**Hive Assignment 2**

Will the reducer work or not if you use “Limit 1” in any HiveQL query?

In Hive, when you use the "LIMIT 1" clause in a query, it restricts the result set to only return one row.

Whether the reducer will work or not depends on the specific scenario and the execution plan chosen by Hive.

If the query involves aggregations, sorting, or other operations that require data shuffling and redistribution among reducers, the reducer will still be involved in the processing. In this case, the reducer will handle the reduced amount of data resulting from the "LIMIT 1" clause.

However, if the query does not require any data shuffling or redistribution, such as a simple "SELECT" query without any aggregations or sorting, Hive may optimize the query execution by using a Map-only job. In this case, the reducer phase might not be involved at all.

In summary, whether the reducer works or not when using "LIMIT 1" in HiveQL depends on the complexity of the query, the specific execution plan chosen by Hive, and whether data shuffling or redistribution is required for the query.

Suppose I have installed Apache Hive on top of my Hadoop cluster using default metastore configuration. Then, what will happen if we have multiple clients trying to access Hive at the same time?

If you have Apache Hive installed on a Hadoop cluster with the default metastore configuration and multiple clients trying to access Hive simultaneously, the following would typically occur:

1.Metastore: The Hive metastore is a central component that stores metadata information about tables, partitions, columns, and other Hive objects. The default configuration uses a relational database (such as MySQL or PostgreSQL) as the metastore backend.

2.Concurrent Access: When multiple clients try to access Hive simultaneously, they will interact with the metastore to perform various operations like creating tables, querying data, modifying schemas, etc. The metastore is designed to handle concurrent access from multiple clients.

3.Concurrency Control: The underlying database used as the metastore backend should handle concurrency control mechanisms to ensure data consistency and prevent conflicts between concurrent transactions. The specific behavior and performance will depend on the concurrency control mechanism employed by the database system.

4.Locking: Hive uses locking mechanisms to control access to metadata and prevent conflicts. When clients perform certain operations, such as altering table structures or modifying data, Hive may acquire locks to ensure that only one client can make changes at a time.

5.Connection Pooling: Hive uses connection pooling to efficiently manage connections to the metastore database. This allows multiple clients to share and reuse connections, minimizing the overhead of establishing new connections for each client.

Overall, with the default metastore configuration in Apache Hive, multiple clients can access Hive concurrently. The concurrency control mechanisms and locking provided by the underlying database system, along with connection pooling, help ensure that multiple clients can interact with Hive and perform operations simultaneously while maintaining data consistency and avoiding conflicts

Suppose, I create a table that contains details of all the transactions done by the customers: CREATE TABLE transaction\_details (cust\_id INT, amount FLOAT, month STRING, country STRING) ROW FORMAT DELIMITED FIELDS TERMINATED BY ‘,’ ;

Now, after inserting 50,000 records in this table, I want to know the total revenue generated for each month. But, Hive is taking too much time in processing this query. How will you solve this problem and list the steps that I will be taking in order to do so?

By partitioning the table, bucketing the data, and optimizing the query, you can significantly improve the processing time for calculating the total revenue generated for each month in Hive. These techniques help reduce the amount of data processed and improve data locality, leading to more efficient query execution.

How can you add a new partition for the month December in the above partitioned table?

ALTER TABLE transaction\_details ADD PARTITION (month='Dec');

I am inserting data into a table based on partitions dynamically. But, I received an error – FAILED ERROR IN SEMANTIC ANALYSIS: Dynamic partition strict mode requires at least one static partition column. How will you remove this error?

set hive.exec.dynamic.partition.mode=nonstrict;

Suppose, I have a CSV file – ‘sample.csv’ present in ‘/temp’ directory with the following entries:

id first\_name last\_name email gender ip\_address

How will you consume this CSV file into the Hive warehouse using built-in SerDe?

CREATE TABLE sample\_table (

id INT,

first\_name STRING,

last\_name STRING,

email STRING,

gender STRING,

ip\_address STRING

)

ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'

WITH SERDEPROPERTIES (

"separatorChar" = ",",

"quoteChar" = "\""

)

STORED AS TEXTFILE;

Suppose, I have a lot of small CSV files present in the input directory in HDFS and I want to create a single Hive table corresponding to these files. The data in these files are in the format: {id, name, e-mail, country}. Now, as we know, Hadoop performance degrades when we use lots of small files.

So, how will you solve this problem where we want to create a single Hive table for lots of small files without degrading the performance of the system?

To solve the problem of having lots of small CSV files in HDFS and create a single Hive table without degrading system performance, you can follow these steps:

1.Combine the small CSV files: Use Hadoop's **getmerge** command to merge the small CSV files into a larger consolidated file. This command concatenates the contents of multiple files into a single file. For example:

hadoop fs -getmerge /input\_directory/\*.csv /temp/merged.csv

This command retrieves all the CSV files from the **/input\_directory** in HDFS, merges them into a single file named **merged.csv**, and stores it in the **/temp** directory.

Create an external table in Hive: Create an external table in Hive that points to the merged CSV file. This allows Hive to access the data without physically moving or duplicating the data.

LOAD DATA LOCAL INPATH ‘Home/country/state/’

OVERWRITE INTO TABLE address;

The following statement failed to execute. What can be the cause?

By verifying the path, permissions, file existence, and format compatibility, you can identify the cause of the failure and address the issue accordingly.

Is it possible to add 100 nodes when we already have 100 nodes in Hive? If yes, how?

In Hive, the number of nodes in a cluster is typically managed by the underlying Hadoop cluster manager, such as YARN or Apache Mesos. Adding nodes to the cluster is a configuration and management task that is handled at the Hadoop cluster level, rather than within Hive itself.

If you have an existing Hive cluster with 100 nodes and you want to add 100 more nodes to the cluster, you would need to follow the appropriate procedures for your specific Hadoop cluster manager. Here are some general steps you may need to take:

1. Provision new nodes: Set up and configure the additional nodes that you want to add to the cluster. This typically involves installing the necessary software, configuring network settings, and ensuring connectivity to the existing cluster.
2. Add nodes to the cluster manager: Use the administration tools provided by your cluster manager (e.g., YARN or Mesos) to add the new nodes to the cluster. This may involve updating configuration files, modifying cluster settings, or using specific commands or APIs provided by the cluster manager.
3. Configure Hive: Once the new nodes are successfully added to the cluster, you may need to update Hive's configuration to take advantage of the increased resources. This may involve adjusting resource allocation, parallelism settings, or other Hive-specific configurations.
4. Restart necessary services: Depending on your cluster manager and configuration, you may need to restart relevant services (such as YARN or Hive services) to apply the changes and ensure that the new nodes are recognized by the cluster.

It's important to note that adding nodes to a cluster requires proper planning and coordination to ensure seamless integration with the existing infrastructure. It's recommended to consult the documentation and resources specific to your Hadoop cluster manager for detailed instructions on adding nodes to the cluster.

**Hive Practical questions:**

Hive Join operations

Create a table named CUSTOMERS(ID | NAME | AGE | ADDRESS | SALARY)

Create a Second table ORDER(OID | DATE | CUSTOMER\_ID | AMOUNT

)

Now perform different joins operations on top of these tables

(Inner JOIN, LEFT OUTER JOIN ,RIGHT OUTER JOIN ,FULL OUTER JOIN)

CREATE TABLE CUSTOMERS (

ID INT ,

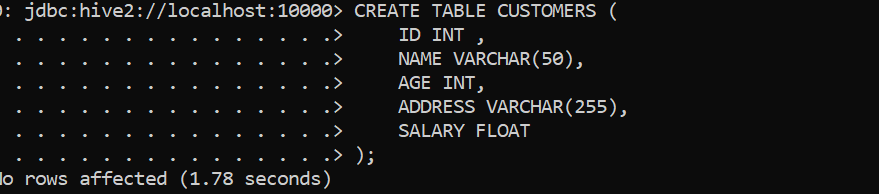
NAME VARCHAR(50),

AGE INT,

ADDRESS VARCHAR(255),

SALARY FLOAT

);



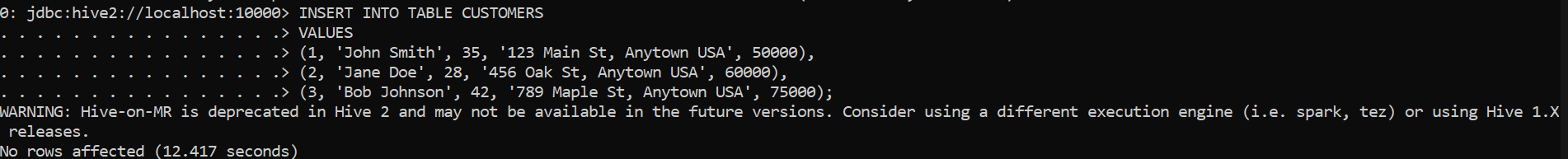
INSERT INTO TABLE CUSTOMERS

VALUES

(1, 'John Smith', 35, '123 Main St, Anytown USA', 50000),

(2, 'Jane Doe', 28, '456 Oak St, Anytown USA', 60000),

(3, 'Bob Johnson', 42, '789 Maple St, Anytown USA', 75000);



CREATE TABLE ORDERS (

OID INT,

DATE\_CREATED DATE,

CUSTOMER\_ID INT,

AMOUNT FLOAT

);



INSERT INTO TABLE ORDERS

VALUES

(1, '2022-01-01', 1, 100.00),

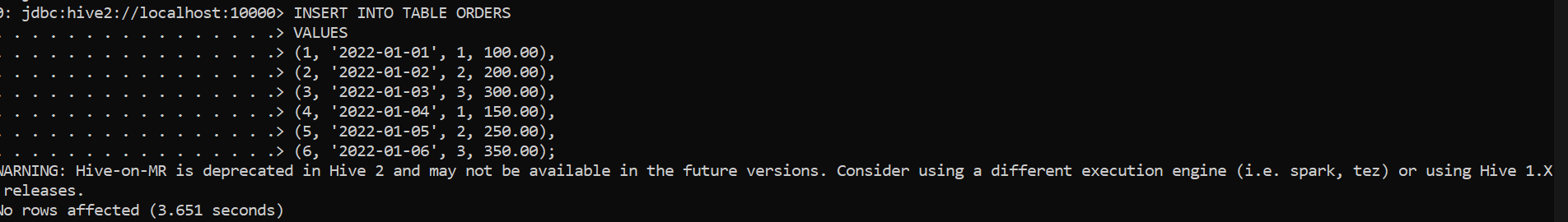
(2, '2022-01-02', 2, 200.00),

(3, '2022-01-03', 3, 300.00),

(4, '2022-01-04', 1, 150.00),

(5, '2022-01-05', 2, 250.00),

(6, '2022-01-06', 3, 350.00);

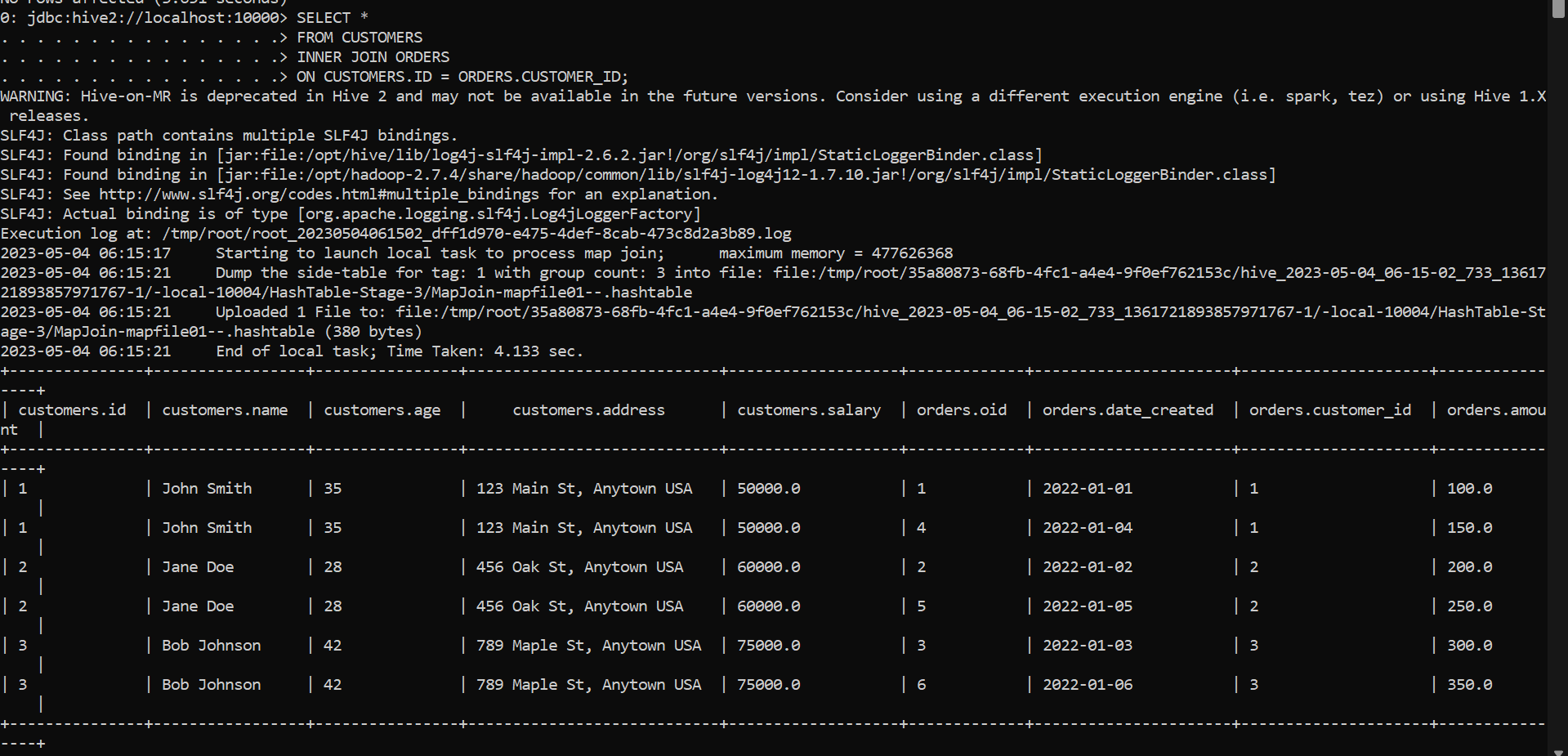


SELECT \*

FROM CUSTOMERS

INNER JOIN ORDERS

ON CUSTOMERS.ID = ORDERS.CUSTOMER\_ID;

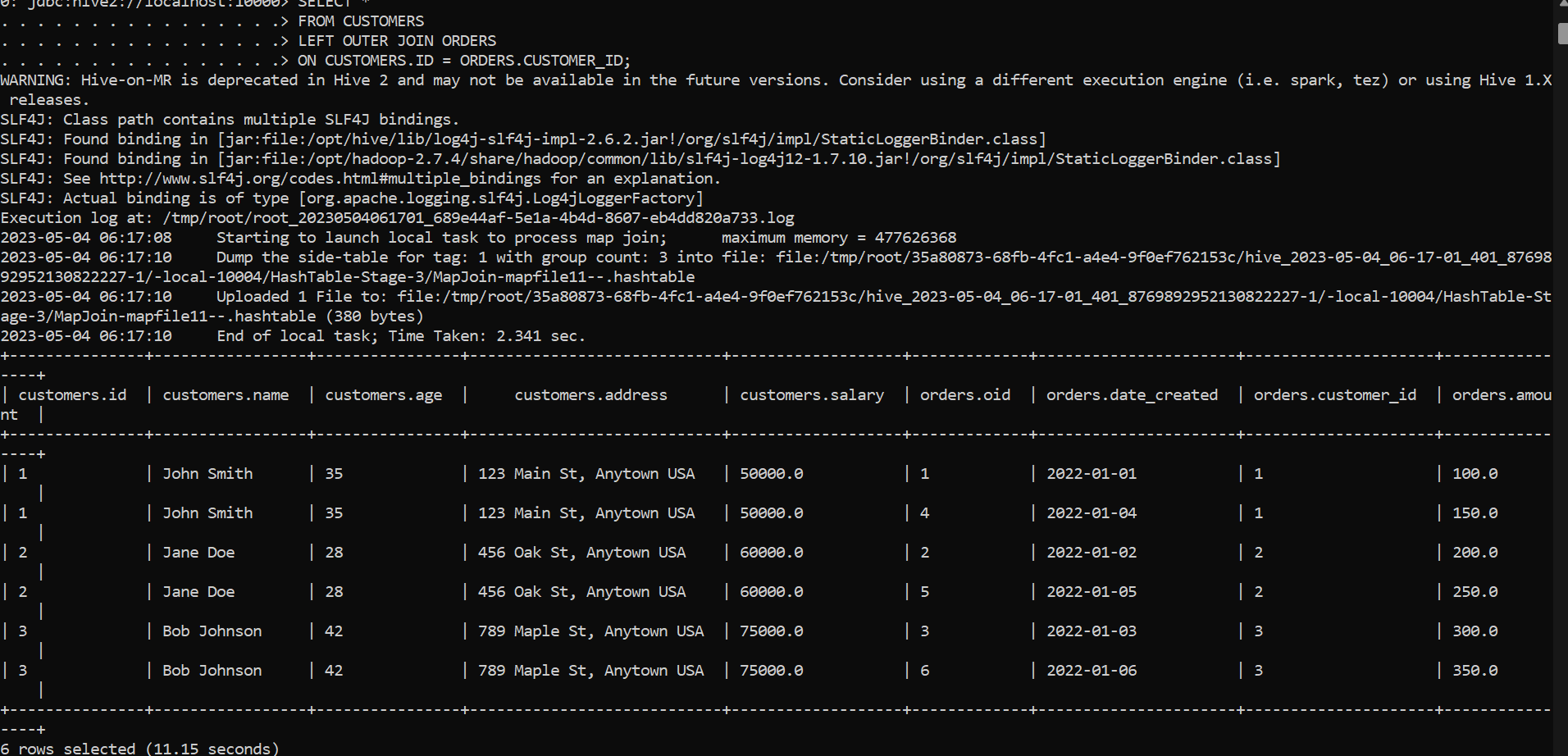


SELECT \*

FROM CUSTOMERS

LEFT OUTER JOIN ORDERS

ON CUSTOMERS.ID = ORDERS.CUSTOMER\_ID;

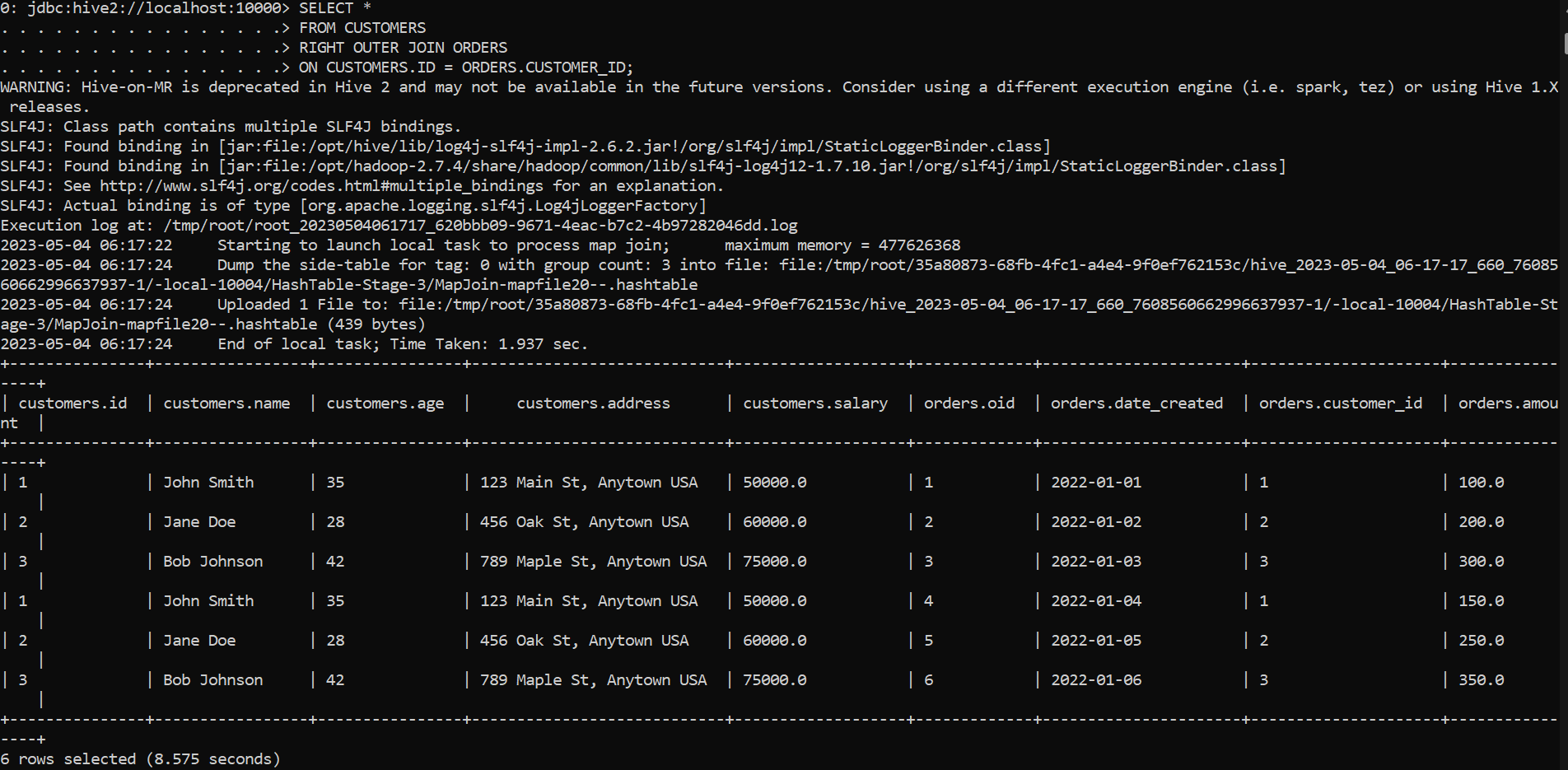


SELECT \*

FROM CUSTOMERS

RIGHT OUTER JOIN ORDERS

ON CUSTOMERS.ID = ORDERS.CUSTOMER\_ID;

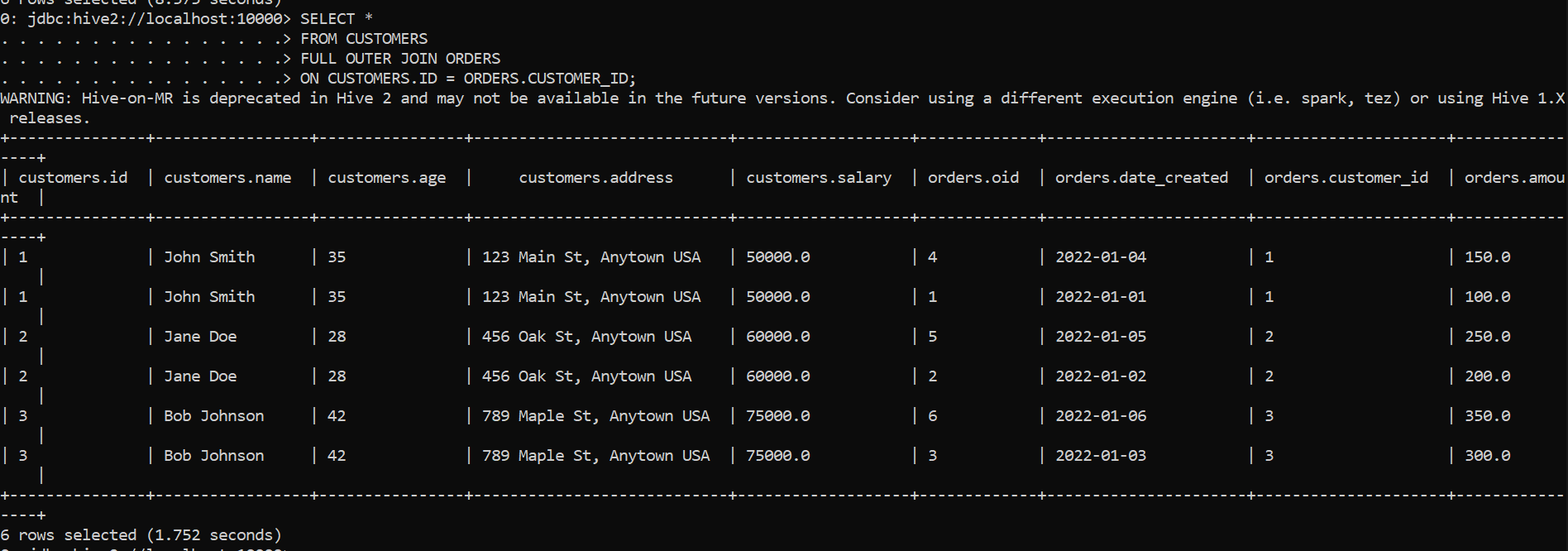


SELECT \*

FROM CUSTOMERS

FULL OUTER JOIN ORDERS

ON CUSTOMERS.ID = ORDERS.CUSTOMER\_ID;



BUILD A DATA PIPELINE WITH HIVE

Download a data from the given location -

https://archive.ics.uci.edu/ml/machine-learning-databases/00360/

1. Create a hive table as per given schema in your dataset

2. try to place a data into table location

3. Perform a select operation

CREATE EXTERNAL TABLE airdata (

new\_date STRING,

time string,

abc FLOAT,

def INT,

ghi INT,

jkl FLOAT,

mno INT,

pqr INT,

stu INT,

vwx INT,

yza INT,

bcd INT,

efg FLOAT,

hij FLOAT,

klm FLOAT

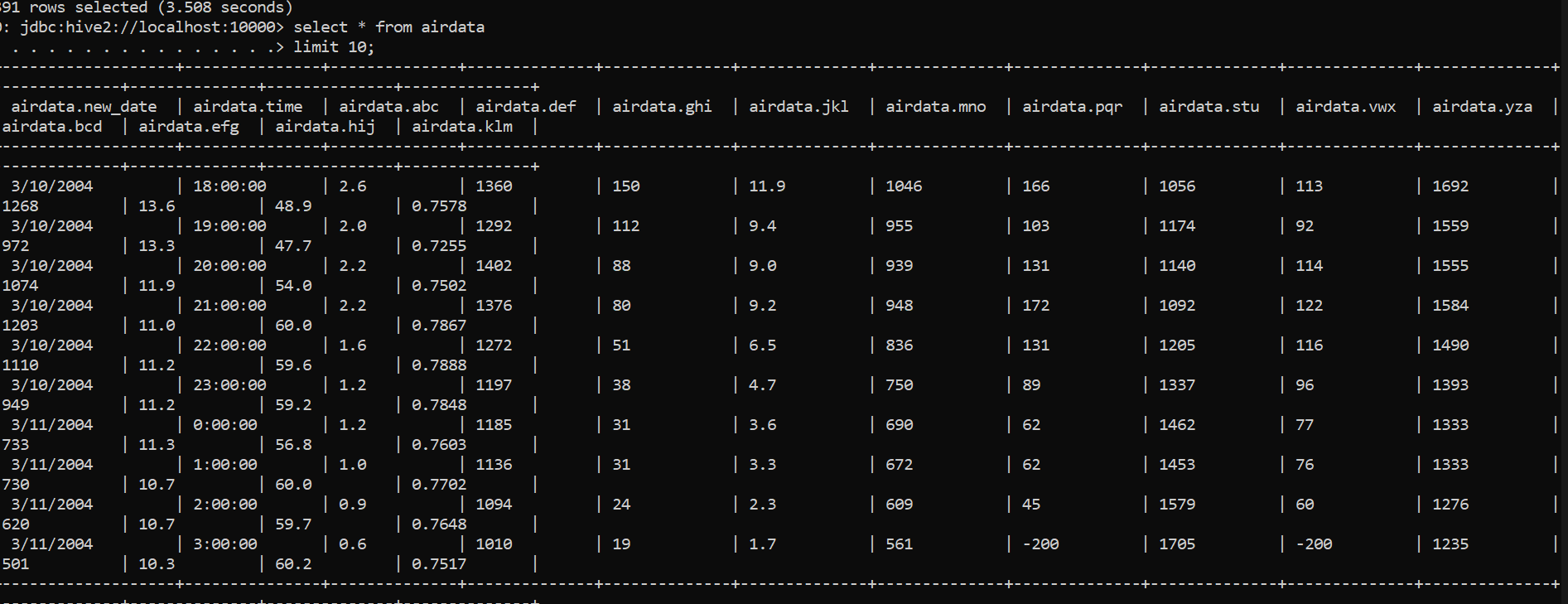
)

row format delimited

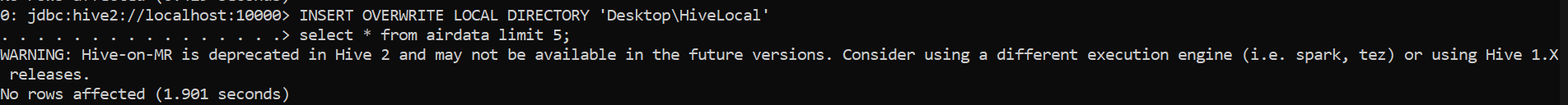
fields terminated by ','

LOCATION '/hive\_data/air'

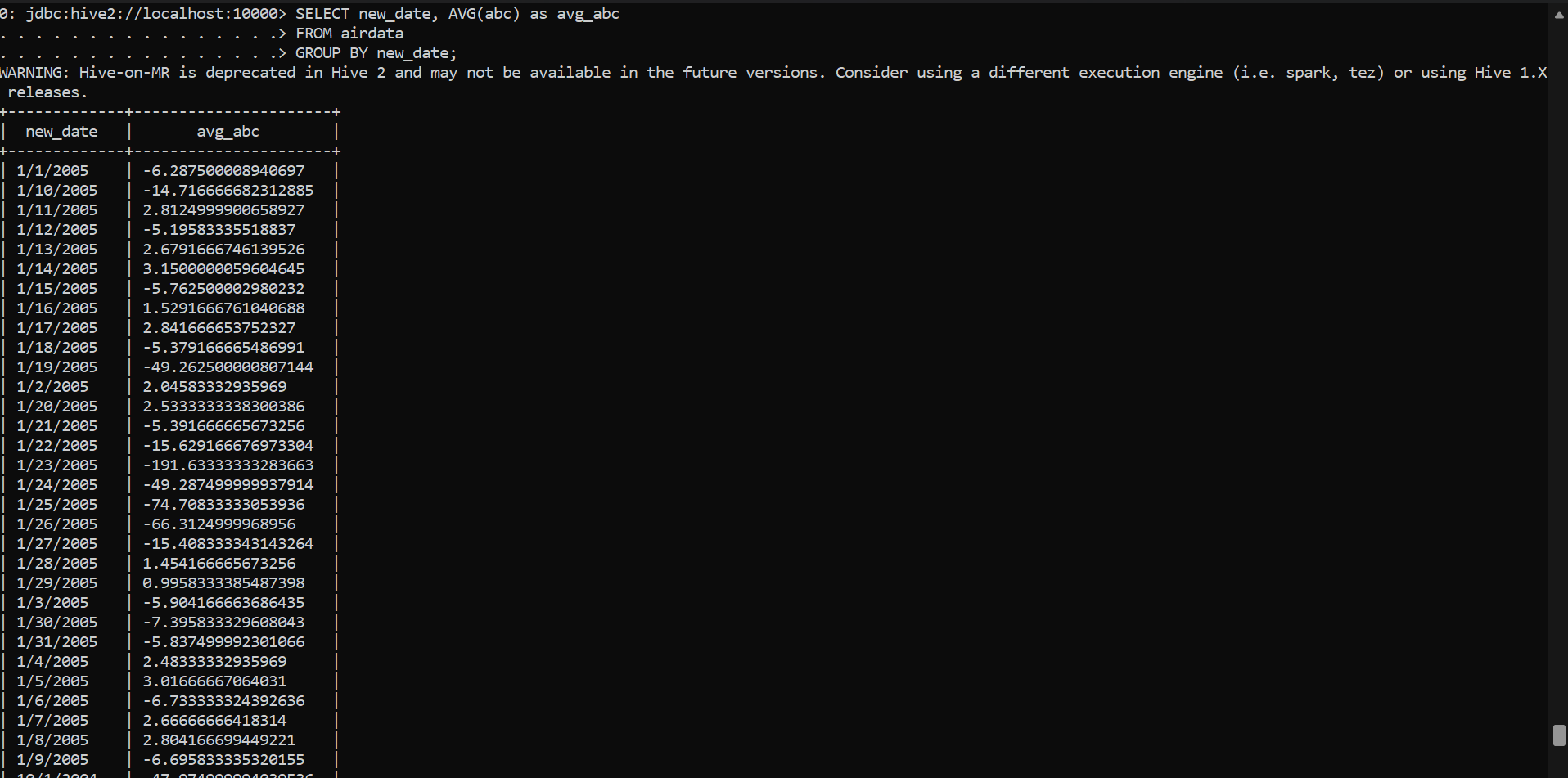
TBLPROPERTIES ("skip.header.line.count"="1");



4. Fetch the result of the select operation in your local as a csv file .

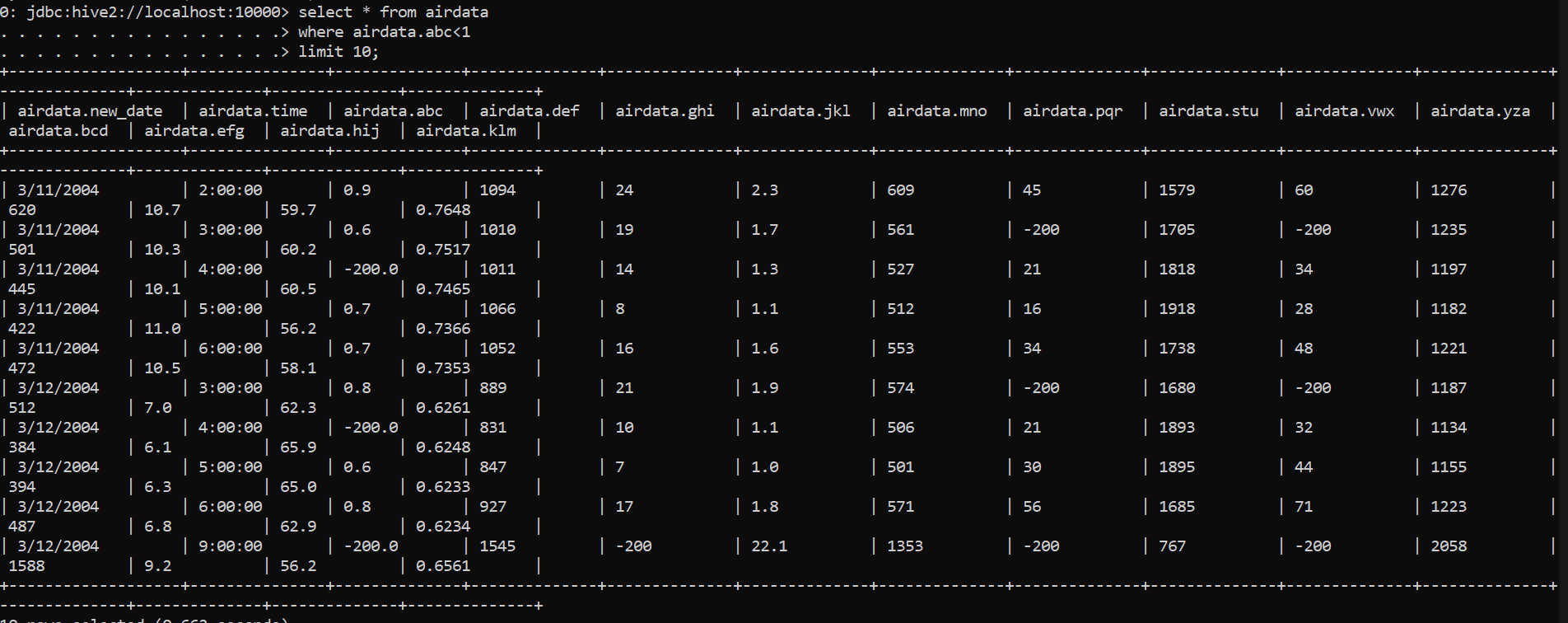


5. Perform group by operation .

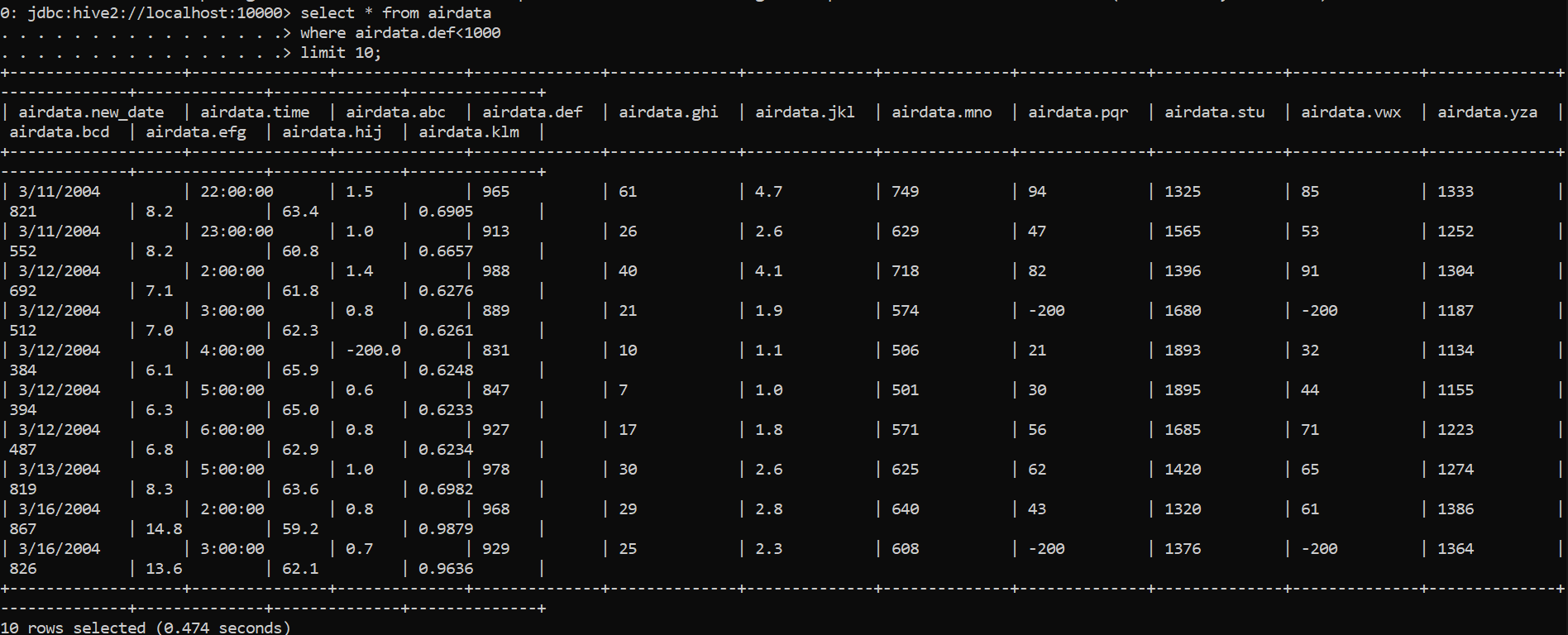


7. Perform filter operation at least 5 kinds of filter examples .

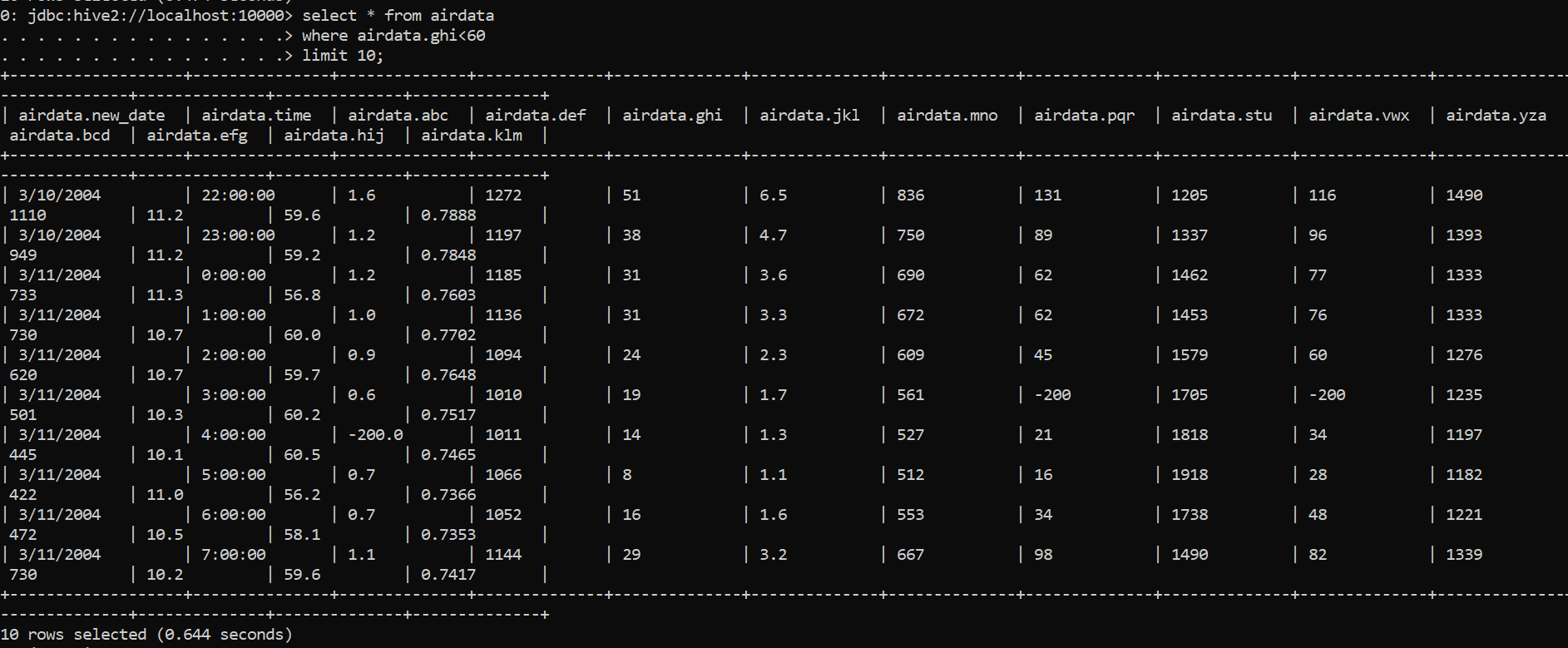
Filter operation 1



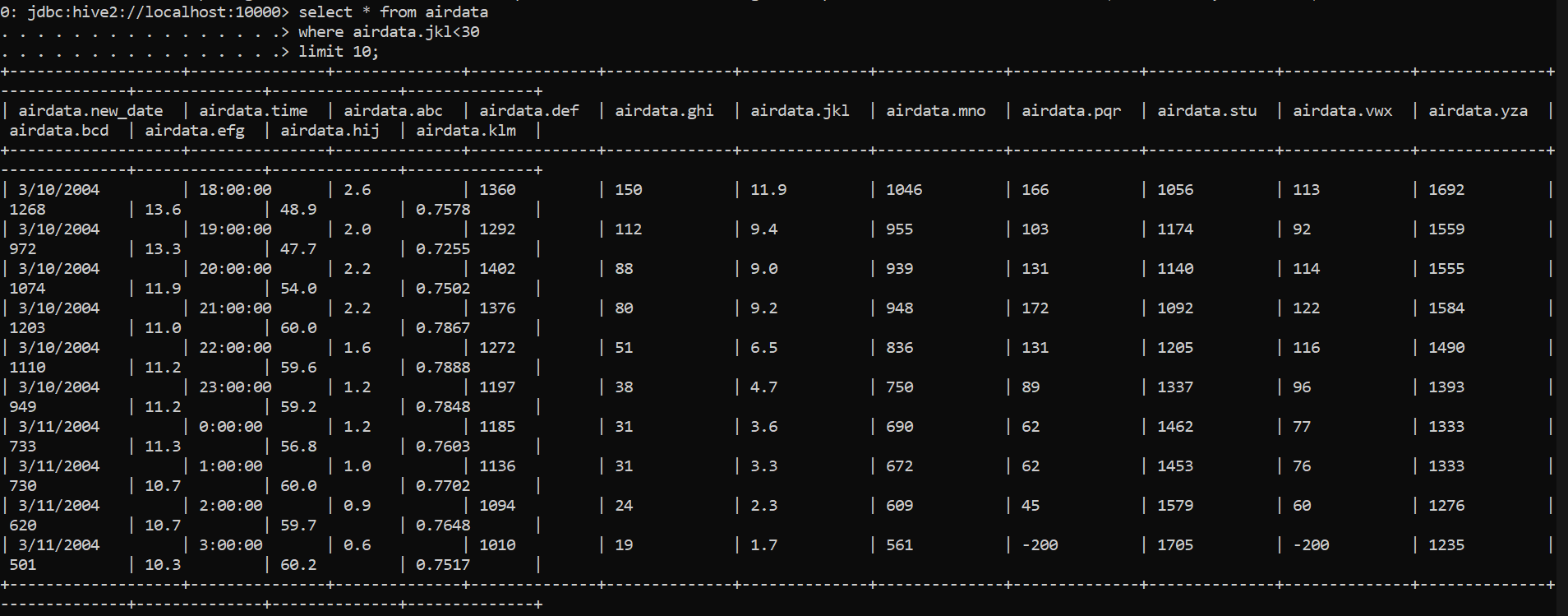
Filter operation 2



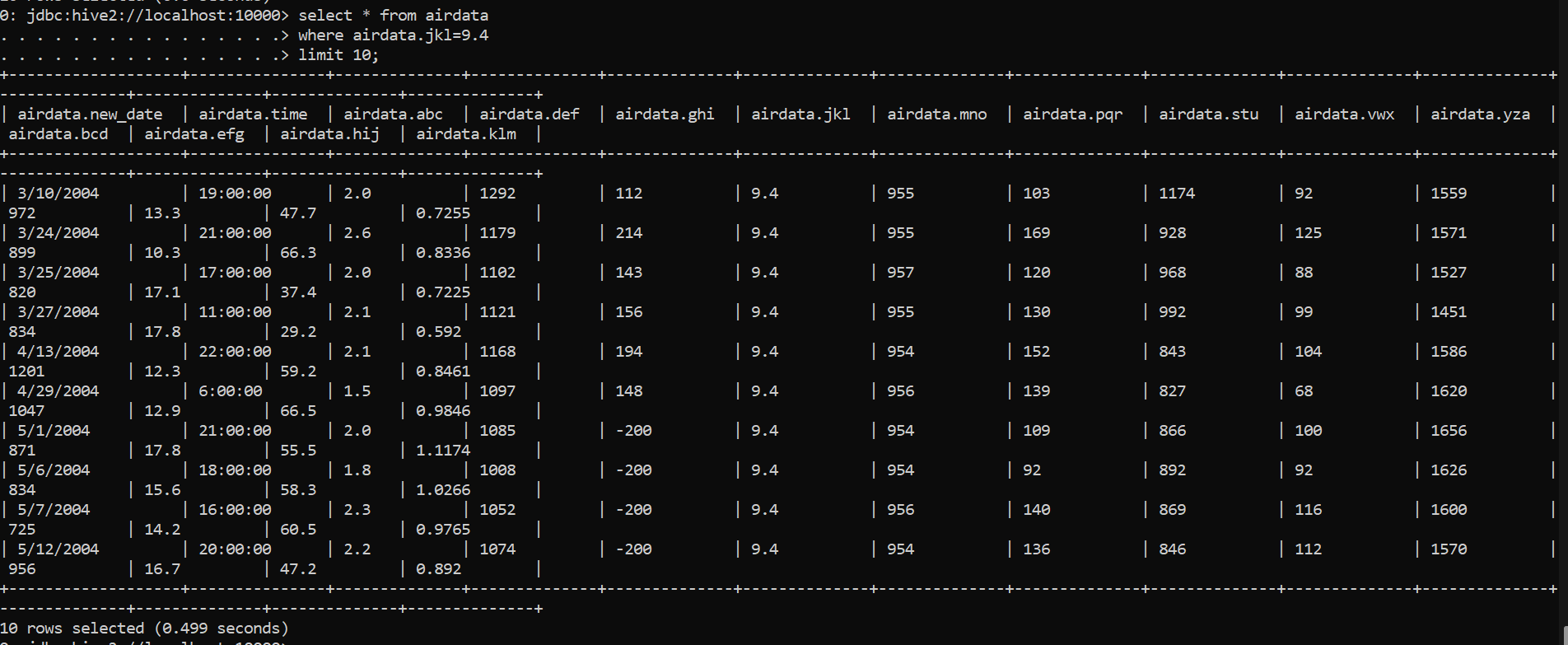
Filter Operation 3



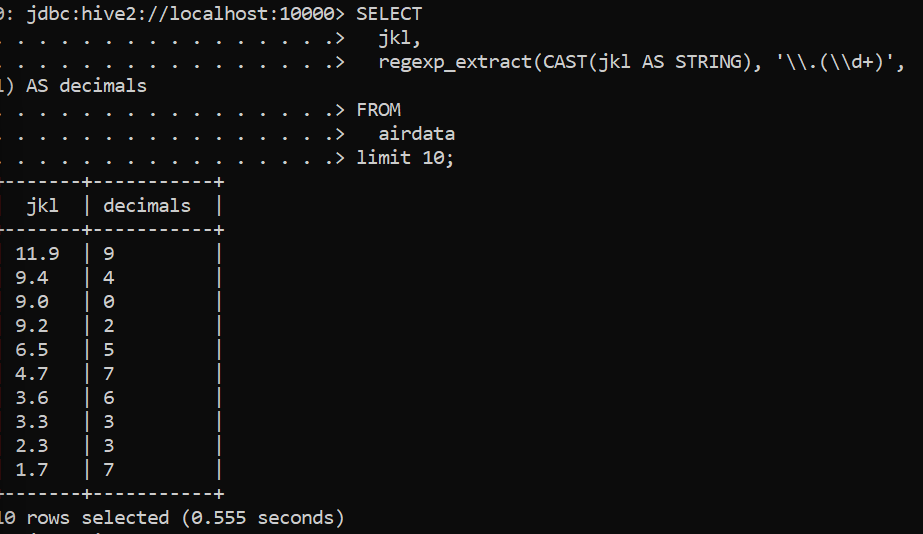
Filter operation 4



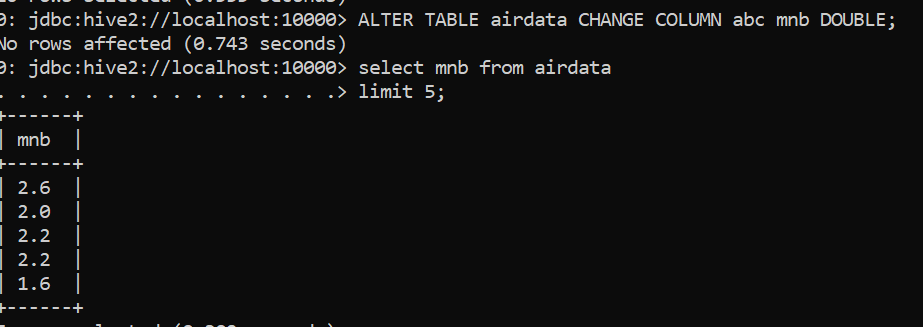
Filter Operation 5



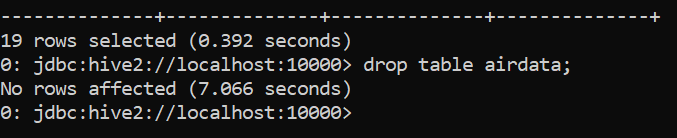
8. show and example of regex operation

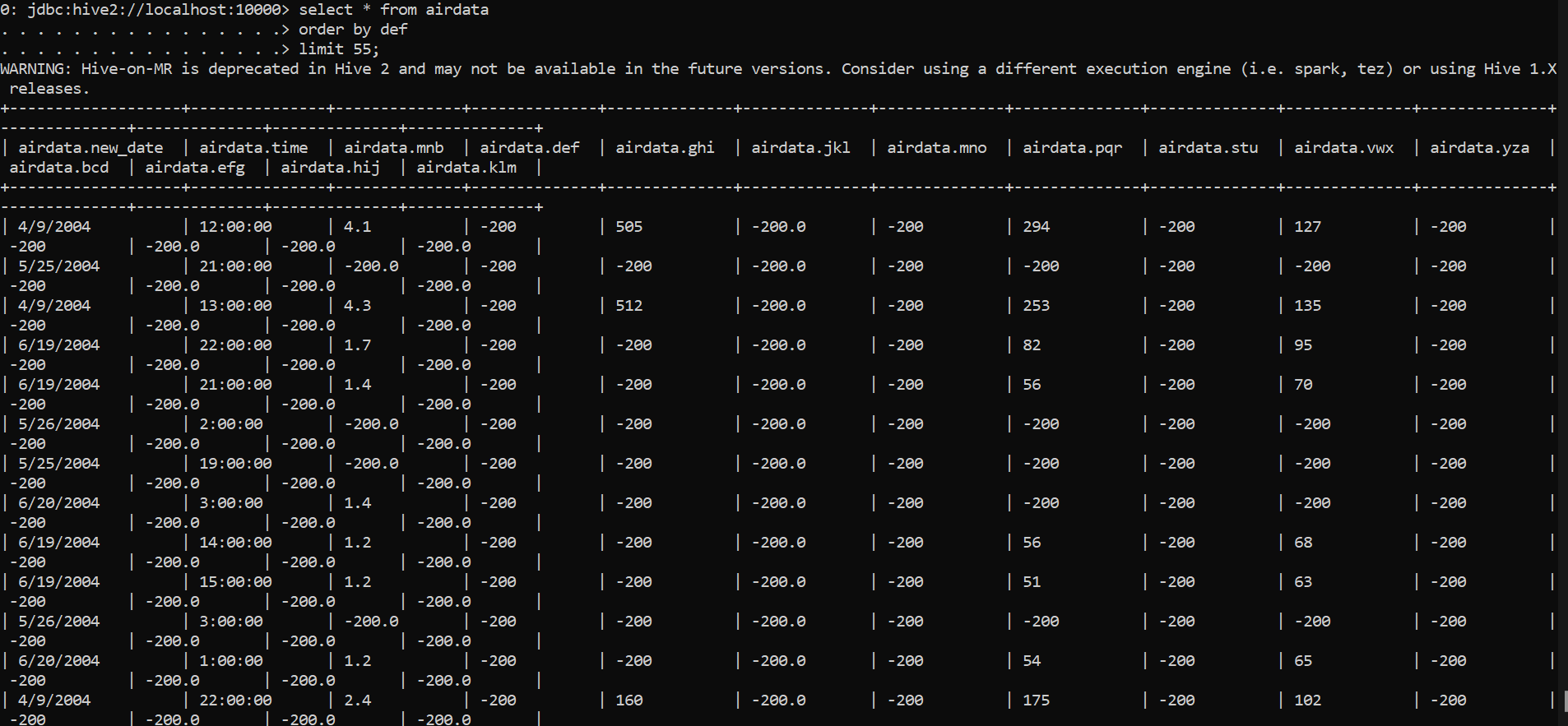


9. alter table operation

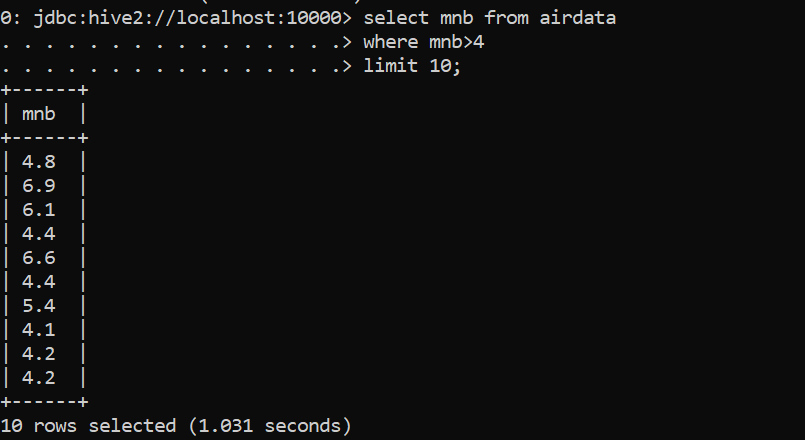


10 . drop table operation

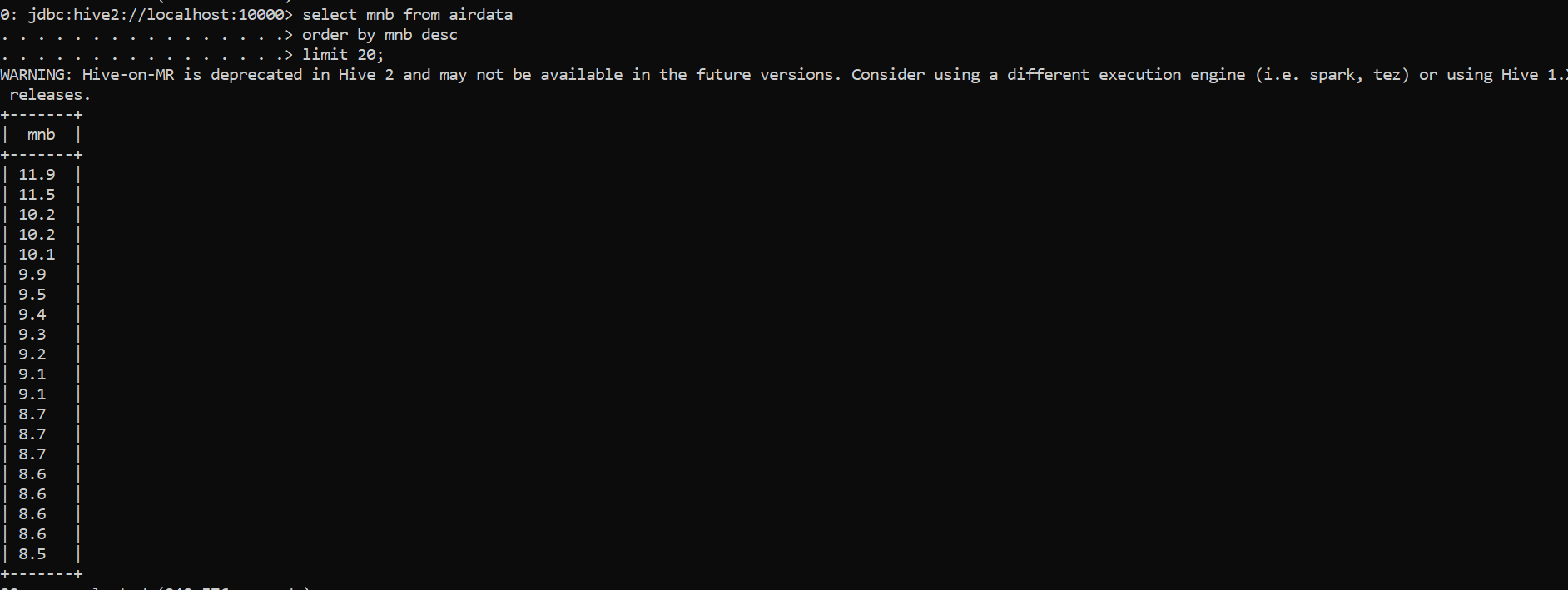


12 . order by operation . 

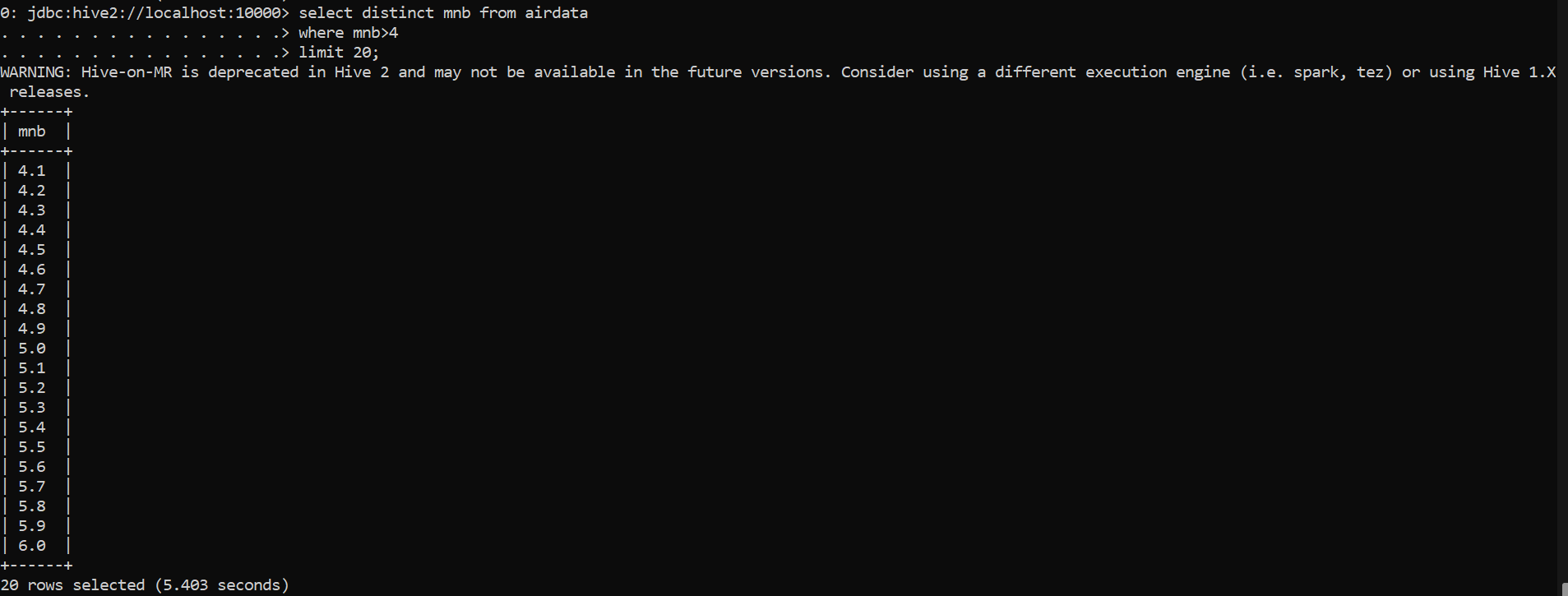
13 . where clause operations you have to perform .



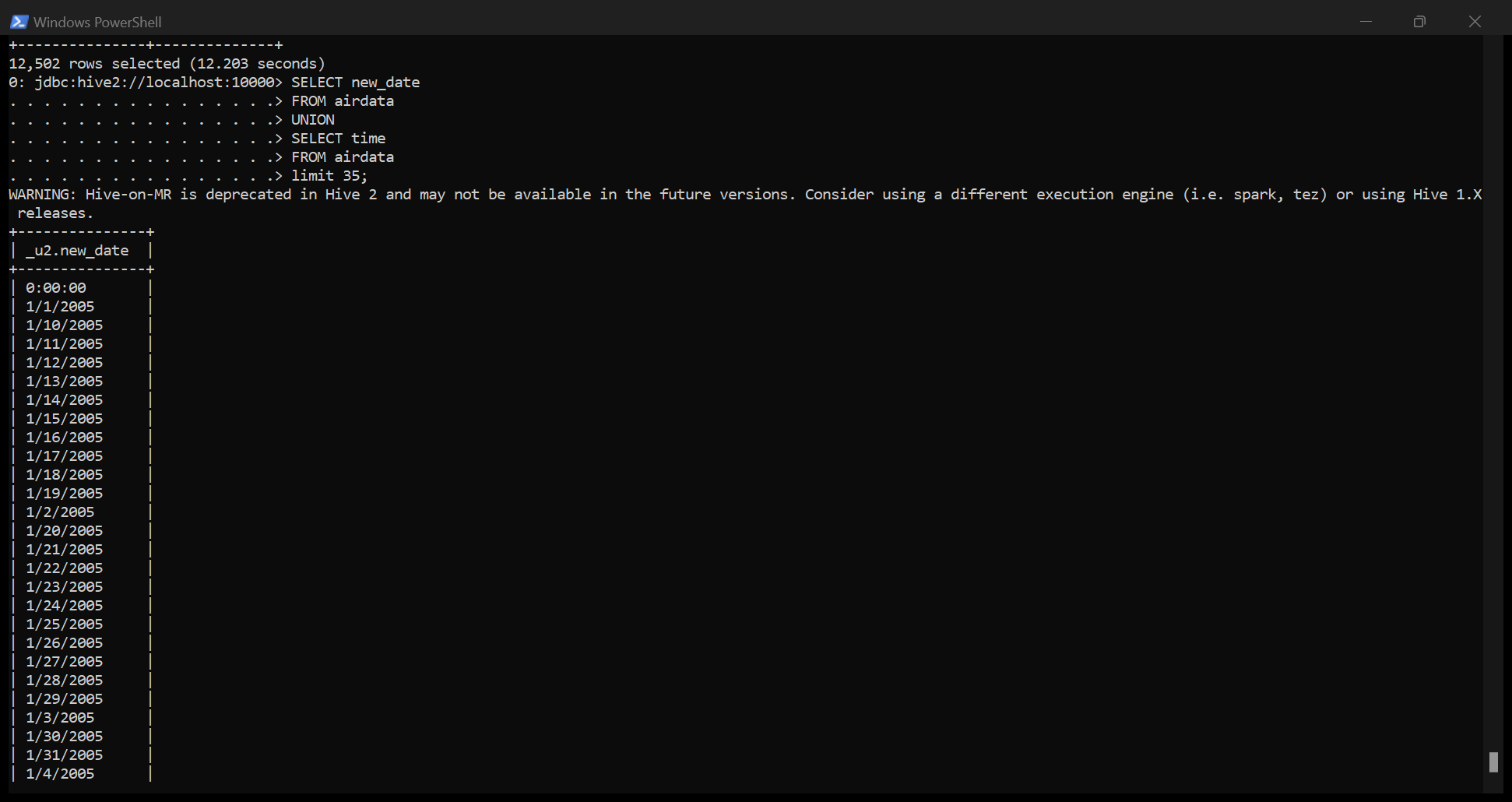
14 . sorting operation you have to perform .



15 . distinct operation you have to perform .



17 . union operation you have to perform .



18 . table view operation you have to perform .

