BIGDATA TEAM

HW #07: Spark Advanced

автор задания:

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

редактор задания[1]:

- Виктория Мукукенова
- Big Data Mentor @ BigData Team
- Data Scientist @ Tele2

[1] Хочешь стать ментором и оставить след в истории Big Data? Тогда хорошо учись, помогай другим и дай нам знать о своем желании. Смело пиши автору задания или менеджеру учебного курса.

1. Описание задания: (Task ID: spark.nested_crud) Spark Nested CRUD

Помимо привычных и тривиальных случаев, когда датафрейм имеет вид:

Набор колонок (столбцов) → данные

Бывают случаи, когда данные имеют вложенную структуру. Примеры:

- для товаров в интернет-магазине можно выделить категорию “еда” и подкатегорию (вложенную структуру) “рыба”. Можно пойти ещё дальше и указать вид приготовления (свежая/ солёная/ копчёная). Тогда структура такого датафрейма будем иметь вид:

  товары → категория(еда) → подкатегория (рыба) → вид приготовления → данные

- автомобили, где вложенной структурой можно указать их тип - легковой/ грузовой, уточнить бренд, указать модель авто и учесть, что модель бывает с разными коробками передач:

  автомобили → тип → бренд → модель → тип коробки передач → данные

В данном ДЗ с помощью Spark нужно решить 1 задачу по трансформации данных с вложенной структурой.

1.1. Требования к реализации

Решать будем наиболее часто встречающуюся проблему с Dataframes API - 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:

- 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 тип `pyspark.sql.Column`.

Если колонка является структурой (то есть, имеет тип `struct` в `printSchema`), то 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 получите 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 необходимо 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
```

1.2. Описание данных

Тест ниже поможет 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.

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 --ignored-modules=pyspark.sql.functions`
    - 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 дополнительную посылку в тестирующую систему (всего можно делать до 3-х посылок без штрафа):

Пример работы системы штрафов:

День	Посылка	Штраф
День 1	Посылка 1	Без штрафа
День 1	Посылка 2	Без штрафа
День 1	Посылка 3	Без штрафа
День 1	Посылка 4	-5%
День 2	Посылка 5	-5%
День 3	Посылка 6	-5%
Итоговый штраф: -15%		

Для подсчета финальной оценки **всегда** берется **последняя** оценка из Grader.

3. Правила оформления задания

Перед отправкой задания оставьте, пожалуйста, отзыв о домашнем задании по ссылке: https://rebrand.ly/bdmade2022q2_feedback_hw. 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.

Оформление задания:

- Код задания (Short name): **HW07:Spark_advanced**.
- PySpark-скрипт (**отправляем файл, а не zip-архив**) с реализованной функциональностью назвать **`BD_MADE_2022_Q2_<Surname>_<Name>_SparkNestedCRUD.py`**
- Важно, чтобы **в глобальной области видимости не создавался Spark Context**, поскольку он будет создаваться внешним скриптом, а реализованная функциональность (update_df) будет импортироваться. Если есть необходимость создавать SparkContext - используйте клаузу “__main__”, чтобы этот код не запускался при импорте.
- Не забывайте импортировать необходимые для работы вашей функции библиотеки
- Для того, чтобы сдать задание, необходимо:
  - Зарегистрироваться и залогиниться в сервисе Everest
  - Перейти на страницу приложения: MADE BigData Grader
  - Выбрать вкладку Submit Job (если отображается иная).
  - Выбрать в качестве “Task”: **HW07:Spark_advanced**[2]
  - Загрузить в качестве “Task solution” файл с решением
  - В качестве Access Token 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 Access Token система вернет -2 и информацию о том, что его нужно поправить);
  * система 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: “**HW07:Spark_advanced**. Иванов Иван Иванович.”

[2] Сервисный ID: spark.nested_crud

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

Любые вопросы / комментарии / предложения можно писать в телеграм-канал курса или на почту bd_made2022q2@bigdatateam.org.
Всем удачи!

4. FAQ (часто задаваемые вопросы)

"You are not allowed to run this application", что делать?
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.

Grader показывает 0 или < 0, а отчет (Grading report) не помогает решить проблему
Ситуации: * система оценивания показывает оценку (Grade) < 0, а отчет (Grading report) не помогает решить проблему (пример помощи: в случае неправильно указанного Access Token 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: “**HW07:Spark_advanced**. Иванов Иван Иванович.” Пример ссылки: https://everest.distcomp.org/jobs/67893456230000abc0123def

Что в отчете Grader означает проверка X ?
Как читать отчет: Для каждого теста • `raw_score` - балл за конкретный тест. Может быть как бинарным (1\0), так и находиться в интервале от 0 до 1 • `score - raw_score * weight` (вес теста в общей оценке). Вес указан для каждого теста ниже Итоговая оценка: смотрите строку Score (сумма Score всех индивидуальных тестов) внизу отчета. **Правильность решения задачи:** test_2nested_struct: (weight = 0.3) - получен корректный результат, если на вход подан датафрейм 2-ого уровня вложенности

test_10nested_struct: (weight = 0.2) - получен корректный результат, если на вход подан датафрейм 10-ого уровня вложенности
test_col_modification_not_lit: (weight = 0.1) - получен корректный результат, если на вход в качестве значения для колонки подана встроенная функция
test_col_parallel_struct: (weight = 0.1) - получен корректный результат, если на вход подан датафрейм с одинаковыми полями в разных частях своей структуры

Поддерживаемость и читаемость кода:
test_py_files_min_lint_score: (weight = 0.3) - качество кода в .py файлах оценивается выше 8.0

Appendix. Подсказки (если не получается решить ДЗ).

Как получить схему Dataframe в формате json?

```
df.schema.jsonValue()
```

Как изменить вложенность структуры Dataframe[3]?

Пример изменения датафрейма с одной вложенной структурой в другую:

```
df.select(
    struct(
        col("car.brand").alias("brand"),
        col("car.transmission.wheel_drive").alias("wheel_drive")
    ).alias("car_specifications")
).show()
```

[3] Пример dataframe из п. 1.2 Описание данных