In [ ]:
```
from pyspark.sql import SparkSession
from pyspark.sql import types as t
from pyspark.sql.functions import col
from pyspark.sql.functions import coalesce
```

In [ ]:
```
spark = SparkSession.builder.appName('lect_13_home_task').getOrCreate()
```

```
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLo
gLevel(newLevel).
22/10/22 17:23:55 WARN NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
```

Домашнє завдання на тему Spark SQL

Задачі з домашнього завдання на SQL потрібно розв'язати за допомогою Spark SQL DataFrame API.

Дампи таблиць знаходяться в папці `data`. Можете створювати стільки нових клітинок, скільки вам необхідно.

Розв'язок кожної задачі має бути відображений в самому файлі (використати метод `.show()`)

Увага! Використовувати мову запитів SQL безпосередньо забороняється, потрібно використовувати виключно DataFrame API!

1. Вивести кількість фільмів в кожній категорії. Результат відсортувати за спаданням.

In [ ]:
```
df = spark.read.options(header='True', inferSchema='True').csv("data/film_ca
df.groupBy('category_id').count().sort('count', ascending=False).show()
```

```
+-----------+-----+
|category_id|count|
+-----------+-----+
|         15|   74|
|          9|   73|
|          8|   69|
|          6|   68|
|          2|   66|
|          1|   64|
|         13|   63|
|          7|   62|
|         10|   61|
|         14|   61|
|          3|   60|
|          5|   58|
|         16|   57|
|          4|   57|
|         11|   56|
|         12|   51|
+-----------+-----+
```

2. Вивести 10 акторів, чиї фільми брали на прокат найбільше. Результат відсортувати за спаданням.

In [ ]:
```
actor_film = spark.read.options(header='True', inferSchema='True').csv("data
rental = spark.read.options(header='True', inferSchema='True').csv("data/ren
inventory = spark.read.options(header='True', inferSchema='True').csv("data/

top_movie = rental.join(inventory, on='inventory_id').groupBy('film_id').cou
actor_film.join(top_movie, on='film_id')\
    .sort('count', ascending=False)\
    .select('actor_id')\
    .limit(10)\
    .sort('actor_id', ascending=False)\
    .show()
```

```
+--------+
|actor_id|
+--------+
|     195|
|     194|
|     193|
|      92|
|      89|
|      51|
|      42|
|      35|
|      32|
|      26|
+--------+
```

3. Вивести категорія фільмів, на яку було витрачено найбільше грошей в прокаті

In [ ]:
```
rental = spark.read.options(header='True', inferSchema='True').csv("data/ren
inventory = spark.read.options(header='True', inferSchema='True').csv("data/
film_category = spark.read.options(header='True', inferSchema='True').csv("d
categories = spark.read.options(header='True', inferSchema='True').csv("data
payment = spark.read.options(header='True', inferSchema='True').csv("data/pa

sales = rental.join(payment, on='rental_id')\
    .groupBy('inventory_id')\
    .sum('amount')\
    .withColumnRenamed("sum(amount)", "income")

sales_by_film_category = sales.join(inventory, on='inventory_id')\
    .join(film_category, on='film_id')\
    .groupby('category_id')\
    .sum('income')\
    .withColumnRenamed("sum(income)", "income")

sales_by_film_category.join(categories, on='category_id')\
    .sort('income', ascending=False)\
    .select('name')\
    .limit(1)\
    .show()
```

```
+------+
|  name|
+------+
|Sports|
+------+
```

4. Вивести назви фільмів, яких немає в inventory. Запит має бути без оператора IN

In [ ]:
```
film = spark.read.options(header='True', inferSchema='True').csv("data/film.
inventory = spark.read.options(header='True', inferSchema='True').csv("data/

film.join(inventory, on='film_id', how='left')\
    .filter("inventory_id is null")\
    .select('title')\
    .show()
```

```
+--------------------+
|               title|
+--------------------+
|      ALICE FANTASIA|
|         APOLLO TEEN|
|      ARGONAUTS TOWN|
|       ARK RIDGEMONT|
|ARSENIC INDEPENDENCE|
|   BOONDOCK BALLROOM|
|       BUTCH PANTHER|
|       CATCH AMISTAD|
| CHINATOWN GLADIATOR|
|      CHOCOLATE DUCK|
|COMMANDMENTS EXPRESS|
|    CROSSING DIVORCE|
|     CROWDS TELEMARK|
|    CRYSTAL BREAKING|
|          DAZED PUNK|
|DELIVERANCE MULHO...|
|   FIREHOUSE VIETNAM|
|       FLOATS GARDEN|
|FRANKENSTEIN STRA...|
|  GLADIATOR WESTWARD|
+--------------------+
only showing top 20 rows
```

5. Вивести топ 3 актори, які найбільше з'являлись в категорії фільмів "Children"

In [ ]:
```
film_actor = spark.read.options(header='True', inferSchema='True').csv("data
film_category = spark.read.options(header='True', inferSchema='True').csv("d
categories = spark.read.options(header='True', inferSchema='True').csv("data

child_cat = film_actor.join(film_category, on='film_id')\
    .join(categories, on='category_id')\
    .filter("name == 'Children'")\
    .groupby('actor_id')\
    .count()\
    .sort('count', ascending=False)\
    .limit(3)\
    .select('actor_id').show()
```

```
+--------+
|actor_id|
+--------+
|      17|
|      80|
|     140|
+--------+
```

6. Вивести міста з кількістю активних та неактивних клієнтів (в активних customer.active = 1). Результат відсортувати за кількістю неактивних клієнтів за спаданням.

In [ ]:
```
customer = spark.read.options(header='True', inferSchema='True').csv("data/c
address = spark.read.options(header='True', inferSchema='True').csv("data/ad
city = spark.read.options(header='True', inferSchema='True').csv("data/city.

customer_city_list = customer.join(address, on='address_id')\
    .join(city, on='city_id')\
    .select(['city_id', 'active'])

city_active = customer_city_list.filter("active == 1")\
    .groupby('city_id')\
    .count()\
    .withColumnRenamed("count", "active")

city_inactive = customer_city_list.filter("active == 0")\
    .groupby('city_id')\
    .count()\
    .withColumnRenamed("count", "inactive")

city_active.join(city_inactive, on='city_id', how='full').na.fill(0).sort('i
```

```
+-------+------+--------+
|city_id|active|inactive|
+-------+------+--------+
|    111|     0|       1|
|    241|     1|       1|
|    259|     0|       1|
|    125|     0|       1|
|     24|     0|       1|
|    407|     0|       1|
|    495|     0|       1|
|    512|     0|       1|
|    554|     0|       1|
|    577|     0|       1|
|    578|     0|       1|
|    281|     0|       1|
|    283|     0|       1|
|     57|     0|       1|
|    356|     0|       1|
|    139|     0|       1|
|      1|     1|       0|
|      2|     1|       0|
|      3|     1|       0|
|      4|     1|       0|
+-------+------+--------+
only showing top 20 rows
```