빅데이터 플랫폼

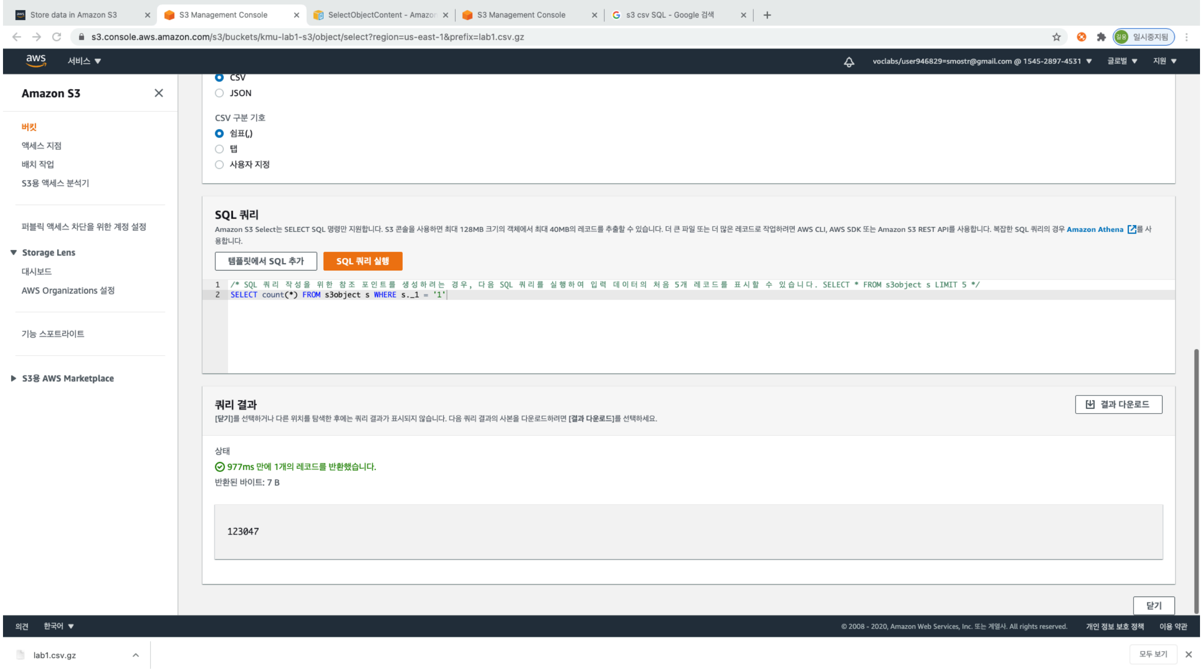
Data Analytics Challenge Questions 답안 제출

소프트웨어전공
20152791 강길웅

Lab 1 : Store data in Amazon S3

Challenge one

The taxi data includes data for two different vendors. The **vendorid** field has two possible values: *1* and *2*. Write a query to count the number of rides for vendor 1.



● CSV
○ JSON

CSV 구분 기호
● 쉼표(,)
○ 탭
○ 사용자 지정

SQL 쿼리

Amazon S3 Select는 SELECT SQL 명령만 지원합니다. S3 콘솔을 사용하면 최대 128MB 크기의 객체에서 최대 40MB의 레코드를 추출할 수 있습니다. 더 큰 파일 또는 더 많은 레코드로 작업하려면 AWS CLI, AWS SDK 또는 Amazon S3 REST API를 사용합니다. 복잡한 SQL 쿼리의 경우 Amazon Athena를 사용합니다.

```
/* SQL 쿼리 작성을 위한 참조 포인트를 생성하려는 경우, 다음 SQL 쿼리를 실행하여 입력 데이터의 처음 5개 레코드를 표시할 수 있습니다. SELECT * FROM s3object s LIMIT 5 */
SELECT count(*) FROM s3object s WHERE s._1 = '1'
```

쿼리 결과

[닫기]를 선택하거나 다른 위치를 탐색한 후에는 쿼리 결과가 표시되지 않습니다. 다음 쿼리 결과의 사본을 다운로드하려면 **[결과 다운로드]**를 선택하세요.

상태
977ms 만에 1개의 레코드를 반환했습니다.
반환된 바이트: 7 B

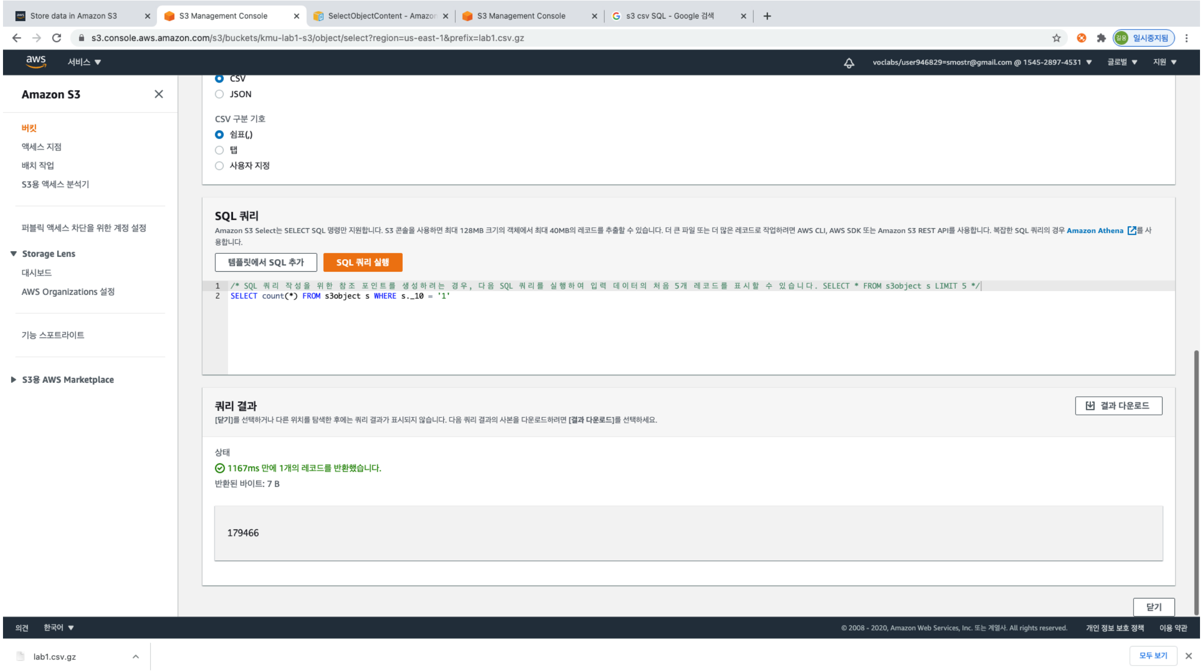
```
123047
```

```
SELECT count(*) FROM s3object s WHERE s._1 = '1'
```

s._1 (vendorID) 가 1인 row의 count를 출력하도록 한다.

Challenge two

The taxi data includes data for payment type. Payment type *1* is for payments that are made by credit card. Write a query to total the number of trips that were paid for by credit card.



s3.console.aws.amazon.com/s3/buckets/kmu-lab1-s3/object/select?region=us-east-1&prefix=lab1.csv.gz
Amazon S3
버킷
액세스 지점
배치 작업
S3용 액세스 분석기
퍼블릭 액세스 차단을 위한 계정 설정
Storage Lens
대시보드
AWS Organizations 설정
기능 스포트라이트
S3용 AWS Marketplace

● CSV
○ JSON

CSV 구분 기호
● 쉼표(,)
○ 탭
○ 사용자 지정

SQL 쿼리

Amazon S3 Select는 SELECT SQL 명령만 지원합니다. S3 콘솔을 사용하면 최대 128MB 크기의 객체에서 최대 40MB의 레코드를 추출할 수 있습니다. 더 큰 파일 또는 더 많은 레코드로 작업하려면 AWS CLI, AWS SDK 또는 Amazon S3 REST API를 사용합니다. 복잡한 SQL 쿼리의 경우 Amazon Athena를 사용합니다.

```
/* SQL 쿼리 작성을 위한 참조 포인트를 생성하려는 경우, 다음 SQL 쿼리를 실행하여 입력 데이터의 처음 5개 레코드를 표시할 수 있습니다. SELECT * FROM s3object s LIMIT 5 */
SELECT count(*) FROM s3object s WHERE s._10 = '1'
```

쿼리 결과

[닫기]를 선택하거나 다른 위치를 탐색한 후에는 쿼리 결과가 표시되지 않습니다. 다음 쿼리 결과의 사본을 다운로드하려면 [결과 다운로드]를 선택하세요.

상태
✓ 1167ms 만에 1개의 레코드를 반환했습니다.
반환된 바이트: 7 B

```
179466
```

결과 다운로드
닫기
© 2008 - 2020, Amazon Web Services, Inc. 또는 계열사. All rights reserved.
lab1.csv.gz

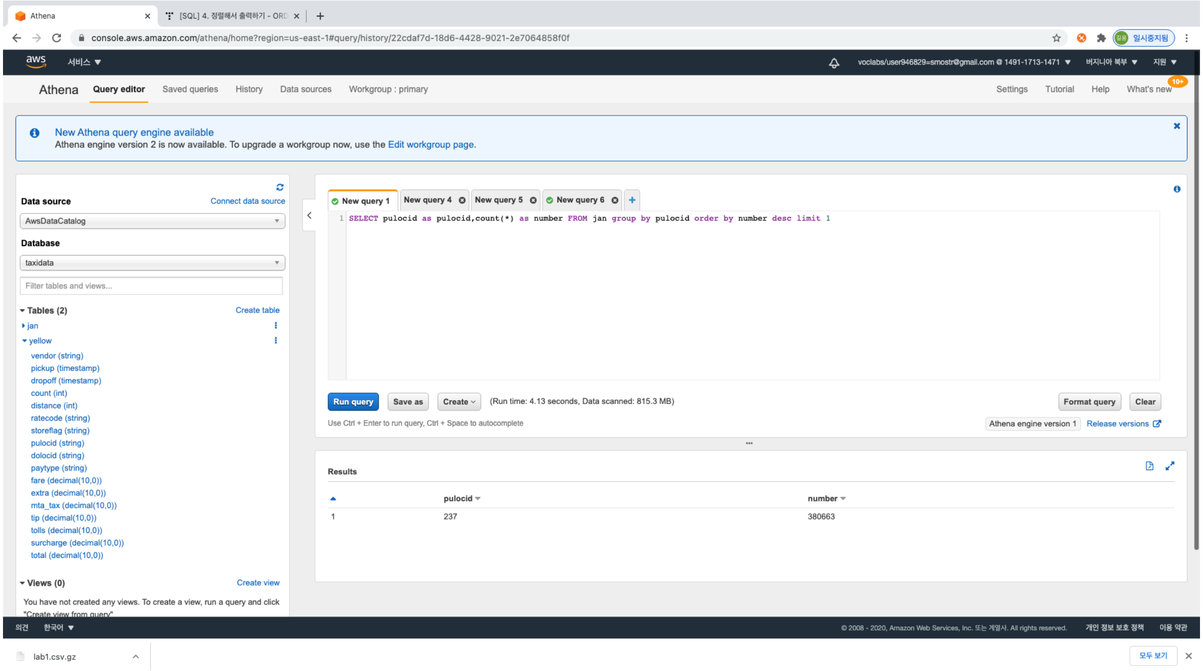
```
SELECT count(*) FROM s3object s WHERE s._10 = '1'
```

s._10(payment type) 가 1인 row의 count를 출력 하도록 한다.

Lab 2 : Query Data in Amazon Athena

Challenge one

Write a query that identifies the most common pickup location in January 2017.



console.aws.amazon.com/athena/home?region=us-east-1#query/history/22cdaf7d-18d6-4428-9021-2e7064858f0f
Athena
Query editor
Saved queries
History
Data sources
Workgroup : primary
Settings
Tutorial
Help
What's new

New Athena query engine available
Athena engine version 2 is now available. To upgrade a workgroup now, use the Edit workgroup page.

Data source: AwsDataCatalog
Database: taxidata

Tables (2)
- jan
- yellow
  - vendor (string)
  - pickup (timestamp)
  - dropoff (timestamp)
  - count (int)
  - distance (int)
  - ratecode (string)
  - storeflag (string)
  - pulocid (string)
  - dolocid (string)
  - paytype (string)
  - fare (decimal(10,0))
  - extra (decimal(10,0))
  - mta_tax (decimal(10,0))
  - tip (decimal(10,0))
  - tolls (decimal(10,0))
  - surcharge (decimal(10,0))
  - total (decimal(10,0))

Views (0)
You have not created any views. To create a view, run a query and click "Create view from query"

```
SELECT pulocid as pulocid,count(*) as number FROM jan group by pulocid order by number desc limit 1
```

(Run time: 4.13 seconds, Data scanned: 815.3 MB)

Use Ctrl + Enter to run query, Ctrl + Space to autocomplete

Athena engine version 1

Results

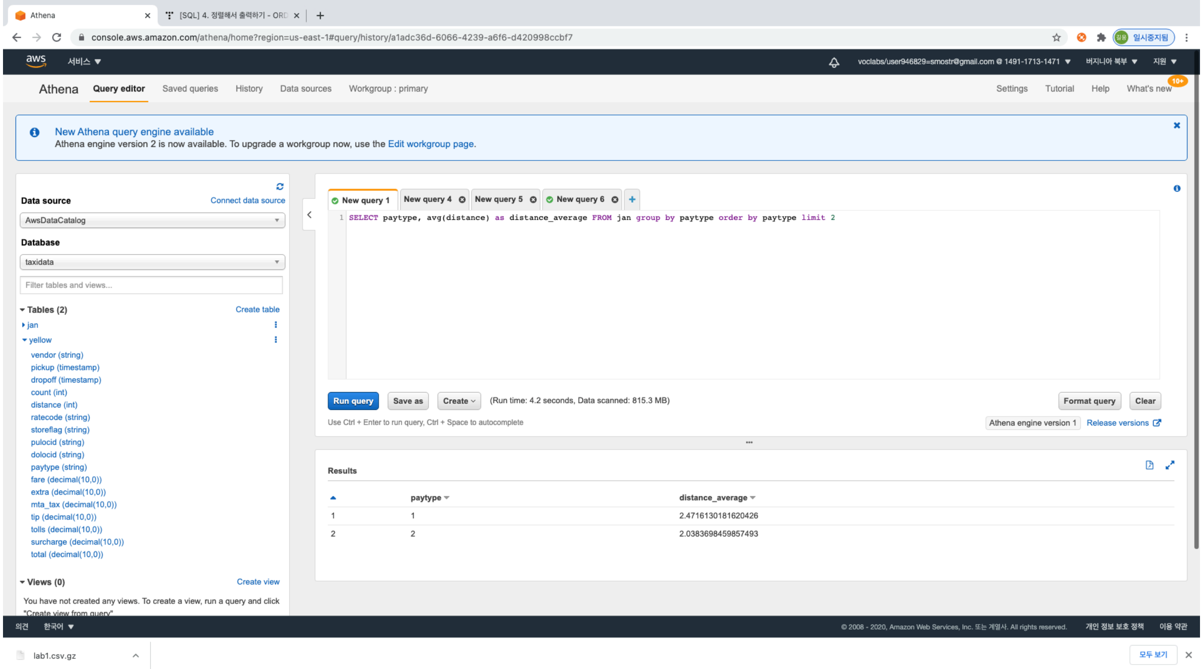
	pulocid	number
1	237	380663

```
SELECT pulocid as pulocid, count(*) as number FROM jan group by pulocid
order by number desc limit 1
```

픽업 위치 id와 그 숫자를 id별로 묶어 가장 큰 값 하나를 출력한다.

Challenge two

Write a query to compare the average distance for trips that were paid with credit cards and the average distance for trips that were paid with cash in January 2017.



New Athena query engine available
Athena engine version 2 is now available. To upgrade a workgroup now, use the Edit workgroup page.

Data source: AwsDataCatalog
Database: taxidata

Tables (2)
- jan
- yellow
  - vendor (string)
  - pickup (timestamp)
  - dropoff (timestamp)
  - count (int)
  - distance (int)
  - ratecode (string)
  - storeflag (string)
  - pulocid (string)
  - dolocid (string)
  - paytype (string)
  - fare (decimal(10,0))
  - extra (decimal(10,0))
  - mta_tax (decimal(10,0))
  - tip (decimal(10,0))
  - tolls (decimal(10,0))
  - surcharge (decimal(10,0))
  - total (decimal(10,0))

Views (0)
You have not created any views. To create a view, run a query and click "Create view from query".

```
SELECT paytype, avg(distance) as distance_average FROM jan group by paytype order by paytype limit 2
```

(Run time: 4.2 seconds, Data scanned: 815.3 MB)

Results

	paytype	distance_average
1	1	2.4716130181620426
2	2	2.0383698459857493

```
SELECT paytype, avg(distance) as distance_average FROM jan group by
paytype order by paytype limit 2
```

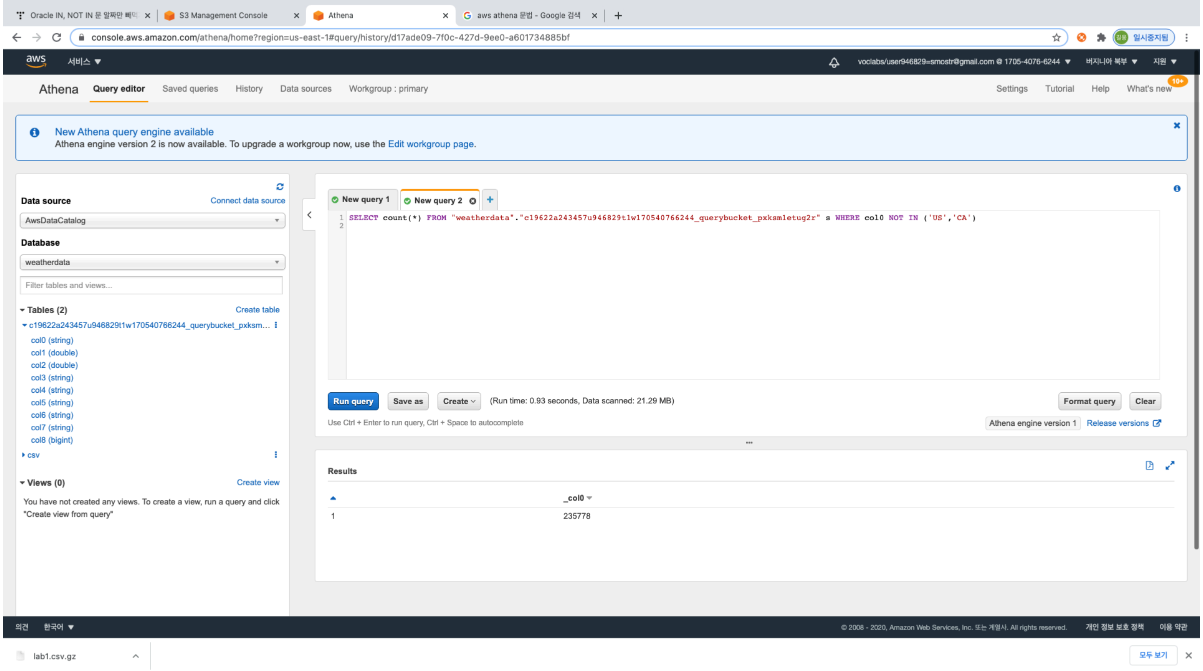
결제 수단 기준으로 거리 평균값을 계산해 결제수단과 거리 평균값을 출력한다. 1,2가 카드와 현금 이므로 결제 수단 기준 오름차 정렬하여 2개만 출력하도록 한다.

Lab 3 : Query data in Amazon S3 with Amazon Athena and AWS Glue

Challenge question

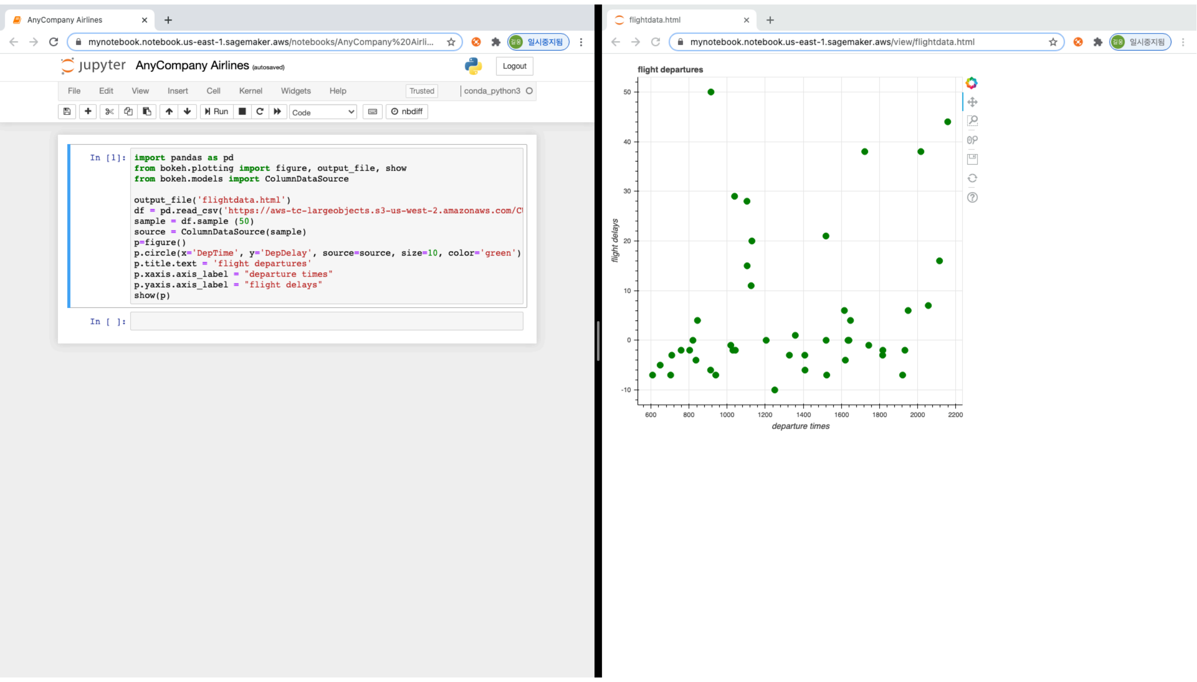
Now that you know how AWS Glue and Athena work together, try the following exercise to test your knowledge. Your lab instructor has a model solution. However, there is more than one way to solve the challenge.

The Global Historical Climatology Network Daily receives data from around the world. You can download data that describes these stations at the following location: ghcnd-stations.txt. The data dictionary for the observation and stations data is available at the following location: Readme.

> **Note** The ghcnd-stations.txt file is a fixed-width text file. Before you use it with AWS Glue, you must convert it to a comma-separated values (CSV) file, or one of the other file formats that AWS Glue supports. One easy way to convert a text file to a .csv file is to open the text file with a spreadsheet program and then save the file in .csv format. You can also find free utility programs on the internet that can help with this process.
>
> For this challenge, do the following tasks:
>
> - Use AWS Glue to create a table for the weather stations.
> - Write a query in Athena to count the number of stations that are not in the US or Canada. The first two characters of the station ID field indicate the country where the station is located. The country codes for the United States is *US* and the country code for Canada is *CA*.
>
> > 요약하자면 ghcnd-stations.txt를 다운받아 .csv파일로 변환하여 AWS Glue를 이용해 weather stations 테이블을 만들고, 미국과 캐나다에 없는 방송국 수를 세는 쿼리를 만들어라.
>
> 
>
>
>
> New Athena query engine available
> Athena engine version 2 is now available. To upgrade a workgroup now, use the Edit workgroup page.
>
> Data source: AwsDataCatalog
> Database: weatherdata
>
> Tables (2)
> - c19622a243457u946829t1w170540766244_querybucket_pxksm...
>   - col0 (string)
>   - col1 (double)
>   - col2 (double)
>   - col3 (string)
>   - col4 (string)
>   - col5 (string)
>   - col6 (string)
>   - col7 (string)
>   - col8 (bigint)
> - csv
>
> Views (0)
> You have not created any views. To create a view, run a query and click "Create view from query"
>
> ```
> SELECT count(*) FROM "weatherdata"."c19622a243457u946829t1w170540766244_querybucket_pxksmletug2r" s WHERE col0 NOT IN ('US','CA')
> ```
>
> (Run time: 0.93 seconds, Data scanned: 21.29 MB)
>
> Use Ctrl + Enter to run query, Ctrl + Space to autocomplete
>
> | | _col0 |
> |---|---|
> | 1 | 235778 |
>
> ```
> SELECT count(*) FROM "weatherdata"."tablename" s WHERE col0 NOT IN ('US','CA')
> // 테이블명이 너무 길어 tablename으로 대체.
> ```
>
> 좌측 테이블 리스트를 보면 테이블이 추가되어 있는데 아카데미에서 생성한 S3에 csv를 넣어 Glue를 이용하려다 보니 테이블 이름이 S3 버킷 이름을 따라가 길어지게 되었다. Column명은 본래 테이블의 헤더가 무엇인지 알지 못하기에 수정하지 않음.
>
> NOT IN구문을 이용해 'US'와 'CA'가 포함되지 않은 데이터의 숫자를 세도록 쿼리문을 만들었다. (col0가 스테이션 ID필드)

Lab 5 : Analyze Data with Amazon SageMaker, Jupyter Notebooks, and Bokeh

> ## Challenge question
>
> Now that you can use the Bokeh Python package to create data visualizations, try the following challenge to apply your skills to a real-world case.
>
> AnyCompany Airlines has collected sample data for flight departures. They asked you to analyze the data to determine if there is an association between departure times and flight delays. You can access the sample data from Amazon S3. Develop a visualization that will describe this association.
>
> > 비행기의 출발시간과 비행 지연시간에 연관성 확인을 위한 시각 자료를 개발.


AnyCompany Airlines
mynotebook.notebook.us-east-1.sagemaker.aws/notebooks/AnyCompany%20Airli...
일시중지됨
jupyter AnyCompany Airlines (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
conda_python3
Run
Code
nbdiff
flightdata.html
mynotebook.notebook.us-east-1.sagemaker.aws/view/flightdata.html
flight departures
50
40
30
20
10
0
-10
flight delays
600
800
1000
1200
1400
1600
1800
2000
2200
departure times

```
In [1]: import pandas as pd
        from bokeh.plotting import figure, output_file, show
        from bokeh.models import ColumnDataSource

        output_file('flightdata.html')
        df = pd.read_csv('https://aws-tc-largeobjects.s3-us-west-2.amazonaws.com/C
        sample = df.sample (50)
        source = ColumnDataSource(sample)
        p=figure()
        p.circle(x='DepTime', y='DepDelay', source=source, size=10, color='green')
        p.title.text = 'flight departures'
        p.xaxis.axis_label = "departure times"
        p.yaxis.axis_label = "flight delays"
        show(p)

In [ ]:
```

> 좌측 코드는 실습 중 Task 5.2 - 44 의 코드를 이용하여 만들었다. 우측은 해당 결과. x축이 출발시간, y축이 비행 지연 시간이다.

Lab 6 : Automate Loading Data with AWS Data Pipeline

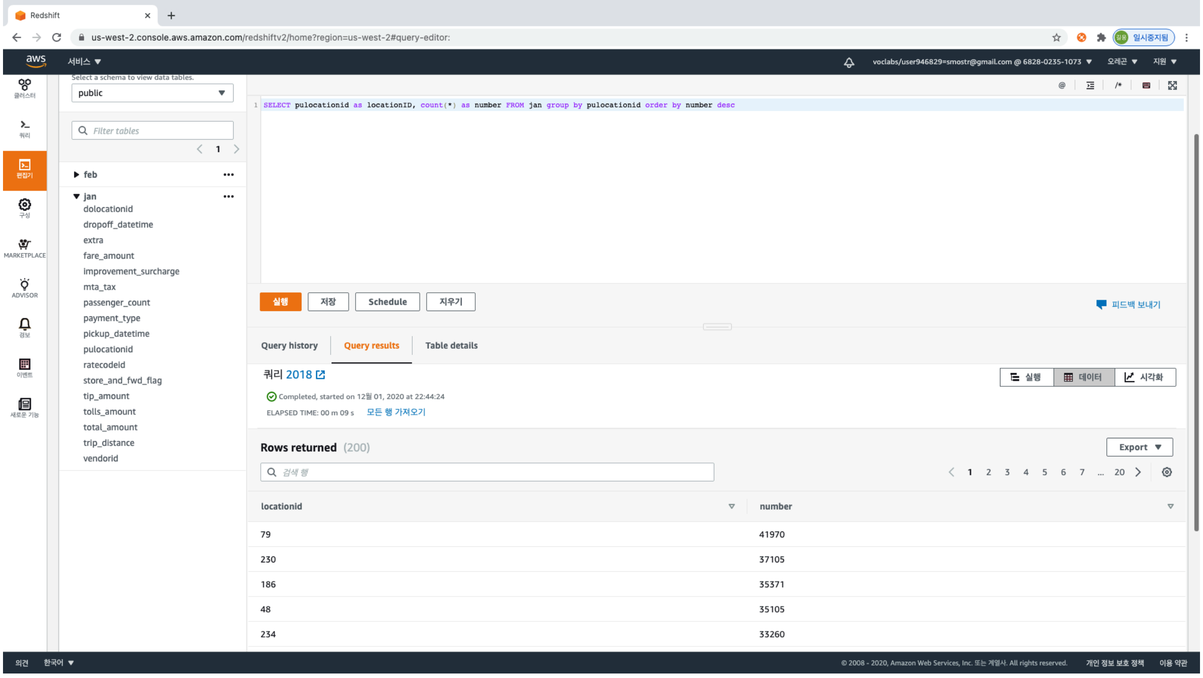
> ## Challenge question
>
> Now that you can automate loading data by using AWS Data Pipeline, try the following challenge to apply your skills to a real-world case.
>
> Your manager is pleased that you automated the process of loading data to Amazon Redshift. He would now like you to do two additional tasks:
>
> - Create a pipeline that will load a second month of data.
> - Determine the most common pickup locations for each of the two months.

The February data is in the following Amazon S3 location:

```
s3://aws-tc-largeobjects/CUR-TF-200-ACBDFO-1/Lab6/February
```

> 2월 데이터를 위한 data pipeline을 제작하고 가장 pickup이 빈번히 발생하는 pickup location을 월별로 각각 구할것.

- 1월



```
SELECT pulocationid as locationID, count(*) as number FROM jan group by pulocationid order by number desc
```

쿼리 2018

Completed, started on 12월 01, 2020 at 22:44:24

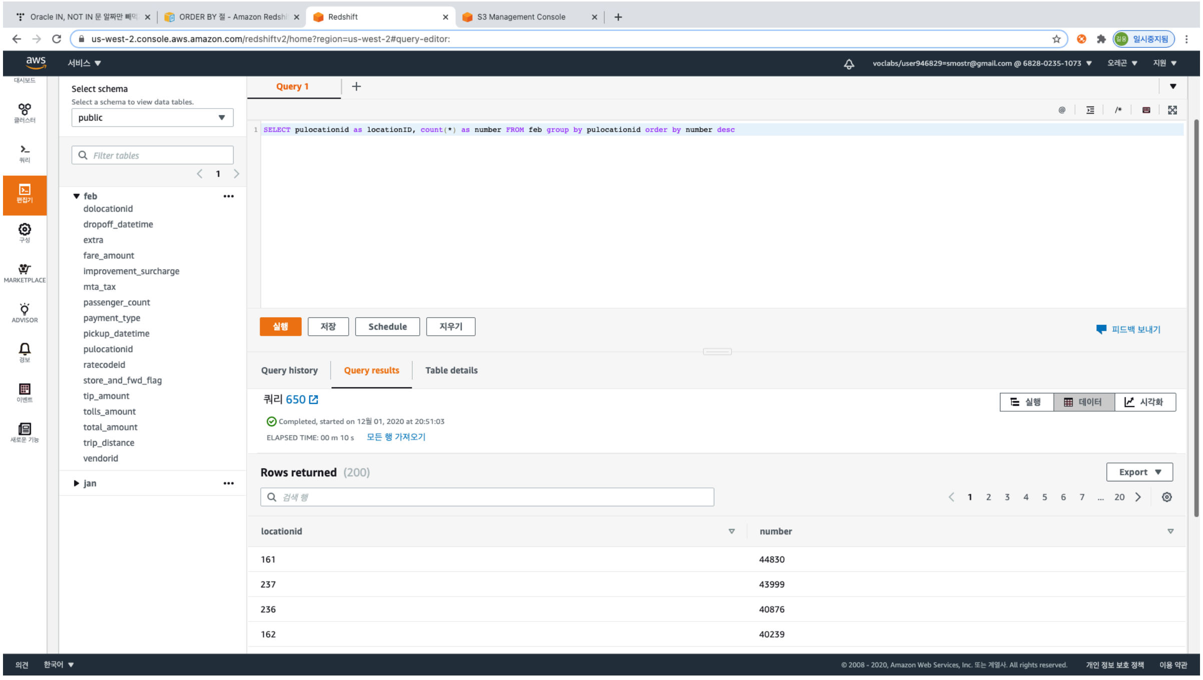
ELAPSED TIME: 00 m 09 s 모든 행 가져오기

Rows returned (200)

locationid	number
79	41970
230	37105
186	35371
48	35105
234	33260

```
SELECT pulocationid as locationID, count(*) as number FROM jan group by
pulocationid order by number desc
```

- 2월



```
SELECT pulocationid as locationID, count(*) as number FROM feb group by pulocationid order by number desc
```

쿼리 650

Completed, started on 12월 01, 2020 at 20:51:03

ELAPSED TIME: 00 m 10 s 모든 행 가져오기

Rows returned (200)

locationid	number
161	44830
237	43999
236	40876
162	40239

> ```
> SELECT pulocationid as locationID, count(*) as number FROM feb group by
> pulocationid order by number desc
> ```
>
> pulocationid(픽업 장소) 로 group화 하여 데이터를 카운트하여 내림차순으로 정렬하여 표기.