HandsOn W05 – MapReduce untuk Data Tabel

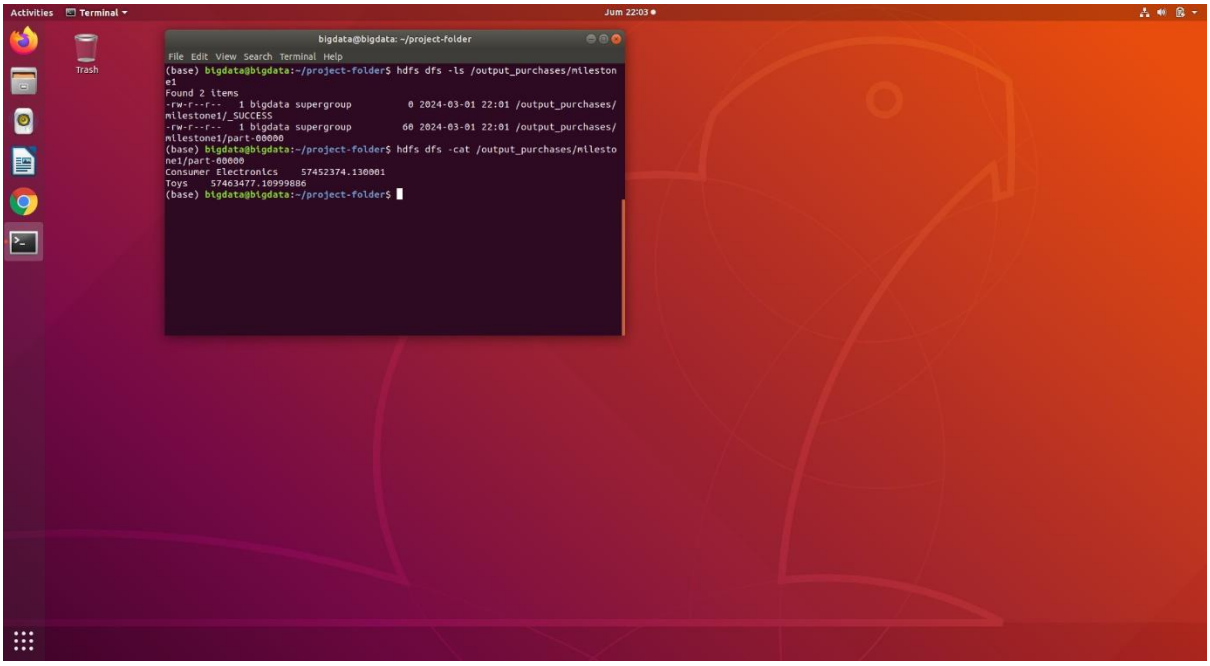
Diberikan dataset yang sama, “purchases.txt”, yang digunakan di ***Example 02*** pada slide, buatlah program map reduce untuk masing-masing milestone di bawah ini. Untuk menjalankan map reduce di Hadoop, file “purchases.txt” tersebut harus sudah ditempatkan di suatu folder di HDFS. Sebelumnya, pastikan Hadoop sudah berjalan di VM yang digunakan (seperti yang telah dilakukan di HandsOn W05 sebelumnya).

A. Milestone 1

1. Tampilkan total nilai penjualan untuk produk: (i) “Toys” dan (ii) “Consumer Electronics”. Sebagai catatan, nama produk dapat bermacam-macam, selama mengandung salah satu dari kedua string, (i) dan (ii), tersebut. Contoh: “Buffalo Toys”. Output dari milestone ini adalah sebagai berikut.

Consumer Electronics 57452374.130001 Toys 57463477.10999886

2. Tampilkan hasil MapReducenya dalam terminal menggunakan perintah `hdfs dfs -cat /folder_output_kamu/file_output`, dan pastekan screenshotnya di bawah.



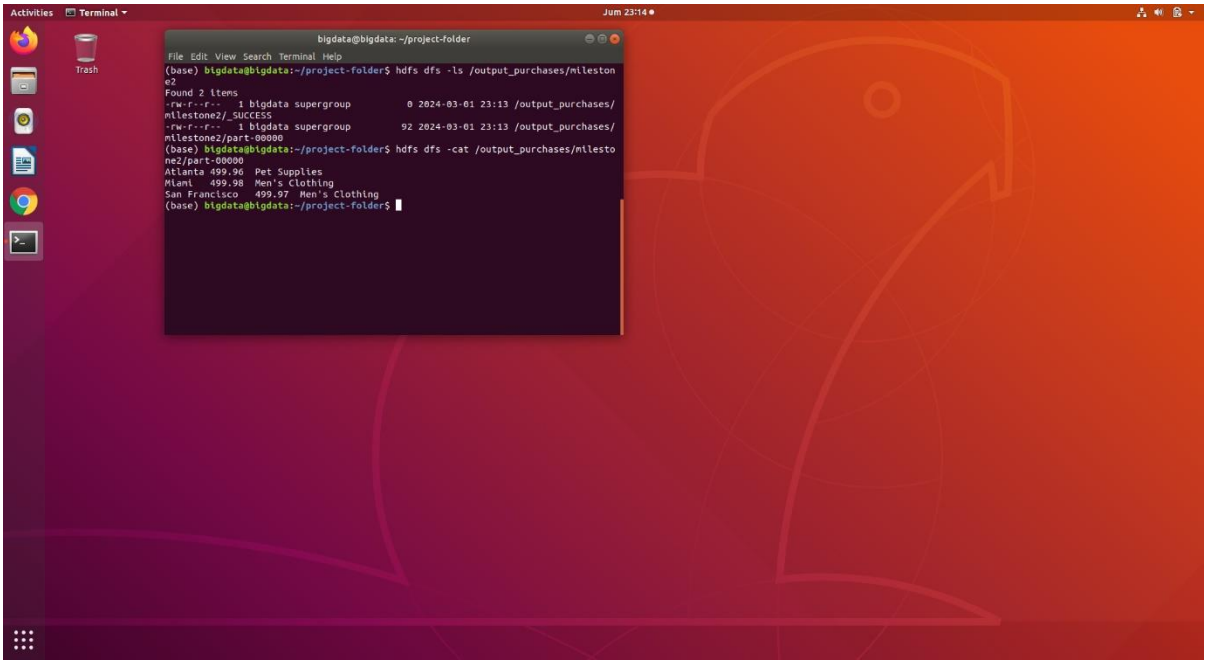
Catatan: semua file python mapper dan reducer yang digunakan selama HandsOn ini, dibutuhkan untuk disubmit. Baca detail format pengumpulannya di bagian paling bawah dari dokumen ini.

B. Milestone 2:

1. Tampilkan nilai penjualan tertinggi beserta item produknya[1] untuk masing-masing toko yang berada di kota: **Miami**, **San Francisco** dan **Atlanta**. Output dari milestone ini adalah sebagai berikut.

Atlanta	499.96	Pet Supplies
Miami	499.98	Men's Clothing
San Francisco	499.97	Men's Clothing

2. Tampilkan hasil MapReducenya dalam terminal menggunakan perintah `hdfs dfs -cat /folder_output_kamu/file_output`, dan pastekan screenshotnya di bawah.



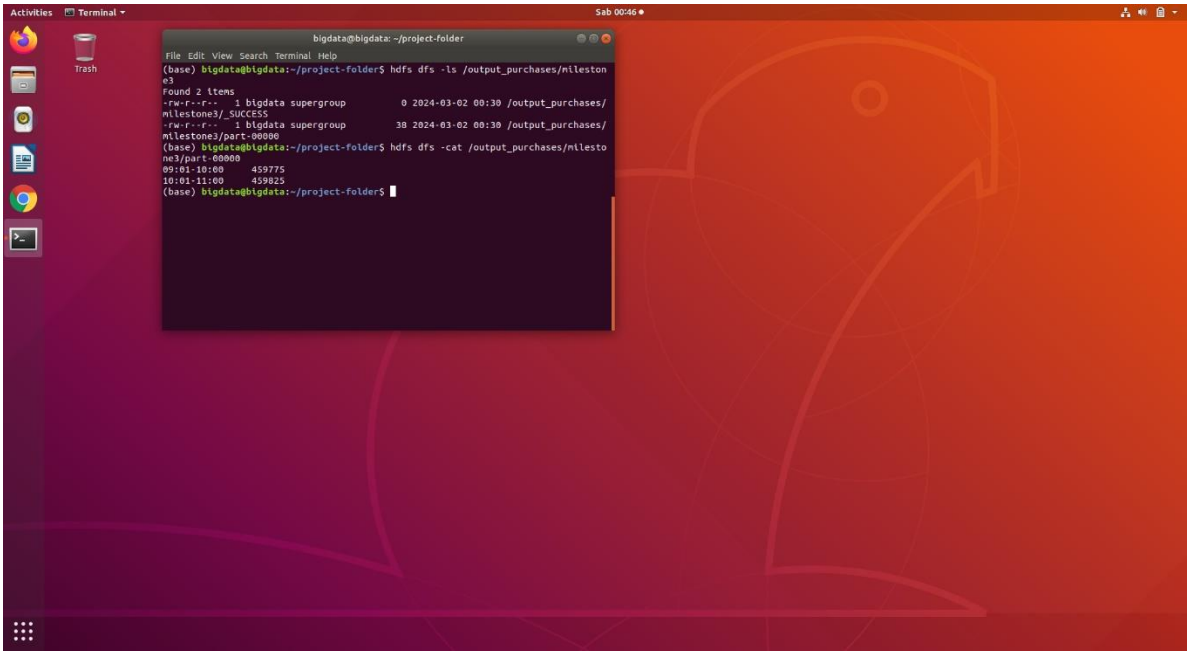
C. Milestone 3:

1. Tampilkan banyaknya penjualan yang terjadi di rentang jam 09:01-10:00 dan jam 10:01-11:00. Output dari milestone ini adalah sebagai berikut.

09:01-10:00	459775
10:01-11:00	459825

3. Tampilkan hasil MapReducenya dalam terminal menggunakan perintah `hdfs dfs -cat /folder_output_kamu/file_output`, dan pastekan screenshotnya di bawah.

[1] Bisa jadi penjualan tertinggi nilainya tidak unik dan terdapat pada beberapa produk. Pada kasus yang demikian, kamu hanya perlu mencantumkan salah satu produknya saja



Pesan antara:

Implementasi MapReduce dengan membuat file kode secara *custom* untuk mapper dan reducer yang dilakukan di atas memberikan keleluasaan programmer untuk mengembangkan programnya. Akan tetapi, hal tersebut memang diperlukan usaha yang relatif besar untuk membawa permasalahan-permasalah yang diberikan ke paradigma "map" dan "reduce". Usaha ini sebanding dengan keuntungan yang bisa kita dapatkan, yaitu mampu mendistribusikan komputasi ke mesin-mesin dalam kluster.

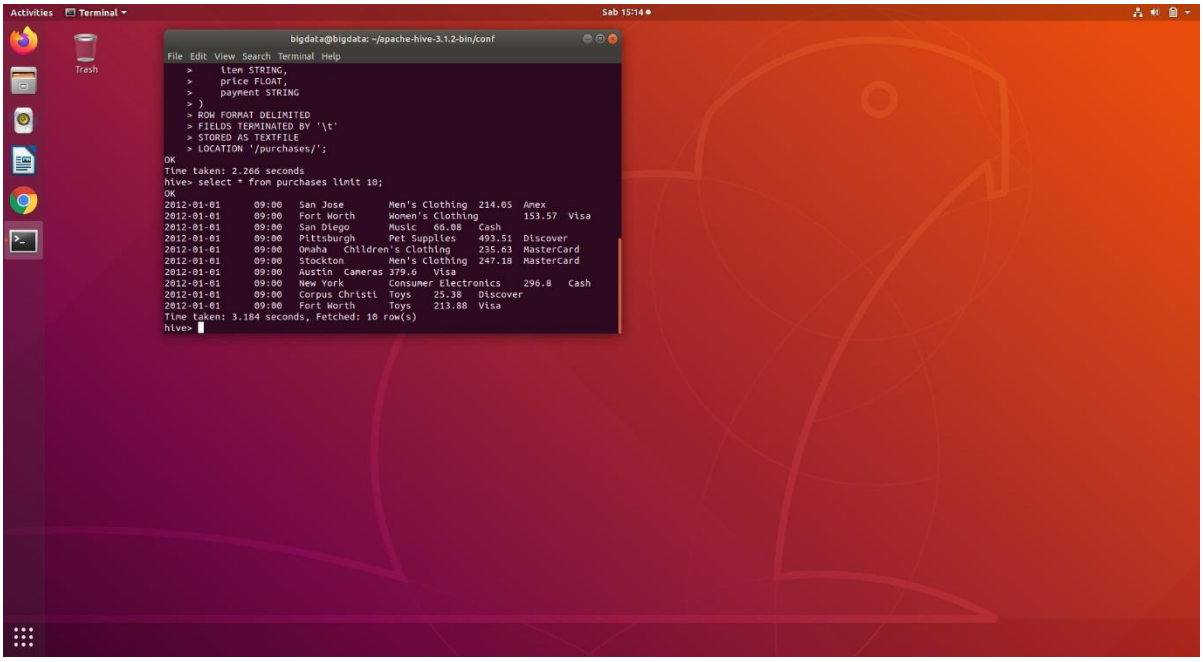
Di dalam ekosistem Hadoop, tersedia sebuah *tool* yang mengubah *SQL-like query* ke komputasi MapReduce yang kemudian dapat didistribusikan ke dalam klaster, yaitu Apache Hive. Dengan menggunakan Apache Hive, seorang programmer dapat mengolah data tabel yang tersimpan (terdistribusi) di klaster layaknya melakukan *query* menggunakan SQL. Sekali lagi, query tersebut kemudian akan dikonversikan ke MapReduce dan akan memproses datanya secara terdistribusi di dalam klaster.

D. Milestone 4

1. Masuk ke Hive dengan cara seperti yang telah dilakukan di HandsOn W04
2. Buatlah tabel "purchases" dari data "purchases.txt" yang telah disimpan di HDFS. Sebagai referensi, untuk membuat tabel "mahasiswa" dari file (dengan tiga kolom, terpisah dengan koma) yang berada di folder HDFS "/mahasiswa", dapat dilakukan dengan kode berikut.

```
CREATE TABLE IF NOT EXISTS mahasiswa(ID int, nama string, ipk float)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE
LOCATION '/mahasiswa/';
```

3. Setelah tabel "purchases" terbuat, tes dengan query `select * from purchases limit 10;`
4. Ambil screenshot (hasil query-nya) dan pastekan di bawah ini.

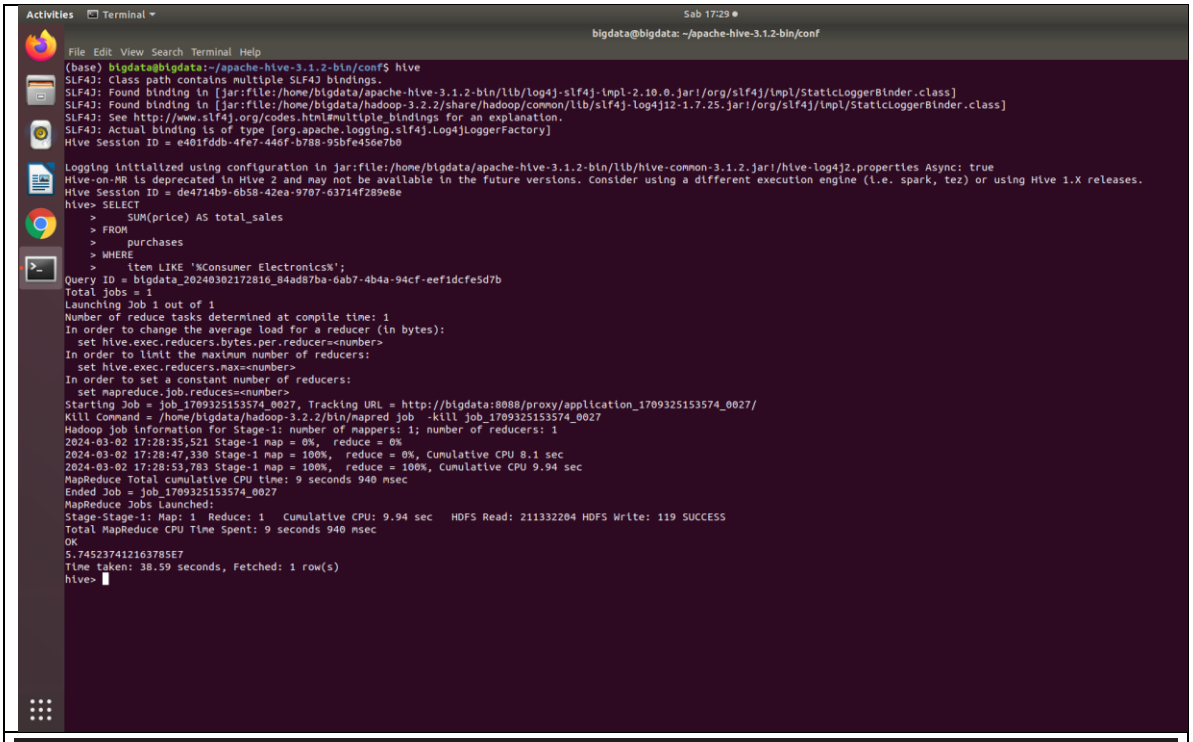


E. Milestone 5

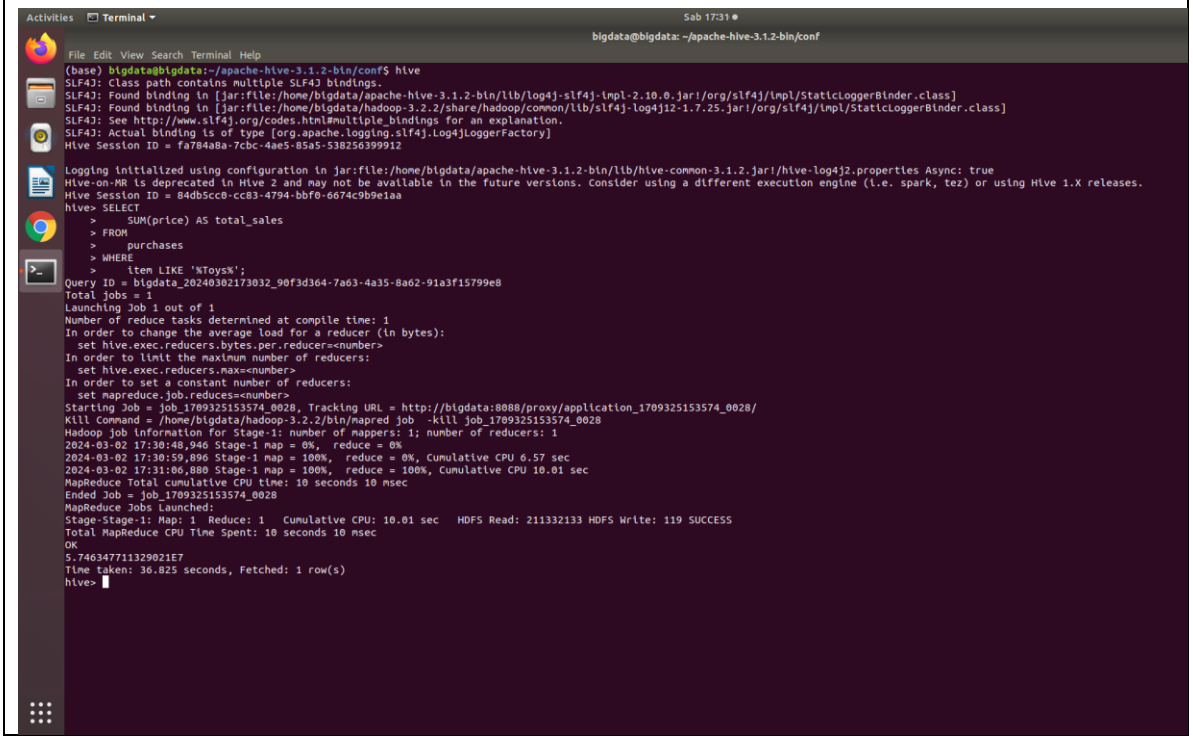
1. Lakukan Milestone 1, akan tetapi menggunakan query Hive dari tabel “puchases” yang telah dibuat.[2]
2. Ambil screenshot (bagian ekspresi SQL dan hasil query-nya), dan pastekan di bawah ini. Hasil Milestone 1 dan 5 seharusnya memberikan keluaran yang sama[3].
   Ekspresi SQL:

```
SELECT
    SUM(price) AS total_sales
FROM
    purchases
WHERE
    item LIKE '%Consumer Electronics%';
```

[2] Bagi VM dengan size RAM kecil, kemungkinan proses akan terhenti di tengah. Jika tidak memungkinan untuk menambahkan size RAM di VM, ambil screenshot “ekspresi SQL yang kamu buat” dan “pesan errornya”.
[3] Hiraukan perbedaan minor string “lowercase” dan “Capital Each Word”.



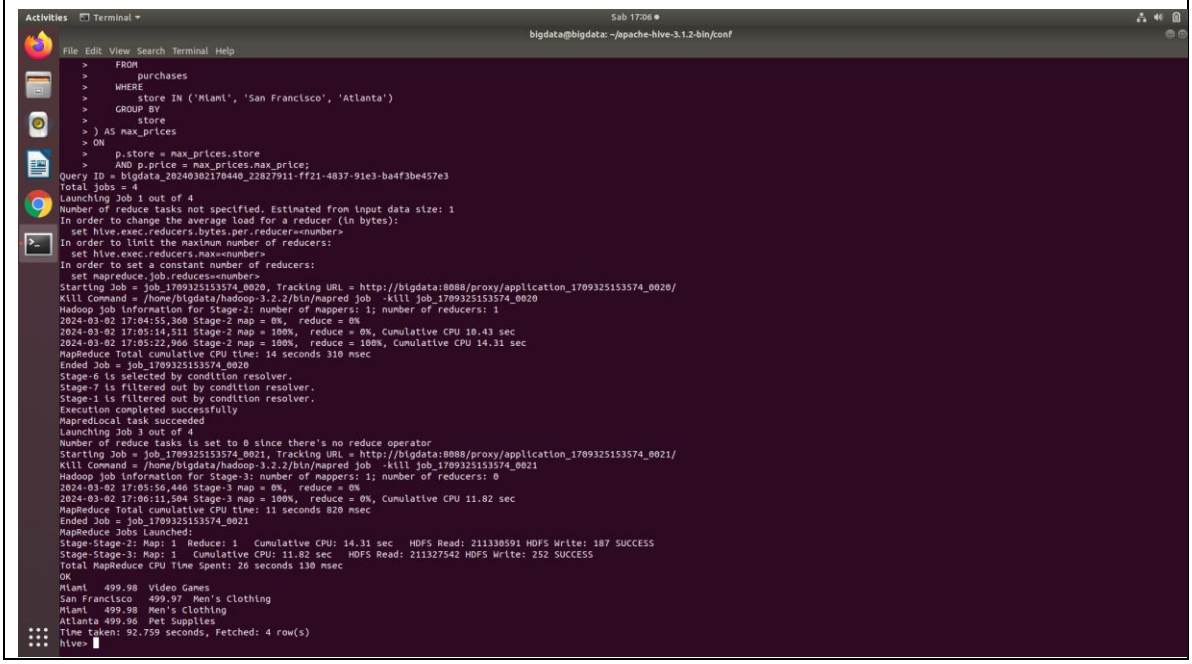

```sql
SELECT
    SUM(price) AS total_sales
FROM
    purchases
WHERE
    item LIKE '%Toys%';
```



F. Milestone 6

1. Lakukan Milestone 2, akan tetapi menggunakan query Hive dari tabel “puchases” yang telah dibuat.
2. Ambil screenshot (bagian ekspresi SQL dan hasil query-nya), dan pastekan di bawah ini. Hasil Milestone 2 dan 6 seharusnya memberikan keluaran yang sama.
   Ekspresi SQL:

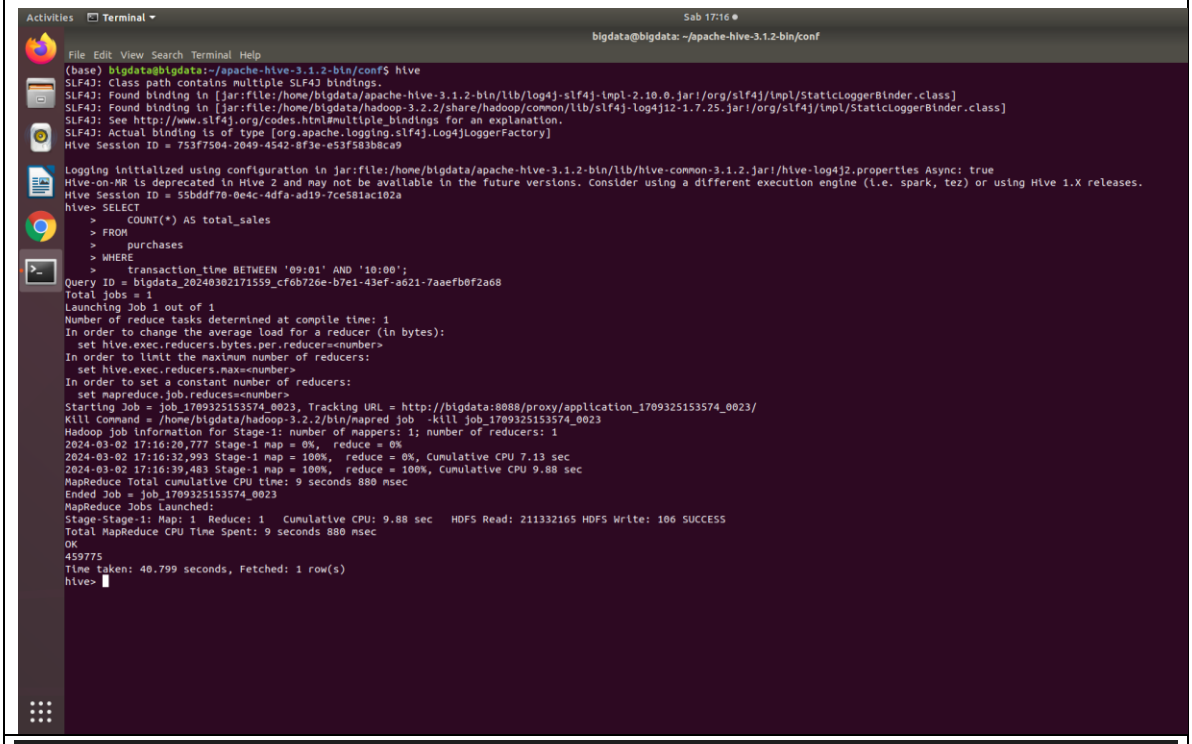
```sql
SELECT
    p.store,
    p.price AS max_price,
    p.item
FROM
    purchases p
JOIN (
    SELECT
        store,
        MAX(price) AS max_price
    FROM
        purchases
    WHERE
        store IN ('Miami', 'San Francisco', 'Atlanta')
    GROUP BY
        store
) AS max_prices
ON
    p.store = max_prices.store
    AND p.price = max_prices.max_price;
```



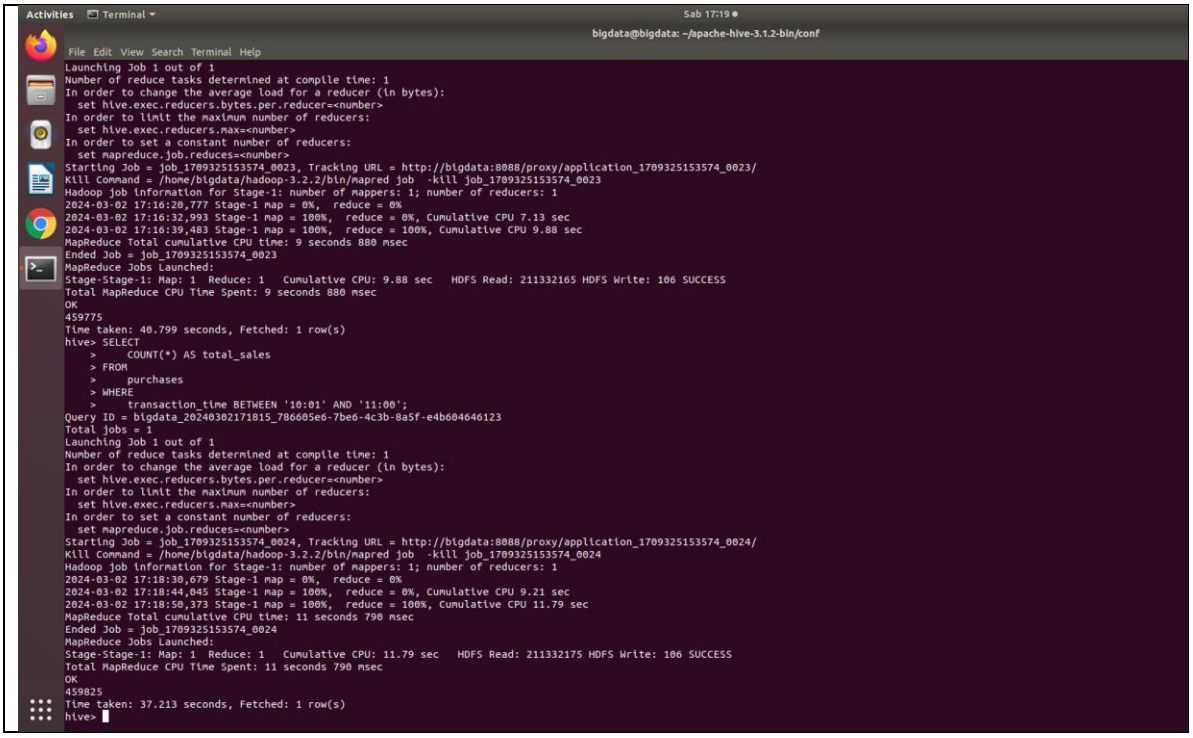
G. Milestone 7

1. Lakukan Milestone 3, akan tetapi menggunakan query Hive dari tabel “puchases” yang telah dibuat.
2. Ambil screenshot (bagian ekspresi SQL dan hasil query-nya), dan pastekan di bawah ini. Hasil Milestone 3 dan 7 seharusnya memberikan keluaran yang sama.
   Ekspresi SQL:

```sql
SELECT
    COUNT(*) AS total_sales
FROM
    purchases
WHERE
    transaction_time BETWEEN '09:01' AND '10:00';
```




```sql
SELECT
    COUNT(*) AS total_sales
FROM
    purchases
WHERE
    transaction_time BETWEEN '10:01' AND '11:00';
```



H. Milestone 8

1. Jika pada Milestone 5, 6 dan 7 hasil yang didapatkan tidak dapat menyamai[4] dari Milestone 1, 2 dan 3, berikan analisa kamu. Jika alasannya adalah terkait keterbatasan SQL-like query, berikan ide/solusinya agar hasil dari Milestone 5, 6 dan 7 secara berturut-turut sama dengan Milestone 1, 2 dan 3.
    - Pada Milestone 1 dan 5 didapatkan hasil yang berbeda, hal ini dikarenakan perbedaan penanganan floating point pada Python dan Hive. Namun, kedua hasilnya saling mendekati sehingga apabila dilakukan pembulatan maka akan menghasilkan nilai yang sama.
    - Pada Milestone 2 dan 6 didapatkan hasil yang sama dimana harga tertinggi untuk Atlanta adalah Pet Supplies, harga tertinggi untuk Miami adalah Men's Clothing, dan harga tertinggi untuk San Fransisco adalah Men's Clothing. Meskipun pada pada Miami terdapat 2 item dengan harga tertinggi yang salah satunya adalah Video Games, namun pada spesifikasi diperbolehkan memilih salah satu item saja sehingga hal ini memenuhi.
    - Pada Milestone 3 dan 7 didapatkan hasil yang sama dimana jumlah penjualan di antara 09.01-10.00 adalah 459775 dan jumlah penjualan di antara 10.01-11.00 adalah 459825.

[4] Hiraukan perbedaan minor string “lowercase” dan “Capital Each Word”.

Setelah semua screenshot dipastekan di masing-masing milestone, upload file zip dengan nama: "W05_NIM_NamaLengkap.zip" ke form submission/assignment di **edunex** yang telah disediakan. Adapun isi dari file zipnya adalah:

1. File pdf dari dokumen ini, dengan nama: "W05_NIM_NamaLengkap.pdf"
2. File mapper dan reducer dari dari Milestone 1-3, dengan format nama "mapper_milestone1.py" dan "reducer_milestone1.py" untuk Milestone 1, begitu seterusnya hingga Milestone 3.

--- done ---