# PO12: Przetwarzanie Big Data 22/23-Z



| | |
|---|---|
| **Rozpoczęto** | środa, 21 grudnia 2022, 20:13 |
| **Stan** | Ukończone |
| **Ukończono** | środa, 21 grudnia 2022, 20:37 |
| **Wykorzystany czas** | 23 min. 28 sek. |
| **Punkty** | 50,00/60,00 |
| **Ocena** | **8,33** (**83,33**%) |

Nawigacja w teście

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | |
| 7 | 8 | 9 | 10 | | | |

Pokaż jedną stronę na raz

Zakończ przegląd

## Zadanie 1

Pytanie **1**
Poprawnie
Punkty: 4,00 z 4,00
Oflaguj pytanie

Utwórz bazę tabelę *Delta Lake* o nazwie `customers`, której lokalizacja będzie `/tmp/delta-customers`.

Kolumny tabeli muszą odpowiadać kolumnom danych źródłowych.
Wszystkie kolumny powinny być ciągami znaków o długości do 200 znaków

```
DROP TABLE IF  EXISTS  customers;


CREATE TABLE  customers  (
 id   VARCHAR(200),
 name VARCHAR(200),
 address VARCHAR(200),
 zipcode VARCHAR(200),
 city VARCHAR(200),
 country VARCHAR(200),
 effectiveDate VARCHAR(200)
 )
USING  DELTA
 LOCATION  "/tmp/delta-customers";
```

## Zadanie 2

Pytanie **2**
Poprawnie
Punkty: 4,00 z 4,00
Oflaguj pytanie

Wprowadź do utworzonej przez Ciebie tabeli dane o naszych klientach obowiązujące na dzień `2021-01-01`.

```
INSERT INTO  customers  (id, name, address, zipcode, city, country, effectiveDate)

 SELECT  id, name, address,  zipcode  , city, country, effectiveDate
from (
    select id, name, address, zipcode, city, country, effectiveDate,
           rank() over (partition by id order by to_date(effectiveDate,'dd-MM-yyyy') desc) as version
    from  source_data 
    where  to_date(effectiveDate,'dd-MM-yyyy') >= date '2021-01-01'
    and    to_date(effectiveDate,'dd-MM-yyyy') < date '2021-01-01'
    )
where version = 1;
```

## Zadanie 3

Pytanie **3**
Poprawnie
Punkty: 4,00 z 4,00
Oflaguj pytanie

Zmień kraj zamieszkania na wartość `Poland` w wszystkich klientów posiadających wartość id mniejszą niż 50.
Zwróć uwagę, że obecnie id jest ciągiem znaków. Użyj wyrażenia `cast(id as int)` aby wykonać zadanie prawidłowo.

```
update  customers 

set  country  =  'Poland'

 WHERE  cast(id as int) <  50  ;
```

## Zadanie 4

Pytanie **4**
Poprawnie
Punkty: 4,00 z 4,00
Oflaguj pytanie

Usuń z tabeli `customers` tych klientów, którzy zmienili swoje dane w styczniu 2021. Identyfikacja klientów odbywać się powinna za każdym razem w oparciu o atrybut `id`.

Jeśli okaże się, że polecenie DELETE nie wspiera podzapytań, skorzystaj z interfejsu w Scali.
Ilub za pomocą poniższego kodu, w którym w jednym paragrafie uzyskaj identyfikatory klientów do usunięcia, a następnie wkomponuj uzyskaną wartość w polecenie usuwające klientów.

```
val usunIDS = spark.sql("""select id as usun
from (
    select id, name, address, zipcode, city, country, effectiveDate,
           rank() over (partition by id order by to_date(effectiveDate,'dd-MM-yyyy') desc) as version
    from  source_data
    where  to_date(effectiveDate,'dd-MM-yyyy') >= date '2021-01-01'
    and    to_date(effectiveDate,'dd-MM-yyyy') < date '2021-02-01'
    )
where version = 1""").collect.map(_.toSeq).flatten.mkString(",")

 spark  .sql(s"""
delete from  customers 

 where  id in (  $usunIDS  ) """)
```

## Zadanie 5

Pytanie **5**
Poprawnie
Punkty: 10,00 z 10,00
Oflaguj pytanie

Korzystając z jednego polecenia `merge` jednocześnie

- wstaw nowych klientów, którzy pojawili się po raz pierwszy w lutym 2021
- zaktualizuj dane adresowe oraz `effectiveDate` tych klientów, którzy w tym samym czasie te dane zmienili.

Poniższe polecenie uzyskuje dane z obu grup klientów:

```
select id, name, address, zipcode, city, country, effectiveDate
from (
    select id, name, address, zipcode, city, country, effectiveDate,
           rank() over (partition by id order by to_date(effectiveDate,'dd-MM-yyyy') desc) as version
    from  source_data
    where  to_date(effectiveDate,'dd-MM-yyyy') >= date '2021-02-01'
    and    to_date(effectiveDate,'dd-MM-yyyy') < date '2021-03-01'
    )
where version = 1
```

```
 MERGE  INTO customers
USING (select id, name, address, zipcode, city, country, effectiveDate
from (
    select id, name, address, zipcode, city, country, effectiveDate,
           rank() over (partition by id order by to_date(effectiveDate,'dd-MM-yyyy') desc) as version
    from  source_data
    where  to_date(effectiveDate,'dd-MM-yyyy') >= date '2021-02-01'
    and    to_date(effectiveDate,'dd-MM-yyyy') < date '2021-03-01'
    )
where version = 1)  new_customers 

ON  customers  .  id  = new_customers.id

WHEN  MATCHED  THEN

 UPDATE  SET customers.address = new_customers.address,
            customers.zipcode = new_customers.zipcode,
            customers.city    = new_customers.city,
            customers.country = new_customers.country,
            customers.effectiveDate = new_customers.effectiveDate
WHEN  NOT MATCHED 

 THEN  INSERT  (id, name,  address  , zipcode, city, country, effectiveDate)

  VALUES  (id, name, address, zipcode, city, country, effectiveDate)
```

## Kilka pytań

Pytanie **6**
Poprawnie
Punkty: 2,00 z 2,00
Oflaguj pytanie

Zwróć uwagę, że niektóre z tych operacji mają flagę `sublinkAppend` zapaloną. **Które to były operacje?**

- ☐ CREATE OR REPLACE TABLE AS SELECT
- ☐ ADD COLUMNS
- ☑ WRITE ✔
- ☐ MERGE
- ☐ DELETE
- ☐ UPDATE
- ☐ CREATE [OR REPLACE] TABLE ✔

Punkty: 1,00 z 1,00

Poprawna odpowiedź to:
- WRITE
- CREATE [OR REPLACE] TABLE

**Czy liczba plików** utworzonych przy **pierwszej operacji** (zaobserwowanych w katalogu tabeli) **zgadza się** z wpisem w uzyskanych powyżej metadanych?

Tak  0  ✔

## Zadanie 6

Pytanie **7**
Poprawnie
Punkty: 10,00 z 10,00
Oflaguj pytanie

Okazało się, że ta operacja zmiany kraju na wartość `Poland` była błędem.
Twoim zadaniem jest przywrócić wartości, które zostały nadpisane przez tą modifikację.
Nie wiemy jednej danych jeśli pojawiły się później(po Twoim poleceniu `update`) aktualizacje adresu (skorzystaj z `effectiveDate`).
Świetnie do takiej naprawy może się przydać operacja `merge`. Tym razem będzie ona miała tylko sekcję `WHEN MATCHED THEN`.

Oczywiście nie korzystaj z danych źródłowych - ich już nie ma. Jest tylko Twoja tabela Delta Lake.

Skorzystaj z DataFrame API.

```
import io.delta.tables._

val deltaTable = DeltaTable.  forPath  (spark, "/tmp/delta-customers")


val poprawneDF = spark.read
  .format  ("delta")
 .option("  versionAsOf  ", "1")
  .load  ("/tmp/delta-customers")

deltaTable.as("old")
  .merge  ( poprawneDF  .as("new"), "old.id = new.id and to_date(old.effectiveDate,'dd-MM-yyyy') < date '2021-01-01'")
  .whenMatched
  .updateExpr  (Map("country" -> "  new  .country"))
  .execute  ()
```

## Zadanie 7

Pytanie **8**
Poprawnie
Punkty: 5,00 z 5,00
Oflaguj pytanie

Kolumna ta jest nam potrzebna i będzie wykorzystywana w następnym zadaniu, dlatego skorzystaj z poniższego kodu oraz wskazówki powyżej, aby podczas wstawiania tych danych wprowadzić brakującą kolumnę do schematu

```
val data032021 = spark.sql("""
select id,
       name, address, zipcode, city, country,
       effectiveDate,
       cast(null as date) as endDate
from (
    select id, name, address, zipcode, city, country, effectiveDate,
           rank() over (partition by id order by to_date(effectiveDate,'dd-MM-yyyy') desc) as version
    from  source_data
    where  to_date(effectiveDate,'dd-MM-yyyy') >= date '2021-03-01'
    and    to_date(effectiveDate,'dd-MM-yyyy') < date '2021-04-01'
    )
where version = 1""")

 data032021  .write.option("  mergeSchema  ", "true").
                    format("  delta  ").
                    mode("  append  ").
                    saveAsTable  ("customers")
```

## Zadanie 8

Pytanie **9**
Poprawnie
Punkty: 7,00 z 7,00
Oflaguj pytanie

Korzystając z tego mechanizmu zmień typy kolumn, o których wspominaliśmy.

Przy okazji wycofaj ostatnią aktualizację. Nie była przemyślana. Pojawiły się duplikaty w naszych danych.

Sprawdź, którą wersję danych potrzebujesz.

```
import io.delta.tables._
val fullHistDF = custDeltaTable.history().select($"version",$"timestamp",$"operation",$"isBlindAppend",$"operationMetrics")
fullHistDF.show(false)

val custDeltaTable = DeltaTable.forPath(spark, "/tmp/delta-customers")


spark.read.format("delta")
 .option("versionAsOf", "10")
 .table("customers")
 .withColumn(" effectiveDate  ", col(" effectiveDate  ").cast("date"))
  .withColumn  (" id  ", col(" id  ").cast("int"))
 .write
 .format("delta")
  .mode  ("overwrite")
 .option("  overwriteSchema  ", "true")
 .saveAsTable("customers")
```

## Zadanie 9

Pytanie **10**
Nie udzielono odpowiedzi
Punkty maks.: 10,00
Oflaguj pytanie

**Zaimplementuj** funkcję, która na podstawie danych źródłowych z kolejnego miesiąca (parametr funkcji) będzie aktualizowała zawartość tabeli `customers` zgodnie z regułami ***SCD Type 2***.

Po zakończonej implementacji **sprawdź jej działanie**.

Jeśli chcesz, aby obecne dane w tabeli customers powinny zostać poprawione, **dokonaj wcześniej stosownych korekt**.

Jeśli chcesz skorzystać ze strony:
https://docs.delta.io/0.4.0/delta-update.html#slowly-changing-data-scd-type-2-operation-into-delta-tables

Zakończ przegląd