Chaitanya Durgesh Nynavarapu
A20561894

ASSIGNMENT-4
BIG DATA (CSP 554)

Exercise 1: Summarizing an Article

Overview

This research evaluated different data organization strategies for Hive-based Big Data Warehouses (BDWs), focusing on partitioning and bucketing techniques. The researchers tested various scenarios using the Star Schema Benchmark (SSB) with different scale factors (30GB, 100GB, 300GB) on both star schema and denormalized table models.

Key Findings
Partitioning Strategies:

- Multiple partitioning showed significant benefits, with processing time decreases of up to 46% in Presto and 42% in Hive for star schemas, and up to 54% in Presto and 45% in Hive for denormalized tables.
- Partitioning by attributes frequently used in query filters (e.g., year and region) proved most effective.

Bucketing Strategies:

- Simple bucketing strategies generally did not improve performance, often increasing processing times
- Bucketing with sorting showed some benefits, particularly for queries using the sorted attribute in GROUP BY and ORDER BY clauses
- Bucketing by join keys in star schemas improved performance significantly in Hive (up to 63% decrease in processing time for SF=300), but not in Presto
- Multiple bucketing did not show any performance benefits and often degraded performance

Combined Strategies:

- Combining partitioning and bucketing showed mixed results, with some scenarios providing benefits while others did not.

CPU Usage:

- CPU usage generally correlated with query processing times, with effective strategies reducing both processing time and CPU usage.

Conclusions and Recommendations

1. Partitioning is generally beneficial for BDWs, especially when aligned with query patterns.
2. Bucketing can be advantageous in specific scenarios, particularly for join operations in Hive, but should be used cautiously.
3. The researchers emphasize the importance of carefully analyzing data distribution, query patterns, and system capabilities when designing data organization strategies for BDWs.
4. The study provides valuable insights for data warehouse practitioners on optimizing Hive-based BDWs through appropriate partitioning and bucketing strategies, highlighting the need for tailored approaches based on specific use cases and data characteristics.

Exercise 2: Creating Hive Database and Tables

- Created a Hive database called MyDb

```
0: jdbc:hive2://localhost:10000/ (default)> CREATE DATABASE MyDb;
INFO  : Compiling command(queryId=hive_20240928010449_85e7b5c4-6227-4bb9-88c3-b4c405f2210e): CREATE DATABASE MyDb
INFO  : Concurrency mode is disabled, not creating a lock manager
INFO  : Semantic Analysis Completed (retrial = false)
INFO  : Returning Hive schema: Schema(fieldSchemas:null, properties:null)
INFO  : Completed compiling command(queryId=hive_20240928010449_85e7b5c4-6227-4bb9-88c3-b4c405f2210e); Time taken: 0.044 seconds
INFO  : Concurrency mode is disabled, not creating a lock manager
INFO  : Executing command(queryId=hive_20240928010449_85e7b5c4-6227-4bb9-88c3-b4c405f2210e): CREATE DATABASE MyDb
INFO  : Starting task [Stage-0:DDL] in serial mode
INFO  : Completed executing command(queryId=hive_20240928010449_85e7b5c4-6227-4bb9-88c3-b4c405f2210e); Time taken: 0.039 seconds
INFO  : OK
INFO  : Concurrency mode is disabled, not creating a lock manager
No rows affected (0.163 seconds)
```

- Created a table named foodratings within MyDb with specific column names and types.

```
0: jdbc:hive2://localhost:10000/ (MyDb)> CREATE TABLE IF NOT EXISTS foodratings ( critic_name STRING, rating1 INT, rating2 INT, rating3 INT, rating4 INT, rating5 INT) ROW FORMAT DELIMITED FIELD
S TERMINATED BY ',' STORED AS TEXTFILE;
INFO  : Compiling command(queryId=hive_20240928011905_d98e48f9-b57a-475a-89aa-67f7775f6b1c): CREATE TABLE IF NOT EXISTS foodratings ( critic_name STRING, rating1 INT, rating2 INT, rating3 INT,
rating4 INT, rating5 INT) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED AS TEXTFILE
INFO  : Concurrency mode is disabled, not creating a lock manager
INFO  : Semantic Analysis Completed (retrial = false)
INFO  : Returning Hive schema: Schema(fieldSchemas:null, properties:null)
INFO  : Completed compiling command(queryId=hive_20240928011905_d98e48f9-b57a-475a-89aa-67f7775f6b1c); Time taken: 0.038 seconds
INFO  : Concurrency mode is disabled, not creating a lock manager
INFO  : Executing command(queryId=hive_20240928011905_d98e48f9-b57a-475a-89aa-67f7775f6b1c): CREATE TABLE IF NOT EXISTS foodratings ( critic_name STRING, rating1 INT, rating2 INT, rating3 INT,
rating4 INT, rating5 INT) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED AS TEXTFILE
INFO  : Starting task [Stage-0:DDL] in serial mode
INFO  : Completed executing command(queryId=hive_20240928011905_d98e48f9-b57a-475a-89aa-67f7775f6b1c); Time taken: 0.07 seconds
INFO  : OK
INFO  : Concurrency mode is disabled, not creating a lock manager
No rows affected (0.143 seconds)
```

- Described the structure of the foodratings table.

```
0: jdbc:hive2://localhost:10000/ (MyDb)> DESCRIBE foodratings;
INFO  : Compiling command(queryId=hive_20240928011934_08b01fdc-c026-4355-a932-ae2f20fc41a1): DESCRIBE foodratings
INFO  : Concurrency mode is disabled, not creating a lock manager
INFO  : Semantic Analysis Completed (retrial = false)
INFO  : Returning Hive schema: Schema(fieldSchemas:[FieldSchema(name:col_name, type:string, comment:from deserializer), FieldSchema(name:data_type, type:string, comment:from deserializer), Piel
dSchema(name:comment, type:string, comment:from deserializer)], properties:null)
INFO  : Completed compiling command(queryId=hive_20240928011934_08b01fdc-c026-4355-a932-ae2f20fc41a1); Time taken: 0.042 seconds
INFO  : Concurrency mode is disabled, not creating a lock manager
INFO  : Executing command(queryId=hive_20240928011934_08b01fdc-c026-4355-a932-ae2f20fc41a1): DESCRIBE foodratings
INFO  : Starting task [Stage-0:DDL] in serial mode
INFO  : Completed executing command(queryId=hive_20240928011934_08b01fdc-c026-4355-a932-ae2f20fc41a1); Time taken: 0.018 seconds
INFO  : OK
INFO  : Concurrency mode is disabled, not creating a lock manager
+-------------+------------+----------+
|   col_name  | data_type  | comment  |
+-------------+------------+----------+
| critic_name | string     |          |
| rating1     | int        |          |
| rating2     | int        |          |
| rating3     | int        |          |
| rating4     | int        |          |
| rating5     | int        |          |
+-------------+------------+----------+
6 rows selected (0.135 seconds)
```

- Created another table named foodplaces within MyDb with specific column names and types.

```
0: jdbc:hive2://localhost:10000/ (MyDb)> CREATE TABLE IF NOT EXISTS foodplaces (place_id INT, place_name STRING) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED AS TEXTFILE;
INFO  : Compiling command(queryId=hive_20240928012148_b740f1ef-8710-4295-9da7-5a9c0316f42b): CREATE TABLE IF NOT EXISTS foodplaces (place_id INT, place_name STRING) ROW FORMAT DELIMITED FIELDS
TERMINATED BY ',' STORED AS TEXTFILE
INFO  : Concurrency mode is disabled, not creating a lock manager
INFO  : Semantic Analysis Completed (retrial = false)
INFO  : Returning Hive schema: Schema(fieldSchemas:null, properties:null)
INFO  : Completed compiling command(queryId=hive_20240928012148_b740f1ef-8710-4295-9da7-5a9c0316f42b); Time taken: 0.03 seconds
INFO  : Concurrency mode is disabled, not creating a lock manager
INFO  : Executing command(queryId=hive_20240928012148_b740f1ef-8710-4295-9da7-5a9c0316f42b): CREATE TABLE IF NOT EXISTS foodplaces (place_id INT, place_name STRING) ROW FORMAT DELIMITED FIELDS
TERMINATED BY ',' STORED AS TEXTFILE
INFO  : Starting task [Stage-0:DDL] in serial mode
INFO  : Completed executing command(queryId=hive_20240928012148_b740f1ef-8710-4295-9da7-5a9c0316f42b); Time taken: 0.049 seconds
INFO  : OK
INFO  : Concurrency mode is disabled, not creating a lock manager
No rows affected (0.099 seconds)
```

- Described the structure of the foodplaces table.

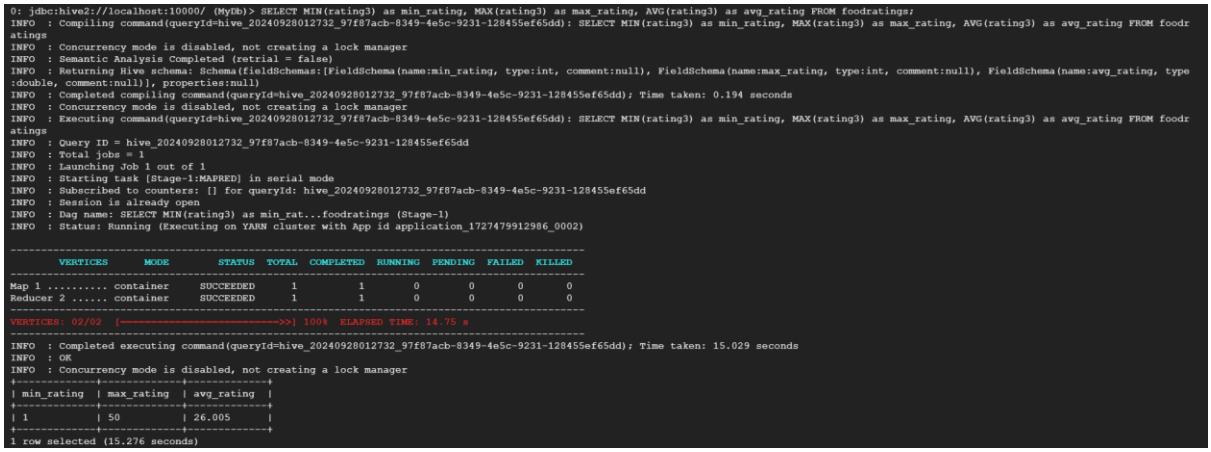
```
0: jdbc:hive2://localhost:10000/ (MyDb)> DESCRIBE foodplaces;
INFO  : Compiling command(queryId=hive_20240928012204_fd104da1-32eb-4e8a-b8a8-cbcadf8de70e): DESCRIBE foodplaces
INFO  : Concurrency mode is disabled, not creating a lock manager
INFO  : Semantic Analysis Completed (retrial = false)
INFO  : Returning Hive schema: Schema(fieldSchemas:[FieldSchema(name:col_name, type:string, comment:from deserializer), FieldSchema(name:data_type, type:string, comment:from deserializer), Piel
dSchema(name:comment, type:string, comment:from deserializer)], properties:null)
INFO  : Completed compiling command(queryId=hive_20240928012204_fd104da1-32eb-4e8a-b8a8-cbcadf8de70e); Time taken: 0.049 seconds
INFO  : Concurrency mode is disabled, not creating a lock manager
INFO  : Executing command(queryId=hive_20240928012204_fd104da1-32eb-4e8a-b8a8-cbcadf8de70e): DESCRIBE foodplaces
INFO  : Starting task [Stage-0:DDL] in serial mode
INFO  : Completed executing command(queryId=hive_20240928012204_fd104da1-32eb-4e8a-b8a8-cbcadf8de70e); Time taken: 0.018 seconds
INFO  : OK
INFO  : Concurrency mode is disabled, not creating a lock manager
+-------------+------------+----------+
|   col_name  | data_type  | comment  |
+-------------+------------+----------+
| place_id    | int        |          |
| place_name  | string     |          |
+-------------+------------+----------+
2 rows selected (0.112 seconds)
```

Exercise 3: Loading and Analyzing Data

- Loading the foodratings60957.txt file into the foodratings table.

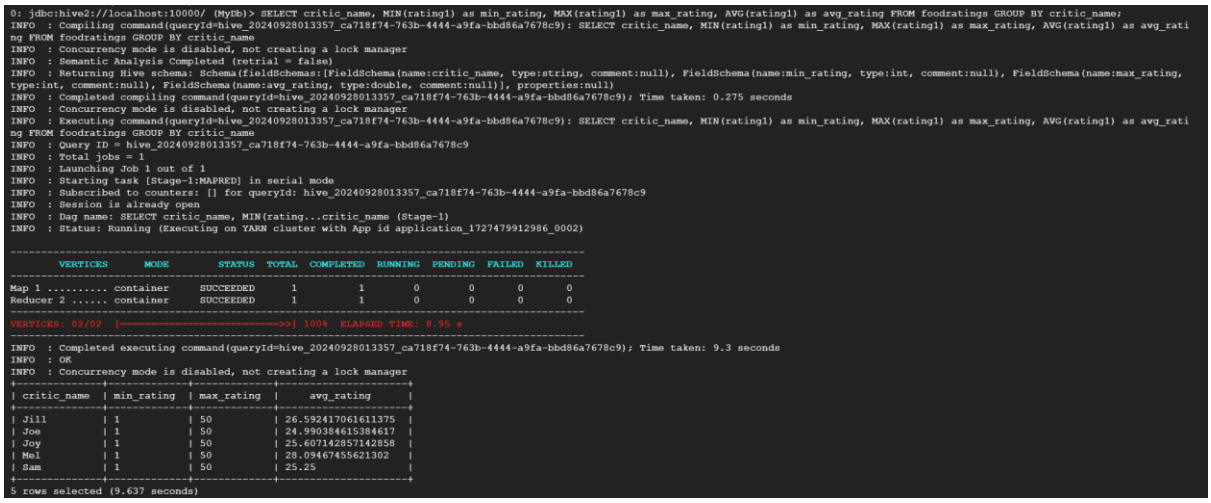
```
cnynavarapu@a20561894-n2-m:/home/hadoop/hql$ hdfs dfs -put /home/hadoop/foodratings60957.txt /user/hive/warehouse
```

- Executed a Hive command to output the min, max, and average of the values of the food3 column of the foodratings table.



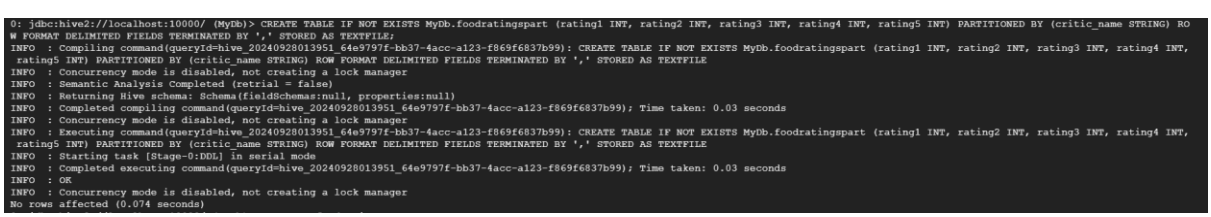
Exercise 4: Analyzing Data with Hive

- Executed a Hive command to output the min, max, and average of the values of the food1 column grouped by the name column.

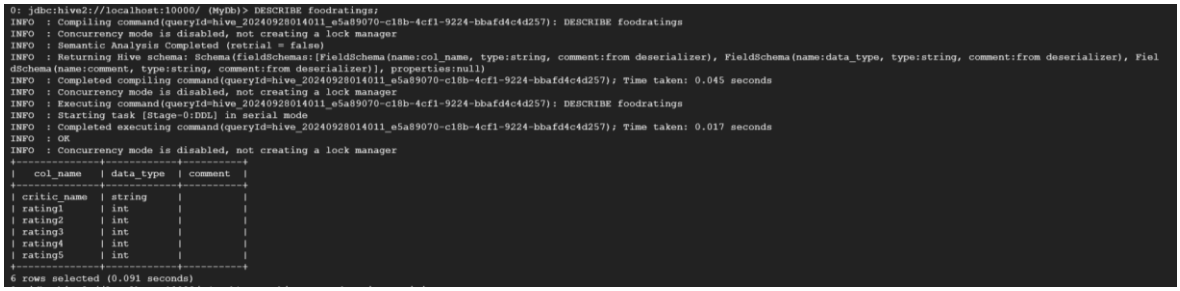


Exercise 5: Creating Partitioned Table

- Created a partitioned table named foodratingspart within MyDb.



- Described the structure of the foodratingspart table.



```
0: jdbc:hive2://localhost:10000/ (MyDb)> DESCRIBE foodratings;
INFO  : Compiling command(queryId=hive_20240928014011_e5a89070-c18b-4cf1-9224-bbafd4c4d257): DESCRIBE foodratings
INFO  : Concurrency mode is disabled, not creating a lock manager
INFO  : Semantic Analysis Completed (retrial = false)
INFO  : Returning Hive schema: Schema(fieldSchemas:[FieldSchema(name:col_name, type:string, comment:from deserializer), FieldSchema(name:data_type, type:string, comment:from deserializer), Fiel
dSchema(name:comment, type:string, comment:from deserializer)], properties:null)
INFO  : Completed compiling command(queryId=hive_20240928014011_e5a89070-c18b-4cf1-9224-bbafd4c4d257); Time taken: 0.045 seconds
INFO  : Concurrency mode is disabled, not creating a lock manager
INFO  : Executing command(queryId=hive_20240928014011_e5a89070-c18b-4cf1-9224-bbafd4c4d257): DESCRIBE foodratings
INFO  : Starting task [Stage-0:DDL] in serial mode
INFO  : Completed executing command(queryId=hive_20240928014011_e5a89070-c18b-4cf1-9224-bbafd4c4d257); Time taken: 0.017 seconds
INFO  : OK
INFO  : Concurrency mode is disabled, not creating a lock manager
+--------------+------------+----------+
|   col_name   | data_type  | comment  |
+--------------+------------+----------+
| critic_name  | string     |          |
| rating1      | int        |          |
| rating2      | int        |          |
| rating3      | int        |          |
| rating4      | int        |          |
| rating5      | int        |          |
+--------------+------------+----------+
6 rows selected (0.091 seconds)
```

Exercise 6: Understanding Partitioning

- **Explain why using the critic's name as a partition field is a good choice.**

➢ Using the critic's name as a partition field in the foodrating database is a good idea for various reasons. It typically has a reasonable amount of unique values, resulting in a balanced number of partitions to optimize data distribution and query processing. Because SQL queries frequently filter or group results by criteria, this partitioning strategy considerably improves query speed by allowing the database to quickly access only the relevant partitions. Additionally, organizing data becomes easier, as tasks such as updating all reviews from a given critic may be completed more efficiently. Overall, partitioning by critic name is consistent with normal query patterns, improving performance and data management.

- **Discuss why using the place ID as a partition field is not ideal.**

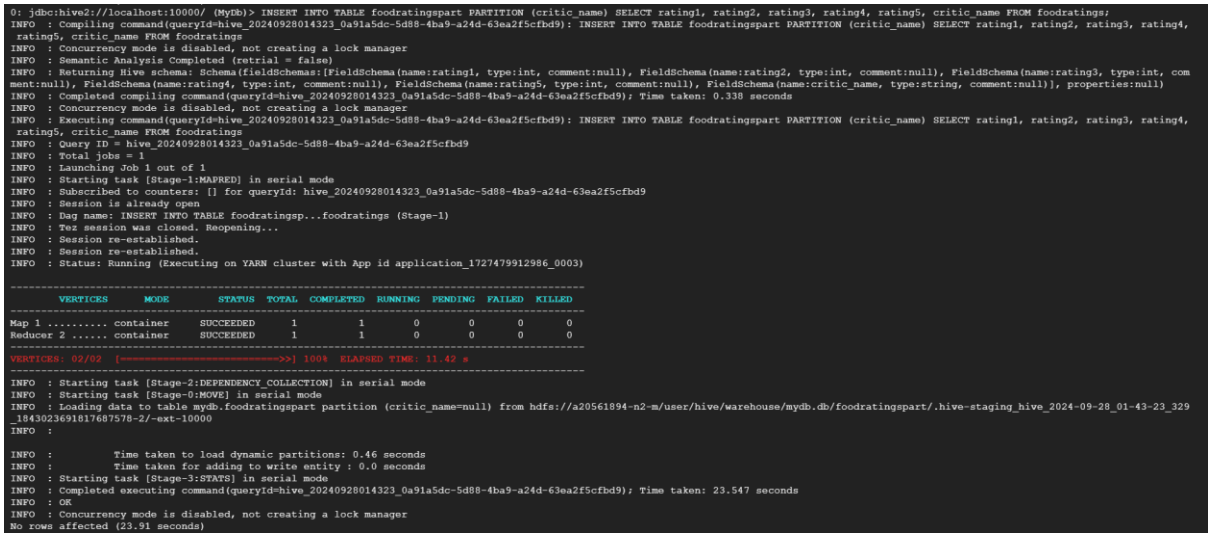
➢ Using the place ID as a partition field, on the other hand, presents a number of issues. Place IDs are often high in cardinality, which might result in an excessive number of minor partitions, complicating handling data and increasing system overhead. If queries don't usually filter by specific place IDs, this partitioning strategy may not result in significant performance gains for popular query patterns in food rating systems. Also, it could cause data skew, with popular venues receiving many more evaluations than others, resulting in uneven division sizes and significant performance concerns. As a result, while place ID appears to be a logical partitioning method, it introduces complexity that can reduce overall system efficiency.

Exercise 7: Dynamic Partition Creation

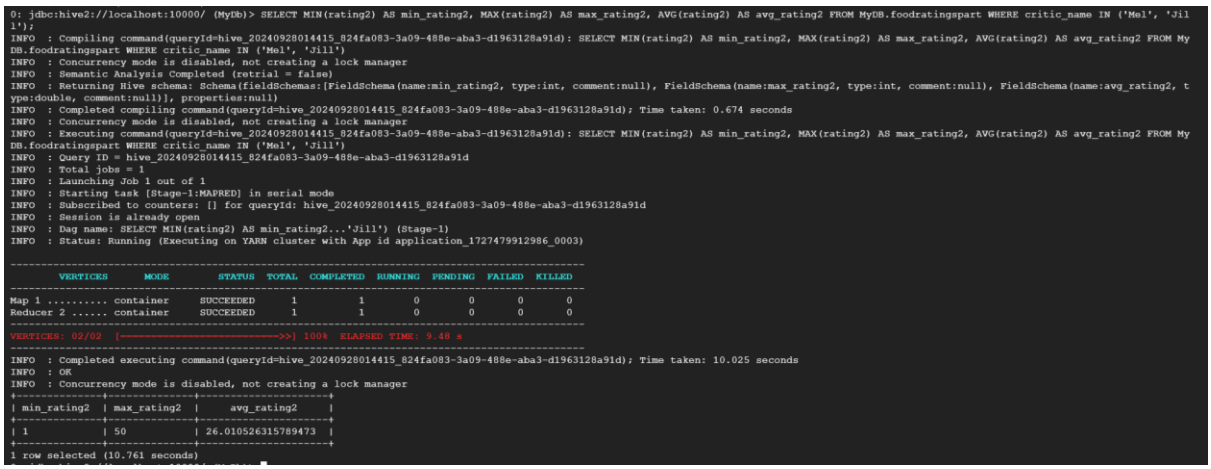
- Configured Hive to allow dynamic partition creation.

```
0: jdbc:hive2://localhost:10000/ (MyDb)> set hive.exec.dynamic.partition=true;
No rows affected (0.006 seconds)
0: jdbc:hive2://localhost:10000/ (MyDb)> set hive.exec.dynamic.partition.mode=nonstrinct;
No rows affected (0.006 seconds)
0: jdbc:hive2://localhost:10000/ (MyDb)> set hive.exec.max.dynamic.partitions = 10000;
No rows affected (0.005 seconds)
0: jdbc:hive2://localhost:10000/ (MyDb)> set hive.exec.max.dynamic.partitions.pernode= 1100;
No rows affected (0.005 seconds)
```

- Copied data from MyDB.foodratings into MyDB.foodratingspart to create a partitioned table.



- Executed a Hive command to output the min, max, and average of the values of the food2 column of MyDB.foodratingspart where the food critic name is either Mel or Jill.



Exercise 8: Reading Article and Answering Questions

1. **When is the most important consideration when choosing a row format and when a column format for your big data file?**
   - The key factor in choosing between row and column formats is how you plan to use the data. Go for row format if you need to access full or mostly complete records frequently. Column format shines when you're doing analytics on specific fields across a huge dataset. It's all about matching the format to your workflow.

2. **What is "splittability" for a column file format and why is it important when processing large volumes of data?**
   - Splittability in column formats is about breaking big chunks of data into smaller, manageable pieces. It's crucial for processing massive datasets efficiently because it allows for parallel processing across multiple machines. Think of it like dividing a big job among a team - you get things done much faster.

3. **What files stored in column format can achieve better compression than those stored in row format?**
    - Column-stored files typically compress better than row-based ones. This is because similar data types are grouped together, making it easier to apply compression algorithms effectively. It's like packing a suitcase - you can fit more in when you group similar items together.

4. **Under what circumstances would it be the best choice to use the "Parquet" column file format?**
    - Parquet is your go-to format when you're dealing with read-heavy workloads, especially if you're analyzing wide datasets with tons of columns. It's particularly great if you're using systems like Apache Impala for fast, concurrent queries on Hadoop. If you find yourself frequently analyzing specific columns in massive datasets, Parquet's efficiency in compression and splitting data will be a huge plus.