1. zaloguj się na vm-cluster-node1
2. wykonaj polecenie "mysql -u root"
3. wykonaj polecenie 'create database hue;'
4. wykonaj polecenie "grant all on hue.* to 'hue'@'localhost' identified by 'hue';"
5. wejdź w konfigurację serwisu hue na cloudera managerze i kliknij w filtrach na database
6. zmień 'Hue Database Type' na 'mysql'
7. 'Hue Database Hostname' na 'localhost'
8. 'Hue Database Port' na '3306'
9. 'Hue Database Password' na 'hue'
10. zrestartuj serwis hue

Dokumentacja hive:
https://cwiki.apache.org/confluence/display/Hive/LanguageManual+DML

Beeline connection string:
!connect jdbc:hive2://vm-cluster-node1:10000 vagrant vagrant

Przykłady zapytań:

```
CREATE TABLE IF NOT EXISTS apache_logs(
        ip STRING,
        minus1 STRING,
        minus2 STRING,
        date STRING,
        request STRING,
        response STRING,
        content_lenght INT,
        referer STRING,
        useragent STRING)
    COMMENT 'apache_logs'
    ROW FORMAT DELIMITED
    FIELDS TERMINATED BY '|'
    STORED AS TEXTFILE
    LOCATION '/user/vagrant/apache_logs_table';

    SHOW CREATE TABLE apache_logs;

CREATE TABLE orc_logs LIKE apache_logs STORED AS orc;
CREATE TABLE avro_logs STORED AS avro AS select * FROM apache_logs;
```

```
CREATE external TABLE IF NOT EXISTS partitioned_logs (
                ip STRING,
                minus1 STRING,
                minus2 STRING,
                date STRING,
                request STRING,
                response STRING,
                content_length INT,
                referer STRING,
                useragent STRING)
        PARTITIONED BY (method STRING)
        ROW FORMAT DELIMITED
        FIELDS TERMINATED BY '|'
        STORED AS TEXTFILE;

SET hive.exec.dynamic.partition.mode=nonstrict;
INSERT INTO partitioned_logs partition(method)
        SELECT *, method_udf(request) AS method FROM apache_logs;

CREATE FUNCTION hello AS 'pl.isa.hadoop.HelloWorldUdf'
        USING JAR 'hdfs:///user/vagrant/hive-udfs-1.0-SNAPSHOT.jar'
```

Zadania:

1. Utwórz tabelę apache_logs. Wrzuć do niej dane z pliku apache_logs. Możesz skorzystać z polecenia 'SHOW CREATE TABLE', żeby namierzyć gdzie hive utworzył bazę na hdfsie i przekopiować plik do katalogu.

2. Utwórz tabelę typu 'external' I załaduj do niej takie same dane. Możesz skorzystać z polecenia 'insert into *tabela* select ...'.

3. Zdropuj obie tabele i zobacz co stało się z danymi na hdfsie.

4. Utwórz tabelę w formacie ORC – na końcu polecenia 'CREATE' dodaj 'STORED AS ORC'. Załaduj do niej dane z tabeli apache_logs. Spróbuj poleceniem 'hdfs dfs -cat ...' wypisać zawartość plików i zobacz, że są binarne.

5. Posumuj content_length po metodzie (PUT, GET) i zobacz, które requesty zajęły najwięcej transferu. Możesz skorzystać z metody hive'owej split(*string*, *pattern*), żeby wyciągnąć metodę z pola request.

6. Utwórz tabelę ip_name z logów załadowanych sqoopem

7. Utwórz tabelę name_useragent będącą wynikiem zjoinowania tabel ip_name i apache_logs

8. Utwórz UDF, który wyciągnie z pola request metodę i spróbuj go użyć zamiast polecenia split. Jara z udfem najprościej umieścić na hdfsie i dodać do hive'a poleceniem CREATE FUNCTION *nazwa_funkcji* AS *pakiet.NazwaKlasy* USING JAR 'hdfs://*ścieżka_na_hdfs*'

9. Utwórz tabelę partycjonowaną po metodzie i wrzuć do niej dane z tabeli apache_logs. Skorzystaj z polecenia SET hive.exec.dynamic.partition=true, żeby odblokować dynamiczne wyznaczanie partycji.

10. Utwórz tabelę partycjonowaną po log_time z eventów, które wrzucił flume