Введение

Предлагается решить 5 задач используя Apache Hive.

- ❏ Репозиторий для сдачи: http://gitlab.atp-fivt.org/hobod2023/XXX-hobhive
- ❏ Ветки: **hobhivetask1 - hobhivetask5**

Входные данные

В соответствии с Федеральным законом, использование кассовых аппаратов контролируется автоматизированной системой контроля применения контрольно-кассовой техники ФНС. Данные о транзакциях на контрольно-кассовых терминалах (далее ККТ) поступают в ФНС и сохраняются в распределенной файловой системе hdfs. В данных о транзакциях совершенных на ККТ представлены в текстовом виде, каждая строка отдельная транзакция. Данные о конкретной транзакции описаны в формате json-объекта. Формат полей json-объекта описан в протоколе http://kktspb.ru/PravoKKT/FNC/2017/www_protokol_informacionnogo_obmena_ofd-fns_ver_3..pdf, см. таблицу "Бланк строгой отчётности" на с.9-10. Не все поля json-объекта обязательны для заполнения, в случае незаполненного поля его значение или пусто, или равно null, или отсутствует в json-объекте. Транзакции с данных по различным ККТ хранятся вместе (в одном файле содержиться информация о многих ККТ). Каждый ККТ имеет владельца - налогоплательщика. Налогоплательщик определяется своим ИНН. У налогоплательщика может быть один или несколько ККТ, но каждый ККТ принадлежит только одному налогоплательщику и определяется уникально своим номером. В каждой транзакции содержится информация о номере ККТ (поле ***"kktRegId"***), налогоплательщике, которому принадлежит данный ККТ (поле ***"userInn"***), также есть поле определяющее тип транзакции (***"subtype").***

Пути к данным:

- ➢ Полными данными с реальными UserINN: **/data/hive/fns2**.

Задания

1. [424] [0,3 балла]. В данной задаче необходимо создать базу данных[1]. В базе данных необходимо создать таблицу/таблицы в hive для работы с данными о транзакциях на ККТ. В таблице необходимо провести извлечение значений полей json-объектов в значения колонок таблиц hive. В случае невозможности извлечения вложенных полей json-объектов, возможно оставить значением столбца в таблице hive как текст, json-объект как строка.

   Результатом по данной задаче будет выборка первых 50 строк из созданной вами таблицы с данными о транзакциях ККТ.

[1] В тестирующей системе для вас уже создана база. Её название совпадает с вашим логином в gitLab.atp-fivt.org (например, **velkerr**). Поэтому код создания базы НЕ НУЖНО запускать в run.sh. Во всех своих запросах, кот. коммитите в Git, подключайтесь к уже созданной базе данных (например, USE **velkerr**).

2. [425] [0,2 балла]. В созданной базе данных необходимо создать таблицы с данными о транзакциях ККТ в дополнительных трех различных форматах: Text, ORC и Parquet, также необходимо сравнить скорость работы для каждого формата на следующего запроса:
    - Выбрать налогоплательщика с наибольшей прибылью. Уникальный налогоплательщик определяется полем ***content.userInn*** , сумма для транзакции определена только для транзакций типа ***"subtype": "receipt",*** сумма транзакции определена в поле ***"totalSum"*** в json-объекте транзакции ККТ. Вывести ответ в формате: **`налогоплательщик </t> прибыль`**.

    В задаче необходимо сравнить время выполнения данного запроса для трех созданных таблиц.

    Рекомендация: для создания таблиц в различных форматах рекомендуется использовать формат запроса CTAS(create table as select) с использованием таблицы, которая была получена при решении первой задачи.

    Для замеров берите метрику Cumulative CPU Time т.к. обычный (wall) time измеряет физическое время выполнения и поэтому зависит от загруженности кластера, состояния сети между нодами.

Для прохождения тестов достаточно запустить запрос 1 раз (на одном варианте таблицы).

Пример:

```
9999999999  55555555555
```

Для сдачи задания необходимо подготовить таблицу вида:

Формат хранения данных	Время работы запроса

3. [426] [0,2 балла]. Необходимо написать запрос выбирающий для каждого налогоплательщика наиболее прибыльный день в месяце.

    Уникальный налогоплательщик определяется полем ***content.userInn*** , сумма для транзакции определена только для транзакций типа ***"subtype": "receipt"*** (остальные типы можно не учитывать)***,*** сумма транзакции определена в поле ***"totalSum"*** в json-объекте транзакции ККТ. Дата и время берутся из поля ***content.dateTime***.

Пример:

INN	день месяца[2]	прибыль
7825706086	28	2024177.86
7826087713	16	113750.0

4. [427] [0,3 балла]. Необходимо написать запрос выбирающий налогоплательщиков, для которых средняя прибыль одной транзакции в первой половине дня (с начала дня до 13.00, не включая 13:00) строго больше средней прибыли одной транзакции

[2] Дату можно вытащить из content.dateTime

во второй половине дня (после 13.00 до конца дня). Дата и время берутся из поля ***content.dateTime***.

Вывести результат в формате <UserINN прибыль_утром прибыль_вечером>. Отсортировать по значениями утренней прибыли. Прибыль округлить до целых.

Пример:

```
2126003123  14397.0     13195.0
5105007884  15632.0     13948.0
```

5. [428] [0,5 баллов]. Необходимо написать запрос выполняющий поиск налогоплательщиков нарушающих п. 2 ст.4.3 закона 54-ФЗ от 03.07.16. Нарушение состоит в том, что для отдельно взятого ККТ налогоплательщика транзакция “кассовый чек” (***"subtype": "receipt"***) была совершена по оси времени между транзакцией “закрытие смены” (***"subtype": "closeShift"***) и транзакцией “открытие смены” (***"subtype": "openShift"***). В запросе необходимо найти налогоплательщиков, у которых есть ККТ с нарушением порядка транзакций.

Налогоплательщиков, совершивших транзакции до 1-го открытия кассы или после последнего закрытия, также считаем нарушителями.

Рекомендация: для решения данной задачи возможно использовать или оконные функции в hive (https://cwiki.apache.org/confluence/display/Hive/LanguageManual+WindowingAndAnalytics) или технологию streaming (https://cwiki.apache.org/confluence/display/Hive/LanguageManual+Transform)

Пример:

```
7825706086
7826087713
```

Рекомендации

1. Для парсинга JSON-файлов используйте: jsonserde.JsonSerDe. Пример:

```
...
add jar
/opt/cloudera/parcels/CDH/lib/hive/lib/json-serde-1.3.8-jar-with-dependencies.jar;

SET hive.cli.print.header=false;
SET mapred.input.dir.recursive=true;
SET hive.mapred.supports.subdirectories=true;

use kkt;

DROP TABLE IF EXISTS kkt_document_json;
CREATE external TABLE kkt_document_json (
    subtype String,
    ofdId String,
    protocolSubversion BIGINT,
```

```
    ...
)
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
...
```

Дополнительные комментарии

1. Во всех задачах кроме 3-й выведите только TOP-50 строк в ответ (чтоб не перегружать систему лишним выводом).
2. Подключить json-serde в HUE:

```
add jar
hdfs:/opt/cloudera/parcels/CDH/lib/hive/lib/json-serde-1.3.8-jar-with-dependencies.jar;
```

3. В датасете есть некорректные строки, поэтому при парсинге данных используйте опцию `"ignore.malformed.json" = "true"`.
4. В задачах 3 и 5 отсортируйте результат по UserInn в порядке возрастания. UserInn строка, поэтому сортируйте лексикографически[3].
5. В тестирующей системе для вас уже создана база данных Hive. Её название совпадает с вашим логином в gitLab.atp-fivt.org (например, **velkerr**). Поэтому код создания базы НЕ НУЖНО запускать в run.sh. Во всех своих запросах, кот. коммитите в Git, подключайтесь к уже созданной базе данных (например, **USE velkerr**).
6. Если ваша база данных Hive, на которой происходит тестирование, испорчена вашими же запросами, заполните форму для её пересоздания.

[3] Например, “100001” < “250”. Order by реализует такую сортировку для строковых типов автоматически.