

## 7.6 분석 파일럿 실행 4단계-제플린 실시간 분석

### 제플린을 이용한 운영 지역 분석

01. 제플린이 구동됐는지 확인하고, 아직 구동되지 않았다면 Server02에서 아래 명령으로 제플린 서버를 실행한다.

```
$ zeppelin-daemon.sh start
```

정상적으로 구동되는 데 1분 정도의 시간이 소요되므로 아래 명령으로 "OK"가 표시되는 것을 확인한다.

```
$ zeppelin-daemon.sh status
```

## ☆ 7.6 분석 파일럿 실행 4단계-제플린 실시간 분석

### ㄹ 제플린을 이용한 운영 지역 분석

02. 크롬 브라우저를 실행하고 제플린 환경에 접속해 홈 화면을 확인한다.

- 제플린 접속 URL: <http://server02.hadoop.com:8081/>

# 7.6 분석 파일럿 실행 4단계-제플린 실시간 분석

## 제플린을 이용한 운행 지역 분석

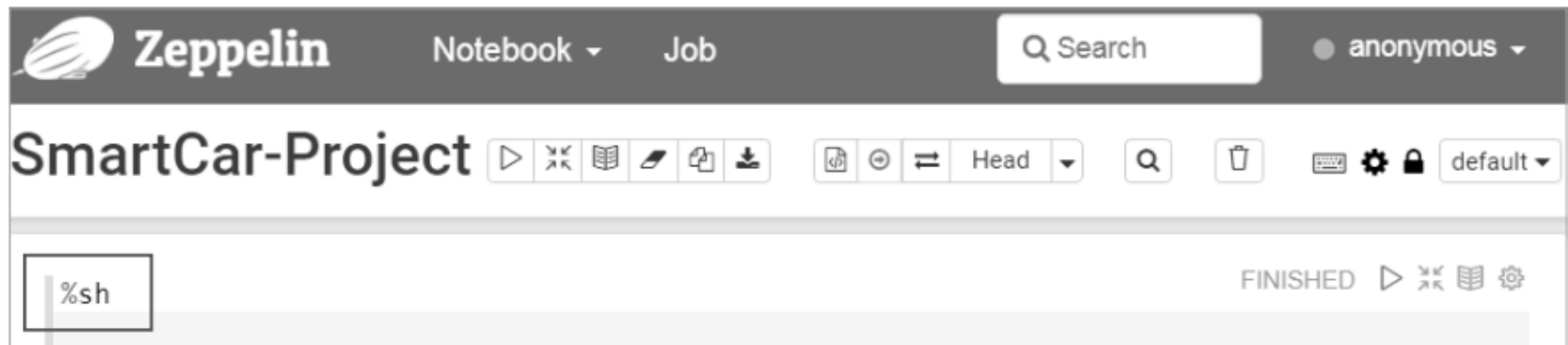
03. 제플린 Notebook을 생성한다. Note Name에는 "SmartCar-Project"를 입력하고 Default Interpreter로는 spark를 선택하고 [Create] 버튼을 클릭한다.

The screenshot shows the Zeppelin web interface. At the top, there's a header with the Zeppelin logo, a 'Notebook' dropdown menu, a 'Job' button, and a search bar. Below the header, a 'Welcome to Zeppelin!' message is visible. A 'Create New Note' dialog box is open in the foreground. The dialog has a title bar with a close button. Inside, there's a 'Note Name' field with the text 'SmartCar-Project' and a 'Default Interpreter' dropdown menu set to 'spark'. At the bottom right of the dialog is a 'Create' button. There are numbered callouts (1, 2, 3, 4) pointing to specific elements: 1 points to the 'Notebook' dropdown, 2 points to the '+ Create new note' button, 3 points to the 'Note Name' input field, and 4 points to the 'Default Interpreter' dropdown.

## 7.6 분석 파일럿 실행 4단계-제플린 실시간 분석

### 제플린을 이용한 운행 지역 분석

04. 먼저 HDFS 명령어를 실행해 분석할 “스마트카 운전자의 운행” 파일을 확인한다. 노트북에서 셀 명령이 가능하도록 바인딩하기 위해 “%sh”로 입력한 후 엔터 키를 누른다.

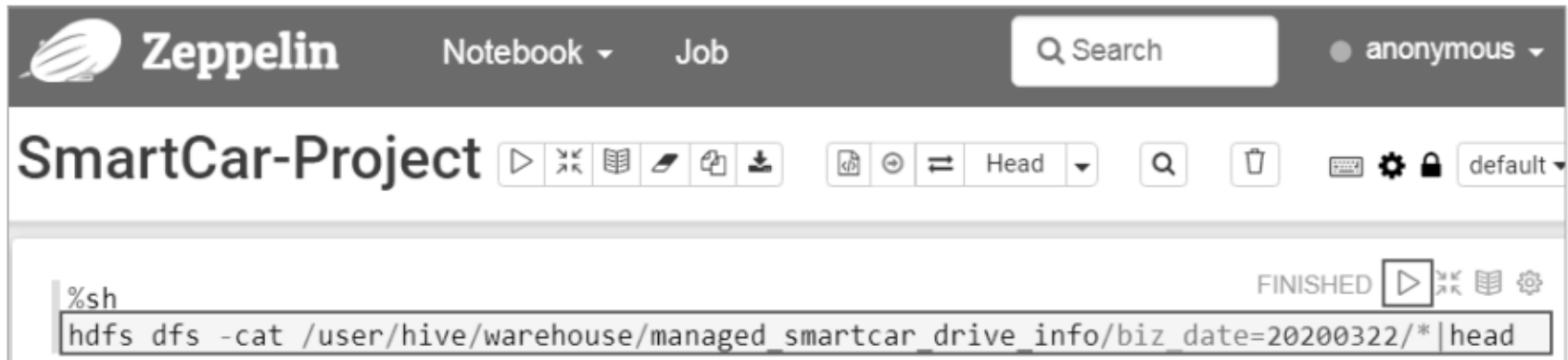


## 7.6 분석 파일럿 실행 4단계-제플린 실시간 분석

### 제플린을 이용한 운행 지역 분석

05. 하이브의 Managed 영역에 만들어져 있는 “스마트카 운전자 운행 정보” 파일을 확인하기 위해 다음 명령어를 입력하고 오른쪽의 [실행] 버튼을 클릭하거나 단축키로 Shift+Enter를 이용한다. biz\_date 정보는 독자의 파일럿 상황에 맞는 날짜로 입력해야 한다.

- `hdfs dfs -cat /user/hive/warehouse/managed_smartcar_drive_info/biz_date=20200322/* | head`



The image shows the Zeppelin Notebook interface. The top bar includes the Zeppelin logo, 'Notebook' and 'Job' tabs, a search bar, and a user dropdown set to 'anonymous'. The main header is 'SmartCar-Project'. Below it is a toolbar with various icons for execution, view, and settings. The central area shows a terminal window with the command `%sh` and `hdfs dfs -cat /user/hive/warehouse/managed_smartcar_drive_info/biz_date=20200322/* | head`. The status 'FINISHED' is displayed next to the command, along with a play button icon and other controls.

# 7.6 분석 파일럿 실행 4단계-제플린 실시간 분석

## 제플린을 이용한 운행 지역 분석

06. 노트북에서 다음과 같은 결과를 확인할 수 있다.

The image shows the Zeppelin Notebook interface. The top bar includes the Zeppelin logo, 'Notebook' and 'Job' tabs, a search bar, and a user dropdown set to 'anonymous'. The main header displays 'SmartCar-Project' with various action icons. Below this, a terminal window shows the execution of a shell command. The command is: `%sh hdfs dfs -cat /user/hive/warehouse/managed_smartcar_drive_info/biz_date=20200322/*|head`. The output is a list of 10 lines of comma-separated data, including IDs, names, ages, genders, locations, professions, and dates. The status 'FINISHED' is shown in the top right of the terminal window.

```
%sh
hdfs dfs -cat /user/hive/warehouse/managed_smartcar_drive_info/biz_date=20200322/*|head
```

```
A0004,남,31,미혼,광주,공무원,2000,2010,D,0,0,F,N,0,F06,20200322000000
A0097,여,38,미혼,대구,전문직,2500,2014,C,1,0,F,N,1,E01,20200322000000
B0006,여,54,기혼,강원,공무원,1500,2001,H,4,0,F,N,20,A05,20200322000000
B0043,여,68,미혼,대전,개인사업,1700,2003,F,1,0,F,N,5,F06,20200322000000
B0051,여,29,미혼,서울,회사원,2000,2011,E,1,0,R1,R,5,D02,20200322000000
B0065,여,33,기혼,경기,프리랜서,1700,2000,C,0,0,F,N,0,B10,20200322000000
B0070,여,59,미혼,대전,공무원,2000,2013,F,1,0,F,N,5,B02,20200322000000
B0094,여,55,기혼,서울,공무원,1500,2005,F,0,1,F,N,0,E05,20200322000000
C0047,여,66,기혼,서울,전문직,1500,2014,C,5,0,F,N,25,D01,20200322000000
C0049,여,22,미혼,경기,주부,1500,2003,E,3,0,R1,R,15,B06,20200322000000
```

# 7.6 분석 파일럿 실행 4단계-제폴린 실시간 분석

## 제폴린을 이용한 운행 지역 분석

07. 앞의 HDFS 데이터셋 확인 명령이 끝나면 하단에 새로운 입력창(Paragraph)이 그림 7.42처럼 활성화된다.

```
%sh
hdfs dfs -cat /user/hive/warehouse/managed_smartcar_drive_info/biz_date=20200322/*|head
```

A0004,남,31,미혼,광주,공무원,2000,2010,D,0,0,F,N,0,F06,20200322000000  
A0097,여,38,미혼,대구,전문직,2500,2014,C,1,0,F,N,1,E01,20200322000000  
B0006,여,54,기혼,강원,공무원,1500,2001,H,4,0,F,N,20,A05,20200322000000  
B0043,여,68,미혼,대전,개인사업,1700,2003,F,1,0,F,N,5,F06,20200322000000  
B0051,여,29,미혼,서울,회사원,2000,2011,E,1,0,R1,R,5,D02,20200322000000  
B0065,여,33,기혼,경기,프리랜서,1700,2000,C,0,0,F,N,0,B10,20200322000000  
B0070,여,59,미혼,대전,공무원,2000,2013,F,1,0,F,N,5,B02,20200322000000  
B0094,여,55,기혼,서울,공무원,1500,2005,F,0,1,F,N,0,E05,20200322000000  
C0047,여,66,기혼,서울,전문직,1500,2014,C,5,0,F,N,25,D01,20200322000000  
C0049,여,22,미혼,경기,주부,1500,2003,E,3,0,R1,R,15,B06,20200322000000  
cat: Unable to write to output stream.

Took 9 sec. Last updated by anonymous at March 29 2020, 1:12:45 AM.

```
READY
```

## 7.6 분석 파일럿 실행 4단계-제플린 실시간 분석

### 제플린을 이용한 운영 지역 분석

08. 스파크의 스칼라 코드를 그림 7.43처럼 작성한다. 작성할 스칼라 코드는 HDFS에서 데이터를 로드하고, 로드한 데이터셋을 대상으로 스파크에서 활용하기 위한 데이터 구조로 만드는 간단한 작업이다. 이때도 역시 biz\_date 정보는 독자의 파일럿 상황에 맞는 날짜로 입력해야 한다. 실행할 스파크-SQL 예제는 C://예제소스/bigdata2nd-master/CH07/Zeppelin-SparkSQL/에 있으니 참고한다. 입력이 완료되면 우측 상단의 [실행] 버튼을 누른다.



```

val url="hdfs://server01.hadoop.com:8020"
val dPath="/user/hive/warehouse/managed_smartcar_drive_info/biz_date=20200322/*"
val driveData=sc.textFile(url + dPath)
case class DriveInfo(car_num: String,          sex: String,          age: String,
                    marriage: String,         region: String,        job: String,
                    car_capacity: String,     car_year: String,       car_model: String,
                    speed_pedal: String,      break_pedal: String,    steer_angle: String,
                    direct_light: String,     speed: String,         area_num: String,
                    date: String)

val drive = driveData.map(sd=>sd.split(",")).map(
    sd=>DriveInfo(sd(0).toString, sd(1).toString, sd(2).toString, sd(3).toString,
                  sd(4).toString, sd(5).toString, sd(6).toString, sd(7).toString,
                  sd(8).toString, sd(9).toString, sd(10).toString, sd(11).toString,
                  sd(12).toString, sd(13).toString, sd(14).toString, sd(15).toString
    )
)

drive.toDF().registerTempTable("DriveInfo")

```

warning: there was one deprecation warning; re-run with -deprecation for details

```
import sqlContext.implicits._
```

```
url: String = hdfs://server01.hadoop.com:8020
```

```
dPath: String = /user/hive/warehouse/managed_smartcar_drive_info/biz_date=20200322/*
```

```
driveData: org.apache.spark.rdd.RDD[String] = hdfs://server01.hadoop.com:8020/user/hive/warehouse/managed_smartcar_drive_info/biz_date=20200322/* MapPartitionsRDD[34] at textFile at <console>:25
```

```
defined class DriveInfo
```

```
drive: org.apache.spark.rdd.RDD[DriveInfo] = MapPartitionsRDD[36] at map at <console>:33
```

## ☆ 7.6 분석 파일럿 실행 4단계-제플린 실시간 분석

### ☞ 제플린을 이용한 운영 지역 분석

09. 정상적으로 실행되고 나면 다음과 같은 메시지를 확인할 수 있다.

```
driveData: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[75] at textFile at <console>:23
defined class DriveInfo
drive: org.apache.spark.rdd.RDD[DriveInfo] = MapPartitionsRDD[77] at map at <console>:27
```

그림 7.44 제플린 노트북 - 스파크 SQL 실행 확인

그림 7.43 예제의 마지막 줄을 보면 로드한 데이터셋을 스파크-SQL로 분석하기 위해 임시 테이블인 DriveInfo를 생성한 것을 확인할 수 있다. 그럼 스파크-SQL을 실행하기 위해 “%spark.sql”을 입력하고 엔터 키를 누른다.

## 7.6 분석 파일럿 실행 4단계-제플린 실시간 분석

### 제플린을 이용한 운행 지역 분석

10. 스파크-SQL 쿼리를 입력하고 실행한다.

다음 쿼리는 스마트카가 운행한 지역의 평균 속도를 구하고, 평균 속도가 높은 순서대로 출력한다.

```
%spark.sql
select T1.area_num, T1.avg_speed
from (select area_num, avg(speed) as avg_speed
      from DriveInfo
      group by area_num
     ) T1
order by T1.avg_speed desc
```

FINISHED   

그림 7.45 제플린 노트북 - 스파크-SQL 조회 쿼리

## 7.6 분석 파일럿 실행 4단계-제폴린 실시간 분석

### 제폴린을 이용한 운행 지역 분석

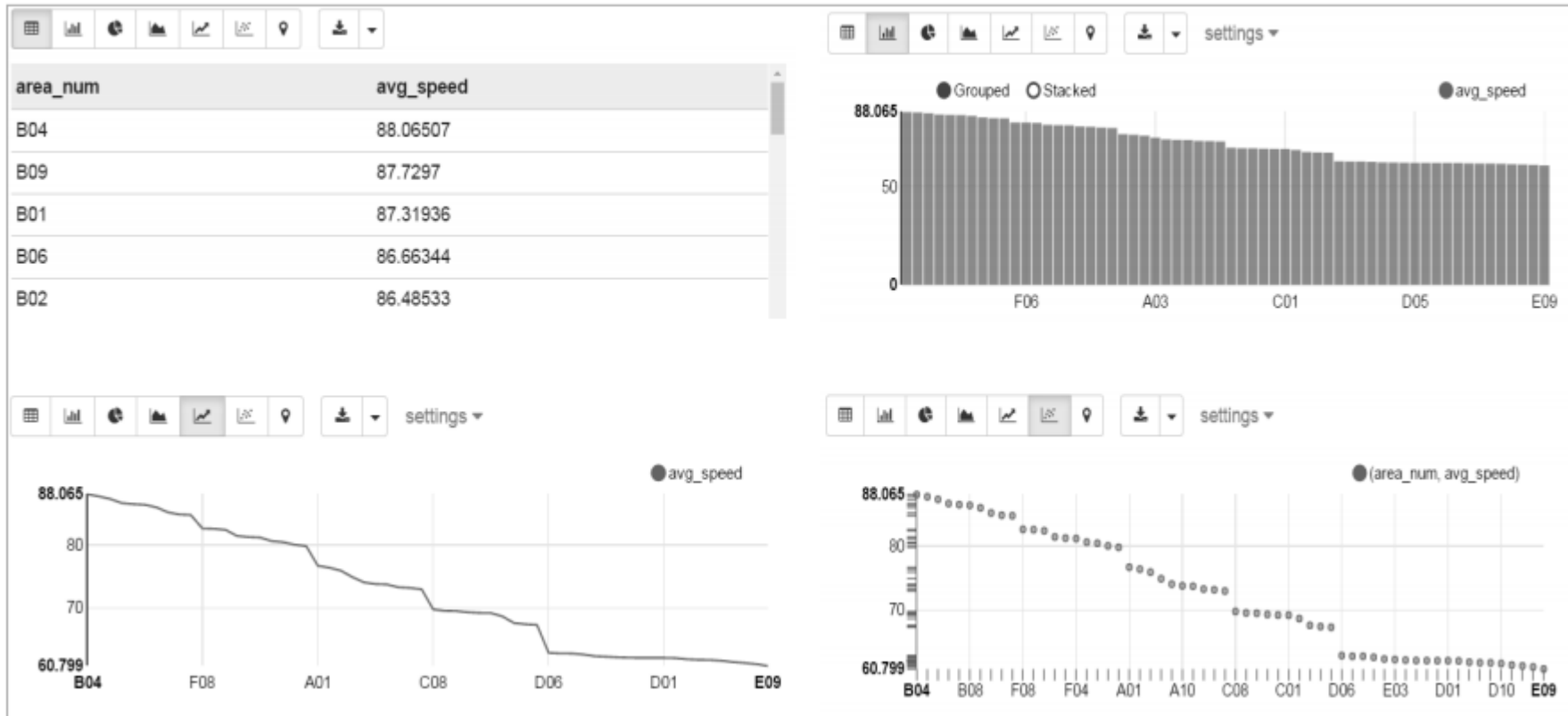
11. 실행 결과가 하단에 테이블 형식으로 출력된다. “B04” 지역이 평균 속도 88km/h로 가장 높게 나왔다.

area_num	avg_speed
B04	88.06507
B09	87.7297
B01	87.31936
B06	86.66344
B02	86.48533
B08	86.38922
B03	85.99351

# 7.6 분석 파일럿 실행 4단계-제폴린 실시간 분석

## 제폴린을 이용한 운행 지역 분석

12. 출력된 결과의 상단에 있는 차트 버튼을 누르면 다양한 형식의 차트로 재구성해서 볼 수 있다.



## 7.6 분석 파일럿 실행 4단계-제플린 실시간 분석

### 제플린을 이용한 운행 지역 분석

13. 또한 기존 스파크-SQL의 조건 변수를 설정해서 좀 더 효율적인 분석을 할 수 있다. Having 절을 추가해서 평균 속도에 대한 동적 변수인 AvgSpeed를 정의하고 실행해 보자.

```
%spark.sql
select T1.area_num, T1.avg_speed
from (select area_num, avg(speed) as avg_speed
      from DriveInfo
      group by area_num having avg_speed >= ${AvgSpeed=60}) T1
order by T1.avg_speed desc
```

FINISHED ▶ ⌵ 📖 ⚙️



## 7.6 분석 파일럿 실행 4단계-제플린 실시간 분석

### 제플린을 이용한 운영 지역 분석

제플린은 다양한 인터프리터(스파크, 하이브, 플링크, R, 카산드라 등)를 이용할 수 있다. 또한 제플린 인터페이스를 구현해서 새로운 인터프리터를 추가 및 확장할 수 있고, 현재도 많은 인터프리터들이 개발 중이다. 특히 제플린에서는 Spark-Group(spark, pyspark, spark.sql, spark.ml, spark.mahout, spark.r 등)을 쉽고 편리하게 사용할 수 있다는 것이 장점이라 할 수 있다.

저사양 파일럿 환경: 제플린 서비스를 정지한다.

- 제플린 서비스: Server02에 SSH로 접속한 후 다음 명령을 실행

```
$ zeppelin-daemon.sh stop
```



## 7.6 분석 파일럿 실행 4단계-제플린 실시간 분석

 제플린을 이용한 운영 지역 분석

# 실습