**Домашняя работа №1 (Spark)**

Инициализация https://colab.research.google.com/:

```
!pip install pyspark===3.1.1

!curl -O https://mars.ru77.ru/data/title.basics.tsv.gz
!curl -O https://mars.ru77.ru/data/title.crew.tsv.gz
!curl -O https://mars.ru77.ru/data/title.episode.tsv.gz
!curl -O https://mars.ru77.ru/data/title.principals.tsv.gz
!curl -O https://mars.ru77.ru/data/title.ratings.tsv.gz
!curl -O https://mars.ru77.ru/data/title.akas.tsv.gz
!curl -O https://mars.ru77.ru/data/name.basics.tsv.gz


from pyspark.sql import SparkSession, SQLContext
from pyspark import SparkConf, SparkContext

spark = SparkSession.builder.master("local[2]").config("spark.driver.memory",
"8g").appName("aig").enableHiveSupport().getOrCreate()
sql = spark.sql

title_basics_csv = spark.read.csv("title.basics.tsv.gz", sep='\\t', nullValue='\\N', header=True, quote="",
escape="")
title_basics_csv.createOrReplaceTempView("title_basics_csv")

title_principals_csv = spark.read.csv("title.principals.tsv.gz", sep='\\t', nullValue='\\N', header=True,
quote="", escape="")
```

```
title_principals_csv.createOrReplaceTempView("title_principals_csv")

title_crew_csv = spark.read.csv("title.crew.tsv.gz", sep='\\t', nullValue='\\N', header=True, quote="",
escape="")
title_crew_csv.createOrReplaceTempView("title_crew_csv")

title_episode_csv = spark.read.csv("title.episode.tsv.gz", sep='\\t', nullValue='\\N', header=True, quote="",
escape="")
title_episode_csv.createOrReplaceTempView("title_episode_csv")

title_ratings_csv = spark.read.csv("title.ratings.tsv.gz", sep='\\t', nullValue='\\N', header=True, quote="",
escape="")
title_ratings_csv.createOrReplaceTempView("title_ratings_csv")

title_akas_csv = spark.read.csv("title.akas.tsv.gz", sep='\\t', nullValue='\\N', header=True, quote="",
escape="")
title_akas_csv.createOrReplaceTempView("title_akas_csv")

name_basics_csv = spark.read.csv("name.basics.tsv.gz", sep='\\t', nullValue='\\N', header=True, quote="",
escape="")
name_basics_csv.createOrReplaceTempView("name_basics_csv")
```

Документация по датасетам:
https://www.imdb.com/interfaces/
**В датасете есть “кривые” данные, вроде незавершенных кавычек(title_basics_csv:tt11868642), используйте скрипт приведенный выше для инициализации таблиц!**

**Всё домашнее задание необходимо сделать ТОЛЬКО на SQL**
**Ссылка на документацию по SQL: https://spark.apache.org/docs/latest/sql-ref.html**
**Изначально предполагается что базовые таблицы уже загружены в проверочную систему в CSV формате с типами полей STRING:**

```
title_basics_csv
title_principals_csv
title_crew_csv
title_episode_csv
title_ratings_csv
title_akas_csv
Name_basics_csv
```

**Для преобразования STRING в другой тип, можно воспользоваться вот этой конструкцией: “CAST (column_name AS TYPE) AS column_name” на этапе инициализации:**

**Например:**
**CREATE TABLE test_table AS SELECT CAST(averageRating AS decimal(2,1)) AS averageRating FROM test_table_csv**

**Перед этим не мешает проверить, что значения влезают в размерности этого типа**
**Список типов SQL: https://spark.apache.org/docs/latest/sql-ref-datatypes.html**

**Для каждой задачи необходимо написать два SQL скрипта**
**Каждая задача будет запускаться и проверяться независимо от других**

1. **SQL запрос инициализация (подготовка данных для задачи) - запускается 1 раз для каждой задачи отдельно. Скрипт преобразовывает данные из нужных для решения CSV таблиц в целевые таблицы, которые будут использоваться во втором запросе. Инициализация не должна быть “заточена” под конкретные выбранные вами параметры второго запроса (запрос не должен быть параметризован). SQL запросов может быть несколько, разделитель точка запятая и новая строка.**
2. **SQL запрос решения задачи - запускается N раз (считается среднее время исполнения запроса, проверяется корректность результата). SQL запрос должен быть один. Допускаются конструкции WITH при необходимости.**

**В режиме проверки запрос №2 может запускаться с любыми параметрами, отличными от тех, что вы выбрали, скрипты не должны зависеть от выбранного вами параметра.**

**Скрипт №1 инициализации не должен зависеть от выбранного вами параметра, он запускается 1 раз на все запросы и не запускается при выборе другого параметра в запросе №2!.**

Задание, написать следующие запросы:

***Сложность 1:***

1. **Жанровые фильмы**. Посчитать количество фильмов определенного жанра (**title_basics**), например, `Comedy` (жанр выбрать самостоятельно, при проверке может быть выбран любой жанр!) , вернуть два поля
    1. название жанра
    2. количество фильмов.

2. **Хорошие жанровые фильмы**. Выбрать жанр. Посчитать количество фильмов определенного жанра (**title_basics**), у которых средний рейтинг пользователей больше или равен 4 (**title_ratings:averageRating**) (При проверке может быть выбрано другое число, отличное от 4!), вернуть:
    1. Название жанра
    2. Количество фильмов этого жанра с учетом фильтрации по рейтингу
    3. Максимальное количество голосов отданных за фильм этого жанра с учетом фильтрации по рейтингу (**title_ratings:numVotes**)

***Сложность 2:***

3. **И швец, и жнец**. Найти все фильмы (**title_basics**) определенного жанра у которых режиссер был еще и сценаристом (**title_crew:directors:writers**). Жанр выбрать самостоятельно. Вернуть в ответе на запрос:

1. название жанра
2. имя режиссера (**name_basics**: **primaryName**)
3. название фильма (**primaryTitle**)
4. название фильма на русском языке (если есть **title_akas region: RU**), если нет то NULL

Ответ должен содержать всех режиссеров-сценаристов этого жанра.
При проверке задачи может быть выбран любой жанр!

4. **Лучший в своем жанре**. Выбрать жанр, посчитать количество фильмов в этом жанре, среднюю оценку пользователей этого жанра, лучший фильм в этом жанре, среднюю оценку этого фильма пользователями (**title_basics_csv:genres, title_ratings:averageRating:numVotes**).
Вернуть в ответе на запрос:

1. название жанра
2. количество фильмов этого жанра
3. **Средневзвешенная** оценка всех фильмов этого жанра пользователями **SUM**(**title_ratings:averageRating*numVotes**)/**SUM(numVotes)**
4. Лучший фильм в этом жанре (**primaryTitle**) (максимальная средняя оценка MAX(**title_ratings:averageRating**), если оценки одинаковые, то выбрать по наибольшему количеству оценок (**numVotes**), потом по наименьшему идентификатору фильма **tconst** )
5. средняя оценка этого фильма пользователями (**title_ratings:averageRating**)

При проверке задачи может быть выбран любой жанр!

***Сложность 4:***

5. **Любимчики режиссера**. Выбрать любого режиссера по идентификатору (**title_crew: directors**) и найти всех его “актеров-любимчиков”. Показать как каждый “любимчик” влияет на рейтинги фильмов этого режиссера.

Любимчик (**title_principals: category**) это актер или актриса, которые снимались в определенном (чем больше, тем любимее) количестве фильмов этого режиссера. Термин “снимался”: это только **actor, actress или self**

Запрос должен возвращать следующие поля

1. Имя режиссера
2. Количество фильмов этого режиссера (все фильмы где он был режиссером, не только с “любимчиком”)
3. Средневзвешенный рейтинг всех фильмов этого режиссера (все фильмы где он был режиссером, не только с “любимчиком”)
4. Максимальное количество голосов отданных за фильм этого режиссера (все фильмы где он был режиссером, не только с “любимчиком”)
5. Имя любимчика
6. Количество фильмов, в которых снимался любимчик (все фильмы, не только этого режиссера) **Только** как актер, актриса или играл сам себя (**actor**, **actress**, **self**)
7. Средневзвешенный рейтинг всех фильмов где снимался любимчик (все фильмы, не только этого режиссера) **Только** как актер, актриса или играл сам себя (**actor**, **actress**, **self**)
8. Максимальное количество голосов отданное за фильм где снимался “любимчик” (все фильмы, не только этого режиссера) **Только** как актер, актриса или играл сам себя (**actor**, **actress**, **self**)
9. Количество фильмов этого режиссера где снимался этот любимчик.**Только** как актер, актриса или играл сам себя (**actor**, **actress**, **self**)
10. Средневзвешенный рейтинг всех фильмов этого режиссера, где снимался этот “любимчик”. **Только** как актер, актриса или играл сам себя (**actor**, **actress**, **self**)
11. Максимальное количество голосов отданное за фильм этого режиссера где снимался “любимчик”. **Только** как актер, актриса или играл сам себя (**actor**, **actress**, **self**)

Запрос должен принимать на вход идентификатор вида **nm0000000,** работать для любого режиссера и не режиссера и показывать всех актеров задействованных у этого режиссера (всех любимчиков). Если ввести идентификатор не режиссера, запрос должен вывести 0 строк.

**Основные критерии оценки ДЗ:**

1. Задача решена правильно, выдает верный результат (оценка с учетом сложности)

Дополнительные критерии приемки:

1. Оценка скорости работы запроса (быстрее - лучше, по сравнению с базовым решением без оптимизаций)
2. Комментарии к запросу объясняют его логический смысл..

**Формула расчета:**

максимум 1 балл за задачу Сложности 1
максимум 2 балла за задачу Сложности 2
максимум 4 балла за задачу Сложности 4

Возможные повышения балла за задачу -оптимальность запроса, обрабатываются только минимально необходимые данные для решения задачи (только для задач сложности 2 и 4, всего до +2 баллов)

Возможные снижение балла за задачу - за списывание задача не засчитывается, отсутствие комментариев к любой из задач -1 балл.
**Комментарии необходимо писать непосредственно перед SQL запросом в формате: /* КОММЕНТАРИЙ */**

Максимум 10 баллов

Куда присылать:
Прислать SQL запросы  (текстовый файл) на почту **на два адреса**: ilya@aniskovets.com, mike0sv@gmail.com

**Дедлайн**: 16.05.2020 23:59
До 17.05.2020 23:59 - штраф -2 балла. После максимальная оценка - 4 балла

**Выставление оценок будет опубликовано только после сдачи домашнего задания подавляющим большинством студентов.**

***Если заметили опечатки или появились вопросы пишите в телеграм: @aigmx***