**Nội dung thực hành**

[***I.***](#_heading=h.30j0zll) ***Thực hành Spark core 1***

[**Bài 1: Nạp (load) và thao tác dữ liệu sử dụng Spark Shell 2**](#_heading=h.1fob9te)

[Nội dung bài 2](#_heading=h.3znysh7)

[Đáp án: 6](#_heading=h.2et92p0)

[**Bài 2: Làm việc với Pair RDD 6**](#_heading=h.tyjcwt)

[Nội dung bài 6](#_heading=h.3dy6vkm)

[Đáp án câu 4 đến 8 8](#_heading=h.1t3h5sf)

[Đáp án câu 9 đến 11 9](#_heading=h.4d34og8)

[**Bài 3: JOIN Pair RDD 10**](#_heading=h.2s8eyo1)

[Nội dung bài 10](#_heading=h.17dp8vu)

[Đáp án: 11](#_heading=h.3rdcrjn)

[***II.***](#_heading=h.26in1rg) ***Thực hành SparkSQL 12***

[**Bài 1: Load / Save data. 12**](#_heading=h.lnxbz9)

[**Bài 2: Sử dụng SQL Queries 14**](#_heading=h.35nkun2)

[**Bài 3: JOIN sử dụng SQL Queries 16**](#_heading=h.1ksv4uv)

[***III.***](#_heading=h.44sinio) ***Spark MLLib 19***

[**Bài 1: Spam detection 19**](#_heading=h.2jxsxqh)

[**Bài 2: Movie Recommendation 20**](#_heading=h.z337ya)

# **Thực hành Spark core**

Chúng ta sẽ thực hành trên file .csv chứa dữ liệu về các phiên đấu giá trực tuyến. Mỗi phiên đấu giá (auction) sẽ có nhiều lần ngã giá cho các sản phẩm gọi là bid. File auctiondata.csv tập hợp của nhiều dòng dữ liệu. Mỗi dòng sẽ là thông tin về bid. Thông tin bid bao gồm

| **Cột** | **Kiểu dữ liệu** | **Mô tả** |
| --- | --- | --- |
| aucid | String | Auction ID. Id của 1phiên đấu giá |
| bid | Float |  |
| bidtime | Float | Thời gian bid tính từ lúc bắt đầu phiên đấu giá |
| bidder | String | Id của người tham giá đấu giá |
| bidrate | Int | Giá người tham giá đấu giá đưa ra |
| openbid | Float | Giá khởi điểm |
| price | Float | Giá chốt |
| itemtype | String | Loại mặt hàng |
| dtl | String | Days to live. Thời gian hiệu lực của phiên đấu giá. |

## **Bài 1: Nạp (load) và thao tác dữ liệu sử dụng Spark Shell**

### Nội dung bài

1. Trên màn hình Desktop của máy ảo bấm vào biểu tượng “Spinup”



1. Bật Terminal window (hoặc từ ssh) rồi chạy Spark shell



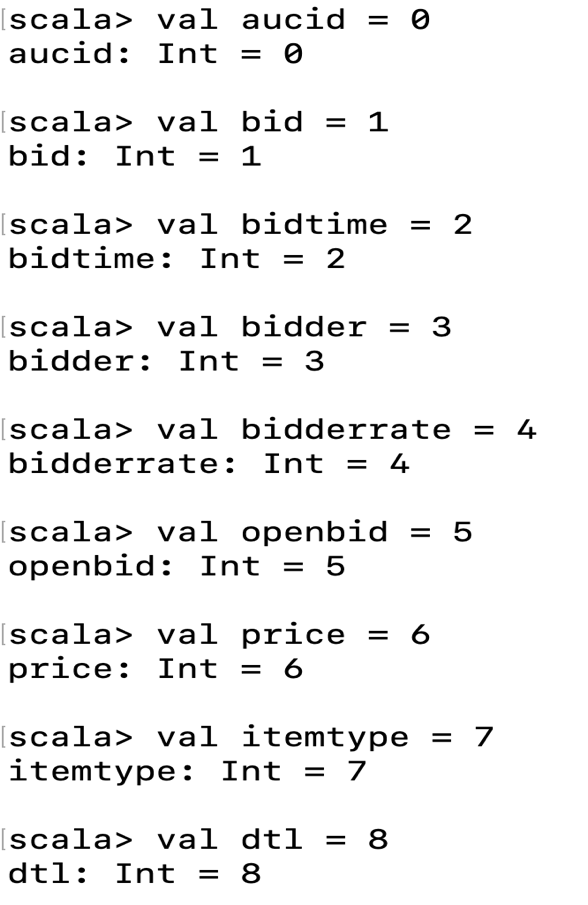
Màn hình sẽ chạy như dưới đây.



1. Dữ liệu trong file csv gồm 9 cột, có index bắt đầu từ 0 đến 8. Index 0 là aucid, index 1 là bid….về sau, khi tao tác với dữ liệu, chúng ta có thể truy xuất giá trị của aucid bằng cách chỉ định index là 0, bid là 1…Tuy nhiên, làm như vậy code sẽ khó đọc. Vì thế chúng ta định nghĩa 1 số biến constant và sẽ sử dụng những constant này.



Viết trong Spark shell, những lệnh trên sẽ như ảnh dưới đây



1. Khi chúng ta chạy Spark shell, tự động Spark context đã được khởi động. Load dữ liệu từ file *auctiondata.csv* vào RDD có tên là *auctionRDD,* bởi vì các trường của file csv được cách nhau bởi dấu phẩy, nên khi load chúng ta sẽ dùng hàm *map* để split data luôn. File *auctiondata.csv* nằm ở thư mục */bigdata/spark/data/*



1. Để đảm bảo chúng ta làm bước 4 đúng. Bạn chạy lệnh sau để lấy ra phần tử đầu tiên của auctionRDD



Nếu kết quả như dưới thì có nghĩa là bước 4 đã ok. Trường hợp báo lỗi, bạn cần xem lại mình gõ đúng hay chưa.

Array[String] = Array(8213034705, 95, 2.927373, jake7870, 0, 95, 117.5, xbox, 3)

*Từ bước 6 các bạn tự dùng những hàm thích hợp của Spark để thỏa mãn yêu cầu đưa ra. Nếu gặp khó khăn có thể tham khảo đáp án ở cuối bài.*

1. Bạn hãy lấy ra 5 dòng đầu tiên của auctionRDD



1. Tính tổng số bids. Kết quả cần là: 10654



1. Tổng số các mặt hàng (item) không trùng nhau (distinct) đã được đấu giá? Kết quả cần là: 627



1. Tổng số các loại mặt hàng (item type) đã được đấu giá? Kết quả cần là 3



1. Tính tổng số lượng bid cho từng loại mặt hàng (item type)? Kết quả cần là:

*(palm, 5917), (cartier, 1953), (xbox, 2784)*



Tiếp theo chúng ta sẽ tính max, min và average số lượng bid cho tất cả các loại mặt hàng đã được đấu giá.

1. Tạo RDD tên là *bidsAuctionRDD* chứa tổng số bid cho từng phiên đấu giá.



1. Trong các loại mặt hàng, mặt hàng nào có nhiều lần bid nhất? (gợi ý, bạn nên sử dụng hàm *Math.max*, và cần import thư viện *import java.lang.Math*). Kết quả cần là 75



1. Trong các loại mặt hàng, mặt hàng nào có ít lần bid nhất? Kết quả cần là 1



1. Tính số lượng bid trung bình. Kết quả cần là 16



### Đáp án:

1. auctionRDD.take(5)
2. val totbids = auctionRDD.count()
3. val totitems = auctionRDD.map(\_(aucid)).distinct.count()
4. val itemtypes = auctionRDD.map(\_(itemtype)).distinct.count()
5. val bids\_itemtype = auctionRDD.map(x=>x(itemtype,1)).reduceByKey((x,y) => (x+y)).collect()
6. val bids\_auctionRDD = get.map(x=>(x(aucid),1).reduceByKey((x,y) => x+y)
7. val maxbids = bids\_auctionRDD.map(x=>x.\_2).reduce((x,y)=>Math.max(x,y))
8. val minbids = bids\_auctionRDD.map(x=>x.\_2).reduce((x,y)=>Math.max(x,y))
9. val avgbids = totbids/totitems

## **Bài 2: Làm việc với Pair RDD**

### Nội dung bài

Trong bài tập này chúng ta làm việc với file csv dữ liệu sở cảnh sát San Francisco về các vụ việc xảy ra (tai nạn, tội phạm…) ở San Francisco gọi là SFPD (San Francisco Police Department), chúng ta sẽ tạo pair RDD và dùng các operation của pair RDD để thao tác dữ liệu. Dữ liệu tai nạn gồm các trường thông tin

| **Cột** | **Kiểu dữ liệu** | **Ví dụ** |
| --- | --- | --- |
| IncidentNum | Mã vụ việc | 150561637 |
| Category | Loại vụ việc | NON-CRIMINAL |
| Descript | Mô tả | FOUND\_PROPERTY |
| Date | Ngày xảy ra vụ việc | 6/28/15 |
| Time | Thời gian xảy ra | 23:50 |
| PdDistrict | Phòng cảnh sát tiếp nhận vụ việc | TARAVAL |
| Resolution | Giải pháp | NONE |
| Address | Địa chỉ xảy ra vụ việc | 1300\_Block\_of\_LA\_PLAYA\_ST |
| X | Tọa độ xảy ra vụ việc | -122.5091348 |
| Y | Tọa độ xảy ra vụ việc | 37.76119777 |
| PdId | Mã phòng cảnh sát | 15056163704013 |

1. Trên màn hình Desktop của máy ảo bấm vào biểu tượng “Spinup”
2. Bật Terminal window (hoặc từ ssh) rồi chạy Spark shell.



1. Dữ liệu trong file csv gồm 12 cột, có index bắt đầu từ 0 đến 11. Index 0 là IncidentNum, index 1 là Category….Về sau, khi tao tác với dữ liệu, chúng ta có thể truy xuất giá trị của IncidentNum bằng cách chỉ định index là 0, Category là 1…Tuy nhiên, làm như vậy code sẽ khó đọc. Vì thế chúng ta định nghĩa 1 số biến constant và sẽ sử dụng những constant này.



1. Load dữ liệu từ file *sfpd.csv* vào RDD có tên là *sfpdRDD,* bởi vì các trường của file csv được cách nhau bởi dấu phẩy, nên khi load chúng ta sẽ dùng hàm *map* để split data luôn. File *sfpd.csv* nằm ở thư mục */bigdata/spark/data/*



***Từ câu 4 đến câu 8 bạn hãy viết lệnh để thỏa mãn yêu cầu. Trường hợp khó khăn bạn có thể tham khảo đáp án ngay phía dưới***

1. Lấy ra bản ghi đầu tiên của sfpdRDD



1. Lấy ra 5 bản ghi đầu tiên của sfpdRDD



1. Tổng số vụ việc (incident) đã xảy ra. Kết quả cần là: 383775



1. Số loại giải pháp (resolution) không trùng lặp (distinct): 17



1. Danh sách các phòng cảnh sát tiếp nhận sự vụ (PdDisticts). Kết quả INGLESIDE, SOUTHERN, PARK, NORTHERN, MISSION, RICHMOND, TENDERLOIN, BAYVIEW, TARAVAL, CENTRAL.



### Đáp án câu 4 đến 8

1. sfpdRDD.first()
2. sfpdRDD.take(5)
3. val totalIncidents = sfpdRDD.count()
4. val totalResolution = sfpdRDD.map(inc => inc(Resolution)).distinct.count
5. val districts = sfpdRDD.map(inc => inc(PdDistrict)).distinct

***Từ câu 9 chúng ta sẽ tạo pair RDD và sử dụng operation của pair RDD***

1. Hãy tìm ra 5 quận (district) có nhiều sự việc (incidents) xảy ra nhất? Chúng ta thấy yêu cầu này khá giống với ví dụ word count (đếm từ) chỉ khác là sau khi đếm xong chúng ta cần sắp xếp số lượng sự vụ giảm dần và lấy ra 5 quận. Như vậy chúng ta cần làm các bước
2. Sử dụng hàm *map* để tạo pair RDD với key là quận (PdDistrict) và value là 1. Có dạng [(PdDistrict, 1)]
3. Sử dụng hàm *reduceByKey((a,b) => a + b)* để tính số lượng sự vụ cho từng quận. Bước này ta sẽ được kết quả là [(PdDistrict, count)]
4. Sử dụng hàm *map* để đảo ngược pair RDD trên thành [(count, PdDistrict)]
5. Sử dụng *sortByKey(false)* trên [(count, PdDistrct)] để sắp xếp số lượng giảm dần.
6. Sử dụng *take(5)* để lấy ra 5 quận có số sự vụ nhiều nhất.
7. Tìm 5 địa chỉ (Address) có số vụ việc xảy ra nhiều nhất? (các bước làm giống câu 10)
8. Lấy ra 3 loại vụ việc (category) hay xảy ra nhất? (các bước làm giống câu 10)

### Đáp án câu 9 đến 11

1. Các bước như sau
2. val pairRDD = sfpdRDD.map(incident => (incident(PdDistrict),1))
3. val countEachDistrict = pairRDD.reduceByKey((x,y) => x + y)
4. val revCountEachDistrict = countEachDistrict.map(item => (item.\_2, item.\_1))
5. val sortedCountEachDistrict = revCountEachDistrict.sortByKey(false)
6. val top5District = sortedCountEachDistrict.take(5)

valtop5District.collect()

Chúng ta có thể viết gộp lại như sau:

val top5District = sfpdRDD.map(incident => (incident(PdDistrict),1)).reduceByKey((x,y) => x + y) .map(item => (item.\_2, item.\_1)) .sortByKey(false) .take(5)

top5District.collect()

1. Chúng ta có thể viết gộp như sau:

val top5Adds = sfpdRDD.map(incident => (incident(Address),1)).reduceByKey((x,y) => x + y) .map(item => (item.\_2, item.\_1)) .sortByKey(false) .take(5)

top5Adds.collect()

1. Chúng ta có thể viết gộp như sau:

val top3Cat = sfpdRDD.map(incident => (incident(Category),1)).reduceByKey((x,y) => x + y) .map(item => (item.\_2, item.\_1)) .sortByKey(false) .take(3)

top3Cat.collect()

## **Bài 3: JOIN Pair RDD**

### Nội dung bài

Chúng ta có 2 file *J\_AddCat.csv* là *J\_AddDist*. File *J\_AddCat.csv* chứa 2 trường là: loại vụ việc (*Category*) và nơi xảy ra vụ việc (*Address*) còn file *J\_AddDist* chứa 2 trường: Phòng cảnh sát tiếp nhận vụ việc (*PdDistrict*) và nơi xảy ra vụ việc (*Address*). Dữ liệu như sau:

1. Load 2 file dữ liệu trên và tạo thành 2 pairRDDs với trường address là key.



1. Liệt kê thông tin như: loại sự việc (Category), phòng cảnh sát tiếp nhận sự việc(PdDistrict) cho từng địa chỉ. Đếm số lượng bản ghi và hiển thị kết quả ra màn hình



1. Liệt kê thông tin như: loại sự việc (Category), phòng cảnh sát tiếp nhận sự việc(PdDistrict) cho từng địa chỉ cho dù địa chỉ đó có PdDistrict hay không?



### Đáp án:

1. 





# **Thực hành SparkSQL**

## **Bài 1: Load / Save data.**

1. Trên màn hình Desktop của máy ảo bấm vào biểu tượng “Spinup”
2. Bật Terminal window và chạy Spark shell.



1. Tạo carDF từ file /bigdata/spark/data/car.json



1. Hiện ra 3 bản ghi của carDF



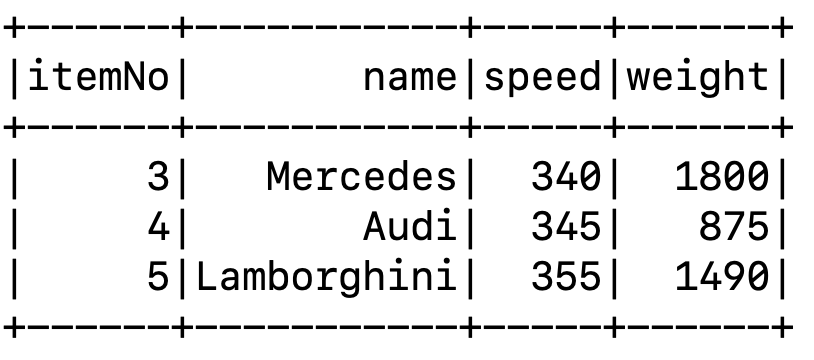
Kết quả:

res3: Array[org.apache.spark.sql.Row] = Array([1,Ferrari,259,800], [2,Jaguar,274,998], [3,Mercedes,340,1800])

1. Tạo fastCarDF chứa các xe có tốc độ lớn hơn 300
2. Hiện dữ liệu của fastCarDF



Kết quả



1. Tạo ra Data Frame projectCar chỉ chứa 2 cột là name và speed, sau đó hiện ra màn hình.



1. Trong các xe chạy tốc độ cao của fastCarDF, hãy tính tốc độ chậm nhất, tốc độ nhanh nhất và tốc độ trung bình rồi hiện ra màn hình.



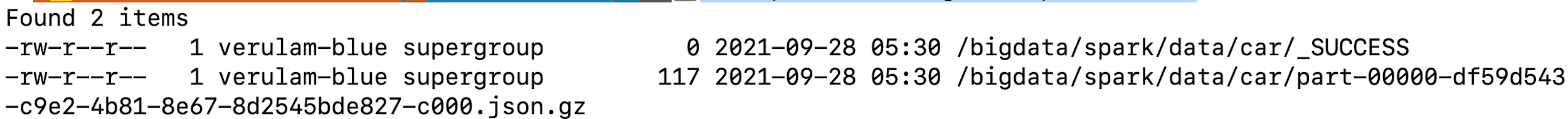
1. Lưu danh sách fastCarDF dưới dạng json và dùng nén gzip vào thư mục /bigdata/spark/data/car



1. Mở Terminal mới rồi kiểm tra xem câu lệnh 9 đã lưu được data trên hdfs hay chưa



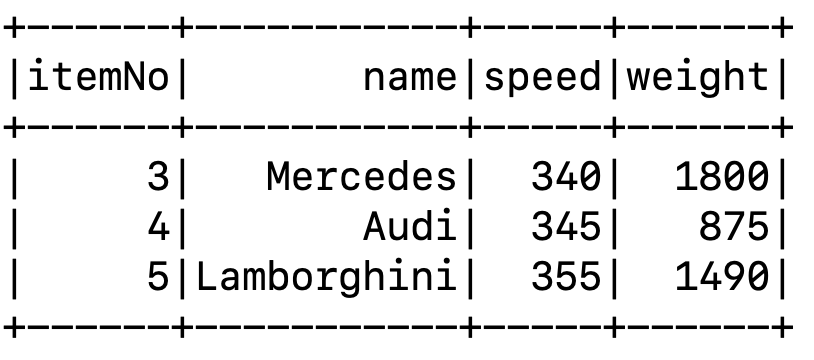
Kết quả cần như dưới



1. Đọc file json nén dạng gzip trong thư mục /bigdata/spark/data/car rồi show kết quả ra màn hình



Kết quả cần như dưới:



## **Bài 2: Sử dụng SQL Queries**

Trong bài tập này chúng ta làm việc với file csv dữ liệu sở cảnh sát San Francisco về các vụ việc xảy ra (tai nạn, tội phạm…) ở San Francisco gọi là SFPD (San Francisco Police Department). Dữ liệu tai nạn gồm các trường thông tin

| **STT** | **Cột** | **Kiểu dữ liệu** | **Ví dụ** |
| --- | --- | --- | --- |
| 0 | IncidentNum | Mã vụ việc | 150561637 |
| 1 | Category | Loại vụ việc | NON-CRIMINAL |
| 2 | Descript | Mô tả | FOUND\_PROPERTY |
| 3 | DayOfWeek | Ngày trong tuần xảy ra tai nạn | Sunday |
| 4 | Date | Ngày xảy ra vụ việc | 6/28/15 |
| 5 | Time | Thời gian xảy ra | 23:50 |
| 6 | PdDistrict | Phòng cảnh sát tiếp nhận vụ việc | TARAVAL |
| 7 | Resolution | Giải pháp | NONE |
| 8 | Address | Địa chỉ xảy ra vụ việc | 1300\_Block\_of\_LA\_PLAYA\_ST |
| 9 | X | Tọa độ xảy ra vụ việc | -122.5091348 |
| 10 | Y | Tọa độ xảy ra vụ việc | 37.76119777 |
| 11 | PdId | Mã phòng cảnh sát | 15056163704013 |

1. Trên màn hình Desktop của máy ảo bấm vào biểu tượng “Spinup”
2. Bật Terminal window và chạy Spark shell.



1. Tạo *sfpdDF* data frame từ file */bigdata/spark/data/sfpd.csv*



Kết quả cần là:

sfpdDF: org.apache.spark.sql.DataFrame = [\_c0: string, \_c1: string ... 10 more fields]

1. Tạo view có tên là sfpd từ sfpdDF data frame.



1. Tổng số vụ việc (incident) đã xảy ra. Kết quả cần là: 383775



1. Số loại giải pháp (resolution) không trùng lặp (distinct). Kết quả là: 17. Chúng ta thấy cột resolution có STT là 7 do vậy:



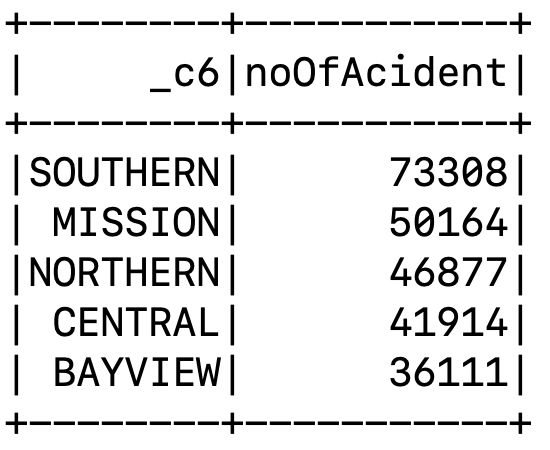
1. Danh sách các phòng cảnh sát tiếp nhận sự vụ (PdDisticts). Kết quả INGLESIDE, SOUTHERN, PARK, NORTHERN, MISSION, RICHMOND, TENDERLOIN, BAYVIEW, TARAVAL, CENTRAL. Ta thấy PdDistrict là cột có STT là 6



1. Hãy tìm ra 5 quận (district) có nhiều sự việc xảy ra nhất?

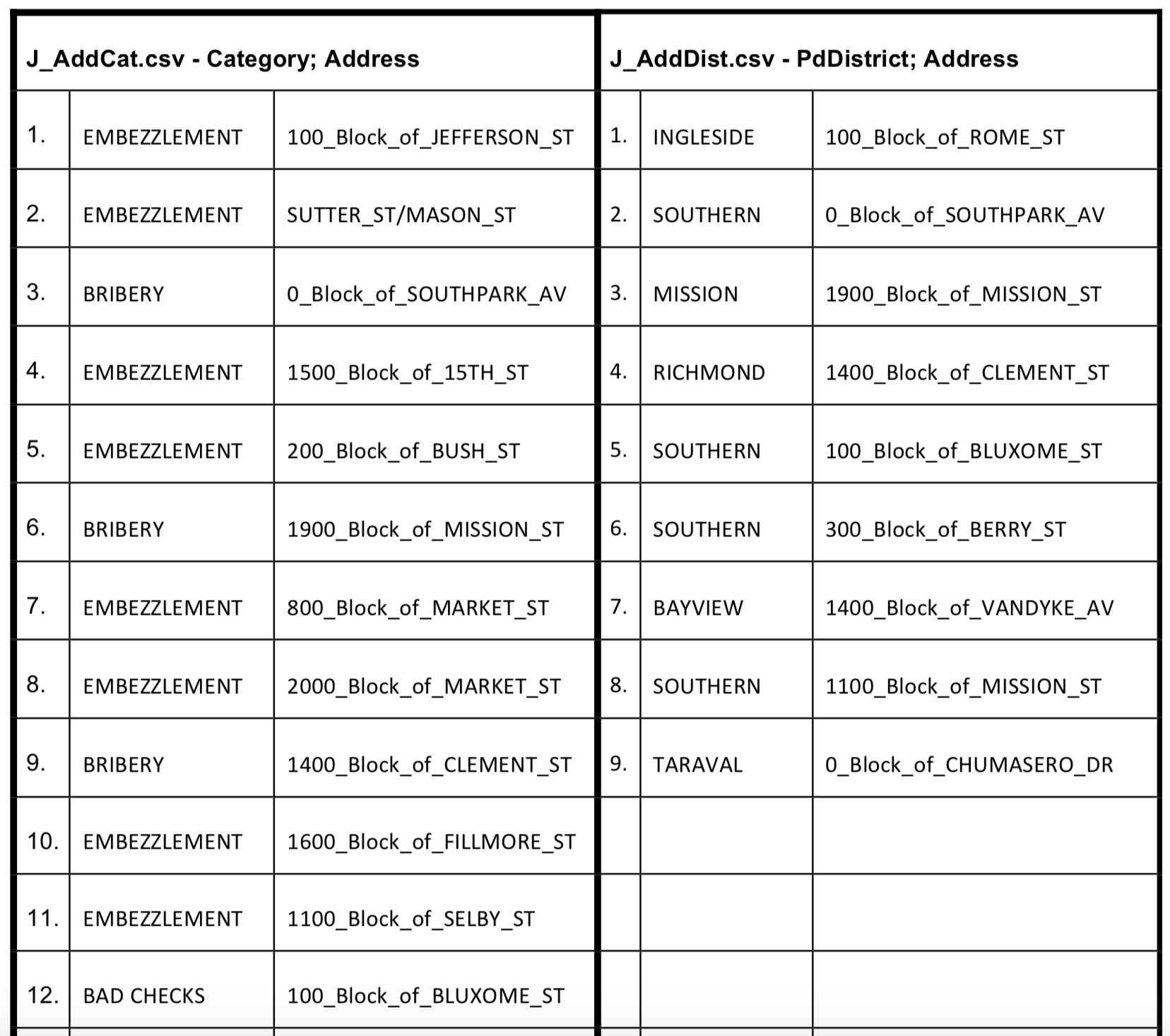


Kết quả sẽ là:



## **Bài 3: JOIN sử dụng SQL Queries**

Chúng ta có 2 file *J\_AddCat.csv* và *J\_AddDist*. File *J\_AddCat.csv* chứa 2 trường là: loại vụ việc (*Category*) và nơi xảy ra vụ việc (*Address*) còn file *J\_AddDist* chứa 2 trường: Phòng cảnh sát tiếp nhận vụ việc (*PdDistrict*) và nơi xảy ra vụ việc (*Address*). Dữ liệu như sau:



1. Trên màn hình Desktop của máy ảo bấm vào biểu tượng “Spinup”
2. Bật Terminal window và chạy Spark shell.



### 

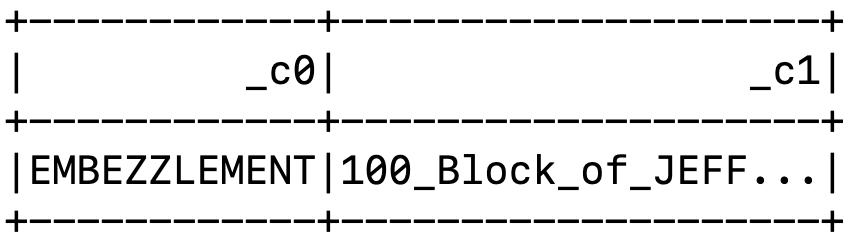
1. Tạo ra view tên *addCat* từ file */bigdata/spark/data/J\_AddCat.csv*



1. Đảm view addCat có data bằng lệnh sau:



Kết quả như sau:



1. Tạo ra view tên *addDist* từ file */bigdata/spark/data/J\_AddDist.csv*



1. Đảm view addDistcó data bằng lệnh sau:



Đảm bảo có kết quả trả ra.

1. Liệt kê thông tin như: loại sự việc (Category), phòng cảnh sát tiếp nhận sự việc(PdDistrict) cho từng địa chỉ.



1. Liệt kê thông tin như: loại sự việc (Category), phòng cảnh sát tiếp nhận sự việc(PdDistrict) cho từng địa chỉ cho dù địa chỉ đó có PdDistrict hay không?



# **Spark MLLib**

## **Bài 1: Spam detection**

Bài tập này chúng ta sẽ sử dụng thuật toán classification để phát hiện 1 email có phải spam hay không

1. Trên màn hình Desktop của máy ảo bấm vào biểu tượng “Spinup”
2. Bật Terminal window và chạy Spark shell.



1. Trong spark shell ta import các thư viện cần thiết
2. Load data spam và không phải spam vào 2 RDDs.



1. Ta sử dụng HashingTF để hashing các từ trong email từ kiểu chữ thành kiểu số, ta lấy số lượng feature là 10000



1. Spark MLLib có 1 class là LabeledPoint cho phép chúng ta label dữ liệu, với những email là spam ta gọi là positiveExamples có label là 1, với những email bình thường sẽ là negativeExamples và có label là 0



1. Tập dữ liệu training sẽ bao gồm cả những spam email và không phải spam, do vậy ta sẽ union cả 2 RDD là positiveExamples và negativeExamples.



1. Xây dựng mô hình sử dụng LogisticRegressionWithSGD



1. Ta thử test mô hình vừa xây với 1 số dữ liệu mẫu



## **Bài 2: Movie Recommendation**

Bài tập này giúp chúng ta xây dựng 1 model để recommend ra các bộ phim cho 1 user.

1. Trên màn hình Desktop của máy ảo bấm vào biểu tượng “Spinup”
2. Bật Terminal window và chạy Spark shell.



1. Import thư viện Spark MLLib



1. Dữ liệu user rating về 1 bộ phim nằm ở file ratings.dat. Cấu trúc của dữ liệu này như sau: userId::productId::rating::timestamp tức là gồm 4 cột cách nhau bởi dấu :: cột đầu tiên là userId kiểu Integer, cột thứ 2 là productId kiểu Integer, cột 3 là rating kiểu Float, cột 4 là timestamp tức thời điểm bộ phim được đánh giá (tạm thời ta chưa quan tâm cột này). Bước 3 này sẽ load dữ liệu từ file ratings.dat rồi split các cột ra.



1. Spark MLLib hỗ trợ Rating class gồm 3 property là userID, productId và rating, ta sẽ chuyển ratingsRDD thành RDD của các Rating object.



1. Ta chia dữ liệu thành 2 tập là training và test theo tỉ lệ 80-20
2. Xây dựng model



1. Recommend 5 bộ phim cho user 4169
2. Dành cho các bạn tự tìm hiểu cách đánh giá hiệu quả của mô hình này trên tập test data