BIGDATA TEAM

HW #07: Spark Advanced

автор задания:

- Andrey Titov, andrey.titov@bigdatateam.org
- Big Data Instructor @ BigData Team
- Senior Spark Engineer @ NVIDIA

1. Описание задания

В данном ДЗ нужно решить **1 задачу**. Решение надо выполнить с **помощью** Spark.

2. Критерии оценивания

Балл за задачу складывается из:

- **70%** - правильное решение задачи
- **30%** - поддерживаемость и читаемость кода
  - в общем случае см. Clean Code и Google Python Style Guide
  - оценка качества будет проводиться автоматическим вызовом pylint:
    - `pylint *.py -d invalid-name,missing-docstring`
    - качество кода должно оцениваться выше 8.0 / 10.0
    - проверяем код **Python версии 3** с помощью `pylint==2.5.3`
- **0%** - эффективность решения (такие как потребляемые CPU-ресурсы, скорость выполнения (в предположении свободного кластера)).

Discounts (скидки и другие акции):

- **100%** за плагиат в решениях (всем участникам процесса)
- **100%** за посылку решения после hard deadline
- **30%** за посылку решения в после soft deadline и до hard deadline
- **5%** за каждую новую посылку (одна дополнительная посылка бесплатно)

Формула подсчета финальной оценки[1]:

$$max(0,\ 0.95^{max(0,\#\text{доп.посылок} - 1)} * (1 - \text{штраф. за. дедлайн. и. списывание})) * \text{последняя. оценка. из. } grader$$

3. (Task ID: spark.nested_crud) Spark Nested CRUD

Одной из частых задач при работе с Dataframes API является CRUD (сокр. от Create, Read, Update, Delete) действия над колонками вашего dataframe. Любая колонка в Spark - это инстанс pyspark.sql.Column.

Чтобы добавить новую (или изменить существующую колонку есть два основных варианта:

[1] результат округляется до целого

- использовать `df.withColumn(name, value)`, где `name` - это имя новой или существующей колонки, а `value` - новое значение колонки, которое имеет тип `pyspark.sql.Column` (например, `lit(1)` - константа 1, `col("b")` значение из колонки b, `rand()` - случайное значение. Все эти функции представлены в пакете `pyspark.sql.functions`;
- использовать `df.select(col1, col2, col3)` - где `col1, col2, col3` - это список колонок, которые нужно "заселектить" в dataframe. При этом это могут быть как новые, так и существующие колонки, имеющие типа `pyspark.sql.Column`.

Если колонка является структурой (то есть имеет тип `struct` в `printSchema`), то выбрать подполя этой колонки можно выбрать, используя `"."` в `col()`, например вот так `col("car.brand")`. К сожалению, в Spark нет встроенной функции, чтобы изменить значение поля brand в колонке `col("car")`. Если вы попытаетесь использовать функцию `df.withColumn("car.brand", new_value)`, то вместо ожидаемого результата вы получете новую колонку с именем `"car.brand"`, которая не будет входить в состав колонки `"car"`:

```
from pyspark.sql.functions import *

cars = \
    spark.range(0,1) \
        .select(
            struct(lit("bmw").alias("brand")).alias("car")
        )

cars.printSchema()

cars.show()

updated = cars.withColumn("car.brand", lit("audi"))
updated.printSchema()
updated.show()
```

В рамках данного домашнего задания вам необходим разработать функцию, которая изменяет любое поле dataframe, включая вложенные поля внутри структур любого уровня вложенности. Ниже представлен скелет этой функции.

```
def update_df(df, columns_dict):
    """
      Updates existing columns or creates new in dataframe df using
columns from columns_dict.
    :param df: input dataframe
```

```
        :type df: pyspark.sql.Dataframe
        :param columns_dict: Key-value dictionary of columns which need to
    be updated. Key is a column name in
        the format of path.to.col
        :type param: Dict[str, pyspark.sql.Column]
        :return: dataframe with updated columns
        :rtype pyspark.sql.DataFrame
        """
        updated_df = df

        # TODO
        return updated_df
```

Этот тест ниже поможет вам понять задачу и проверить правильность работы функции

```
from pyspark.sql.functions import *

input_df = \
    spark.range(0,1) \
        .select(
            struct(
                lit("bmw").alias("brand"),
                lit("220i").alias("model"),
                struct(
                    lit("rear").alias("wheel_drive"),
                    lit("automatic").alias("gear_box")
                ).alias("transmission")
            ).alias("car")
        )

input_df.printSchema()
input_df.show(1, False)

updates = {
    "car.brand": lit("audi"),
    "car.transmission.wheel_drive": lit("all"),
    "car.color": lit("black"),
    "owner.first_name": lit("Ivan"),
    "owner.last_name": lit("Ivanov"),
}
```

```
def check_results(df):
    assert set(df.columns) == set(["car", "owner"])
    assert set(df.select(col("car.*")).columns) == set(["brand",
"model", "transmission", "color"])
    assert set(df.select(col("owner.*")).columns) == set(["first_name",
"last_name"])
    assert df.filter(col("`car`.`transmission`.`wheel_drive`") ==
"all").count() == 1
    assert df.filter(col("`car`.`transmission`.`gear_box`") ==
"automatic").count() == 1
    print("All tests passed!")
```

Ниже приведен пример правильного dataframe, который должна сгенерировать функция:

```
valid_results = \
    spark.range(0,1) \
        .select(
            struct(
                lit("audi").alias("brand"),
                lit("220i").alias("model"),
                struct(
                    lit("all").alias("wheel_drive"),
                    lit("automatic").alias("gear_box")
                ).alias("transmission"),
                lit("black").alias("color")
            ).alias("car"),
            struct(
                lit("Ivan").alias("first_name"),
                lit("Ivanov").alias("last_name")
            ).alias("owner")
        )
valid_results.printSchema()

check_results(valid_results)

# Ваша функция должна успешно пройти тест check_results
check_results(update_df(input_df, updates))
```

P.S. проверять ваши функции мы будем на своих датасетах, поэтому хардкодить имена колонок в функцию не стоит ;) Такие решения засчитывать не будем

4. Правила оформления задания

Оформление задания:

- Код задания (Short name): **HW7:Spark-advanced(Nested_crud)**.
- PySpark-скрипт (**отправляем файл, а не zip-архив**) с реализованной функциональностью назвать **X5BD2021Q1_<Surname>_<Name>_SparkNestedCRUD.py**
- Важно, чтобы **в глобальной области видимости не создавался Spark Context**, поскольку он будет создаваться внешним скриптов, а реализованная функциональность (update_df) будет импортироваться. Если есть необходимость создавать SparkContext - используйте клаузу "__main__", чтобы этот код не запускался при импорте.
- Для того, чтобы сдать задание необходимо:
    - Зарегистрироваться и залогиниться в сервисе Everest
    - Перейти на страницу приложения: BDT-grader-X5-BD
    - Выбрать вкладку Submit Job (если отображается иная).
    - Выбрать в качестве "Task": **HW7:Spark-advanced(Nested_crud)**[2]
    - Загрузить в качестве "Task solution" файл с решением
    - В качестве Sender ID указать тот, который был выслан по почте
- Если Вы видите надпись "You are not allowed to run this application" во вкладке Submit Job в Everest, то на данный момент сдача закрыта (нет доступных для сдачи домашних заданий, по техническим причинам или другое). Попробуйте, пожалуйста, еще раз через некоторое время. Если Вы еще ни разу не сдавали, у коллег сдача работает, но Вы видите такое сообщение, сообщите нам об этом.
- Ситуации:
  * система оценивания показывает оценку (Grade) < 0, а отчет (Grading report) не помогает решить проблему (пример помощи: в случае неправильно указанного Sender ID система вернет -2 и информацию о том, что его нужно поправить);
  * показывает 0 и в отчете (Grading report) не указано, какие тесты не пройдены.
  Если Вы столкнулись с какой-то из них присылайте ссылку на выполненное задание (Job) на почту с темой письма "Short name. ФИО.". Например: "**HW7:Spark-advanced(Nested_crud)**. Иванов Иван Иванович."
  Пример ссылки: https://everest.distcomp.org/jobs/67893456230000abc0123def
  Внимание: Если до дедлайна остается меньше суток, и Вы знаете (сами проверили или коллеги сообщили), что сдача решений сломана, обязательно

[2] Сервисный ID: spark.nested_crud

сдайте свое решение и напишите письмо, как написано выше, чтобы мы видели, какое решение Вы имели до дедлайна и смогли его оценить.

- Перед отправкой задания, оставьте, пожалуйста, отзыв о домашнем задании по сссылке: http://rebrand.ly/x5bd2021q1_feedback_hw07. Это позволит нам скорректировать учебную нагрузку по следующим заданиям (в зависимости от того, сколько часов уходит на решение ДЗ), а также ответить на интересующие вопросы.

Любые вопросы / комментарии / предложения можно писать в телеграм-канал курса или на почту bigdata_x52021q1@bigdatateam.org .

Всем удачи!