CLOUD BI

5. forduló

A kategória támogatója: DXC Technology

Ismertető a feladatlaphoz

A forduló alatt a képek nem töltödtek be megfelelően egy képmásolási hiba miatt, ezért a forduló eredményeit érvénytelenítettük.

1. feladat 0 pont

Az alábbi programnyelvek közül, melyiken írodott a **Spark**?

Válasz

- ○ Java
- ○ Scala
- ○ Python
- ○ R

2. feladat 0 pont

```
Az alábbi lehetőségek közül melyik nem beépített funkciója a
Sparknak?
a.  A benne írodott programok támogatják az in-memory számításokat
b.  A benne írodott programok több fájlrendszerrel kompatibilisek
c.  A benne írodott programok nagy hibatűréssel rendelkeznek
d.  A benne írodott programok a Sparknak köszönhetően
költséghatékonyak
```

Válasz

- A benne írodott programok támogatják az in-memory számításokat
- A benne írodott programok több fájlrendszerrel kompatibilisek
- A benne írodott programok nagy hibatűréssel rendelkeznek
- A benne írodott programok a Sparknak köszönhetően költséghatékonyak

3. feladat 0 pont

Melyik állítás(ok) igaz(ak) az alábbiak közül a **get_dtype** függvényre?

```python
def get_dtype(table_name, column_name):
    df = spark.sql(f"select * from {table_name}")
    return [dtype for name, dtype in df.dtypes if name == column_name][0]
```

Válaszok

- Amennyiben a függvényt olyan táblával hívjuk meg amelyben nem szerepel column_name nevű oszlop akkor a program IndexError-al terminálni fog
- A függvény visszaad egy listát azokról az oszlopokról ahol az adattípus megegyezik a column_name változóban megadott oszlop adattípusával
- A függvény visszaadja az adattípusát a column_name változóban megadott oszlopnak amennyiben az megtalálható a spark.sql ben lefuttatott queryből készített Spark Dataframe dtypes-ai között mint kulcs
- A spark.sql ben megfuttatott query eredményéből készített Spark Dataframet szűri le és adja vissza a column_name változóban megadott oszlopnévvel
- A spark.sql ben megfuttatott queryből a neve ellenére nem egy Spark DataFrame készül el így a .dtypes hívásra a program terminálni fog egy AttributeError hibával

4. feladat 0 pont

```
Adja meg, hogy mennyi az executor szám az alábbi clusteren amennyiben kijelenthetjük,
hogy a tanulmányok alapján az egyidejűleg futtatható feladatok száma (concurrency) 5.

6 Node amelyből egy Node 16core és 64GB ram

(A levezetés mélységétől függően több válasz is elfogadható, de alapvetően egy számot
várunk megoldásnak, pl.: 47)
```

Válaszok

5. feladat 0 pont

```
Adja meg, hogy mennyi az executor memory az alábbi clusteren amennyiben
kijelenthetjük, hogy a tanulmányok alapján az egyidejűleg futtatható feladatok száma
(concurrency) 5.

6 Node amelyből egy Node 16core és 64GB ram

(A levezetés mélységétől függően több válasz is elfogadható, de alapvetően egy egész
számot várunk megoldásnak, az executor memory mértékét GB-ban, pl.: 43)
```

Válaszok

6. feladat 0 pont

Az alábbi kódban egy **parts_summary** nevű listába összegyüjtöttük egy tábláról hogy az egyes partíciói milyen where feltétellel érhetőek el és ott hány fizikális fájl található (HDFS-en). Ennek tudatában milyen funkciót lát el az alábbi kód?

```python
def who_knows_what_i_do(table_name, parts_summary):
    for where_filter, num_files in parts_summary:
        if num_files > 1:
            part_df = spark.sql(f"select * from {table_name} where {where_filter}").checkpoint()
            logger.info(f"select * from {table_name} where {where_filter}")
            (
                part_df
                .coalesce(1)
                .write
                .insertInto(table_name, overwrite=True)
            )

            spark.sql(f"REFRESH TABLE {table_name}")

            logger.info(
                f"num files > 1 ezért lefuttattuk a kódot")
        else:
            logger.info(
                f"num_files 1 volt")
```

Válasz

- A kód minden említett partíciót megkeres és épít belőle egy park_df nevű Spark DataFrame-t amelyet aztán egyben kiír a table_name nevű táblába
- A kód abban az esetben ha a partició helyén levő fájlok száma nagyobb mint 1 ezeket egy part_df nevű DataFrame segítségével újraírja 1 fájllá a coalesce(1) és overwrite=True funkciók segítségével
- A kód minden esetben amikor a fájlok száma nagyobb mint 1 egy part_df nevű DataFrame segítségével összegyüjti az adott partíció adatait majd csak ezeket a partíciókat befrissíti a REFRESH TABLE parancs segítségével
- A kód az összegyüjtött partíciók where feltételeit felhasználva végigiterál a táblán és ahol a fájlok száma nagyobb mint 1 ott a HDFS en a coalesce(1) ben megadott 1 számot követve a REFRESH TABLE hatására 1 fájllá mergeli az ott található összes fájlt.

Megoldások beküldése