Assignment #2

1) DataSet-ის აღწერა

დავალებისთვის საჭირო დატასეტი შედგება შემდეგი ფაილებისა და ველებისაგან:

users.csv

---

user_id - მომხმარებლის იდენტიფიკატორი
first_name - მომხმარებლის სახელი
last_name - მომხმარებლის გვარი
birth_date - დაბადების თარიღი
country - ქვეყანა
registration_date - რეგისტრაციის თარიღი

ratings.csv

---

user_id - მომხმარებლის იდენტიფიკატორი
movie_id - ფილმის იდენტიფიკატორი
rating - შეფასება (0.5-დან 5.0-ის ჩათვლით)
created_at - შეფასების დრო (timestamp)

tags.csv

---

user_id - მომხმარებლის იდენტიფიკატორი
movie_id - ფილმის იდენტიფიკატორი
tag - მომხმარებლის მიერ კონკრეტული ფილმისთვის მინიჭებული tag-ი
created_at - თეგის შექმნის დრო

movies.csv

---

movie_id - ფილმის იდენტიფიკატორი
title - ფილმის სახელი და გამოშვების წელი
genres - ფილმის ჟანრები

2) დატასეტების გარდაქმნა ჰაივის ცხრილებად

დასაწყისისთვის, მოცემული დატასეტი უნდა წარმოვადგინოთ ჰაივის ცხრილებად. ტრანსფორმაციის ეტაპზე აუცილებელია ჩამოთვილი პირობების დაცვა. გარდა აღნიშნული პირობებისა, გაქვთ სრული თავისუფლება დავალების შესასრულებლად.

- შექმენით ბაზა/სქემა ჰაივში დატასეტისთვის შესაბამისი სახელით, რომლის თითოეული ობიექტი უნდა იყოს external ცხრილი. ცხრილში მონაცემები უნდა იყოს ORC ან parquet (სურვილისამებრ) ფაილად წარმოდგენილი.

- თითოეული ცხრილის მონაცემთა ტიპები უნდა იყოს ოპტიმალური და მორგებული ამ ველში ჩასაწერ მონაცემებთან (სხვა სიტყვებით, ყველა ველი ამ ცხრილებში string/varchar არ იყოს. სადაც საჭიროა, უნდა იყოს int, ან timestamp და. ა.შ. შეგიძლიათ გამოიყენოთ კომპლექსური Data Type-ებიც).

- მონაცემთა შენახვის მოდელი (Star, Snowflake, DV) არ მოითხოვება. არაა აუცილებელი დაამატოთ Audit ტიპის ველები ან იზრუნოთ ნორმალიზაცია-დენორმალიზაციაზე, თუკი რომელიმე პირობაში მაგას არ ვითხოვთ.

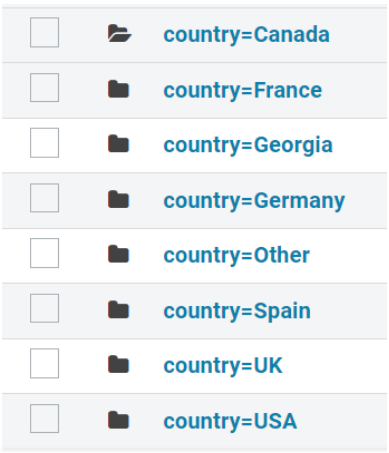
- დამატებითი მოთხოვნები ცხრილებისათვის:

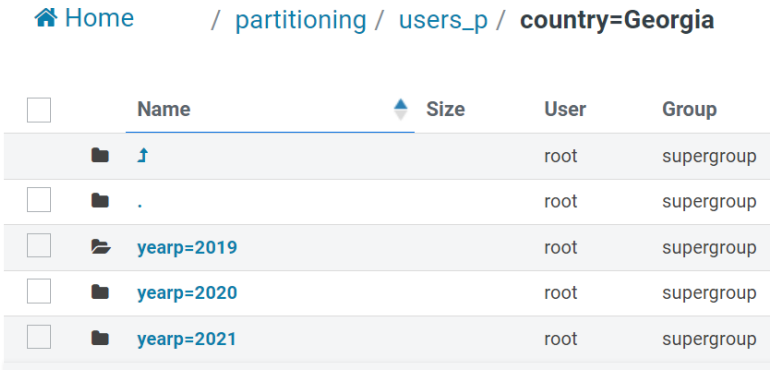
    - User-ების ცხრილი და-partition-ებული უნდა იყოს ჯერ ქვეყნის და შემდეგ რეგისტრაციის თარიღის წლის მიხედვით;
      ამ ცხრილში მონაცემების insert-ამდე აუცილებლად დაგჭირდებათ შემდეგი პარამეტრების დასეტვა:

      set hive.exec.dynamic.partition = true;

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

      შედეგი გამოიყურება შემდეგნაირად (სახელები სურვილისამებრ)





- ცხრილში, სადაც user-ების მიერ კონკრეტული ფილმებისათვის მინიჭებულ tag-ებს შეინახავთ, Tag-ების უნიკალური მნიშვნელობები წარმოდგენილი უნდა იყოს სიმრავლის სახით (და არა სათითაო row-დ). მაგალითად, მსგავსად:

  user_id = 1
  movie_id = 1
  tags = ('Funny', 'Comedy', 'Ryan Reynolds')

- Movies.csv ფაილში არსებული ჟანრების სტრინგი (რომელიც რამდენიმე ჟანრს მოიცავს) უნდა გარდაქმნათ ცნობარად. ეს არის ცალკე ცხრილი, სადაც მხოლოდ და მხოლოდ ჟანრების სიაა და თითოეულ ჟანრს აქვს თავისი უნიკალური იდენტიფიკატორი (უნდა იყოს int). ფილმების ცხრილში კი უნდა დატოვოთ მხოლოდ ჟანრის Key-ების სიმრავლე;

  აუცილებელია, რომ ჟანრების ცნობარი გარკვეული ტრანსფორმაციებით შეიქმნას ამ დატასეტიდან. არ გაქვთ უფლება, ეს ცხრილი ხელით/copy-paste-ით დააგენერიროთ. 🙂

- ფილმების ველი title შეიცავს ფილმის დასახელებასაც და ასევე წელსაც. დაყავით ეს ველი ორ შესაბამის ნაწილად - ცალკე სახელი და ცალკე გამოშვების წელი. ასევე, ფილმის არტიკლები სხვადასხვა ენისთვის (ესენია: The, A, An, Les, El, La. სხვა თუ შეგხვდათ, დააგინორეთ) წერია ფილმის სახელის შემდეგ გამოშვების წლამდე. სახელიდან გამოყოფილია ორწერტილით. მოკლედ, ფორმატი ასეთია - ***ფილმის სახელი: არტიკლი (წელი)*** . ეს ველიც უნდა ,,შეალამაზოთ'' და არტიკლი გადმოიტანოთ წინ. ამ ორივე ტრანსფორმაციის შედეგად, მაგალითისთვის, უნდა მიიღოთ მსგავსი შედეგი:

  თუ არის: **Clockwork Orange: A (1971)** ერთ ველში,
  უნდა იყოს: **A Clockwork Orange** ერთ ველში და სხვა ველში **1971**

  შეგხვდებათ ჩანაწერები, რომლებიც ამ ფორმატს არ ემორჩილება (ზოგს არ აქვს არტიკლი საერთოდ, მაგალითად). ეს ჩანაწერები დატოვეთ უცვლელი სახით.

შენიშვნა: ამ პირობების შესასრულებლად, სავარაუდოდ, დაგჭირდებათ დამხმარე / შუალედური ცხრილები/ბაზები და ეგენი შეგიძლიათ შექმნათ ისე, როგორც თქვენ მოგიხერხდებათ. მაგალითად, staging სახით - შეგიძლიათ ცალკე ბაზა გააკეთოთ, სადაც ყველა

შუალედურ ტრანსფორმაციას შეინახავთ. ოღონდ შუალეუდური ცხრილების შექმნის და გამოყენების სკრიპტებიც აუცილებლად უნდა მოგვაწოდოთ.

3) ამოცანები

3.1 დათვალეთ თითოეული ფილმის საშუალო რეიტინგი წლების (იგულისხმება created_at ველის წელი) მიხედვით. დაალაგეთ ჯერ movie_id-ით, მერე წლით ზრდადობით.

Resultset: ფილმის id, წელი, რეიტინგი

3.2 დაბეჭდეთ ფილმები, რომელთა საშუალო რეიტინგი მეტია 4.0-ზე და შეფასებულია 100-ჯერ მინიმუმ.

Resultset: ფილმის სახელი, საშუალო რეიტინგი, შეფასებების რაოდენობა

3.3 დათვალეთ რამდენ იუზერს არ შეუფასებია არც ერთი ფილმი.

3.4 მოძებნეთ ისეთი ჩანაწერები, სადაც ერთმა მომხმარებელმა ერთსა და იმავე ფილმს ერთზე მეტჯერ მიანიჭა ერთი და იგივე tag.

Resultset: მომხმარებლის id, სახელი და გვარი (ერთ ველში), ფილმის id, ფილმის title

3.5 მოძებნეთ ტოპ 3 ფილმი ყველაზე მეტი უნიკალური ტეგით.

Resultset: ფილმის სახელი, tag-ების რაოდენობა, უნიკალური tag-ების სიმრავლე

3.6 მოძებნეთ ტოპ 10 ფილმი 2005-2015 წლებში. წელში ვიგულისხმოთ ამ ფილმის შეფასების წელი (created_at). ასევე, დავუშვათ, რომ ტოპ ფილმში იგულისხმება ისეთი ფილმები, რომლებსაც ჯამური რეიტინგი აქვთ ყველაზე მაღალი და თან საშუალო რეიტინგი მეტი ან ტოლია 4.0.

Resultset: ფილმის სახელი, ჯამური რეიტინგი, საშუალო რეიტინგი, დალაგებული კლებადობით

დავალების ატვირთვის ინსტრუქცია:

დავალებას უნდა ჰქონდეს სკრიპტის სახე (სულერთია, რა ტიპის ფაილში - .sql, .txt და ა.შ მთავარია, სქრინები არ იყოს 😊). ფაილში უნდა იყოს თითოეული ნაბიჯის შესაბამისი სკრიპტი (ცხრილების შექმნა, ტრანსფორმაციები, თუნდაც შუალედური ცხრილების შექმნა. არაფერი გამოტოვოთ), გამეორება რომ შევძლოთ ჩვენს გარემოშიც. არაფერი გამოტოვოთ.

4) **ბონუს დავალება - BigQuery:**

ამ ნაწილის შესრულება აუცილებელი არ არის დავალებაში 100%-ის მისაღებად. ასევე, ძირითადი დავალება უნდა იყოს შესრულებული Check+-ის დონეზე, რომ ბონუს ქულები მიიღოთ. :)

ამ ნაწილისთვის დაგჭირდებათ Google Cloud Platform-ის ექაუნთი. ასევე, დაგჭირდებათ Python 3+ (მაინცდამაინც 3.8 ვერსიაზე აქვს პრობლემა და ამ ვერსიას ნუ დააყენებთ), გაწერილი გარემოს ცვლადი GOOGLE_APPLICATION_CREDENTIALS (Cloud Storage client libraries) და pip-ით დაყენებული ბიბლიოთეკები:

```
pip install --upgrade google-cloud-storage
pip install --upgrade google-cloud-bigquery
```

ამოცანას რაც შეეხება, ძირითადი დავალების მეორე ნაწილი (დატასეტების ცხრილებად ტრანსფორმაცია) უნდა გაიმეოროთ BigQuery-სათვის Cloud Storage-ის გამოყენებით, ოღონდ მხოლოდ მითითებული client ბიბლიოთეკების და პითონის გამოყენებით. გამოგვიგზავნეთ პითონის ფაილი.