



**Amazon Web Services
Data Engineering Immersion Day**

Bridging Data for AI/ML using Amazon SageMaker
March 2020

Table of Contents

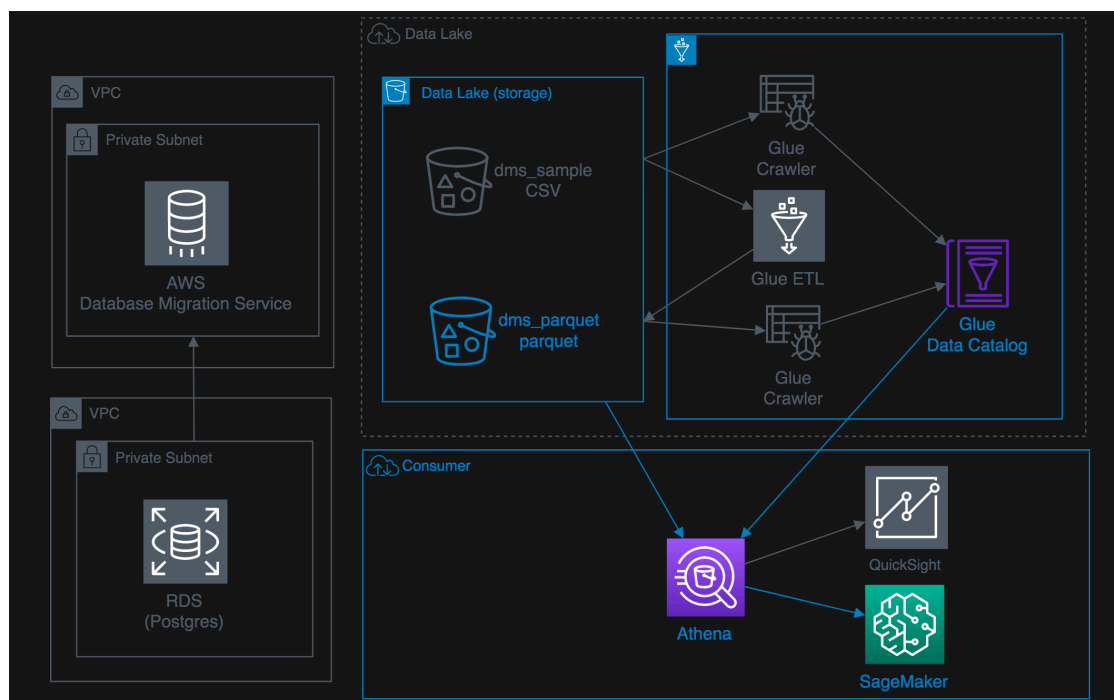
| | |
|--|-----------------|
| <i>Introduction</i> | <i>2</i> |
| <i>Create Amazon SageMaker Notebook Instance.....</i> | <i>3</i> |
| <i>Connect the SageMaker Jupyter notebook to Athena</i> | <i>7</i> |
| <i>Working in Pandas.....</i> | <i>9</i> |

Introduction

Amazon SageMaker is an end-to-end machine learning platform that lets you build, train, and deploy machine learning models in AWS. It is a highly modular service that lets you use each of these components independently of each other.

You will Learn:

- How to use the Jupyter notebook component of Sagemaker to integrate with the data lake using Athena
- Populate data frames for data manipulation.

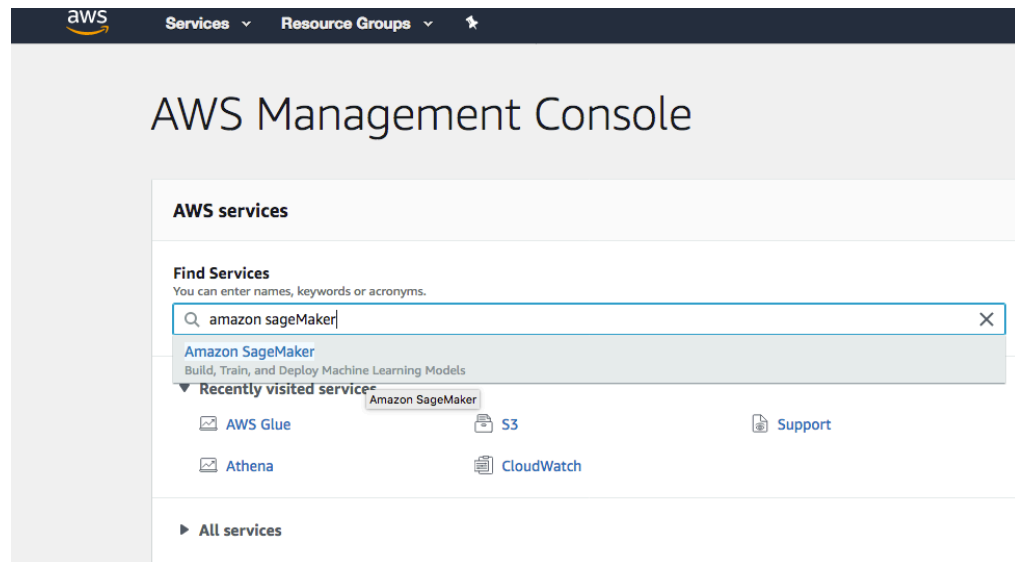


This process to prepare the data to satisfy the needs of ML algorithms is iterative. To prepare the data, we will make the table definitions in Athena available in a Jupyter notebook instance of SageMaker.

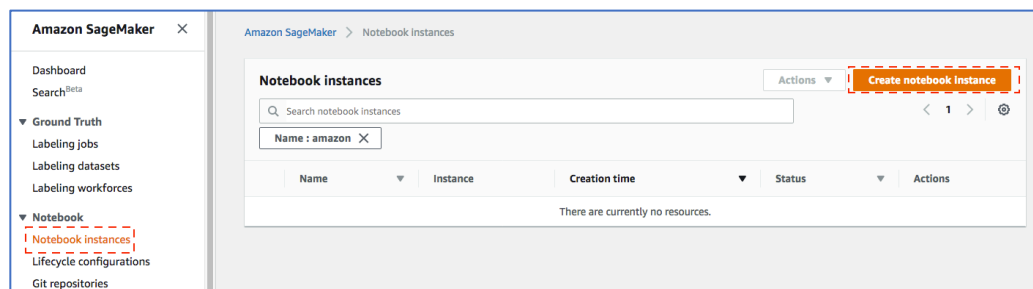
Jupyter notebooks are popular among data scientists and used to visualize data, perform statistical analysis, complete data manipulations, and make the data ready for machine learning work.

Create Amazon SageMaker Notebook Instance

1. Go to the Amazon SageMaker from AWS console



2. In the Amazon SageMaker navigation pane, click **Notebook instances** and Click **Create notebook instance**.



3. Put following values to create instance
 - a. **Give your choice of name for Notebook instance name e.g. "datalake-Sagemaker"**
 - b. You can leave **Notebook instance type** as default value "ml.t2.medium" for this lab.
 - c. Leave Elastic **Inference** as null. This is to add extra resources.
 - d. Choose a role for the notebook instances in Amazon SageMaker to interact with Amazon S3. As role doesn't exist select the **Create new role** option.
 - e. In **Create an IAM Role** pop up window, choose the specific S3 bucket you will grant access to. For this lab, choose **Any S3 bucket** as shown below and click **Create Role**:

- g. There is no Athena permission available in your SageMaker execution role. In this case, it is "AmazonSageMaker-ExecutionRole-20190531T143193". Click **Attach Policies** button in **Permissions** tab.

Search IAM

Dashboard
Groups
Users
Roles
Policies
Identity providers
Account settings
Credential report
Encryption keys

Roles > AmazonSageMaker-ExecutionRole-20190531T143193

Summary Delete role

Role ARN: arn:aws:iam::341259728059:role/service-role/AmazonSageMaker-ExecutionRole-20190531T143193

Role description: SageMaker execution role created from the SageMaker AWS Management Console. | Edit

Instance Profile ARNs: /service-role/

Path: /service-role/

Creation time: 2019-05-31 14:31 PDT

Maximum CLI/API session duration: 1 hour Edit

Permissions Trust relationships Tags Access Advisor Revoke sessions

Permissions policies (2 policies applied)

Attach policies Add inline policy

| Policy name | Policy type | |
|---|--------------------|---|
| AmazonSageMakerFullAccess | AWS managed policy | ✕ |
| AmazonSageMaker-ExecutionPolicy-20190531T143193 | Managed policy | ✕ |

Permissions boundary (not set)

- h. **Filter policies** by "Athena", check **AmazonAthenaFullAccess** managed policy and click **Attach Policy** button at the bottom of the screen. Close the new window/tab.

Add permissions to AmazonSageMaker-ExecutionRole-20190531T143193

Attach Permissions

Create policy Refresh

Filter policies Showing 2 results

| Policy name | Type | Used as | Description |
|--|-------------|------------------------|--|
| <input checked="" type="checkbox"/> AmazonAthenaFullAccess | AWS managed | None | Provide full access to Amazon Athena and scoped access to the depende... |
| <input type="checkbox"/> AWSQuicksightAthenaAccess | AWS managed | Permissions policy (1) | Quicksight access to Athena API and S3 buckets used for Athena query re... |

Cancel **Attach policy**

After Athena policy to IAM Role , you can close this window and go back SageMaker browser window.

- i. Leave all other options default. Click **Create notebook instance**.

Permissions and encryption

IAM role
Notebook instances require permissions to call other services including SageMaker and S3. Choose a role or let us create a role with the [AmazonSageMakerFullAccess](#) IAM policy attached.

AmazonSageMaker-ExecutionRole-20190531T143193 ▼

✓ **Success! You created an IAM role.**
[AmazonSageMaker-ExecutionRole-20190531T143193](#) ✕

Root access - optional

☒ **Enable** - Give users root access to the notebook

☐ **Disable** - Don't give users root access to the notebook
 Lifecycle configurations always have root access

Encryption key - optional
 Encrypt your notebook data. Choose an existing KMS key or enter a key's ARN.

No Custom Encryption ▼

► **Network - optional**

► **Git repositories - optional**

► **Tags - optional**

Cancel Create notebook instance

- Wait for the notebook instances to be created and the Status to change to **Inservice** and Click **Open Jupyter** in from **Actions** column.

Amazon SageMaker

- Dashboard
- Search
- Ground Truth
- Labeling jobs
- Labeling datasets
- Labeling workforces
- Notebook**
 - Notebook instances
 - Lifecycle configurations
 - Git repositories

Amazon SageMaker > Notebook instances

Notebook instances

Search notebook instances

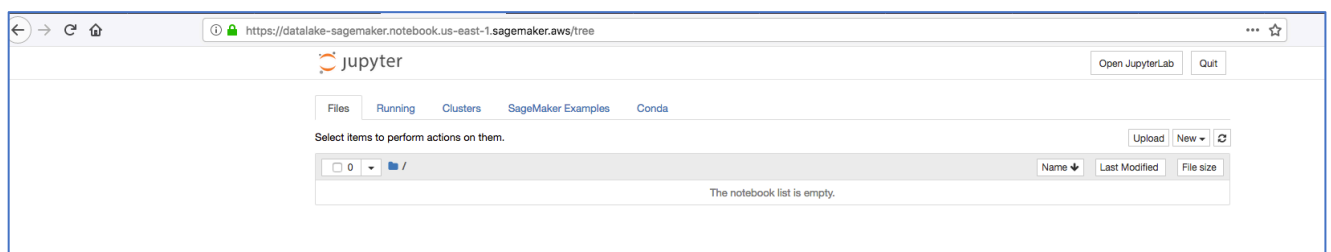
Name: datalake ✕

Actions ▼ Create notebook instance

< 1 > ⓘ

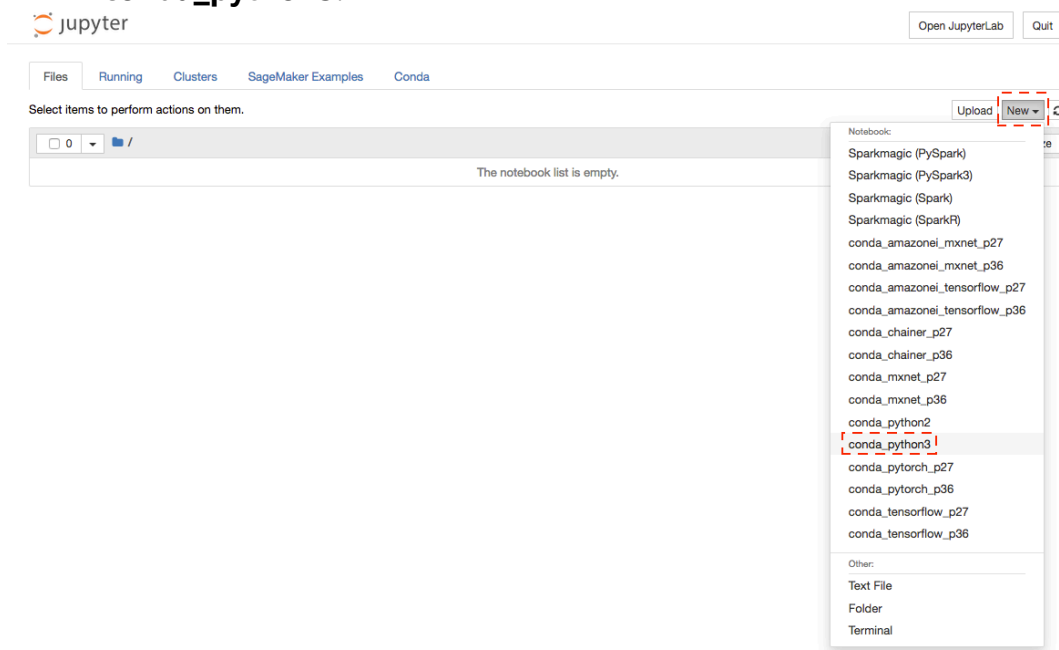
| Name | Instance | Creation time | Status | Actions |
|--------------------|--------------|------------------------|-----------|------------------------------|
| dataLake-Sagemaker | ml.t2.medium | May 31, 2019 21:49 UTC | InService | Open Jupyter Open JupyterLab |

- The notebook interface opens in a separate browser window.



Connect the SageMaker Jupyter notebook to Athena

1. In the Jupyter notebook interface, click **New**. For the kernel, choose **conda_python3**.

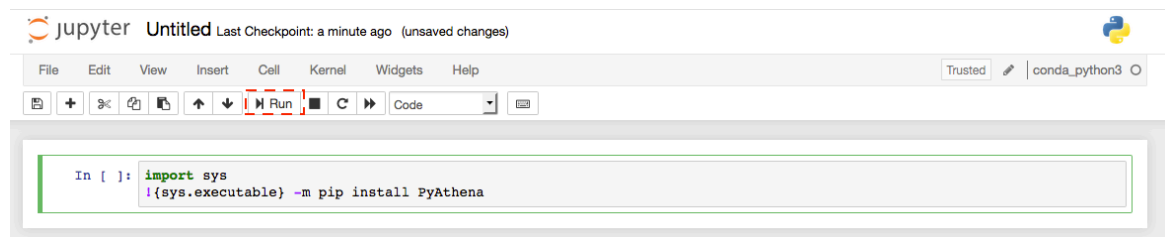


Note: Amazon SageMaker provides several kernels for Jupyter, including support for Python 2 and Python 3, MXNet, TensorFlow, and PySpark. This exercise uses Python because it includes the Pandas library.

2. Within the notebook, execute the following commands to install the Athena JDBC driver and In the top toolbar, click **Run**. (PyAthena is a Python [DB API 2.0 \(PEP 249\)](#) compliant client for the [Amazon Athena JDBC driver](#).)

Enter -> `import sys`

Enter -> `!{sys.executable} -m pip install PyAthena`



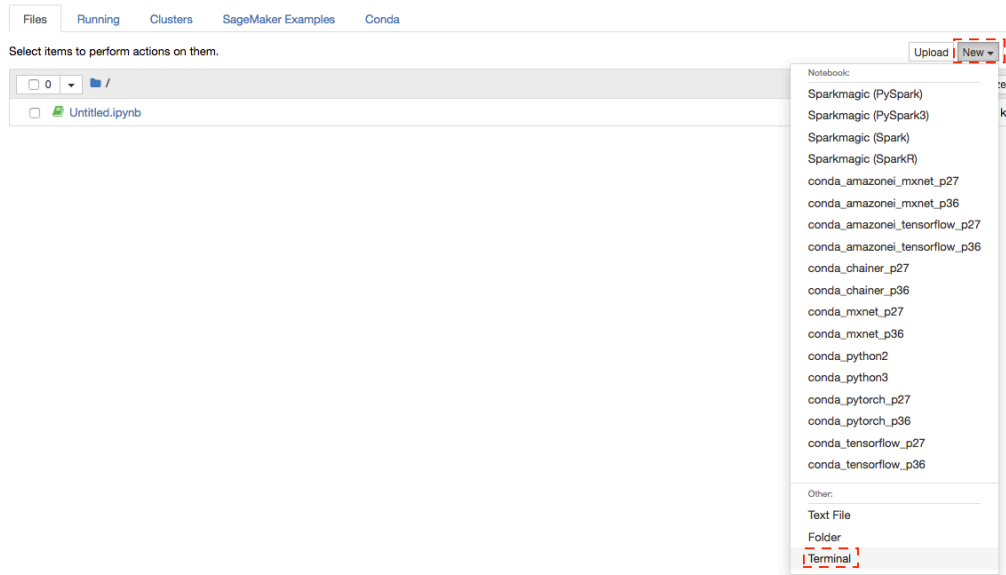
Observe success message in log:


```
In [1]: import sys
        !{sys.executable} -m pip install PyAthena

Collecting PyAthena
  Downloading https://files.pythonhosted.org/packages/32/b6/841b79b42dad604429596bfed6f669131b59328fa30171538d85adf
fba/PyAthena-1.6.1-py2.py3-none-any.whl (50kB)
    100% |#####| 51kB 18.3MB/s ta 0:00:01
Requirement already satisfied: botocore>=1.5.52 in /home/ec2-user/anaconda3/envs/python3/lib/python3.6/site-packages
(from PyAthena) (1.12.146)
Collecting future (from PyAthena)
  Downloading https://files.pythonhosted.org/packages/90/52/e20466b8500a181e1e144fd8305caf2cf475e2f9674e797b222f8105
f5f/future-0.17.1.tar.gz (829kB)
    100% |#####| 829kB 21.3MB/s ta 0:00:01
Collecting tenacity>=4.1.0 (from PyAthena)
  Downloading https://files.pythonhosted.org/packages/6a/93/dfcf5b1b46ab29196274b78dcb69fab5e54b6dc303a7eed90a79194d
277/tenacity-5.0.4-py2.py3-none-any.whl
Requirement already satisfied: boto3>=1.4.4 in /home/ec2-user/anaconda3/envs/python3/lib/python3.6/site-packages (fro
m PyAthena) (1.9.146)
Requirement already satisfied: jmespath<1.0.0,>=0.7.1 in /home/ec2-user/anaconda3/envs/python3/lib/python3.6/site-pac
kages (from botocore>=1.5.52->PyAthena) (0.9.4)
Requirement already satisfied: docutils>=0.10 in /home/ec2-user/anaconda3/envs/python3/lib/python3.6/site-packages (f
rom botocore>=1.5.52->PyAthena) (0.14)
Requirement already satisfied: python-dateutil<3.0.0,>=2.1; python_version >= "2.7" in /home/ec2-user/anaconda3/envs/
python3/lib/python3.6/site-packages (from botocore>=1.5.52->PyAthena) (2.7.3)
Requirement already satisfied: urllib3<1.25,>=1.20; python_version >= "3.4" in /home/ec2-user/anaconda3/envs/python3/
lib/python3.6/site-packages (from botocore>=1.5.52->PyAthena) (1.23)
Requirement already satisfied: six>=1.9.0 in /home/ec2-user/anaconda3/envs/python3/lib/python3.6/site-packages (from
tenacity>=4.1.0->PyAthena) (1.11.0)
Requirement already satisfied: s3transfer<0.3.0,>=0.2.0 in /home/ec2-user/anaconda3/envs/python3/lib/python3.6/site-p
ackages (from boto3>=1.4.4->PyAthena) (0.2.0)
Building wheels for collected packages: future
  Running setup.py bdist_wheel for future ... done
  Stored in directory: /home/ec2-user/.cache/pip/wheels/0c/61/d2/d6b7317325828fbb39ee6ad559dbe4664d0896da4721bf379e
Successfully built future
Installing collected packages: future, tenacity, PyAthena
Successfully installed PyAthena-1.6.1 future-0.17.1 tenacity-5.0.4
```

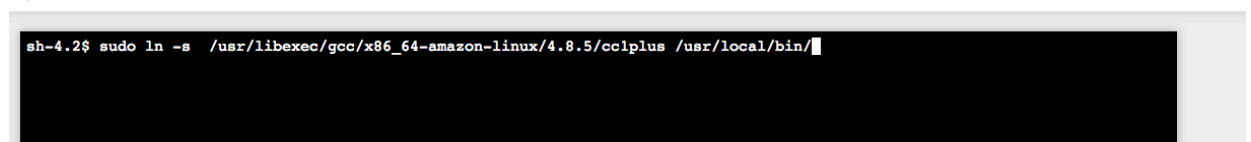
Note: If an error occurs, you may be required to set a file system path for an application that helps build the driver software. If this is the case, follow these steps.

a. In your home notebook window, select **New > Terminal**.



b. Copy following command and press **Enter**.

```
sudo ln -s /usr/libexec/gcc/x86_64-amazon-linux/4.8.5/cc1plus /usr/local/bin/
jupyter
```



c. Run the original paragraph code from Step 2 again to build the AthenaJDBC driver .

Working in Pandas

After the Athena driver is installed, you can use the JDBC connection to connect to Athena and populate the Pandas data frames. For data scientists, working with data is typically divided into multiple stages: ingesting and cleaning data, analyzing and modeling data, then organizing the results of the analysis into a form suitable for plotting or tabular display. Pandas is the ideal tool for all of these tasks.

1. You can load Athena table data from Data lake to Pandas data frame and apply machine learning. Copy following code in your notebook and replace following:
 - a. Account number : Open this link <https://console.aws.amazon.com/billing/home?#/account> and copy account ID.
 - b. Your region : Throughout this lab you are working in **N. Virginia** region which is us-east-1. Refer link here to get region name mapping - <https://docs.aws.amazon.com/general/latest/gr/rande.html>
 - c. Your Athena Database name, this is the Glue Database name which you created during previous lab e.g. "ticketdata"

```
from pyathena import connect
import pandas as pd
conn = connect(s3_staging_dir='s3://aws-athena-query-
results-<youraccountnumber>-<yourregion>/',
               region_name='<yourregion>')

df = pd.read_sql('SELECT * FROM
"<yourathenadatabase>".nfl_stadium_data" order by stadium
limit 10;', conn)
df
```

2. Click **Run** and data frame will display query output.

```
In [3]: from pyathena import connect
import pandas as pd
conn = connect(s3_staging_dir='s3://aws-athena-query-results-[redacted]-us-east-1',
               region_name='us-east-1')

df = pd.read_sql('SELECT * FROM "ticketdata".nfl_stadium_data" order by stadium limit 10;', conn)
df
```

Out[3]:

| | stadium | seating_capacity | location | surface | roof | team | opened | sport_location_id |
|---|-------------------------|------------------------|---------------|-----------------|----------------------------------|-------------|--------|-------------------|
| 0 | AT&T Stadium | +8.000000000000000e+04 | "Arlington | Texas" | Matrix RealGrass artificial turf | Retractable | None | 2009 |
| 1 | Arrowhead Stadium | +7.641600000000000e+04 | "Kansas City | Missouri | Latitude 36 Bermuda Grass | Open | None | 1972 |
| 2 | Bank of America Stadium | +7.541900000000000e+04 | "Charlotte | North Carolina" | Voyager Bermuda Grass | Open | None | 1996 |
| 3 | CenturyLink Field | +6.800000000000000e+04 | "Seattle | Washington" | FieldTurf Revolution | Open | None | 2002 |
| 4 | EverBank Field | +6.724600000000000e+04 | "Jacksonville | Florida" | Tifway 419 Bermuda Grass | Open | None | 1995 |
| 5 | FedExField | +8.200000000000000e+04 | "Landover | Maryland" | Latitude 36 Bermuda Grass | Open | None | 1997 |
| 6 | FirstEnergy Stadium | +6.743100000000000e+04 | "Cleveland | Ohio" | Kentucky Bluegrass | Open | None | 1999 |
| 7 | Ford Field | +6.500000000000000e+04 | "Detroit | Michigan" | FieldTurf Classic HD | Fixed | None | 2002 |
| 8 | Georgia Dome | +7.125000000000000e+04 | "Atlanta | Georgia" | FieldTurf Classic HD | Fixed | None | 1992 |
| 9 | Gillette Stadium | +6.682900000000000e+04 | "Foxborough | Massachusetts" | FieldTurf Revolution | Open | None | 2002 |

In this query , you are loading all nfl stadium information to panda dataframe from table `nfl_stadium_data`.

Note:

if you get a SageMaker does not have Athena execution permissions error issue. You need to add Athena Access to the Sagemaker role as steps provide in previous section.

3. You can use apply different ML algorithm in populated Pandas data frames. For example, draw a plot. Copy following code in your notebook and replace Your Athena Database name, this is the Glue Database name which you created during previous lab e.g. "ticketdata".
In this query , you are loading all even ticket information to panda dataframe from table `sporting_event_ticket_info`.

```
df = pd.read_sql('SELECT sport,          \
                  event_date_time,       \
                  home_team,away_team,   \
                  city,                  \
                  count(*) as tickets,    \
                  sum(ticket_price) as total_tickets_amt, \
                  avg(ticket_price) as avg_ticket_price, \
                  max(ticket_price) as max_ticket_price, \
                  min(ticket_price) as min_ticket_price   \
FROM
"<yourathenadatabase>".sporting_event_ticket_info" \
group by 1,2,3,4,5 \
order by 1,2,3,4,5 limit 1000;', conn)

df
```

4. Click **Run** and data frame will display query output

```
In [15]: df = pd.read_sql('SELECT sport,
                        event_date_time,
                        home_team,away_team,
                        city,
                        count(*) as tickets,
                        sum(ticket_price) as total_tickets_amt,
                        avg(ticket_price) as avg_ticket_price,
                        max(ticket_price) as max_ticket_price,
                        min(ticket_price) as min_ticket_price
FROM "ticketdata"."sporting_event_ticket_info" \
group by 1,2,3,4,5
order by 1,2,3,4,5 limit 1000;', conn)
df
```

```
Out[15]:
```

| | sport | event_date_time | home_team | away_team | city | tickets | total_tickets_amt | avg_ticket_price | max_ticket_price | min_ticket_price |
|---|----------|---------------------|----------------------|---------------------|--------------------------|---------|-------------------|------------------|------------------|------------------|
| 0 | baseball | 2019-04-07 00:00:00 | Chicago Cubs | Milwaukee Brewers | Chicago Illinois | 2792 | 130163.04 | 46.620000 | 77.70 | 31.08 |
| 1 | baseball | 2019-04-07 00:00:00 | Cincinnati Reds | San Diego Padres | Cincinnati Ohio | 2888 | 92322.14 | 31.967500 | 33.65 | 26.92 |
| 2 | baseball | 2019-04-07 00:00:00 | Colorado Rockies | St. Louis Cardinals | Denver Colorado | 3600 | 262764.00 | 72.990000 | 145.98 | 48.66 |
| 3 | baseball | 2019-04-07 00:00:00 | Kansas City Royals | Colorado Rockies | Kansas City Missouri | 2488 | 140024.64 | 56.280000 | 75.04 | 37.52 |
| 4 | baseball | 2019-04-07 00:00:00 | San Francisco Giants | Cleveland Indians | San Francisco California | 2848 | 109505.60 | 38.450000 | 61.52 | 30.76 |
| 5 | baseball | 2019-04-07 00:00:00 | Tampa Bay Rays | New York Yankees | St. Petersburg Florida | 1888 | 91983.36 | 48.720000 | 64.96 | 32.48 |
| 6 | baseball | 2019-04-07 00:00:00 | Texas Rangers | Toronto Blue Jays | Arlington Texas | 3400 | 178415.00 | 52.475000 | 83.96 | 41.98 |

5. In new execution line copy following code

```
import matplotlib.pyplot as plt
df.plot(x='event_date_time',y='avg_ticket_price')
```

6. Click Run and you will see data plot which got draw using matplotlib library.

```
In [17]: import matplotlib.pyplot as plt
df.plot(x='event_date_time',y='avg_ticket_price')
```

```
Out[17]: <matplotlib.axes._subplots.AxesSubplot at 0x7fc1c0b26d8>
```

