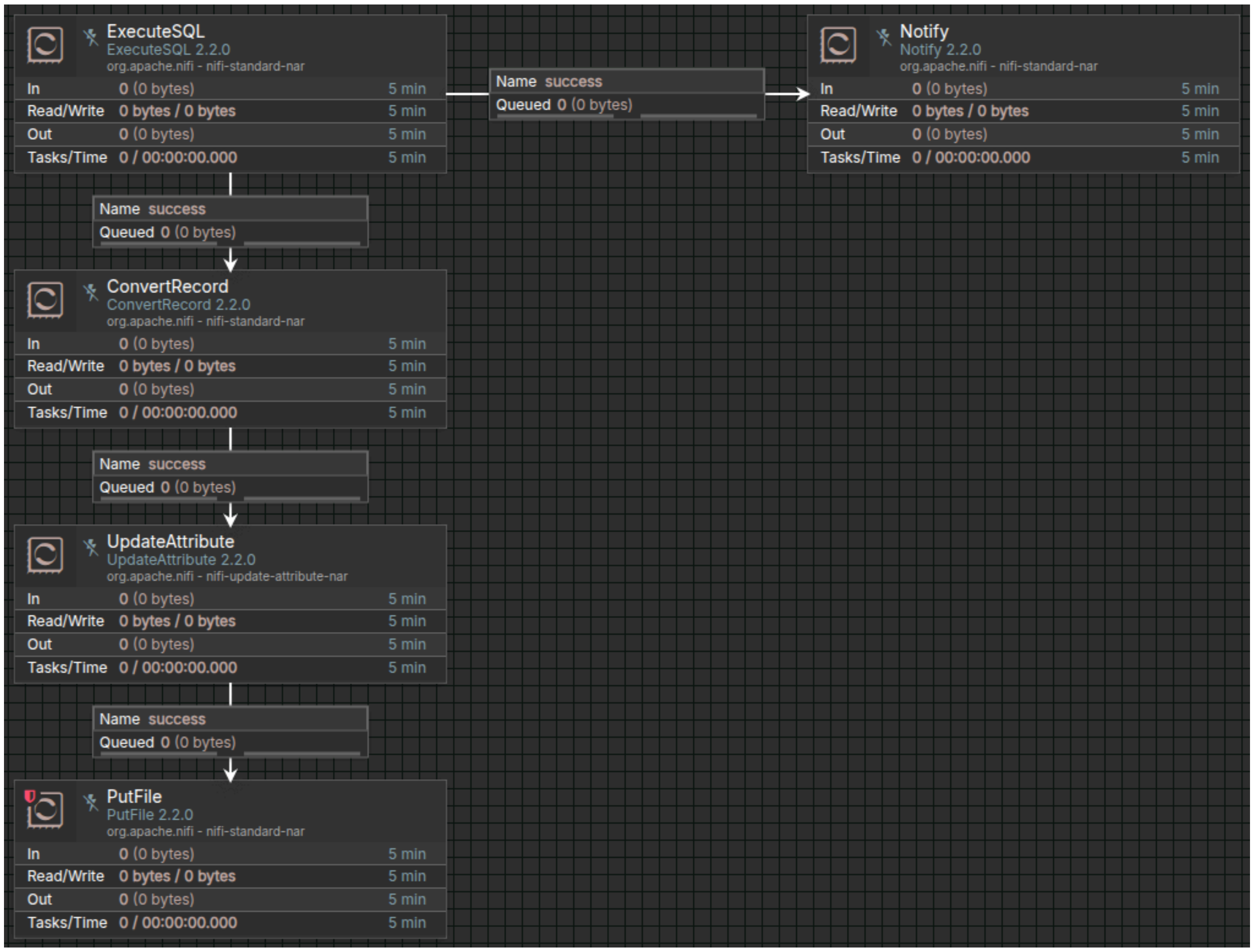
In	0 (0 bytes)	5 min
Read/Write	0 bytes / 0 bytes	5 min
Out	0 (0 bytes)	5 min
Tasks/Time	0 / 00:00:00.000	5 min

Name success
Queued 0 (0 bytes)

PutFile
PutFile 2.2.0
org.apache.nifi - nifi-standard-nar

In	0 (0 bytes)	5 min
Read/Write	0 bytes / 0 bytes	5 min
Out	0 (0 bytes)	5 min
Tasks/Time	0 / 00:00:00.000	5 min

- DIM_DATE



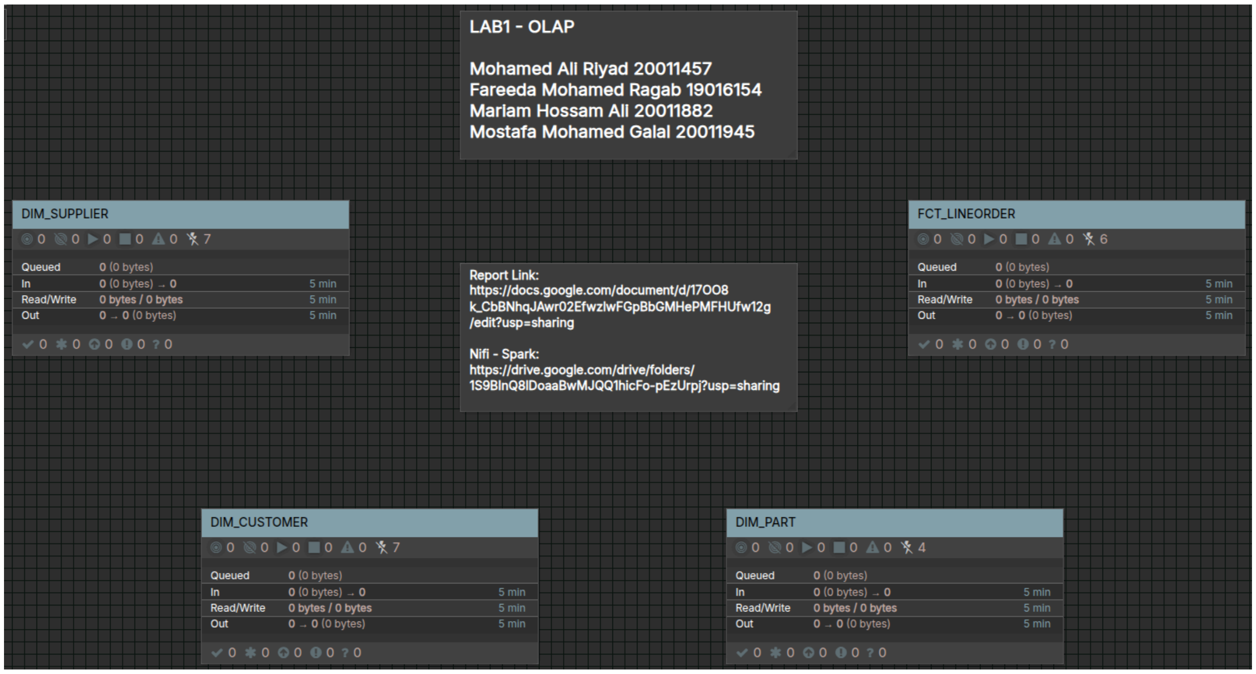
2. LookupRecord processor:

Instead of using the ExecuteSQL processor, which relies on an SQL connection to query the MySQL server, we switched to the LookupRecord processor. This processor leverages the DBLookupService, which acts as a cache where the reference column serves as the key, and the columns of interest are stored as values.

While this approach still requires a MySQL connection to fetch data, the join operation is now performed within NiFi. Additionally, depending on the cache size, the number of fetch operations can be reduced, making it a more efficient alternative to our initial approach.

We also decided to remove the date dimension, as splitting it into a separate table and then rejoining it for queries introduced unnecessary overhead. Instead, we retained the other three dimensions outlined in the star schema while keeping the same fact table structure.

- Data flow

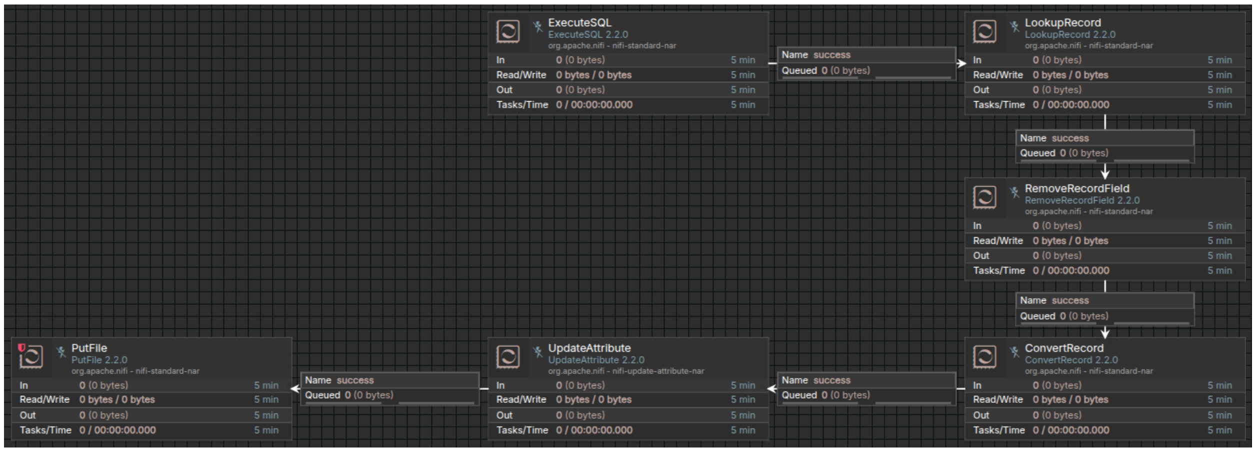

LAB1 - OLAP

Mohamed Ali Riyad 20011457
Fareeda Mohamed Ragab 19016154
Mariam Hossam Ali 20011882
Mostafa Mohamed Galal 20011945

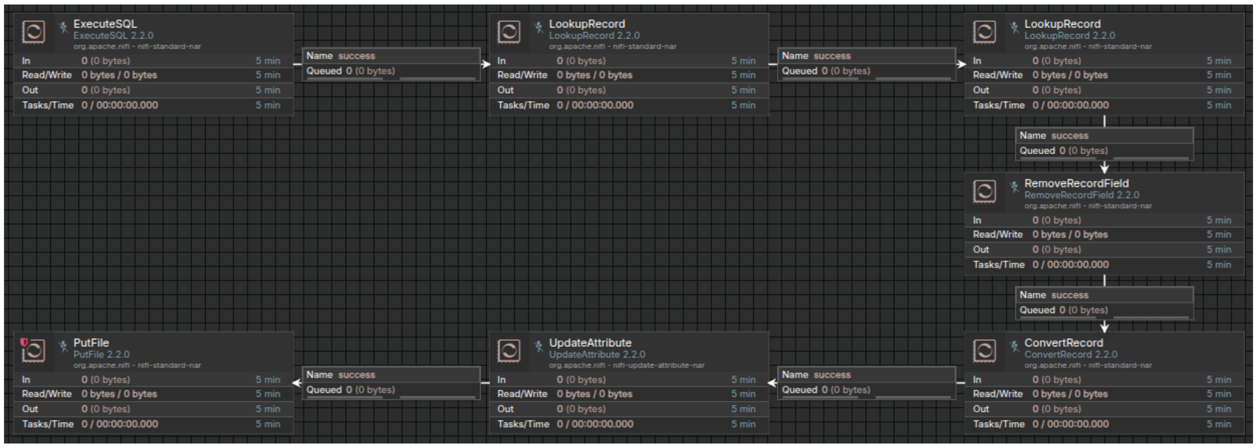
Report Link:
https://docs.google.com/document/d/17OO8
k_CbBNhqJAwr02EfwzlwFGpBbGMHePMFHUfw12g
/edit?usp=sharing

Nifi - Spark:
https://drive.google.com/drive/folders/
1S9BlnQ8lDoaaBwMJQQ1hicFo-pEzUrpj?usp=sharing

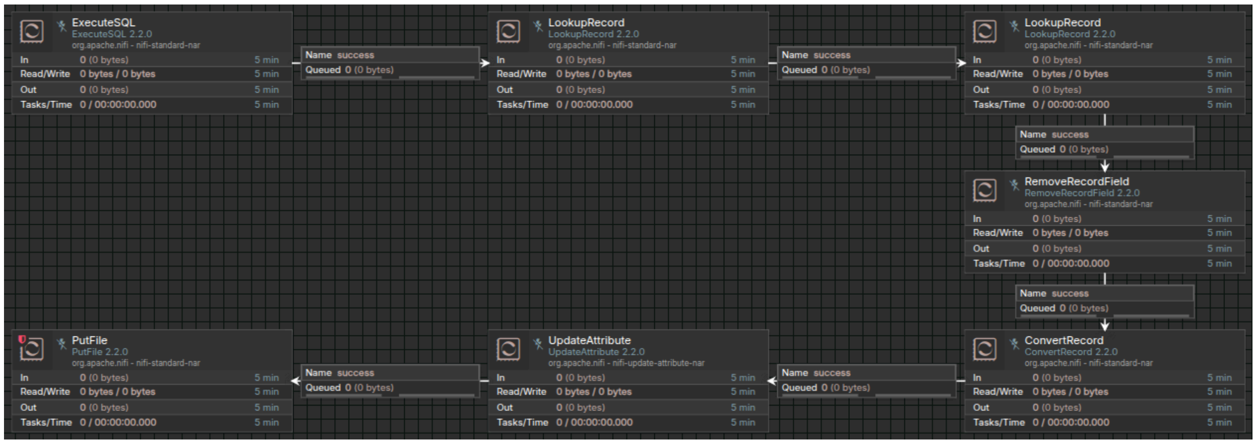
- FCT_LINEORDER



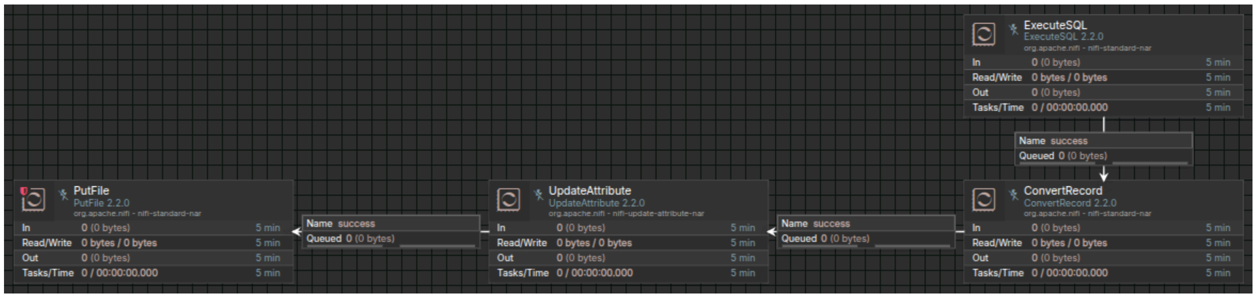
- DIM_SUPPLIER



- DIM_CUSTOMER



- DIM_PART

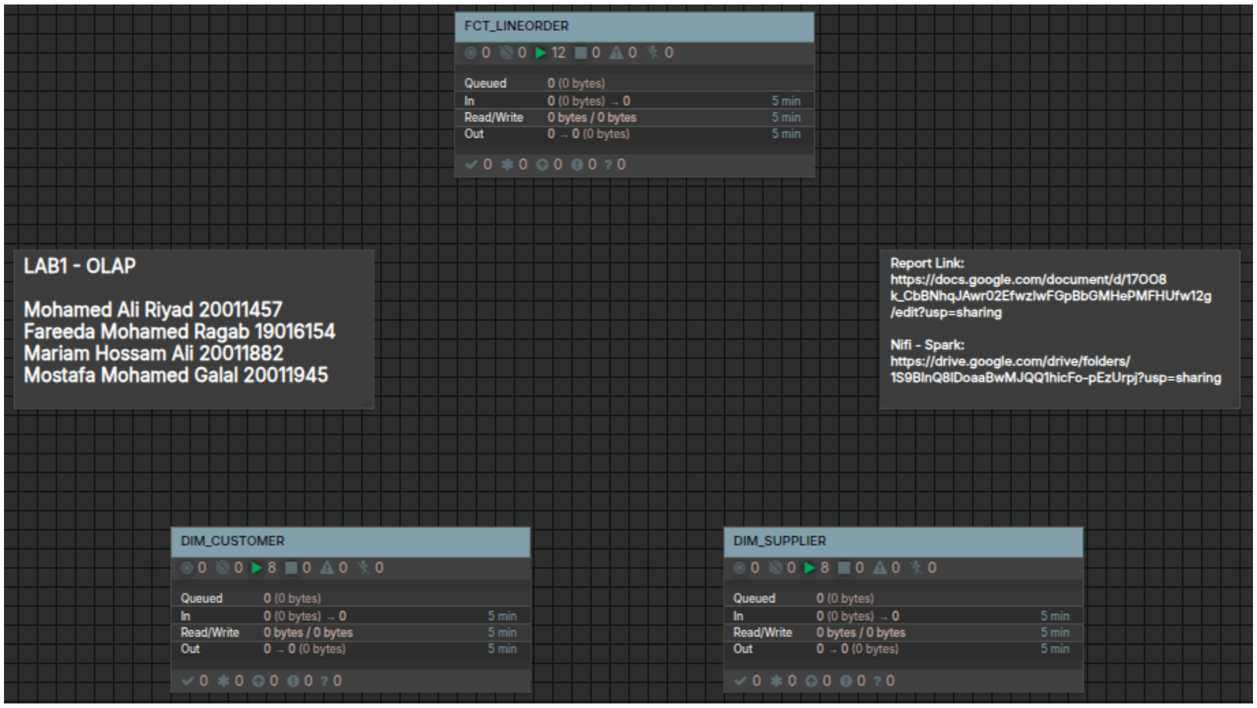


3. JoinEnrichment processor:

We here used the JoinEnrichment processor which needs flow files to be annotated with certain attributes which are ORIGINAL and ENRICHMENT in order to merge them.
One problem that we faced during using Enrichment processor with Lineitem table is that it took a lot of heap and won't finish in reasonable time due to its large number of records (6 M records), So we added GenerateTableFetch and batched the Lineitem table into 6 batches each of them would be joined with Orders table after that we merge these batched flowfiles in one single flowfile convert it from Avro to Parquet.
Another problem is that JoinEnricment uses whatever flowfiles in the queues connected to it so it can take two flowfiles from same queue (meaning trying to join two flowfiles of same table whether Lineitem or Orders) so we needed one table of each time present in each queue while JoinEnrichment tries to fetch the flow files so we added Rate controller and do some inspection to see optimal time such that we are sure that JoinEnrichment has done the joining operation and starts to fetch next batch tables to be joined. We reached the result that 30 sec per batch is the fastest and safest option so the whole process took about 4 min.

For the sake of simplicity and just using tables meant for the query we made a minimal Star Schema derived from the previous one In which we included only Lineorder, Supplier and Customer only.
The reset of the process is the same as other approaches.

- Dataflow



FCT_LINEORDER

```
0  0  12  0  0  0
Queued      0 (0 bytes)
In          0 (0 bytes) → 0          5 min
Read/Write  0 bytes / 0 bytes        5 min
Out         0 → 0 (0 bytes)          5 min
0  0  0  0  0
```

LAB1 - OLAP

Mohamed Ali Riyad 20011457
Fareeda Mohamed Ragab 19016154
Mariam Hossam Ali 20011882
Mostafa Mohamed Galal 20011945

Report Link:
https://docs.google.com/document/d/17OO8k_CbBNhqJAwr02EfwzlwFGpBbGMHePMFHUfw12g/edit?usp=sharing

Nifi - Spark:
https://drive.google.com/drive/folders/1S9BlnQ8lDoaaBwMJQQ1hicFo-pEzUrpj?usp=sharing

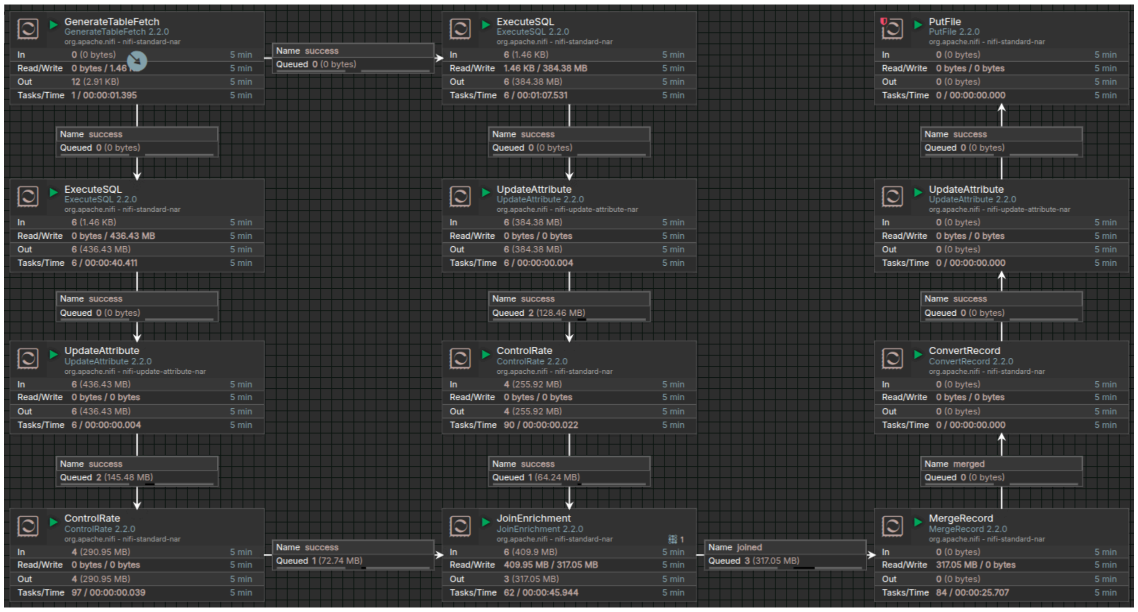
DIM_CUSTOMER

```
0  0  8  0  0  0
Queued      0 (0 bytes)
In          0 (0 bytes) → 0          5 min
Read/Write  0 bytes / 0 bytes        5 min
Out         0 → 0 (0 bytes)          5 min
0  0  0  0  0
```

DIM_SUPPLIER

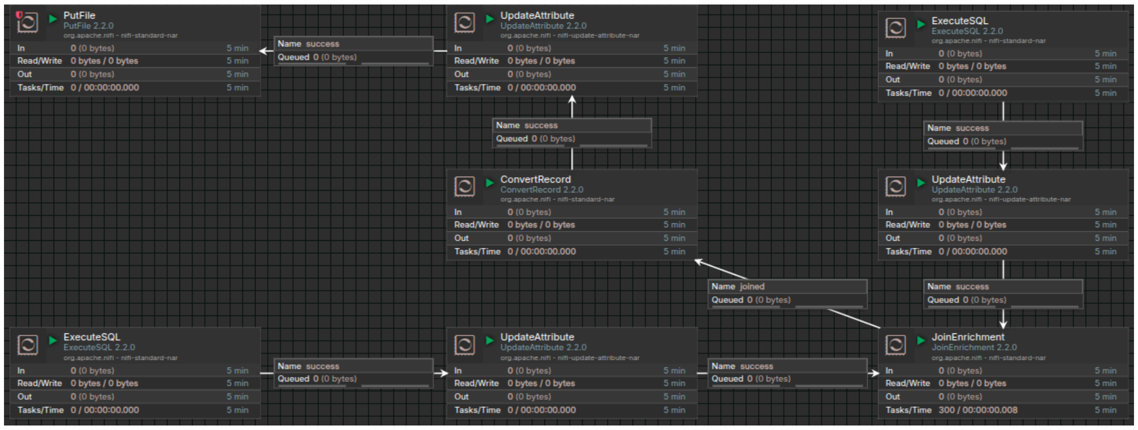
```
0  0  8  0  0  0
Queued      0 (0 bytes)
In          0 (0 bytes) → 0          5 min
Read/Write  0 bytes / 0 bytes        5 min
Out         0 → 0 (0 bytes)          5 min
0  0  0  0  0
```

- FCT_LINEORDER



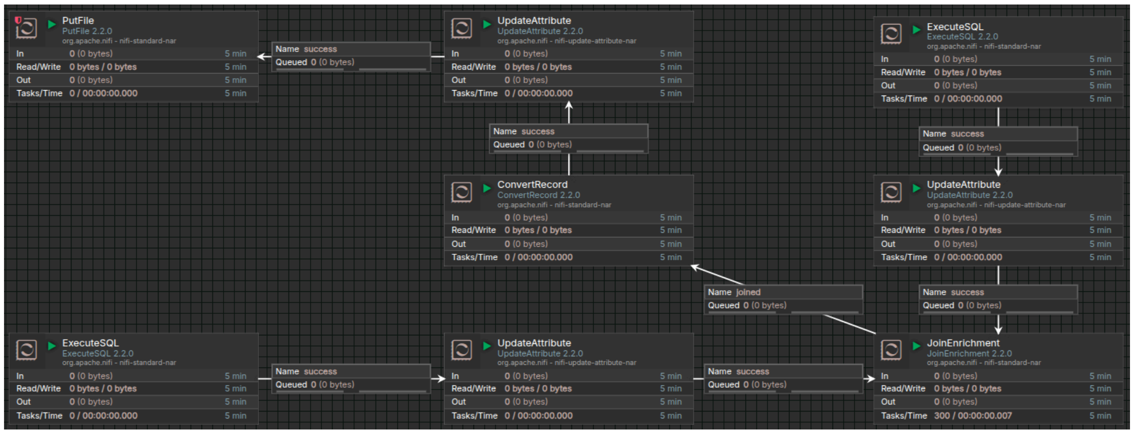
```
GenerateTableFetch
GenerateTableFetch 2.2.0
org.apache.nifi - nifi-standard-nar
In          0 (0 bytes)              5 min
Read/Write  0 bytes / 1.46           5 min
Out         12 (2.91 KB)             5 min
Tasks/Time  1 / 00:00:01.395         5 min

Name success
Queued 0 (0 bytes)

ExecuteSQL
ExecuteSQL 2.2.0
org.apache.nifi - nifi-standard-nar
In          6 (1.46 KB)              5 min
Read/Write  1.46 KB / 384.38 MB      5 min
Out         6 (384.38 MB)            5 min
Tasks/Time  6 / 00:01:07.531         5 min

PutFile
PutFile 2.2.0
org.apache.nifi - nifi-standard-nar
In          0 (0 bytes)              5 min
Read/Write  0 bytes / 0 bytes        5 min
Out         0 (0 bytes)              5 min
Tasks/Time  0 / 00:00:00.000         5 min

Name success
Queued 0 (0 bytes)

Name success
Queued 0 (0 bytes)

Name success
Queued 0 (0 bytes)

ExecuteSQL
ExecuteSQL 2.2.0
org.apache.nifi - nifi-standard-nar
In          6 (1.46 KB)              5 min
Read/Write  0 bytes / 436.43 MB      5 min
Out         6 (436.43 MB)            5 min
Tasks/Time  6 / 00:00:40.411         5 min

UpdateAttribute
UpdateAttribute 2.2.0
org.apache.nifi - nifi-update-attribute-nar
In          6 (384.38 MB)            5 min
Read/Write  0 bytes / 0 bytes        5 min
Out         6 (384.38 MB)            5 min
Tasks/Time  6 / 00:00:00.004         5 min

UpdateAttribute
UpdateAttribute 2.2.0
org.apache.nifi - nifi-update-attribute-nar
In          0 (0 bytes)              5 min
Read/Write  0 bytes / 0 bytes        5 min
Out         0 (0 bytes)              5 min
Tasks/Time  0 / 00:00:00.000         5 min

Name success
Queued 0 (0 bytes)

Name success
Queued 2 (128.46 MB)

Name success
Queued 0 (0 bytes)

UpdateAttribute
UpdateAttribute 2.2.0
org.apache.nifi - nifi-update-attribute-nar
In          6 (436.43 MB)            5 min
Read/Write  0 bytes / 0 bytes        5 min
Out         6 (436.43 MB)            5 min
Tasks/Time  6 / 00:00:00.004         5 min

ControlRate
ControlRate 2.2.0
org.apache.nifi - nifi-standard-nar
In          4 (255.92 MB)            5 min
Read/Write  0 bytes / 0 bytes        5 min
Out         4 (255.92 MB)            5 min
Tasks/Time  90 / 00:00:00.022        5 min

ConvertRecord
ConvertRecord 2.2.0
org.apache.nifi - nifi-standard-nar
In          0 (0 bytes)              5 min
Read/Write  0 bytes / 0 bytes        5 min
Out         0 (0 bytes)              5 min
Tasks/Time  0 / 00:00:00.000         5 min

Name success
Queued 2 (145.48 MB)

Name success
Queued 1 (64.24 MB)

Name merged
Queued 0 (0 bytes)

ControlRate
ControlRate 2.2.0
org.apache.nifi - nifi-standard-nar
In          4 (290.95 MB)            5 min
Read/Write  0 bytes / 0 bytes        5 min
Out         4 (290.95 MB)            5 min
Tasks/Time  97 / 00:00:00.039        5 min

Name success
Queued 1 (72.74 MB)

JoinEnrichment
JoinEnrichment 2.2.0
org.apache.nifi - nifi-standard-nar                1
In          6 (409.9 MB)             5 min
Read/Write  409.95 MB / 317.05 MB    5 min
Out         3 (317.05 MB)            5 min
Tasks/Time  62 / 00:00:45.944        5 min

Name joined
Queued 3 (317.05 MB)

MergeRecord
MergeRecord 2.2.0
org.apache.nifi - nifi-standard-nar
In          0 (0 bytes)              5 min
Read/Write  317.05 MB / 0 bytes      5 min
Out         0 (0 bytes)              5 min
Tasks/Time  84 / 00:00:25.707        5 min
```

- DIM_SUPPLIER



```
PutFile
PutFile 2.2.0
org.apache.nifi - nifi-standard-nar
In          0 (0 bytes)              5 min
Read/Write  0 bytes / 0 bytes        5 min
Out         0 (0 bytes)              5 min
Tasks/Time  0 / 00:00:00.000         5 min

Name success
Queued 0 (0 bytes)

UpdateAttribute
UpdateAttribute 2.2.0
org.apache.nifi - nifi-update-attribute-nar
In          0 (0 bytes)              5 min
Read/Write  0 bytes / 0 bytes        5 min
Out         0 (0 bytes)              5 min
Tasks/Time  0 / 00:00:00.000         5 min

ExecuteSQL
ExecuteSQL 2.2.0
org.apache.nifi - nifi-standard-nar
In          0 (0 bytes)              5 min
Read/Write  0 bytes / 0 bytes        5 min
Out         0 (0 bytes)              5 min
Tasks/Time  0 / 00:00:00.000         5 min

Name success
Queued 0 (0 bytes)

Name success
Queued 0 (0 bytes)

ConvertRecord
ConvertRecord 2.2.0
org.apache.nifi - nifi-standard-nar
In          0 (0 bytes)              5 min
Read/Write  0 bytes / 0 bytes        5 min
Out         0 (0 bytes)              5 min
Tasks/Time  0 / 00:00:00.000         5 min

UpdateAttribute
UpdateAttribute 2.2.0
org.apache.nifi - nifi-update-attribute-nar
In          0 (0 bytes)              5 min
Read/Write  0 bytes / 0 bytes        5 min
Out         0 (0 bytes)              5 min
Tasks/Time  0 / 00:00:00.000         5 min

Name joined
Queued 0 (0 bytes)

Name success
Queued 0 (0 bytes)

ExecuteSQL
ExecuteSQL 2.2.0
org.apache.nifi - nifi-standard-nar
In          0 (0 bytes)              5 min
Read/Write  0 bytes / 0 bytes        5 min
Out         0 (0 bytes)              5 min
Tasks/Time  0 / 00:00:00.000         5 min

Name success
Queued 0 (0 bytes)

UpdateAttribute
UpdateAttribute 2.2.0
org.apache.nifi - nifi-update-attribute-nar
In          0 (0 bytes)              5 min
Read/Write  0 bytes / 0 bytes        5 min
Out         0 (0 bytes)              5 min
Tasks/Time  0 / 00:00:00.000         5 min

Name success
Queued 0 (0 bytes)

JoinEnrichment
JoinEnrichment 2.2.0
org.apache.nifi - nifi-standard-nar
In          0 (0 bytes)              5 min
Read/Write  0 bytes / 0 bytes        5 min
Out         0 (0 bytes)              5 min
Tasks/Time  300 / 00:00:00.008       5 min
```

- DIM_CUSTOMER



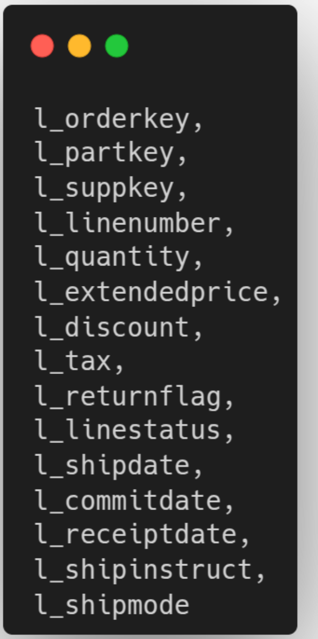
```
PutFile
PutFile 2.2.0
org.apache.nifi - nifi-standard-nar
In          0 (0 bytes)              5 min
Read/Write  0 bytes / 0 bytes        5 min
Out         0 (0 bytes)              5 min
Tasks/Time  0 / 00:00:00.000         5 min

Name success
Queued 0 (0 bytes)

UpdateAttribute
UpdateAttribute 2.2.0
org.apache.nifi - nifi-update-attribute-nar
In          0 (0 bytes)              5 min
Read/Write  0 bytes / 0 bytes        5 min
Out         0 (0 bytes)              5 min
Tasks/Time  0 / 00:00:00.000         5 min

ExecuteSQL
ExecuteSQL 2.2.0
org.apache.nifi - nifi-standard-nar
In          0 (0 bytes)              5 min
Read/Write  0 bytes / 0 bytes        5 min
Out         0 (0 bytes)              5 min
Tasks/Time  0 / 00:00:00.000         5 min

Name success
Queued 0 (0 bytes)

Name success
Queued 0 (0 bytes)

ConvertRecord
ConvertRecord 2.2.0
org.apache.nifi - nifi-standard-nar
In          0 (0 bytes)              5 min
Read/Write  0 bytes / 0 bytes        5 min
Out         0 (0 bytes)              5 min
Tasks/Time  0 / 00:00:00.000         5 min

UpdateAttribute
UpdateAttribute 2.2.0
org.apache.nifi - nifi-update-attribute-nar
In          0 (0 bytes)              5 min
Read/Write  0 bytes / 0 bytes        5 min
Out         0 (0 bytes)              5 min
Tasks/Time  0 / 00:00:00.000         5 min

Name joined
Queued 0 (0 bytes)

Name success
Queued 0 (0 bytes)

ExecuteSQL
ExecuteSQL 2.2.0
org.apache.nifi - nifi-standard-nar
In          0 (0 bytes)              5 min
Read/Write  0 bytes / 0 bytes        5 min
Out         0 (0 bytes)              5 min
Tasks/Time  0 / 00:00:00.000         5 min

Name success
Queued 0 (0 bytes)

UpdateAttribute
UpdateAttribute 2.2.0
org.apache.nifi - nifi-update-attribute-nar
In          0 (0 bytes)              5 min
Read/Write  0 bytes / 0 bytes        5 min
Out         0 (0 bytes)              5 min
Tasks/Time  0 / 00:00:00.000         5 min

Name success
Queued 0 (0 bytes)

JoinEnrichment
JoinEnrichment 2.2.0
org.apache.nifi - nifi-standard-nar
In          0 (0 bytes)              5 min
Read/Write  0 bytes / 0 bytes        5 min
Out         0 (0 bytes)              5 min
Tasks/Time  300 / 00:00:00.007       5 min
```

SQL queries

- FCT_LINEORDER

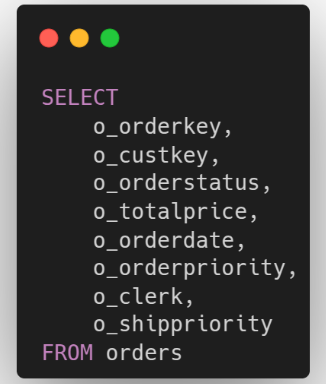
  Lineitem Select:

  Lineitem is returned in batches considering only these columns



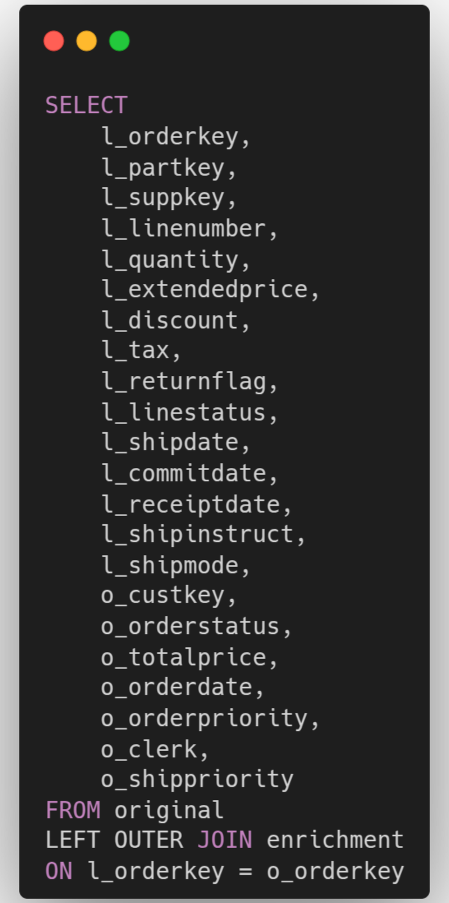
```
l_orderkey,
l_partkey,
l_suppkey,
l_linenumber,
l_quantity,
l_extendedprice,
l_discount,
l_tax,
l_returnflag,
l_linestatus,
l_shipdate,
l_commitdate,
l_receiptdate,
l_shipinstruct,
l_shipmode
```

Orders Select:



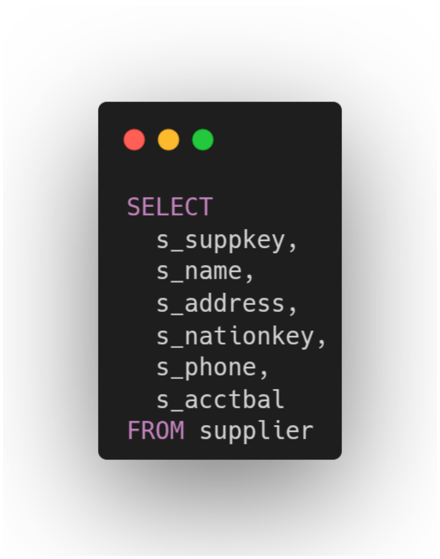
```
SELECT
    o_orderkey,
    o_custkey,
    o_orderstatus,
    o_totalprice,
    o_orderdate,
    o_orderpriority,
    o_clerk,
    o_shippriority
FROM orders
```

LineOrder Join:



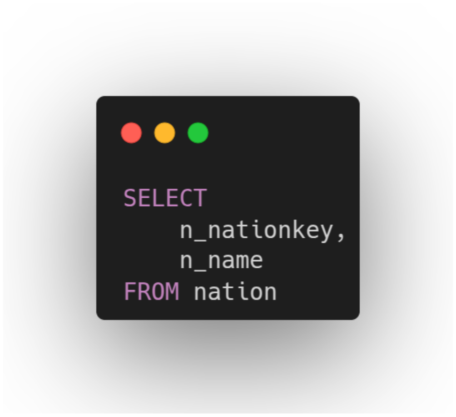
```
SELECT
    l_orderkey,
    l_partkey,
    l_suppkey,
    l_linenumber,
    l_quantity,
    l_extendedprice,
    l_discount,
    l_tax,
    l_returnflag,
    l_linestatus,
    l_shipdate,
    l_commitdate,
    l_receiptdate,
    l_shipinstruct,
    l_shipmode,
    o_custkey,
    o_orderstatus,
    o_totalprice,
    o_orderdate,
    o_orderpriority,
    o_clerk,
    o_shippriority
FROM original
LEFT OUTER JOIN enrichment
ON l_orderkey = o_orderkey
```

- DIM_SUPPLIER
  Supplier Select:



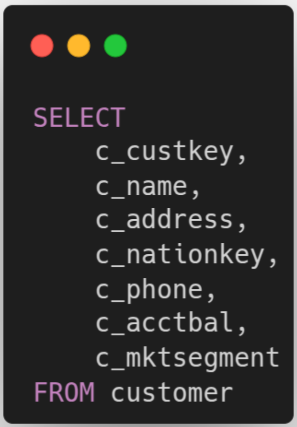
```
SELECT
  s_suppkey,
  s_name,
  s_address,
  s_nationkey,
  s_phone,
  s_acctbal
FROM supplier
```

Nation Select:



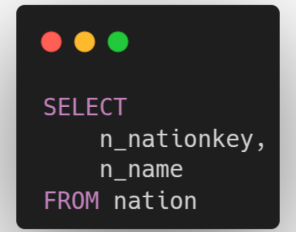
```
SELECT
    n_nationkey,
    n_name
FROM nation
```

- DIM_CUSTOMER
  Customer Select:



```sql
SELECT
    c_custkey,
    c_name,
    c_address,
    c_nationkey,
    c_phone,
    c_acctbal,
    c_mktsegment
FROM customer
```

  Nation Select:

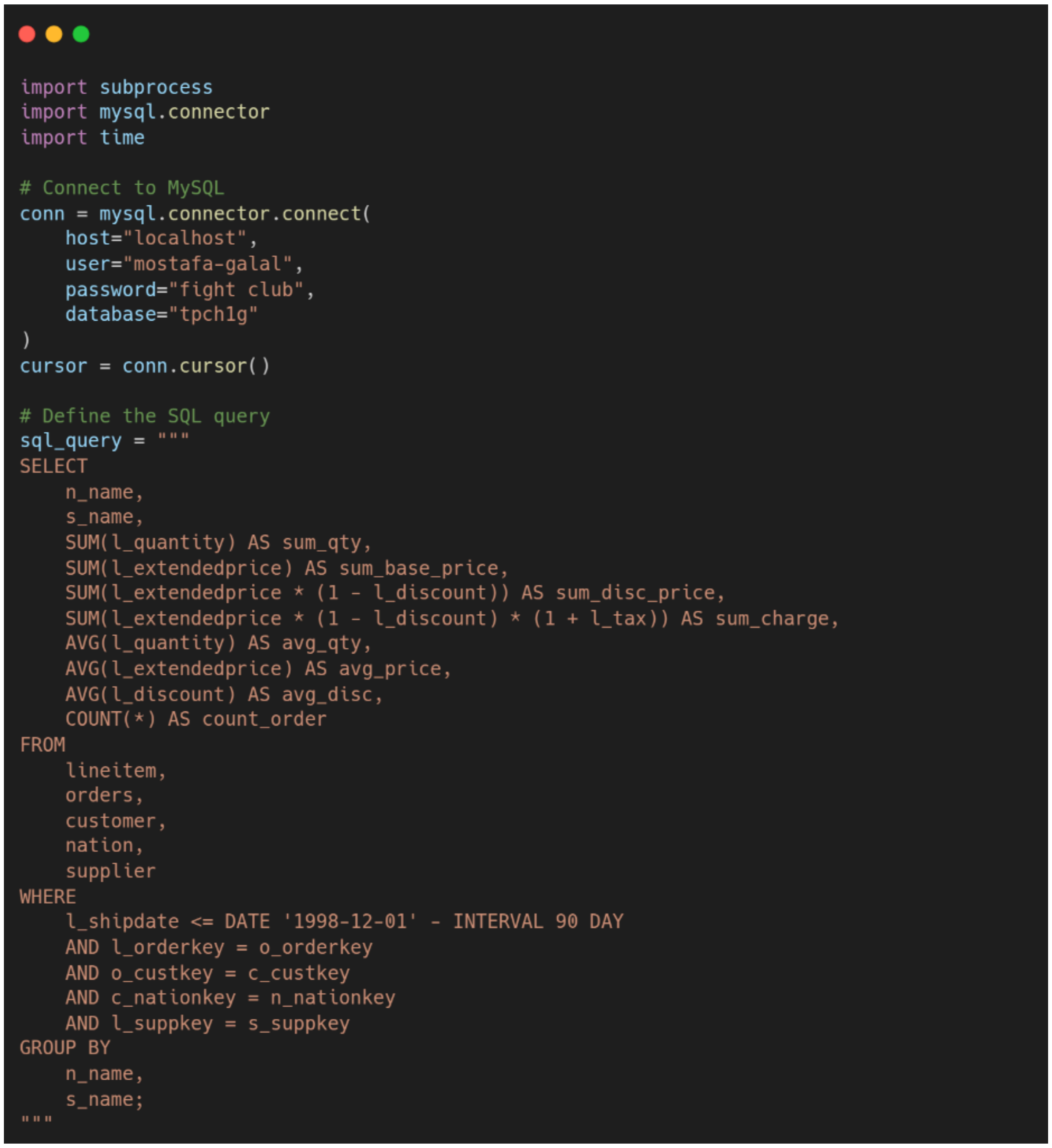


```sql
SELECT
    n_nationkey,
    n_name
FROM nation
```

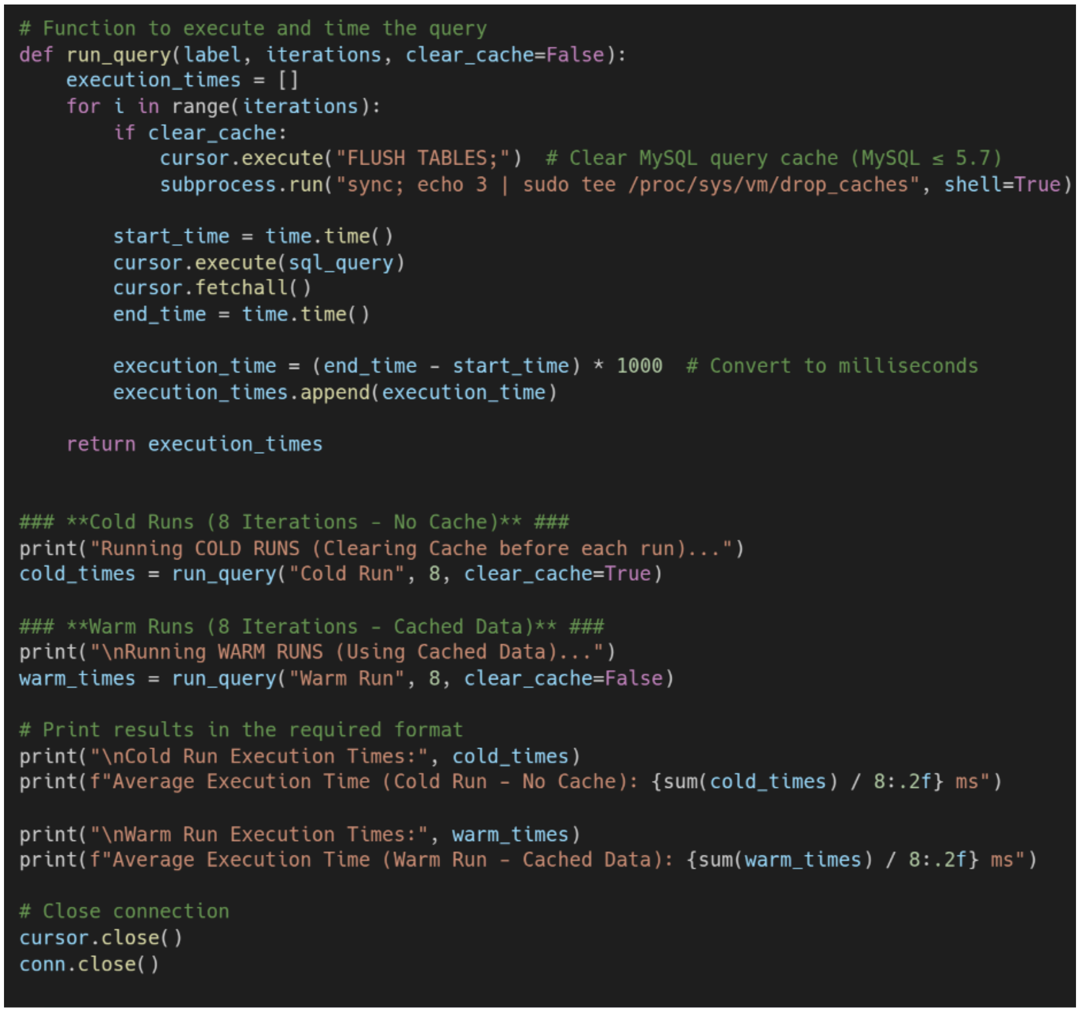
Benchmark

1. MySQL Benchmark

- Code snippet:

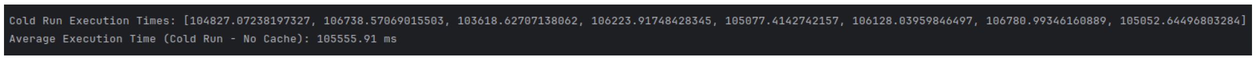


```
import subprocess
import mysql.connector
import time

# Connect to MySQL
conn = mysql.connector.connect(
    host="localhost",
    user="mostafa-galal",
    password="fight club",
    database="tpch1g"
)
cursor = conn.cursor()

# Define the SQL query
sql_query = """
SELECT
    n_name,
    s_name,
    SUM(l_quantity) AS sum_qty,
    SUM(l_extendedprice) AS sum_base_price,
    SUM(l_extendedprice * (1 - l_discount)) AS sum_disc_price,
    SUM(l_extendedprice * (1 - l_discount) * (1 + l_tax)) AS sum_charge,
    AVG(l_quantity) AS avg_qty,
    AVG(l_extendedprice) AS avg_price,
    AVG(l_discount) AS avg_disc,
    COUNT(*) AS count_order
FROM
    lineitem,
    orders,
    customer,
    nation,
    supplier
WHERE
    l_shipdate <= DATE '1998-12-01' - INTERVAL 90 DAY
    AND l_orderkey = o_orderkey
    AND o_custkey = c_custkey
    AND c_nationkey = n_nationkey
    AND l_suppkey = s_suppkey
GROUP BY
    n_name,
    s_name;
"""
```



```python
# Function to execute and time the query
def run_query(label, iterations, clear_cache=False):
    execution_times = []
    for i in range(iterations):
        if clear_cache:
            cursor.execute("FLUSH TABLES;")  # Clear MySQL query cache (MySQL ≤ 5.7)
            subprocess.run("sync; echo 3 | sudo tee /proc/sys/vm/drop_caches", shell=True)

        start_time = time.time()
        cursor.execute(sql_query)
        cursor.fetchall()
        end_time = time.time()

        execution_time = (end_time - start_time) * 1000  # Convert to milliseconds
        execution_times.append(execution_time)

    return execution_times


### **Cold Runs (8 Iterations - No Cache)** ###
print("Running COLD RUNS (Clearing Cache before each run)...")
cold_times = run_query("Cold Run", 8, clear_cache=True)

### **Warm Runs (8 Iterations - Cached Data)** ###
print("\nRunning WARM RUNS (Using Cached Data)...")
warm_times = run_query("Warm Run", 8, clear_cache=False)

# Print results in the required format
print("\nCold Run Execution Times:", cold_times)
print(f"Average Execution Time (Cold Run - No Cache): {sum(cold_times) / 8:.2f} ms")

print("\nWarm Run Execution Times:", warm_times)
print(f"Average Execution Time (Warm Run - Cached Data): {sum(warm_times) / 8:.2f} ms")

# Close connection
cursor.close()
conn.close()
```

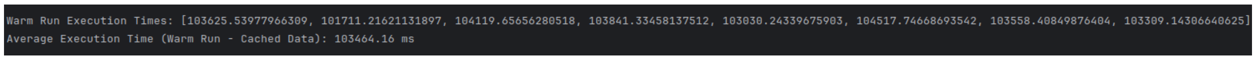
- MySQL (Cold run execution time):



```
Cold Run Execution Times: [104827.07238197327, 106738.57069015503, 103618.62707138062, 106223.91748428345, 105077.4142742157, 106128.03959846497, 106780.99346160889, 105052.64496803284]
Average Execution Time (Cold Run - No Cache): 105555.91 ms
```

Cold average execution time: 105555.91 ms = 1.7593 min

- MySQL (Warm run execution time):

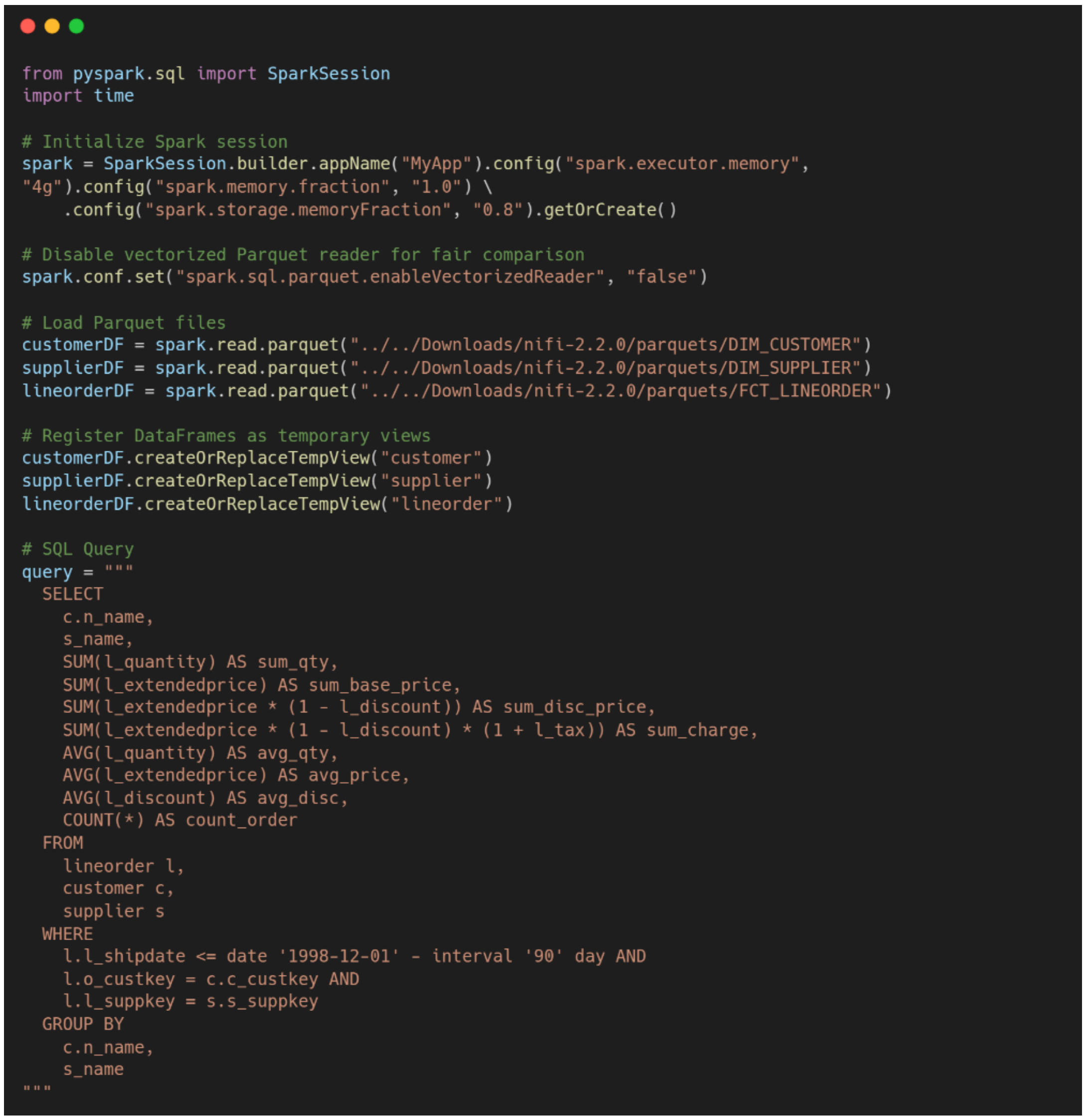


```
Warm Run Execution Times: [103625.53977966309, 101711.21621131897, 104119.65656280518, 103841.33458137512, 103030.24339675903, 104517.74668693542, 103558.40849876404, 103309.14306640625]
Average Execution Time (Warm Run - Cached Data): 103464.16 ms
```

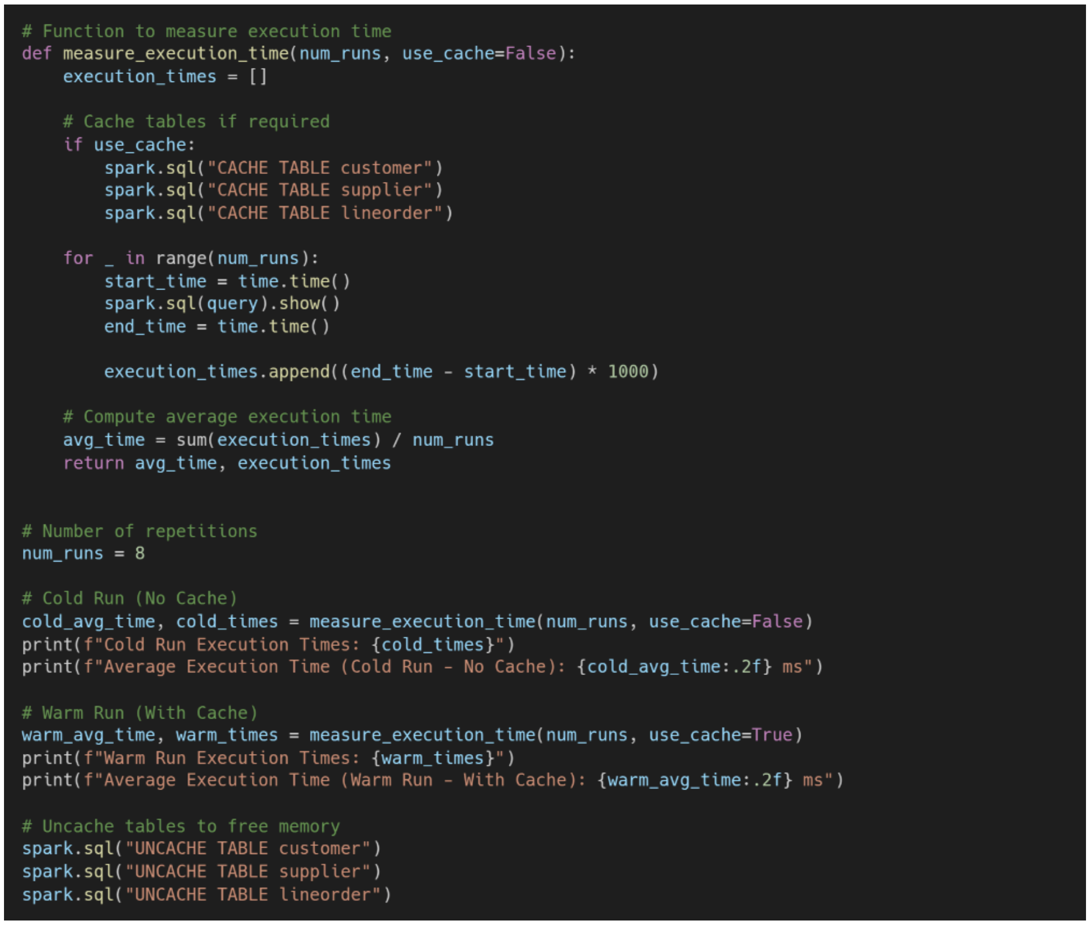
Warm average execution time: 103464.16 ms = 1.7244 min

2. Spark Benchmark

- Code snippet:

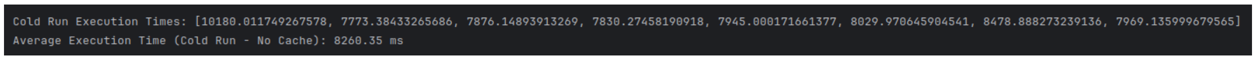


```python
from pyspark.sql import SparkSession
import time

# Initialize Spark session
spark = SparkSession.builder.appName("MyApp").config("spark.executor.memory",
"4g").config("spark.memory.fraction", "1.0") \
    .config("spark.storage.memoryFraction", "0.8").getOrCreate()

# Disable vectorized Parquet reader for fair comparison
spark.conf.set("spark.sql.parquet.enableVectorizedReader", "false")

# Load Parquet files
customerDF = spark.read.parquet("../../Downloads/nifi-2.2.0/parquets/DIM_CUSTOMER")
supplierDF = spark.read.parquet("../../Downloads/nifi-2.2.0/parquets/DIM_SUPPLIER")
lineorderDF = spark.read.parquet("../../Downloads/nifi-2.2.0/parquets/FCT_LINEORDER")

# Register DataFrames as temporary views
customerDF.createOrReplaceTempView("customer")
supplierDF.createOrReplaceTempView("supplier")
lineorderDF.createOrReplaceTempView("lineorder")

# SQL Query
query = """
  SELECT
    c.n_name,
    s_name,
    SUM(l_quantity) AS sum_qty,
    SUM(l_extendedprice) AS sum_base_price,
    SUM(l_extendedprice * (1 - l_discount)) AS sum_disc_price,
    SUM(l_extendedprice * (1 - l_discount) * (1 + l_tax)) AS sum_charge,
    AVG(l_quantity) AS avg_qty,
    AVG(l_extendedprice) AS avg_price,
    AVG(l_discount) AS avg_disc,
    COUNT(*) AS count_order
  FROM
    lineorder l,
    customer c,
    supplier s
  WHERE
    l.l_shipdate <= date '1998-12-01' - interval '90' day AND
    l.o_custkey = c.c_custkey AND
    l.l_suppkey = s.s_suppkey
  GROUP BY
    c.n_name,
    s_name
"""
```



```python
# Function to measure execution time
def measure_execution_time(num_runs, use_cache=False):
    execution_times = []

    # Cache tables if required
    if use_cache:
        spark.sql("CACHE TABLE customer")
        spark.sql("CACHE TABLE supplier")
        spark.sql("CACHE TABLE lineorder")

    for _ in range(num_runs):
        start_time = time.time()
        spark.sql(query).show()
        end_time = time.time()

        execution_times.append((end_time - start_time) * 1000)

    # Compute average execution time
    avg_time = sum(execution_times) / num_runs
    return avg_time, execution_times


# Number of repetitions
num_runs = 8

# Cold Run (No Cache)
cold_avg_time, cold_times = measure_execution_time(num_runs, use_cache=False)
print(f"Cold Run Execution Times: {cold_times}")
print(f"Average Execution Time (Cold Run - No Cache): {cold_avg_time:.2f} ms")

# Warm Run (With Cache)
warm_avg_time, warm_times = measure_execution_time(num_runs, use_cache=True)
print(f"Warm Run Execution Times: {warm_times}")
print(f"Average Execution Time (Warm Run - With Cache): {warm_avg_time:.2f} ms")

# Uncache tables to free memory
spark.sql("UNCACHE TABLE customer")
spark.sql("UNCACHE TABLE supplier")
spark.sql("UNCACHE TABLE lineorder")
```

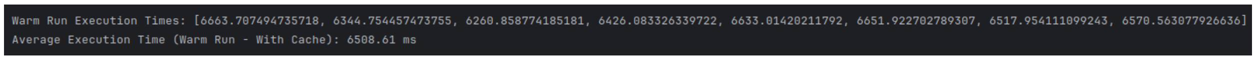
- Spark (Cold run execution time):



```
Cold Run Execution Times: [10180.011749267578, 7773.38433265686, 7876.14893913269, 7830.27458190918, 7945.000171661377, 8029.970645904541, 8478.888273239136, 7969.135999679565]
Average Execution Time (Cold Run - No Cache): 8260.35 ms
```

Warm average execution time: 8260.35 ms = 0.1377 min

- Spark (Warm run execution time):



```
Warm Run Execution Times: [6663.707494735718, 6344.754457473755, 6260.858774185181, 6426.083326339722, 6633.01420211792, 6651.922702789307, 6517.954111099243, 6570.563077926636]
Average Execution Time (Warm Run - With Cache): 6508.61 ms
```

Warm average execution time: 6508.61 ms = 0.1085 min

3. Analysis

- **Cold Run Performance:**

  MySQL took **105,555.91 ms (1.7593 min)**, whereas Spark took **8,260.35 ms (0.1377 min)**. Spark outperformed MySQL by a factor of approximately **12.8x**.

- **Warm Run Performance:**

  MySQL took **103,464.16 ms (1.7244 min)**, whereas Spark took **6,508.61 ms (0.1085 min)**. Spark outperformed MySQL by a factor of approximately **15.9x**.

- **Overall Performance Difference:**

  Spark demonstrates a significant advantage over MySQL, with execution times differing by **two orders of magnitude**.

  The performance gap increases further in the warm run, highlighting Spark's superior efficiency in handling cached data.

4. Conclusion

The benchmark results are summarized in the table below:

Execution Type	MySQL (ms)	MySQL (min)	Spark (ms)	Spark (min)
Cold Run	105,555.91	1.7593	8,260.35	0.1377
Warm Run	103,464.16	1.7244	6,508.61	0.1085

Apache Spark exhibits a dramatically faster execution time than MySQL in both cold and warm runs. The significant reduction in execution time suggests that Spark is a more suitable option for processing large-scale data, particularly in scenarios requiring high-speed data retrieval and analysis. This benchmark reaffirms Spark's efficiency, making it a preferable choice for big data applications compared to MySQL.