

DOMÁCÍ ÚKOL NA BI-BIG

Vypracoval Matyáš Skalický 15.12.2018

OBSAH

ÚVOD	2
DATASET.....	2
ZDROJ DATASETU:.....	2
SPUŠTĚNÍ DATABÁZOVÉHO CLUSTERU	4
<i>SPARK</i>	4
BUILD IMAGE PRO SPARK	4
SPUŠTĚNÍ MASTER A WORKER NODE	4
SPARK-SHELL.....	4
PŘIPOJENÍ SPARK-SHELL NA MASTER.....	4
<i>HDFS KONTEJNER</i>	4
IMPORT DAT DO DATABÁZOVÉHO CLUSTERU	5
AGREGACE	6
1. VYTVOŘIT NOVÝ DATASET, KTERÝ BUDE AGREGOVAT DATA Z JEDNOHO PŮVODNÍHO DATASETU	6
2. VYTVOŘIT NOVÝ DATASET, KTERÝ BUDE AGREGOVAT DATA ZE DVOU PŮVODNÍCH DATASETŮ NAJEDNOU	6
3. VYTVOŘIT NOVÝ DATASET, KTERÝ BUDE AGREGOVAT DATA ZE DVOU DATASETŮ NAJEDNOU, Z ČEHOŽ JEDEN BUDE VÝSLEDKEM PŘEDCHOZÍ AGREGACE A ULOŽIT HO ZPĚT DO DATABÁZE/NA FILE SYSTEM	6
EXPORT DAT ZE SPARK CLUSTERU	6
VYHLADÁVACÍ INDEX	7
<i>NAHRÁNÍ DAT A TVORBA INDEXU</i>	7
<i>DOTAZY NA INDEX</i>	8
FILTROVÁNÍ - VYHLEDÁNÍ VŠECH MUŽŮ:	8
TRÍDĚNÍ - SEŘADIT ZÁKAZNÍKY PODLE ÚTRATY.....	8
WILDCARD HLEDÁNÍ - VŠICHNI ZÁKAZNÍCI CO PŘIŠLI Z GOOGLE.....	8
<i>DASHBOARD</i>	9
ZÁVĚR	9

ÚVOD

Cílem tohoto úkolu je zopakovat si postupy probrané na cvičení na vlastních datech.

DATASET

Dataset obsahuje 550 000 záznamů z Black Friday z amerického obchodního řetězce. Obsahuje jak numerické, tak kategorické proměnné. Obsahuje chybějící hodnoty. Bylo dogenerován sloupec Refferal a upraveno Product_ID na numerickou hodnotu (odstraněním předpony P a úvodních nul). Úpravy datasetu jsou popsány v souboru [DATASETTTRANSFORM.IPYNB](#). Dataset byl pro potřeby úkolu rozdělen na 3 části.

ZDROJ DATASETU:

[HTTPS://WWW.KAGGLE.COM/MEHDIDAG/BLACK-FRIDAY](https://www.kaggle.com/mehdidag/black-friday)

DATASET USERS.CSV

User_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Refferal
1000001	F	0-17	10	A	2	0	www.baidu.com
1000001	F	0-17	10	A	2	0	www.samsung.com
1000001	F	0-17	10	A	2	0	www.google.com.ar
1000001	F	0-17	10	A	2	0	www.google.ca
1000002	M	55+	16	C	4+	0	www.accuweather.com

RANGEINDEX: 537577 ENTRIES, 0 TO 537576

DATA COLUMNS (TOTAL 8 COLUMNS):

USER_ID	537577 INT64 - UNIKÁTNÍ IDENTIFIKÁTOR UŽIVATELE.
GENDER	537577 OBJECT - POHLAVÍ ZÁKAZNÍKA (M/F).
AGE	537577 OBJECT - VĚKOVÁ SKUPINA ZÁKAZNÍKA.
OCCUPATION	537577 INT64 - KATEGORIE DLE TYPU ZAMĚSTNÁNÍ (MASKOVANÁ, ČÍSLO).
CITY_CATEGORY	537577 OBJECT - KATEGORIE MĚSTA (MASKOVANÁ - HODNOTY A, B, C).
STAY_IN_CURRENT_CITY_YEARS	537577 OBJECT - DÉLKA POBYTU V SOUČASNÉM MĚSTĚ V LETECH.
MARITAL_STATUS	537577 INT64 - MANŽELSKÝ STATUS (1=ŽENATÝ, 0=JINAK).
REFFERAL	537577 OBJECT - STRÁNKA, ODKUD UŽIVATEL PŘIŠEL, KDYŽ SE REGISTROVAL

MEMORY USAGE: 32.8+ MB

DATASET PRODUCTS.CSV

User_ID	Product_ID	Product_Category_1	Product_Category_2	Product_Category_3
1e+06	69042	3	nan	nan
1e+06	248942	1	6	14
1e+06	87842	12	nan	nan
1e+06	85442	12	14	nan
1e+06	285442	8	nan	nan

RANGEINDEX: 537577 ENTRIES, 0 TO 537576

DATA COLUMNS (TOTAL 5 COLUMNS):

USER_ID 537577 INT64 - UNIKÁTNÍ IDENTIFIKAČNÍ ČÍSLO UŽIVATELE.

PRODUCT_ID 537577 INT64 - UNIKÁTNÍ IDENTIFIKAČNÍ ČÍSLO PRODUKTU.

PRODUCT_CATEGORY_1 537577 FLOAT64 - PRIMÁRNÍ KATEGORIE PRODUKTU. (MASKOVANÁ, ČÍSLO)

PRODUCT_CATEGORY_2 370591 FLOAT64 - SEKUNDÁRNÍ KATEGORIE PRODUKTU. (MASKOVANÁ, ČÍSLO)

PRODUCT_CATEGORY_3 164278 FLOAT64 - TERCERNÍ KATEGORIE PRODUKTU. (MASKOVANÁ, ČÍSLO)

MEMORY USAGE: 20.5 MB

DATASET USER_PURCHASE.CSV

User_ID	Purchase
1e+06	8370
1e+06	15200
1e+06	1422
1e+06	1057
1e+06	7969

RANGEINDEX: 537577 ENTRIES, 0 TO 537576

DATA COLUMNS (TOTAL 2 COLUMNS):

USER_ID 537577 NON-NULL INT64 - UNIKÁTNÍ IDENTIFIKAČNÍ ČÍSLO UŽIVATELE.

PURCHASE 537577 NON-NULL INT64 - PŘEDPOKLÁDÁM ŽE KUMULATIVNÍ SUMA ČÁSTKY UTRACENÉ UŽIVATELEM V OBCHODĚ.

DATASET NEOBSAHUJE INFORMACE O MĚNĚ.

MEMORY USAGE: 8.2 MB

SPUŠTĚNÍ DATABÁZOVÉHO CLUSTERU

Následující postup vychází z [NÁVODU PREZENTOVANÉHO NA 5. CVIČENÍ](#). Postup vyžaduje nainstalovaný a plně funkční [DOCKER-COMPOSE](#).

SPARK

BUILD IMAGE PRO SPARK

Otevřeme terminál ve složce spark. Bude vytvořen image *spark* který bude využit dle parametrů při spuštění jak pro worker, tak pro master node.

```
DOCKER BUILD -F SPARK.DF -T SPARK .
```

SPUŠTĚNÍ MASTER A WORKER NODE

```
DOCKER-COMPOSE UP
```

SPARK-SHELL

SPUŠTĚNÍ SPARK-SHELL

Následující příkaz spustí kontejner s přístupem do bashu na popředí.

```
DOCKER RUN -IT -P 8088:8088 -P 8042:8042 -P 4041:4040 --NAME DRIVER -H DRIVER SPARK:LATEST BASH
```

PŘIPOJENÍ SPARK-SHELL NA MASTER

V běžícím bashi kontejneru se připojíme na master node spark clusteru.

```
SPARK-SHELL --MASTER SPARK://<IP ADRESA MASTERA>:7077
```

HDFS KONTEJNER

Pro spuštění kontejneru s Hadoop HDFS použijeme následující příkaz:

```
DOCKER RUN --NAME HADOOP -T -I SEQUENCEIQ/HADOOP-DOCKER /ETC/BOOTSTRAP.SH -BASH
```

Pro zjednodušení práce si přidáme do cesty image odkaz na hadoop.

```
EXPORT PATH=$PATH:/USR/LOCAL/HADOOP/BIN/
```

IMPORT DAT DO DATABÁZOVÉHO CLUSTERU

Vytvoříme složku *data* v rootu image a v rootu HDFS filesystemu.

```
MKDIR /DATA
HDFS DFS -MKDIR /DATA
```

Kopie dat do kontejneru na kterém běží HDFS.

```
DOCKER CP DATA/PRODUCTS.CSV HADOOP:/DATA/PRODUCTS.CSV
DOCKER CP DATA/USER_PURCHASE.CSV HADOOP:/DATA/USER_PURCHASE.CSV
DOCKER CP DATA/USERS.CSV HADOOP:/DATA/USERS.CSV
```

Vložení dat z filesystemu image do HDFS.

```
HDFS DFS -PUT /DATA/PRODUCTS.CSV /DATA/PRODUCTS.CSV
HDFS DFS -PUT /DATA/USER_PURCHASE.CSV /DATA/USER_PURCHASE.CSV
HDFS DFS -PUT /DATA/USERS.CSV /DATA/USERS.CSV
```

Načtení csv souboru z HDFS do sparku

```
VAL USERS = SPARK.SQLCONTEXT.READ.FORMAT("CSV").OPTION("HEADER", "TRUE").OPTION("INFERSHEMA", "TRUE").LOAD("HDFS://172.17.0.5:9000/DATA/USERS.CSV")
VAL USER_PURCHASE = SPARK.SQLCONTEXT.READ.FORMAT("CSV").OPTION("HEADER", "TRUE").OPTION("INFERSHEMA", "TRUE").LOAD("HDFS://172.17.0.5:9000/DATA/USER_PURCHASE.CSV")
VAL PRODUCTS = SPARK.SQLCONTEXT.READ.FORMAT("CSV").OPTION("HEADER", "TRUE").OPTION("INFERSHEMA", "TRUE").LOAD("HDFS://172.17.0.5:9000/DATA/PRODUCTS.CSV")
```

AGREGACE

1. VYTVOŘIT NOVÝ DATASET, KTERÝ BUDE AGREGOVAT DATA Z JEDNOHO PŮVODNÍHO DATASETU

Zjistíme, kolik maximálně uživatel utratil. Vzhledem k tomu, že předpokládám, že se jedná o kumulativní sumu tedy i kolik celkem v obchodě utratil.

```
// ZJISTÍME MAXIMÁLNÍ HODNOTY SLOUPCE PURCHASE
VAL USER_PURCHASE_MAX = USER_PURCHASE.GROUPBY("USER_ID").MAX("PURCHASE")
// ZOBRAZÍME VÝSLEDEK
USER_PURCHASE_MAX.SHOW()
```

2. VYTVOŘIT NOVÝ DATASET, KTERÝ BUDE AGREGOVAT DATA ZE DVOU PŮVODNÍCH DATASETŮ NAJEDNOU

Zjistíme, kteří zákazníci si koupili nejméně produktů

```
FROM USERS U FULL JOIN PRODUCTS P USING(USER_ID) GROUP BY USER_ID ORDER BY PRODUCT_COUNT ASC")
```

```
// ZOBRAZÍME SI VÝSLEDEK. VOLÁNÍ PROVEDE DOTAZ.
USERS_ORDER_COUNT.SHOW()
```

3. VYTVOŘIT NOVÝ DATASET, KTERÝ BUDE AGREGOVAT DATA ZE DVOU DATASETŮ NAJEDNOU, Z ČEHOŽ JEDEN BUDE VÝSLEDKEM PŘEDCHOZÍ AGREGACE A ULOŽIT HO ZPĚT DO DATABÁZE/NA FILE SYSTÉM

```
// REGISTRACE DATAFRAMU NA SQL TABULKU
USER_PURCHASE_MAX.WITHCOLUMNRENAMED("MAX(PURCHASE)",
"PURCHASE").REGISTERTABLE("USER_PURCHASE_MAX").
USERS_ORDER_COUNT.REGISTERTABLE("USERS_ORDER_COUNT")

// DOTAZ
VAL USER_AVERAGE_PURCHASE_AMOUNT = SPARK.SQLCONTEXT.SQL("SELECT USER_ID, PURCHASE / PRODUCT_COUNT AS
AVERAGE_PURCHASE FROM USERS_ORDER_COUNT U JOIN USER_PURCHASE_MAX USING (USER_ID) ORDER BY
AVERAGE_PURCHASE DESC")

// ZOBRAZÍME VÝSLEDEK
USER_AVERAGE_PURCHASE_AMOUNT.SHOW()
```

EXPORT DAT ZE SPARK CLUSTERU

Na HDFS se vytvoří složka /data/users_order_count s výsledným csv.

```
USERS_ORDER_COUNT.COALESCED(1).WRITE.FORMAT("COM.DATABRICKS.SPARK.CSV").OPTION("HEADER", "TRUE").SAVE("HDFS:
//172.17.0.5:9000/DATA/USERS_ORDER_COUNT")
```

Kopie dat z HDFS do filesystemu kontejneru na kterém běží HDFS.

```
HDFS DFS -GET /DATA/USERS_ORDER_COUNT /DATA/USERS_ORDER_COUNT
```

Kopie dat z filesystemu kontejneru do filesystemu počítače.

```
DOCKER CP HADOOP:/DATA/USERS_ORDER_COUNT .
```

VYHLEDÁVACÍ INDEX

NAHRÁNÍ DAT A TVORBA INDEXU

Použijeme kontejner s technologií Elasticsearch pro indexaci csv souboru. Zároveň spustíme i kontejner Kibana, který slouží pro vizualizaci a dotazování nad daty. Data do Elasticsearch dostaneme za pomoci docker kontejneru LogStash. **POSTUP VYCHÁZÍ ZE CVIČENÍ ČÍSLO 9.**

Jdeme do složky logstash. Kontejnery spustíme na pozadí s použitím příkazu:

```
DOCKER-COMPOSE UP -D
```

Data se načtou dle definice konfigurace v souboru *logstash.conf*:

```
INPUT {
  FILE {
    PATH => "/DATASETS/BLACKFRIDAYEDIT.CSV"
    START_POSITION => "BEGINNING"
  }
}

FILTER {
  CSV {
    SEPARATOR => ",",
    COLUMNS => ["USER_ID", "PRODUCT_ID", "GENDER", "AGE", "OCCUPATION", "CITY_CATEGORY", "STAY_IN_CURRENT_CITY_YEARS", "MARITAL_STATUS", "PRODUCT_CATEGORY_1", "PRODUCT_CATEGORY_2", "PRODUCT_CATEGORY_3", "PURCHASE", "REFERRAL"]
  }
  MUTATE {CONVERT => ["USER_ID", "INTEGER"]}
  MUTATE {CONVERT => ["PRODUCT_ID", "INTEGER"]}
  MUTATE {CONVERT => ["OCCUPATION", "INTEGER"]}
  MUTATE {CONVERT => ["PRODUCT_CATEGORY_1", "INTEGER"]}
  MUTATE {CONVERT => ["PRODUCT_CATEGORY_2", "INTEGER"]}
  MUTATE {CONVERT => ["PRODUCT_CATEGORY_3", "INTEGER"]}
  MUTATE {CONVERT => ["PURCHASE", "INTEGER"]}
}

OUTPUT {
  ELASTICSEARCH {
    HOSTS => "HTTP://ELASTICSEARCH:9200"
    INDEX => "BLACKFRIDAY"
  }
}
```

Uživatelské rozhraní Kibana je na adrese: [HTTP://127.0.0.1:5601](http://127.0.0.1:5601). Pro vytvoření indexu zvolíme *Management -> Index Patterns* a do pole "Index pattern" okopírujeme název indexu, tedy "blackfriday". Následně klikneme next a v time filter zvolíme možnost "I don't want to use Time Filter" jelikož náš dataset neobsahuje časové značky. Klikneme na Create index pattern a index je tak vytvořen.

DOTAZY NA INDEX

Dotazy v Kibaně lze provádět v záložce *Discover*. Konzole se nachází na *Dev Tools* -> *Console*.

FILTROVÁNÍ - VYHLEDÁNÍ VŠECH MUŽŮ:

Dev Tools:

```
GET /BLACKFRIDAY/DOC/_SEARCH
{
  "QUERY": {
    "MATCH": {
      "GENDER": "M"
    }
  }
}
```

Kibana: GENDER:M

TŘÍDĚNÍ - SEŘADIT ZÁKAZNÍKY PODLE ÚTRATY

Dev Tools:

```
GET /BLACKFRIDAY/DOC/_SEARCH
{
  "SORT": {
    "PURCHASE": {
      "ORDER": "DESC"
    }
  }
}
```

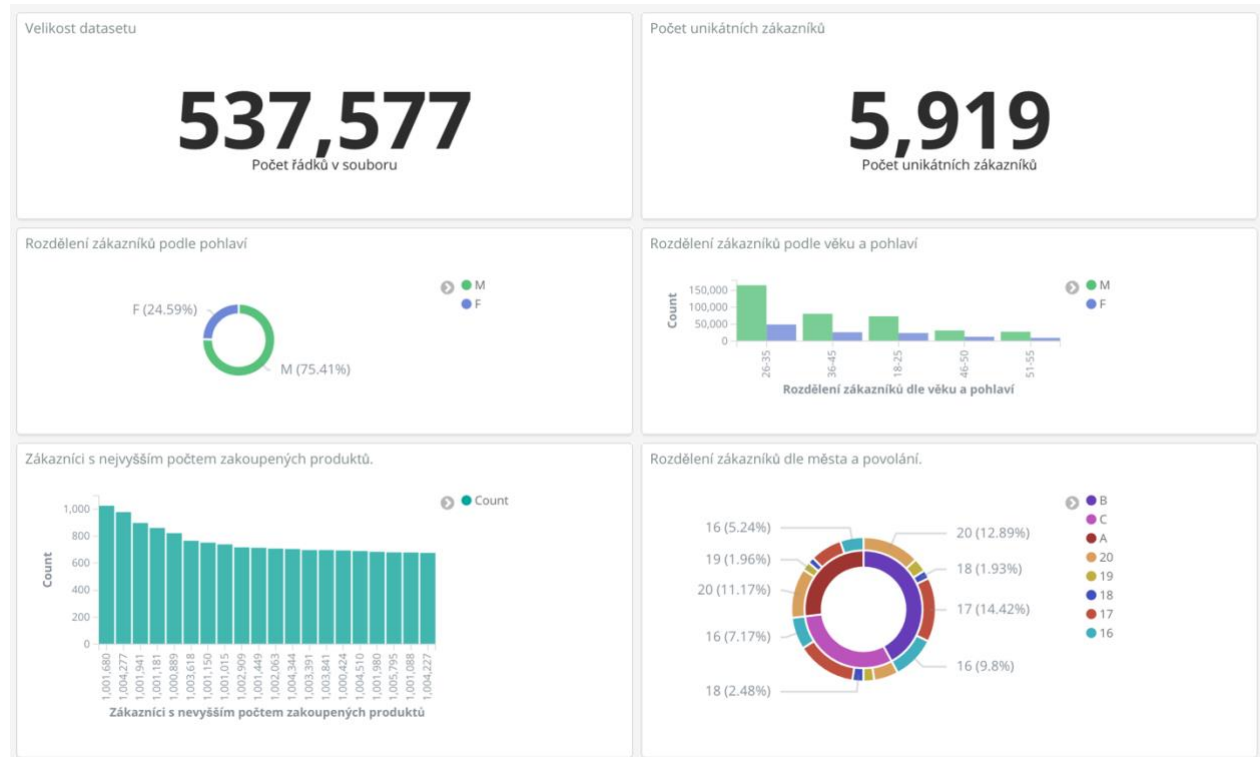
WILDCARD HLEDÁNÍ - VŠICHNI ZÁKAZNÍCI CO PŘIŠLI Z GOOGLE

Dev Tools:

```
GET /BLACKFRIDAY/DOC/_SEARCH
{
  "QUERY": {
    "WILDCARD": {
      "REFFERAL": "WWW.GOOGLE.*"
    }
  }
}
```

Kibana: REFFERAL:WWW.GOOGLE.*

DASHBOARD



Export dashboardu je uložen ve formátu JSON ve složce logstash.

ZÁVĚR

Vyzkoušel jsem si vytvořit vlastní Spark cluster s Hadoop HDFS úložištěm a následně ve spark-console udělat pár jednoduchých transformací. Dataset jsem taktéž importoval přes LogStash do ElasticSearch, vytvořil pár vyhledávacích dotazů a v Kibaně následně připravil dashboard s vizualizacemi. Vypracování úkolu mě moc nebavilo, protože jsem prováděl velmi podobné postupy jako na cvičení a v UseCasech. I tak mi tato semestrální práce zabrala větší množství času, než by mi připadalo užitečné. Mám pocit, že celá práce byla spíš než o big data o psaní dokumentace a o vymýšlení SQL dotazů.