Hive

Tym razem naszym zbiorem danych będzie *Craft Beers Dataset*. Zbiór ten zawiera dane o ponad 2 tysiącach tzw. piw rzemieślniczych (*craft canned beers*) ze Stanów Zjednoczonych oraz ponad pięciuset browarów produkujących te piwa. Dane te zostały uzupełnione o informacje na temat stanów Ameryki Północnej obejmujące liczbę ludności, powierzchnię, pełną nazwę i stolicę.

Zbiór danych pochodzi z `https://www.kaggle.com/nickhould/craft-cans`
Dane dotyczące stanów Ameryki Północnej zostały uzyskane z
`https://pl.wikipedia.org/wiki/Podzia%C5%82_terytorialny_Stan%C3%B3w_Zjednoczonych`

Struktura plików wchodzących w skład zestawu danych jest następująca

Piwa (`beers.csv`):

- `abv` – zawartość alkoholu w procentach (0 – piwo bezalkoholowe, 1 – 100% alkoholu) – `Numeric`
- `ibu` – międzynarodowa jednostka gorzkości, która określa jak bardzo gorzkie jest piwo – `Numeric`
- `id` – identyfikator piwa – `Numeric`
- `name` – nazwa piwa – `String`
- `style` – typ/styl piwa (lager, ale, IPA, itp.) – `String`
- `brewery_id` – identyfikator browaru będącego producentem piwa – `Numeric`
- `ounces` – objętość piwa w uncjach (1 ~ 30cm^3) – `Numeric`

Browary (`breweries.csv`):

- `brewery_id` – identyfikator browaru – `Numeric`
- `name` – nazwa browaru – `String`
- `city` – miasto, w którym browar ma swoją siedzibę – `String`
- `state` – symbol stanu, w którym browar ma swoją siedzibę – `String`

Stanów Ameryki Północnej (`states.csv`):

- `id` – identyfikator stanu – `Numeric`
- `name` – pełna nazwa stanu – `String`
- `abbrev` – symbol stanu – `String`
- `area` – powierzchnia – `Numeric`
- `population` – ludność (w 2006 roku) – `Numeric`
- `capital` – stolica – `String`
- `statehood` – data utworzenia stanu

Przygotowanie środowiska

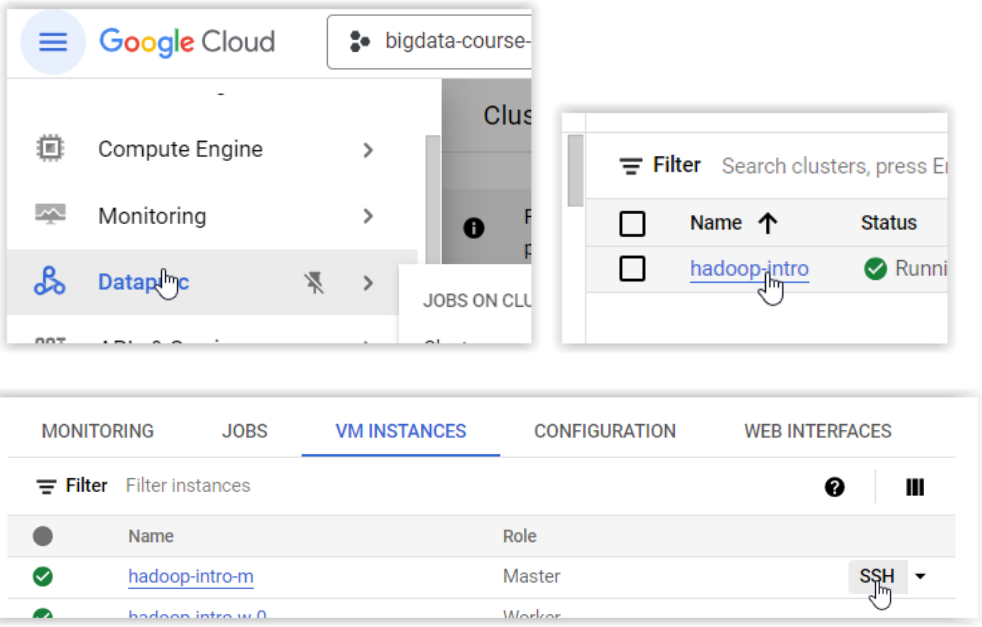
Na początku przygotujemy środowisko przetwarzania danych.

Wykorzystamy w tym celu środowisko udostępniane przez firmę Google – Google Cloud Platform.

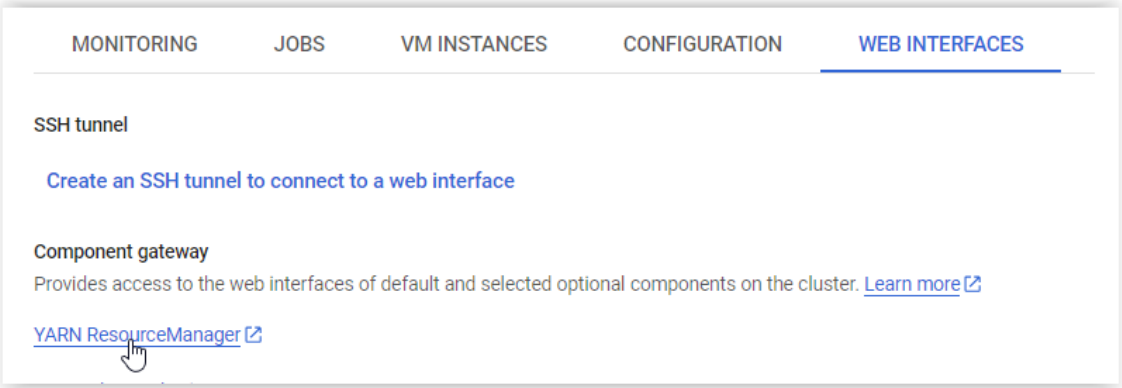
1. Korzystając z poniższego polecenia i konsoli Cloud Shell utwórz klaster.
   Podmień w poleceniu przed jego wykonaniem stosowne fragmenty.

```
gcloud dataproc clusters create ${CLUSTER_NAME} \
 --enable-component-gateway \
 --region ${REGION} \
 --master-machine-type n1-standard-2 --master-boot-disk-size 50 \
 --num-workers 2 --worker-machine-type n1-standard-2 --worker-boot-disk-size 50 \
 --image-version 2.1-debian11 --optional-components ZEPPELIN \
 --project ${PROJECT_ID} --max-age=2h
```

2. Przejdź na zakładkę z instancjami utworzonych maszyn wchodzących w skład naszego klastra i podłącz się za pomocą terminala SSH do węzła master w naszym klastrze.



3. Dodatkowo, korzystając z interfejsów sieciowych, otwórz stronę managera zasobów YARN



Analiza ustawień platformy Hive

4. Zanim zaczniemy przetwarzanie danych, „rozglądniemy się” w ustawieniach platformy Hive. Korzystając z terminala uruchom klienta linii poleceń `beeline`. Jest to zwykły klient JDBC obsługujący jednak dodatkowo dialekt Hive SQL oraz dodatkowe komendy np. dotyczące działań na *Hive Metastore*.

```
beeline -n ${USER} -u jdbc:hive2://localhost:10000/default --silent
```

Zwróć uwagę na wersję platformy Hive.

```
Connecting to jdbc:hive2://localhost:10000/default
Connected to: Apache Hive (version 3.1.3)
Driver: Hive JDBC (version 3.1.3)
Transaction isolation: TRANSACTION_REPEATABLE_READ
Beeline version 3.1.3 by Apache Hive
0: jdbc:hive2://localhost:10000/default>
```

5. Za pomocą poniższych poleceń dowiedz się:
   a. jako kto zostałeś zalogowany
   b. jakie są dostępne bazy danych
   c. jakie tabele dostępne są w bieżącej bazie danych

```
select logged_in_user();
show databases;
show tables;
```

```
0: jdbc:hive2://localhost:10000/default> select logged_in_user();
+------------------------+
|          _c0           |
+------------------------+
| jankiewicz_krzysztof   |
+------------------------+
0: jdbc:hive2://localhost:10000/default> show databases;
+----------------+
| database_name  |
+----------------+
| default        |
+----------------+
0: jdbc:hive2://localhost:10000/default> show tables;
+-----------+
| tab_name  |
+-----------+
+-----------+
```

6. Korzystając z poniższego polecenia uzyskaj odpowiedź na poniższe pytania:

```
set {option_name};
```

   a. jaki silnik wykonawczy jest wykorzystywany (`hive.execution.engine`)?
   b. czy wykorzystywany jest optymalizator kosztowy (`hive.cbo.enable`)?
   c. czy mechanizm scalania plików jest uruchamiany (`hive.compactor.initiator.on`)?
   d. czy dla nowo tworzonych tabel będą wyliczane statystyki (`hive.stats.autogather`)?

Zapewne Hive nie odniósłby aż takiego sukcesu i nie wytyczył kierunków dla wielu innych narzędzi Big Data gdyby nie jego pomysł na repozytorium metadanych – Metastore. Sprawdźmy jak wygląda konfiguracja usługi Metastore w naszym przypadku.

7. Nadal korzystając z polecenia `set` sprawdź:
   a. Z jakiej bazy danych korzysta Metastore (`javax.jdo.option.ConnectionURL`)?
   b. Za pomocą jakiego URI możemy połączyć się z usługą Metastore (`hive.metastore.uris`)?

8. Wywołaj jeszcze jedno polecenie, które sprawdzi czy rzeczywiście zewnętrzna w stosunku do Hive usługa hive-metastore jest uruchomiona.

```
!sh systemctl list-units hive-metastore.service
```

Na niektórych dystrybucjach/platformach istnieją interfejsy graficzne dla platformy Hive np. Hortonworks udostępniał kiedyś *Hive View*, a później *Data Analytics Studio.* Cloudera ma *Hue*.
Platforma Azure ma usługę HDInsight i widok *Hive*. Na GCP pozostaje:

- Interfejs CLI – `beeline`
- Notatniki Zeppelina

Na notatniki przyjdzie jeszcze czas. Na razie pozostaniemy w naszym `beeline`.

Przesłanie analizowanych danych

Jeśli chcesz możesz wyjść z klienta beeline i uruchomić go ponownie bez opcji `--silent`. Używając jej wyniki są "ładne", "czyste". Jednak jednocześnie tracimy mnóstwo ważnych informacji o tym co dzieje się "pod maską".

9. Utwórz bazę danych o nazwie `beers`.
   Ustaw nowoutworzoną bazę danych `beers` jako bazę domyślna

```
0: jdbc:hive2://localhost:10000/default> create database beers;
0: jdbc:hive2://localhost:10000/default> use beers;
0: jdbc:hive2://localhost:10000/default> show databases;
+----------------+
| database_name  |
+----------------+
| beers          |
| default        |
+----------------+
```

10. Przed załadowaniem danych na platformę Hive musimy dostarczyć te dane na maszynę, na której Hive działa. Hive potrafi dane załadować zarówno z poziomu lokalnego systemu plików jak i z poziomu HDFS. Załaduj potrzebne nam dane

```
!sh mkdir /tmp/source
!sh wget http://jankiewicz.pl/bigdata/hadoop-intro/beers.zip
!sh unzip beers.zip -d /tmp/source
```

11. Podczas przetwarzania danych chcemy skorzystać dobrodziejstw formatu ORC. Jak zapewne wiemy, dane do tabel, które oparte są na tym formacie, mogą być załadowane za pomocą poleceń DML, a to oznacza, że potrzebujemy pośrednika. Niech naszym pośrednikiem, przy ładowaniu danych dotyczących piw, będzie tabela w formacie tekstowym – pliki, które posiadamy odpowiadają jej definicji. Utwórz stosowną tabelę

```
create table beers_sf(line int, abv float, ibu float, id int,
    name string, style string, brewery_id int, ounces float)
    ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED AS TEXTFILE;
```

12. Za pomocą polecenia `load data` załaduj dane do nowoutworzonej tabeli

```
load data local inpath "/tmp/source/beers/beers.csv"
into table beers_sf;
```

Sprawdź czy Twoje dane znalazły się w tabeli `beers_sf`. Wykonaj w tym celu polecenie

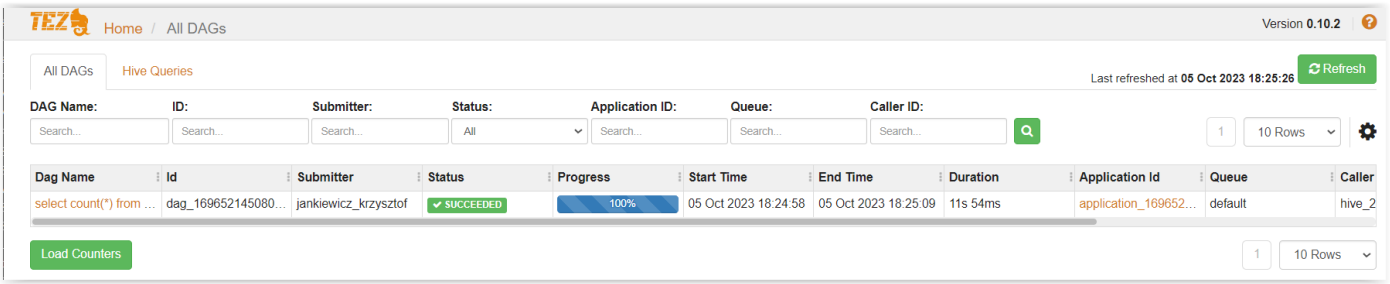
```
select count(*) from beers_sf;
```

```
+-------+
|  _c0  |
+-------+
| 2411  |
+-------+
1 row selected (17.561 seconds)
```

Kilka pytań:

a. Dlaczego to tak długo trwało? Co w tym czasie się działo?
b. Rzuć okiem na interfejs managera zasobów klastra YARN, jaki typ aplikacji jest obecnie uruchomiony?

Przejdź do interfejsu silnika TEZ. Czy zauważasz DAG, który został wykonany w ramach silnika TEZ?



c. Przejdź do szczegółów tego DAG. Z ilu węzłów składa się ten DAG?
d. Jaka klasa przetwarzania (*Processor Class*) odpowiadała za implementację pierwszego z węzłów?
e. Jaka klasa przetwarzania odpowiadała za implementację drugiego z węzłów?
f. Czy jest dla Ciebie jasne dlaczego akurat takie klasy zostały użyte i jak wyglądało to przetwarzanie? Ile różnych wartości klucza utworzył mapper aby reduktor mógł obliczyć nasz wynik?

13. Kontynuujemy naszą pracę z danymi. Za chwilę utworzymy docelową tabelę w formacie ORC. Tabele mogą składać się z wielu partycji i/lub kubełków. Dzięki nim dane są dzielone na wiele plików, a to ułatwia rozpraszanie przetwarzania i skalowalność.

    Partycje definiujemy w szczególności wówczas, gdy atrybut lub zbiór atrybutów jest często wykorzystywany w warunkach selekcji i jednocześnie liczba różnych wartości tych atrybutów jest ograniczona (np. 2-100). W naszych danych nie ma takiego atrybutu, dlatego ograniczymy się jedynie do utworzenia kubełków.

    Oprzemy je o klucz klastrujący składający się z kolumny `brewery_id`. Podobnie postąpimy w przypadku tabeli z browarami. Takie rozwiązanie pozwoli na wykorzystanie połączeń świadomych partycji (*partition-wise join*).

    Ponadto w ramach plików dane uporządkujemy względem poziomu ich gorzkości. Przyspieszy to selekcję danych w oparciu o atrybut `ibu`.

```
create table beers_orc(line int, abv float, ibu float, id int,
    name string, style string, brewery_id int, ounces float)
    CLUSTERED BY (brewery_id) SORTED BY (ibu) INTO 32 BUCKETS
    STORED AS ORC;
```

14. Przepisz dane z tabeli tekstowej do docelowej korzystając z polecenia DML.

```
insert into beers_orc select * from beers_sf where id is not null;
```

    Znowu trochę to trwało. Zaglądnij do interfejsu silnika TEZ. Zobacz do szczegółów nowego DAG.

    a. Ile tym razem było węzłów?
    b. W jakim celu zostały wykorzystane mechanizmy redukcji korzystające z 32 jednostek zadań? Czy nie wystarczyło skorzystać tylko z mapowania? Przecież niczego nie grupowaliśmy, nie sortowaliśmy ani wyliczaliśmy żadnych agregatów.

15. Sprawdź ile wierszy znalazło się w tabeli docelowej.

```
select count(*) from beers_orc;
```

```
+-------+
|  _c0  |
+-------+
| 2410  |
+-------+
1 row selected (0.322 seconds)
```

    a. Dlaczego tym razem wynik uzyskany został tak szybko?
    b. Zaglądnij do interfejsu silnika TEZ. Dlaczego tym razem silnik TEZ nie został użyty?

16. Dane do następnej z tabel załadujemy w nieco inny sposób. Skorzystamy z pośrednictwa tabeli zewnętrznej. Tabela zewnętrzna pozwala na udostępnienie danych bezpośrednio z poziomu HDFS.
    Za pomocą poniższego polecenia utwórz tabelę zewnętrzną udostępniającą dane z pliku breweries.csv

```
CREATE EXTERNAL TABLE IF NOT EXISTS breweries_ext(
        id INT,        name STRING,        city STRING,        state STRING)
    COMMENT 'craft breweries'
    ROW FORMAT DELIMITED
    FIELDS TERMINATED BY ','
    STORED AS TEXTFILE
    location '/tmp/breweries';
```

17. Sprawdź czy katalog `breweries` w HDFS został utworzony. Jeśli tak załaduj do niego plik `breweries.csv`.

```
!sh ls /tmp/source/beers
!sh hadoop fs -copyFromLocal /tmp/source/beers/breweries.csv /tmp/breweries/
```

Sprawdźmy czy nasze dane są w ten sposób dostępne.

```
select count(*) from breweries_ext;
```

```
0: jdbc:hive2://localhost:10000/default> select count(*) from breweries_ext;
+------+
| _c0  |
+------+
| 559  |
+------+
```

18. Utwórz tabelę w formacie ORC i przepisz do niej dane bezpośrednio z pliku.

```
CREATE TABLE IF NOT EXISTS breweries_orc(
        id INT,        name STRING,        city STRING,        state STRING)
    COMMENT 'craft breweries'
    CLUSTERED BY (id) INTO 32 BUCKETS
    STORED AS ORC;

INSERT OVERWRITE TABLE breweries_orc
SELECT * FROM breweries_ext
WHERE id IS NOT NULL;
```

Sprawdź czy wszystko jest OK

```
0: jdbc:hive2://localhost:10000/default> select count(*) from breweries_orc;
+------+
| _c0  |
+------+
| 558  |
+------+
```

19. Danych dotyczących stanów w ogóle nie będziemy ładowali. Będą one dostępne jako tabela zewnętrzna udostępniająca dane z katalogu `/temp/states`. Podmieniając zawartość katalogu będziemy mogli "aktualizować" zawartość tabeli.
    Poniżej znajdziesz **tylko szablon** polecenia `create table`. Samodzielnie skonstruuj właściwe polecenia.

```
CREATE TABLE IF NOT EXISTS states (
        id INT, name STRING, abbrev STRING, area FLOAT,
        population FLOAT, capital STRING, statehood STRING)
```

    Koniecznie sprawdź czy dane znalazły się w "docelowej" tabeli

```
select sum(population), count(*) from states_ext;
```

```
0: jdbc:hive2://localhost:10000/default> select sum(population), count(*) from states_ext;
+--------------------+------+
|        _c0         | _c1  |
+--------------------+------+
| 299.3709993362427  | 52   |
+--------------------+------+
```

20. Pamiętaj, że tabela ta zawiera wszystkie wiersze ze źródłowych plików, w tym wiersze nagłówka.
    W jaki sposób moglibyśmy z tej tabeli usunąć te wiersze nagłówka?

Podczas optymalizacji zapytań ważne są aktualne statystyki. Przypomnij sobie jaka była wartość parametru, który odpowiadał za ich wyliczanie podczas wstawiania danych. Sprawdź czy dobrze to pamiętasz.

21. Sprawdź statystyki na poziomie tabel. Czy liczby wierszy zostały podliczone?

```
describe formatted beers_orc;
```

22. Sprawdź statystyki na poziomie kolumn. Czy wyliczone zostały wartości minimalne, maksymalne, liczby unikalnych wartości itp.? Jeśli nie widzisz tych statystyk, to być Twoje polecenie wyliczające statystyki nie było kompletne.

```
describe formatted beers_orc ibu;
```

    Gdyby z jakiegokolwiek powodu tych statystyk nie było, wówczas możesz skorzystać z polecenia ANALYZE TABLE, które powinno być Ci znane. Jeśli masz wątpliwości, zaglądnij na:
    `https://cwiki.apache.org/confluence/display/Hive/StatsDev`

23. Zanim zaczniemy przetwarzanie naszych danych sprawdźmy jeszcze organizację tabeli `beers_orc`.
    Uruchom ponownie polecenie wyświetlające statystyki na poziomie tabeli
    A następnie poszukaj stosownych informacji i odpowiedz na następujące pytania:
    a. Ile plików wykorzystywanych jest dla danych tabeli `beers_orc`? *Zastanów się nad tym, kiedy ta liczba plików może wzrosnąć, oraz czy gdy wzrośnie, to czy może także zmaleć?*
    b. Korzystając z przeglądarki plików HDFS dostępnej w ramach interfejsu węzła nazw znajdź katalog zawierający dane tej tabeli. Jak nazywa się ten katalog?
    c. Ile jest w nim plików? Czy jest to zgodne z liczbą uzyskaną ze statystyk?
    d. Jakiego typu są to pliki (bazowe czy zawierające deltę zmian)?
    e. Czy zawartość tabeli `beers_orc` jest skompresowana?
    f. Czy tabela została podzielona na kubełki?
    g. Czy dane zostały posortowane?
    h. Jaka biblioteka SerDe odpowiada za serializację i deserializację danych?

Zadania

BigData i SQL, to jak połączenie nowego ze starym. Nie tylko *Hive* dostarcza interfejsu SQL. Big Data jest pełne SQLa.

Tak czy inaczej możemy czuć się jak na starych śmieciach. Hive udostępnia bardzo bogaty zakres funkcjonalności SQL włącznie ze stosunkowo bogatym zestawem funkcji, w tym funkcji analitycznych. Nie bez powodu implementowane są za jego pomocą hurtownie danych oraz procesy ETL. Szczegóły:
`https://cwiki.apache.org/confluence/display/Hive/LanguageManual`

W porównaniu do asemblerowego MapReduce, czy nawet do nieco obcego *Pig Latin*, teraz możemy poczuć się dopieszczeni. Dlatego też bez dalszych wstępów bierzmy się do roboty.

Jeśli chcesz, możesz skorzystać z notatnika Zeppelin. Stosowne instrukcje znajdziesz w dodatku na końcu tego warsztatu.

Rozwiąż poniższe zadania:

24. Na początek coś prostego. Jakie 3 craftowe marki piw w USA mają największą wartość `ibu`?
    (3 piwa o największej goryczce).

25. Jaka jest średnia ilość alkoholu w craftowych markach piw w USA?

26. Jakie cztery typy/style piw charakteryzują się największą średnią ilością alkoholu?

27. W którym mieście warzy się markę piwa o największej ilości alkoholu?

28. Uporządkuj nazwy stanów pod względem malejącej największej wartości `ibu` w piwach warzonych na ich terenie. Podaj nazwy trzech pierwszych.

29. W którym stanie warzonych jest największa liczba marek piw w przeliczeniu na jednego mieszkańca?

```
select count(beer.line)/state.population as beer_density, state.name
from beers_orc beer
join breweries_orc brewery on brewery.id = beer.brewery_id
join states_ext state on brewery.state = state.abbrev
group by state.name, state.population
order by beer_density desc limit 1
```

Zatrzymajmy się na chwilę na zapytaniu rozwiązującym ostatnie zadanie.

30. Wygeneruj za pomocą polecenia `EXPLAIN` realizacji tego zapytania. Zauważasz elementy optymalizacji wdrożonej przez Hive? Np. *filter down* albo projekcja na wczesnym etapie przetwarzania?
    a. Która z tabel została rozesłana w całości (*broadcasted*) do wszystkich węzłów w celu wykonania operacji połączenia?
    b. Ile operacji mapowania zostało wykorzystanych do realizacji tego zapytania.
    c. Ile operacji redukcji było wykorzystanych w tym samym zapytaniu?
    d. Czy mechanizm podziału na kubełki został wykorzystany podczas połączenia?
    e. Zidentyfikuj operację mapowania odpowiadającą za połączenie, a następnie odszukaj odpowiadający jej węzeł w szczegółach dotyczących zadania TEZ wykorzystanego do wykonania takiego samego zapytania. Ile było użytych jednostek zadań dla tej operacji?
    f. Które klauzule w zapytaniu odpowiadały za operacje redukcji?

    Jeśli czegoś nie rozumiesz podyskutuj o tym z prowadzącym.

Zadania dla zaawansowanych (opcjonalne)

31. Jeśli znasz funkcje analityczne wykonaj zapytanie, które znajdzie trzy najbardziej gorzkie piwa produkowane przez browary w każdym ze stanów.

Do tej pory przetwarzaliśmy dane, których kolumny są prostych typów. Należy jednak pamiętać, że świat SQL w systemach Big Data, praktycznie bez wyjątku rozszerza relacyjny model i umożliwia obsługę kolumn, które są typów złożonych. To odpowiedź na wymagania świata Big Data odnoszące się do terminu *variety*.

Obsługiwane są kolumny, których typy to: rekordy, tablice asocjacyjne (mapy), czy po prostu tablice składające z dowolnej liczby wartości (prostych lub złożonych). Takie bogactwo typów wymaga ponad podstawowej znajomości SQL.

32. Pobierz plik zawierający informacje na temat krajów

```
wget http://jankiewicz.pl/bigdata/hadoop-intro/mondial.countries.json
```

Poniżej przykładowy wiersz z tego pliku (sformatowany dla czytelności)

```
{
   "_id":{
      "$oid":"581cb5a519ec2deb4ba71b3d"
   },
   "name":"France",
   "code":"F",
   "capital":"F-Île-de-France-Paris",
   "area":547030,
   "gdp":2739000,
   "inflation":0.9,
   "unemployment":10.2,
   "independence":null,
   "government":"republic",
   "population":[
      {
         "year":1946,
         "value":40502513
      },
      {
         "year":1954,
         "value":42777162
      },
. . .
      {
         "year":2006,
         "value":60825000
      },
      {
         "year":2011,
         "value":64933400
      }
   ]
}
```

33. Przekopiuj pobrany plik do systemu plików HDFS

```
!sh hadoop fs -mkdir /tmp/countries
!sh hadoop fs -copyFromLocal mondial.countries.json /tmp/countries/
```

34. A następnie zdefiniuj odpowiadającą jego strukturze tabelę.

```
CREATE EXTERNAL TABLE IF NOT EXISTS countries_ext
(
   `_id` struct<`oid`: string>,
   `name` string,
   `code` string,
   `capital` string,
   `area` int,
   `gdp` int,
   `inflation` double,
   `unemployment` double,
   `independence` string,
   `government` string,
   `population` array<struct<`year`:int, `value`:int>>
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.JsonSerDe'
LOCATION '/tmp/countries';
```

35. Na początek coś prostego. Trzy największe kraje na świecie. Tak, bardziej dla sprawdzenia poprawności definicji tabeli, niż znajomości SQL.

36. Zadziałało? Jeśli tak, to świetnie. Zatem coś trudniejszego. Podaj dla tych krajów, oprócz nazwy i wielkości, także procentowy udział tej wielkości w sumarycznej wielkości wszystkich krajów na świecie.

37. Do tej pory trzymaliśmy się "bezpiecznych" prostych typów. Czas jednak wskoczyć na głęboką wodę.
    Które kraje mają określoną liczbę ludności w roku 2011?

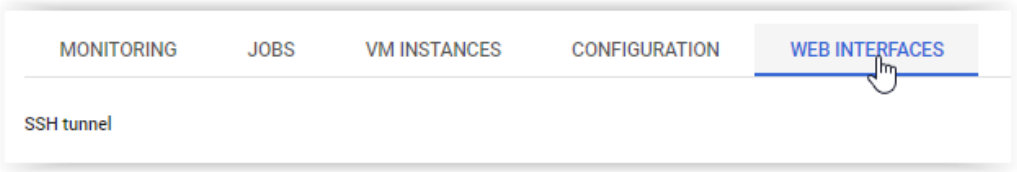
38. No to jeszcze jedno. Oblicz sumaryczną ludność wszystkich krajów świata na rok 2011. Nie wszystkie kraje mają określoną ludność w roku 2011. Jeśli jej brakuje, wykorzystaj najnowszą jaka jest dostępna, oczywiście na rok 2011.
    Zadanie można wykonać prosto odwołując się do tabeli dwukrotnie i dokonując połączenia obu zbiorów danych. Czy zdajesz sobie jednak sprawę czym to skutkuje? To co w zwykłej bazie danych na stosunkowo niedużych danych jest do zaakceptowania, to w Big Data zaakceptowane być nie może.
    Połączenie dwóch potencjalnie petabajtowych zbiorów danych? Nigdy nie idź tą drogą jeśli tylko masz alternatywę. Tu ona oczywiście jest. Odwołaj się do tabeli `countries_ext` tylko jeden raz.

Alkohol szkodzi.

Dodatek – konfiguracja Zeppelina

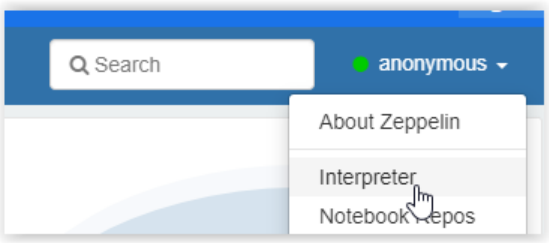
Jeśli chcesz skorzystać z graficznego edytora poleceń, a także wizualizować dane np. za pomocą wykresów, skorzystaj z poniższej instrukcji.

1. W konsoli GCP przejdź na stronie szczegółów uruchomionego klastra na zakładkę z interfejsami sieciowymi

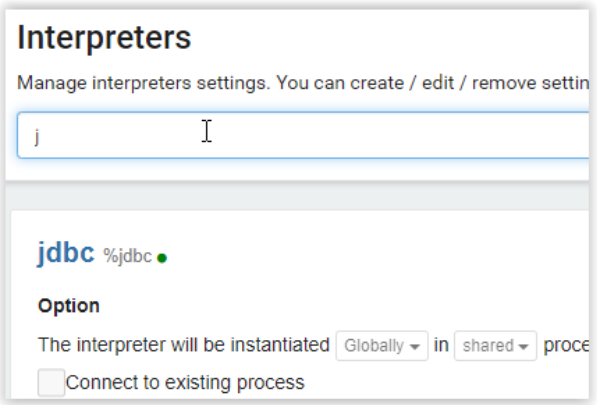


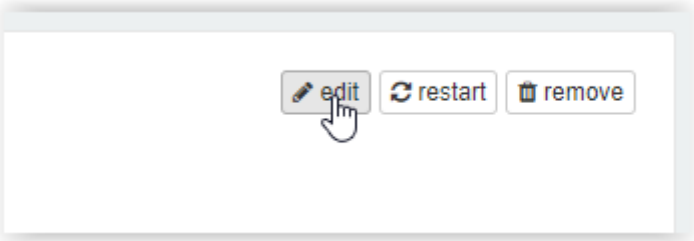
2. Wybierz link prowadzący Cię do platformy notatnika Zeppelin

3. Zanim zaczniemy tworzyć notatkę musimy skonfigurować odpowiedni interpreter. W przypadku *Hive* potrzebny interpreter to jdbc. Przejdź na stronę z konfiguracją interpreterów.

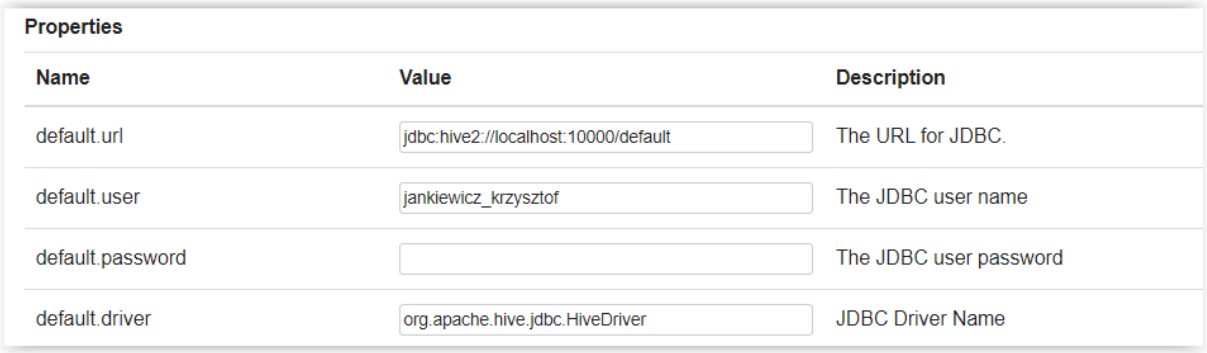


4. Wyszukaj interpreter jdbc, a następnie włącz jego edycję.





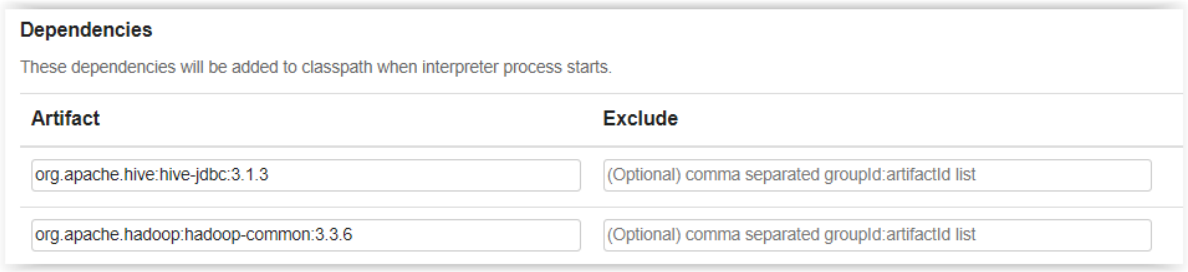
5. Zmień wartości poniższych parametrów konfiguracyjnych na podane wartości
   - `default.url – jdbc:hive2://localhost:10000/beers`
   - `default.driver – org.apache.hive.jdbc.HiveDriver`
   - `default.user – {nazwa_uzytkownika}`
   - `default.password –`



6. Na dole ustawień w zależnościach dodaj
   - `org.apache.hive:hive-jdbc:3.1.3`
   - `org.apache.hadoop:hadoop-common:3.3.6`

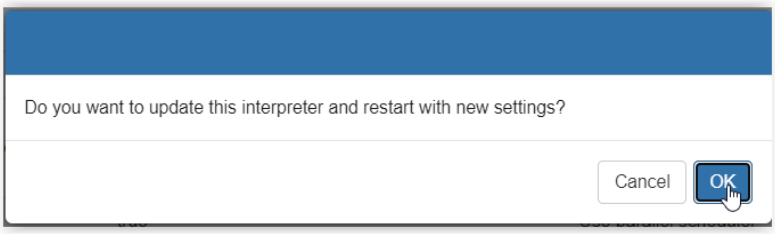
   Możesz dowiedzieć się z jakich wersji potrzebujesz za pomocą poniższych poleceń

```
hive --version
hadoop version
```

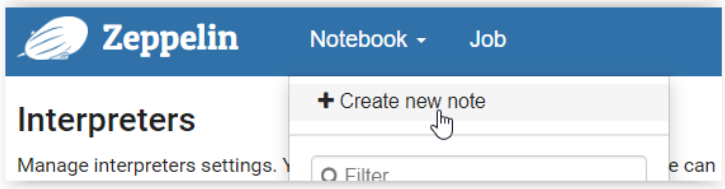


7. Zapisz zmiany za pomocą przycisku *Save*

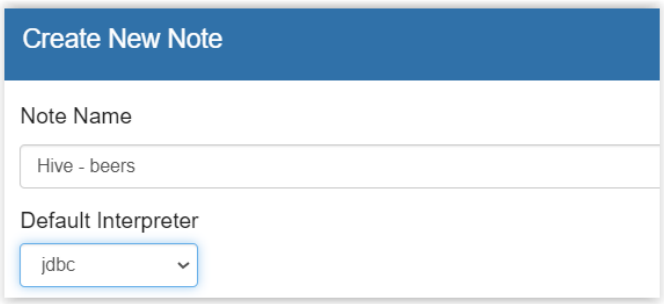
8. Potwierdź chęć modyfikacji ustawień i restartu interpretera



9. Korzystając z menu Notebook utwórz nową notatkę.

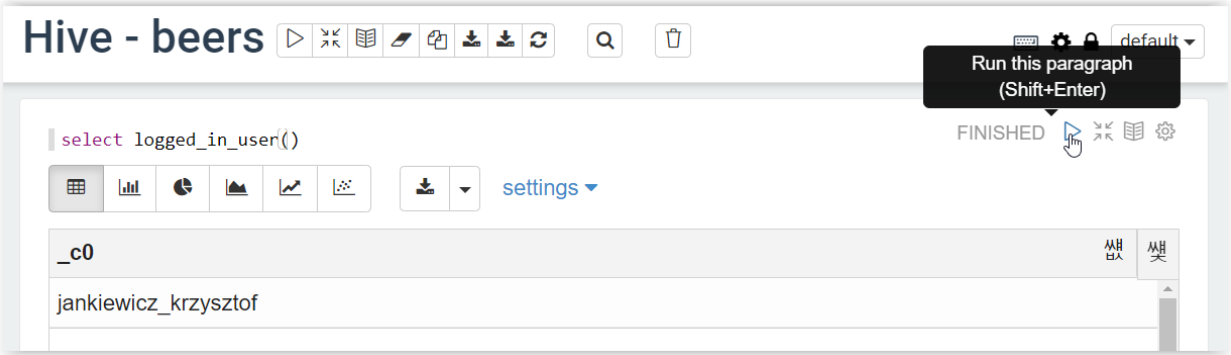


10. Nadaj tytuł notatce oraz określ jaki interpreter ma być używany jako domyślny



11. Wprowadź do pierwszego paragrafu nasze pierwsze polecenie wykonane w ramach tego warsztatu, a następnie go uruchom.

```
select logged_in_user()
```



12. Spróbuj też prezentacji wyników za pomocą wykresów. Prawda, że proste?

```
select avg(ibu) as avg_ibu, style
from beers.beers_orc
group by style
having count(*) > 100
```