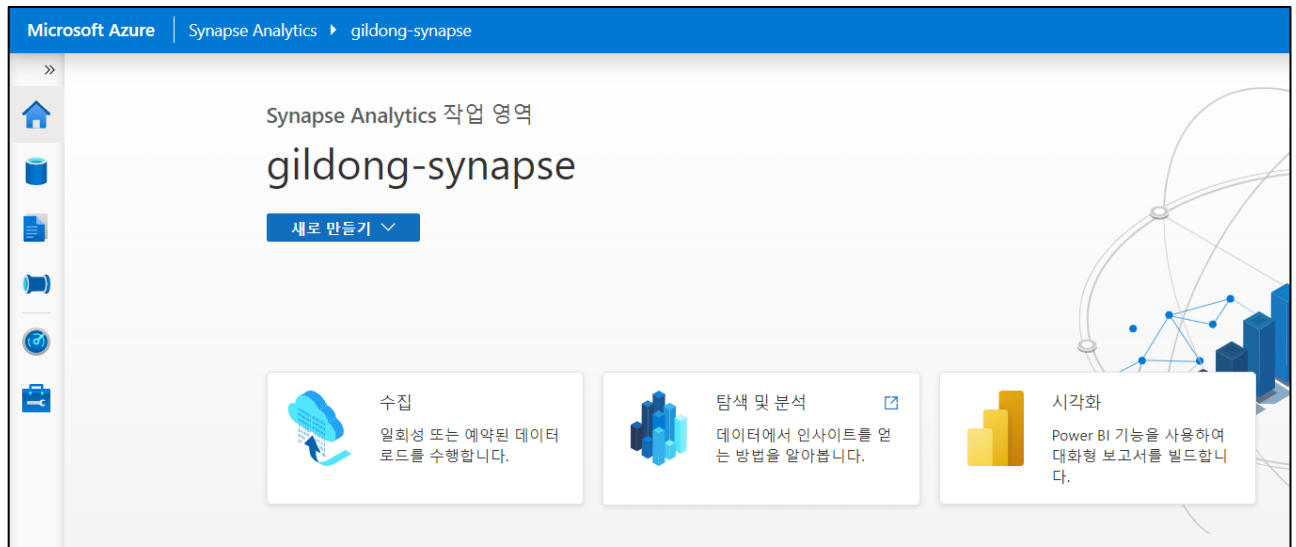


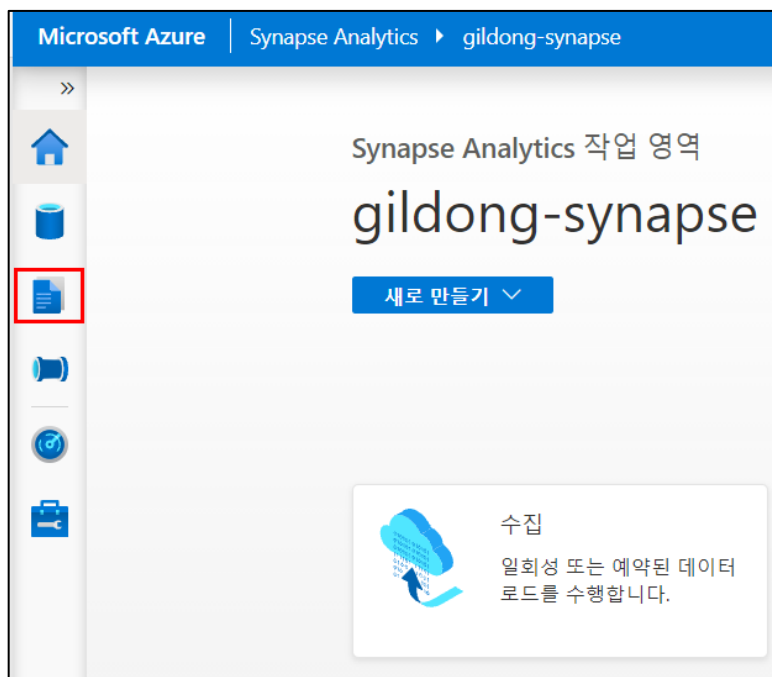
Lab 7 – Synapse Spark

Task 1 : Import Notebook files

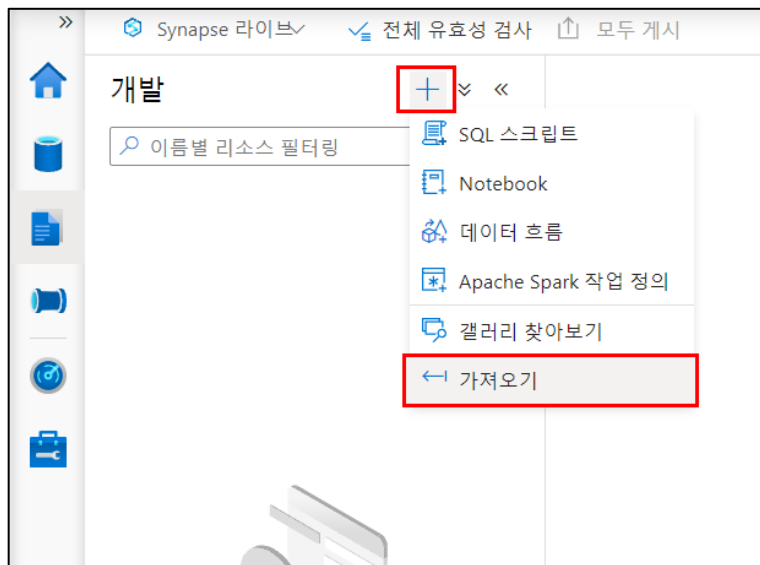
1. 구성해 놓은 **Synapse Analytics** 에 접속합니다.



2. 개발(Develop) Hub 로 이동합니다.



3. 상단의 + 표시를 클릭한 후 드롭다운 메뉴 중 가져오기를 클릭합니다.



4. MZC Azure Synapse/LAB File/Spark Notebook/Demo01 ~ 04 번 파일을 선택하여 파일을 가져옵니다.

이름	수정된 날짜	유형	크기
01_Read_Write_Data_And_Chart	2021-11-01 오후 7:23	IPYNB 파일	660KB
02_01_Create_Spark_Table_With_NYC-Ta...	2021-11-01 오후 7:23	IPYNB 파일	24KB
02_02_NYC_Taxi_Fare_AutoML_Noteboo...	2021-11-01 오후 7:23	IPYNB 파일	11KB
03_Predict_NYC_Taxi_Tips_ONNX	2021-11-01 오후 7:23	IPYNB 파일	60KB
Demo_01_Read write data and chart	2021-11-01 오후 7:23	IPYNB 파일	660KB
Demo_02_IoT Retail Data AutoML	2021-11-01 오후 7:23	IPYNB 파일	5,608KB
Demo_03_Create Spark Table with NYC ...	2021-11-01 오후 7:23	IPYNB 파일	24KB
Demo_04_NYC Taxi Fare AutoML Noteb...	2021-11-01 오후 7:23	IPYNB 파일	11KB
Demo_05_Predict NYC Taxi Tips ONNX	2021-11-01 오후 7:23	IPYNB 파일	60KB

5. 파일들을 확인하고 모두 게시를 눌러 현재까지의 상태(파일 가져오기)를 저장합니다.

Microsoft Azure

Synapse Analytics

gildong-synapse

검색

Synapse 라이브

전체 유효성 검사

모두 게시 4

개발

이름별 리소스 필터링

Notebook 4

Demo_01_Read write data and chart

Demo_02_IoT Retail Data AutoML

Demo_03_Create Spark Table with N...

Demo_04_NYC Taxi Fare AutoML Not...

Demo_01_Read write...

Demo_02_IoT Retail ...

Dem...

모두 실행

실행 취소

게시

개요

시작되지 않음

셀을 실행하기 전에 연결할 Spark 풀을 선택하세요.

Import Library

+ 코드

```

1 from azureml.opendatasets import NycTlcG
2 import pandas as pd
3 from datetime import datetime
4 from dateutil.relativedelta import relat

```

[3] ✓

모두 게시

보류 중인 모든 변경 내용을 라이브 환경에 게시하려고 합니다. [자세한 정보](#)

보류 중인 변경 내용 (4)

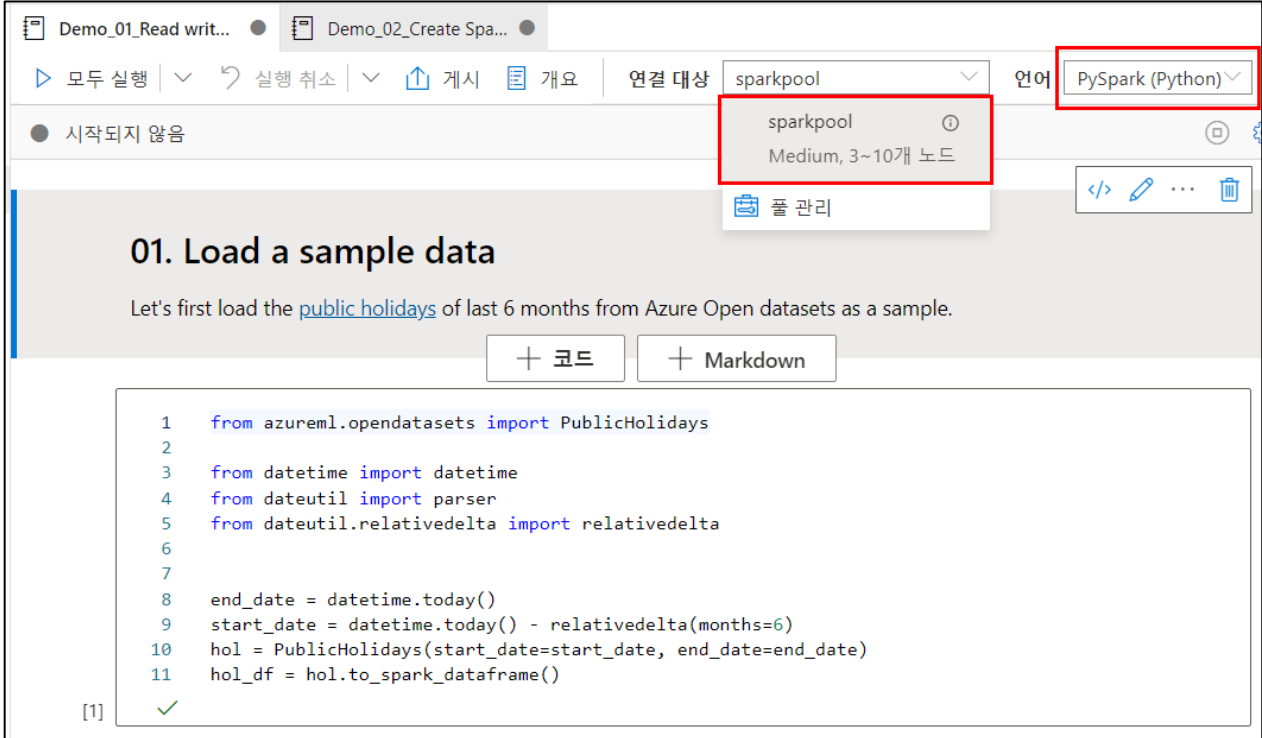
이름	변경	기준
▲ 노트북		
Demo_01_Read write data ... (신규)		-
Demo_02_IoT Retail Data ... (신규)		-
Demo_03_Create Spark Ta... (신규)		-
Demo_04_NYC Taxi Fare A... (신규)		-

게시

취소

Task 2 : Demo 01. Read Write Data & Chart

1. **01 번 파일**을 선택합니다. 연결대상으로 생성한 **spark pool** 을 선택합니다.
언어는 **PySpark** 로 설정합니다.



Demo_01_Read writ... Demo_02_Create Spa...

모두 실행 실행 취소 게시 개요 연결 대상 sparkpool 언어 PySpark (Python)

시작되지 않음

sparkpool
Medium, 3~10개 노드

풀 관리

01. Load a sample data

Let's first load the [public holidays](#) of last 6 months from Azure Open datasets as a sample.

+ 코드 + Markdown

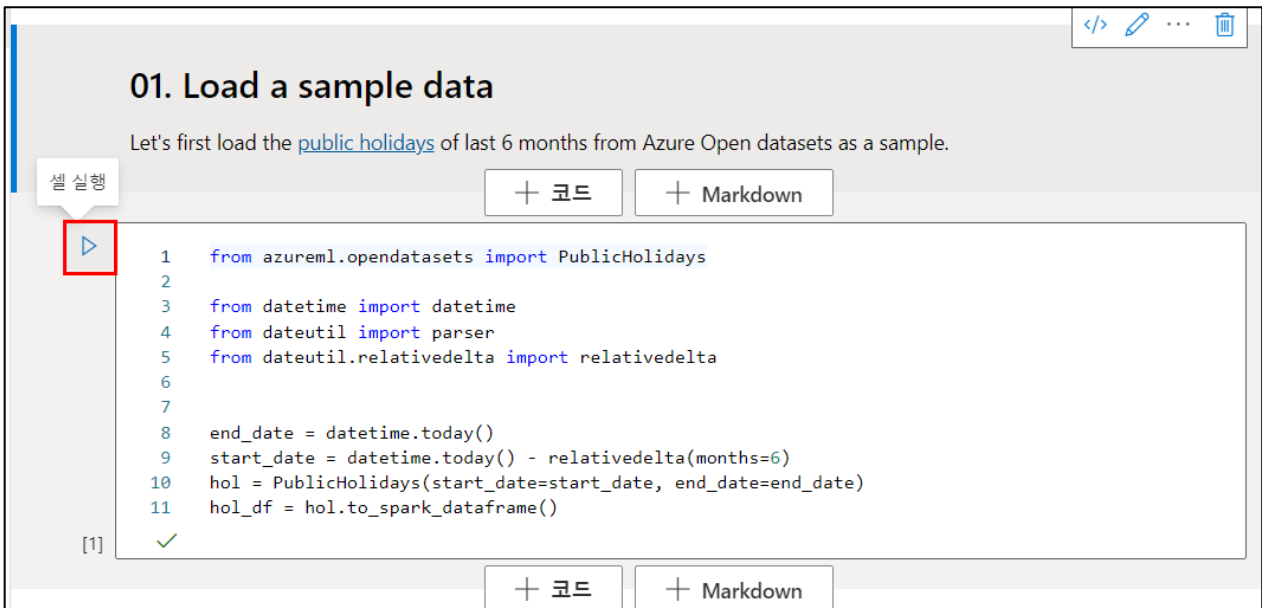
```

1 from azureml.opendatasets import PublicHolidays
2
3 from datetime import datetime
4 from dateutil import parser
5 from dateutil.relativedelta import relativedelta
6
7
8 end_date = datetime.today()
9 start_date = datetime.today() - relativedelta(months=6)
10 hol = PublicHolidays(start_date=start_date, end_date=end_date)
11 hol_df = hol.to_spark_dataframe()

```

[1] ✓

2. **Demo 01** 번 파일을 실행합니다. 각 셀마다 **실행** 단추를 눌러 실행하며 **2 번 셀까지** 실행합니다. Spark pool 세션이 시작되는데 시간이 몇 분 소요될 수 있습니다.



01. Load a sample data

Let's first load the [public holidays](#) of last 6 months from Azure Open datasets as a sample.

+ 코드 + Markdown

실행

```

1 from azureml.opendatasets import PublicHolidays
2
3 from datetime import datetime
4 from dateutil import parser
5 from dateutil.relativedelta import relativedelta
6
7
8 end_date = datetime.today()
9 start_date = datetime.today() - relativedelta(months=6)
10 hol = PublicHolidays(start_date=start_date, end_date=end_date)
11 hol_df = hol.to_spark_dataframe()

```

[1] ✓

+ 코드 + Markdown

3. 3 번 셀에 ADLS **account name**, **container name**, **relative path** 를 아래와 같이 입력합니다.

```

1  from pyspark.sql import SparkSession
2  from pyspark.sql.types import *
3
4  # Primary storage info
5  account_name = '<ADLS Gen2 Account Name>' # fill in your primary account name
6  container_name = 'sparkdemo' # fill in your container name
7  relative_path = 'sparkdemo01/' # fill in your relative folder path
8
9  adls_path = 'abfss://%s@%s.dfs.core.windows.net/%s' % (container_name, account_name, relative_path)
10 print('Primary storage account path: ' + adls_path)

```

[] Press shift + enter to execute cells

Account name : <ADLS Gen2 Account Name>

Container name: sparkdemo

Relative path: sparkdemo01/

4. 5 번 셀까지 실행합니다. **ADLS** 에 데이터가 저장되었는지 확인합니다. **데이터 허브**에서 바로 연결된 **ADLS** 를 조회할 수 있으며, 아래와 같이 **.csv**, **.json**, **.parquet** 형식의 파일이 각 폴더에 생성된 것을 확인합니다.

<div> <div>새 SQL 스크립트</div> <div>새 데이터 흐름</div> <div>새 통합 데이터 세트</div> <div>업로드</div> <div>다운로드</div> <div>새 폴더</div> <div>모두 선택</div> <div>기타</div> </div>		
<div> <div>← → ∨ ↑</div> <div>sparkdemo > sparkdemo01</div> </div>		
이름	마지막으로 수정한 날짜	콘텐츠 형식
<div> <div>📁 holiday.csv</div> <div>2021. 10. 12. 오후 3:43:03</div> <div>Folder</div> </div>		
<div> <div>📁 holiday.json</div> <div>2021. 10. 12. 오후 3:43:00</div> <div>Folder</div> </div>		
<div> <div>📁 holiday.parquet</div> <div>2021. 10. 12. 오후 3:42:49</div> <div>Folder</div> </div>		

<div> <div>새 SQL 스크립트</div> <div>업로드</div> <div>다운로드</div> <div>새 폴더</div> <div>모두 선택</div> <div>이름 바꾸기</div> </div>		
<div> <div>← → ∨ ↑</div> <div>sparkdemo > sparkdemo01 > holiday.csv</div> </div>		
이름	마지막으로 수정한 날짜	
<div> <div>📄 _SUCCESS</div> <div>2021. 10. 12. 오후 3:43:06</div> </div>		
<div> <div>📄 part-00000-3bb39bed-a6ea-4607-97fd-175d68b34807-c000.csv</div> <div>2021. 10. 12. 오후 3:43:06</div> </div>		

5. **6, 7, 8 번 셀**을 실행하여 ADLS 에서 다시 데이터를 읽어와 봅니다.
6. **9, 10, 11 번 셀**을 실행하여 Delta Lake table 을 만들고 저장합니다. **ADLS 에서 Delta 테이블을 확인**합니다.
(설정된 기간에 따라 opendatasets 가 제공하는 데이터 목록이 달라질 수 있습니다.)

새 SQL 스크립트 새 데이터 흐름 새 통합 데이터 세트 업로드 다운로드	
← → ↓ ↑ sparkdemo > sparkdemo01	
이름	마지막으로 수정한 날짜
delta	2021. 10. 12. 오후 3:51:25
holiday.csv	2021. 10. 12. 오후 3:43:03
holiday.json	2021. 10. 12. 오후 3:43:00
holiday.parquet	2021. 10. 12. 오후 3:42:49

← → ↓ ↑ sparkdemo > sparkdemo01 > delta > holiday	
이름	마지막으로 수정한 날짜
_delta_log	2021. 10. 12. 오후 3:51:25
holidayName=Gandhi Jayanti	2021. 10. 12. 오후 3:51:30
holidayName=Independence Day	2021. 10. 12. 오후 3:51:30
holidayName=Labour Day	2021. 10. 12. 오후 3:51:30

7. **12 번 셀**을 실행해서 다시 delta table 을 읽어옵니다.
8. **13 번 셀**을 실행하고 다시 한번 **ADLS** 를 확인합니다. **CountryRegionCode** 가 **JP** 인 데이터로 **Overwrite** 된 결과를 확인합니다.
(설정된 기간에 따라 opendatasets 가 제공하는 데이터 목록이 달라질 수 있습니다.)

새 SQL 스크립트 새 데이터 흐름 새 통합 데이터 세트 업로드 다운로드

sparkdemo > sparkdemo01 > delta > holiday

이름	마지막으로 수정한 날짜
_delta_log	2021. 10. 12. 오후 3:51:25
holidayName=Gandhi Jayanti	2021. 10. 12. 오후 3:51:30
holidayName=Independence Day	2021. 10. 12. 오후 3:51:30
holidayName=Labour Day	2021. 10. 12. 오후 3:51:30
holidayName=こどもの日	2021. 10. 12. 오후 3:58:25
holidayName=みどりの日	2021. 10. 12. 오후 3:58:25
holidayName=体育の日	2021. 10. 12. 오후 3:58:25
holidayName=山の日	2021. 10. 12. 오후 3:58:25
holidayName=憲法記念日	2021. 10. 12. 오후 3:58:25
holidayName=敬老の日	2021. 10. 12. 오후 3:58:25
holidayName=昭和の日	2021. 10. 12. 오후 3:58:25
holidayName=海の日	2021. 10. 12. 오후 3:58:25
holidayName=秋分の日	2021. 10. 12. 오후 3:58:25

9. 14 번 셀을 실행해서 Overwrite 된 버전을 읽어옵니다.

10. 18 번 셀까지 차례대로 실행해보면서 조건에 따라 **Delta table** 을 편집해봅니다.

11. 20 번 셀까지 차례대로 실행합니다. 21 번 셀에 **SQL pool 이름** 을 아래와 같이 입력합니다.

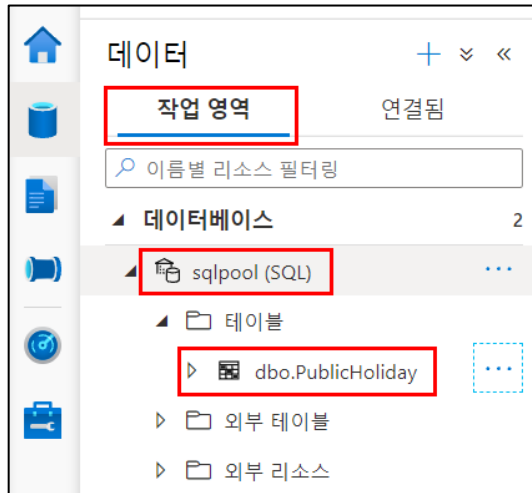
```

1 %%spark
2 // Write the dataframe into your sql pool
3 import org.apache.spark.sql.SqlAnalyticsConnector._
4 import com.microsoft.spark.sqlanalytics.utils.Constants
5
6 val sql_pool_name = "sqlpool" //fill in your sql pool name
7
8 holiday_nodate.write.sqlanalytics(s"$sql_pool_name.dbo.PublicHoliday", Constants.INTERNAL)
9
✓

```

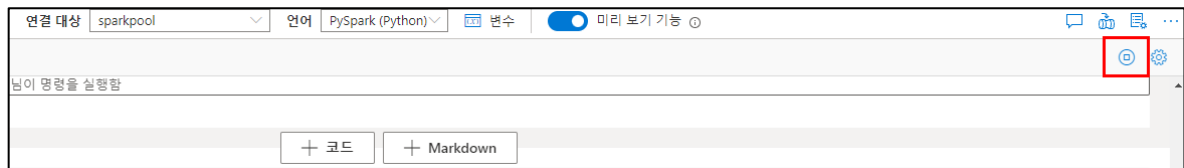
SQL pool name : sqlpool

12. **22 번 셀까지** 차례대로 실행합니다. SQL pool 에 데이터를 저장하고 읽어와 봅니다. 데이터 허브에서 SQL pool 에 테이블이 생성된 것을 확인합니다.



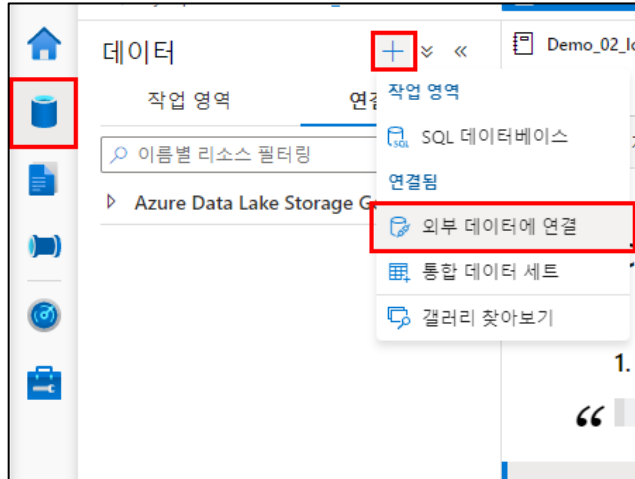
13. **나머지 셀**을 차례대로 실행합니다. 여러 시각화 차트 결과를 확인합니다.

14. 노트북을 닫기 전에 **세션 종료** 버튼을 눌러 세션을 종료합니다.

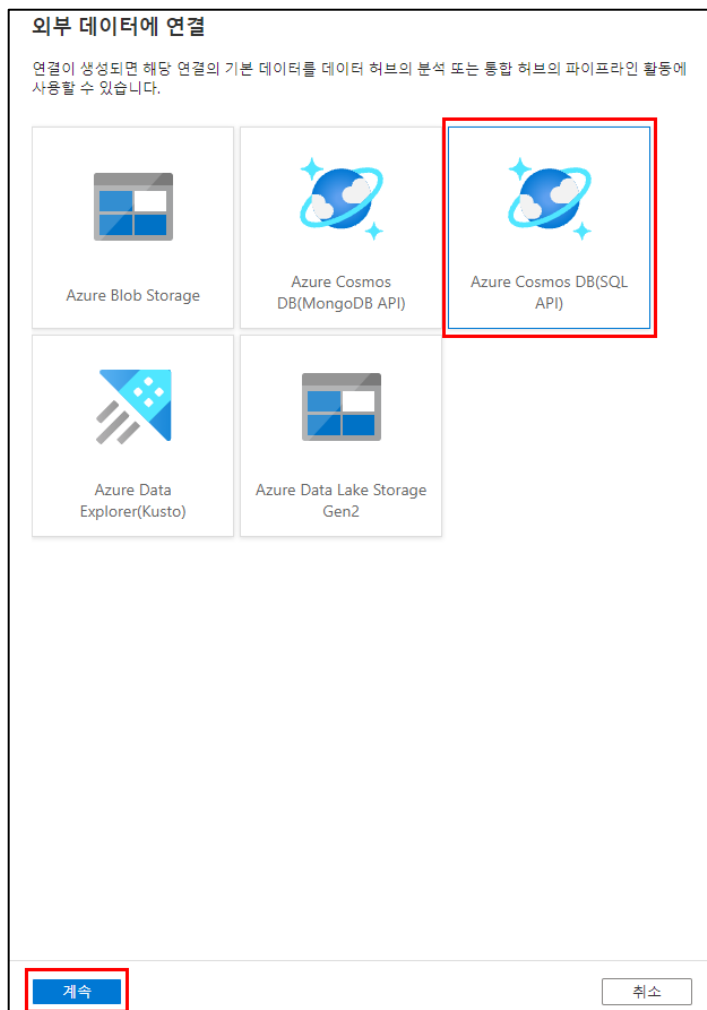


Task 3 : Demo 02. IoT Retail Data + AutoML

1. 데이터 허브로 이동합니다. 상단의 +를 클릭하고 외부 데이터에 연결을 클릭합니다.



2. Azure Cosmos DB(SQL API)를 선택합니다. 계속을 클릭합니다.



3. **Linked Service** 의 이름과 **Cosmos DB** 의 정보를 입력합니다. 연결 테스트를 진행하고 성공적으로 연결이 되는 것을 확인한 후, **만들기**를 클릭합니다.

새 연결된 서비스(Azure Cosmos DB(SQL API))

i 연결된 서비스의 이름을 선택합니다. 이 이름은 나중에 업데이트할 수 없습니다.

이름 *

설명

통합 런타임을 통해 연결 * ①

인증 방법

연결 문자열 Azure Key Vault

계정 선택 방법 ①

☒ Azure 구독에서 ☐ 수동으로 입력

Azure 구독 ①

Azure Cosmos DB 계정 이름 * ①

데이터베이스 이름 *

추가 연결 속성

+ 새로 만들기

주석

+ 새로 만들기

▶ 매개 변수

▶ 고급 ①

만들기 뒤로 연결 테스트 취소

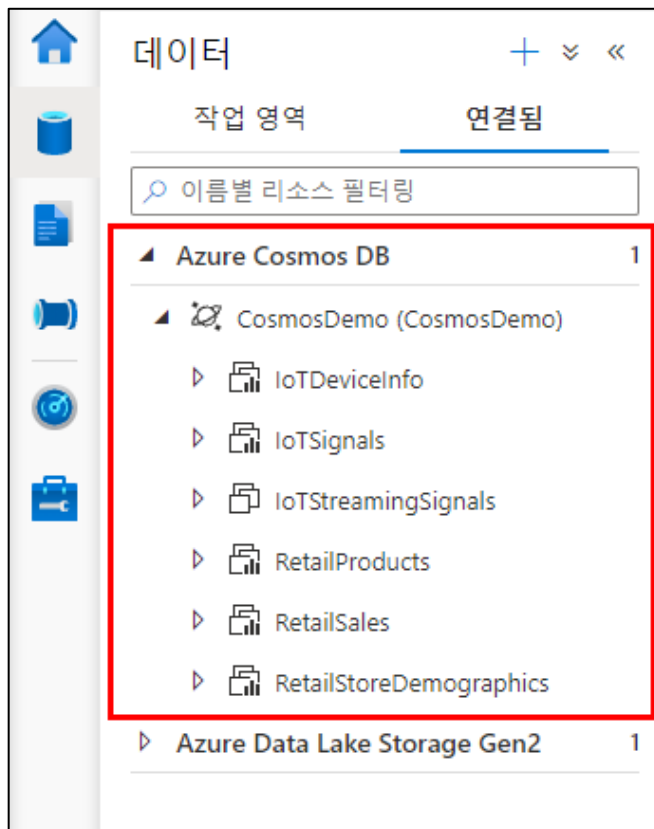
이름 : CosmosDemo

Azure 구독 : Cosmos DB 를 만든 구독 선택(현재 구독)

Azure Cosmos DB 계정 이름 : <ADLS Gen2 Account Name>

데이터베이스 이름 : CosmosDemo

4. **Refresh** 하여 생성된 Link 를 확인합니다.



5. **Demo 02** 번 파일도 Demo 01 파일과 같은 단계를 거쳐서 실행합니다.
연결대상에 spark pool 을 할당하고 실행하면서 결과를 조회합니다.



6. 2 번 셀까지 ADLS Gen2 Account 를 변경합니다.

```

1 dfDeviceInfo = (spark
2     .read
3     .csv('abfss://cosmosdemo@gildongadls.dfs.core.windows.net/SynapseDemoIoT/IoTDeviceInfo.csv', header=True)
4 )
5
6 dfSignals = (spark
7     .read
8     .csv('abfss://cosmosdemo@gildongadls.dfs.core.windows.net/SynapseDemoIoT/IoTSignals.csv', header=True)
9 )
10
11 dfProducts = (spark
12     .read
13     .csv('abfss://cosmosdemo@gildongadls.dfs.core.windows.net/SynapseDemoRetail/Products.csv', header=True)
14 )
15
16 dfRetailSales = (spark
17     .read
18     .csv('abfss://cosmosdemo@gildongadls.dfs.core.windows.net/SynapseDemoRetail/RetailSales.csv', header=True)
19 )
20
21 dfStoreDemographics = (spark
22     .read
23     .csv('abfss://cosmosdemo@gildongadls.dfs.core.windows.net/SynapseDemoRetail/StoreDemoGraphics.csv', header=True)
24 )
25

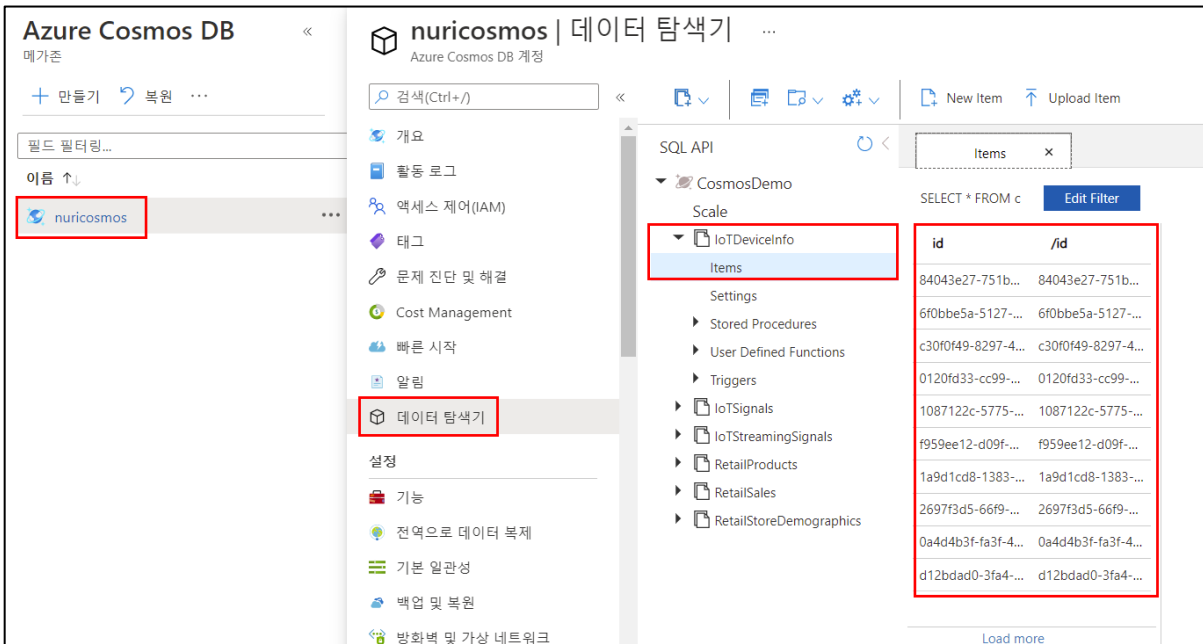
```

abfss://cosmosdemo@<ADLS Gen2 Account Name>.

dfs.core.windows.net/SynapseDemoIoT/IoTDeviceInfo.csv

7. 4 번 셀까지 순차적으로 실행합니다. Cosmos DB 의 데이터 탐색기 혹은 Storage Explorer 에서 들어온 데이터를 확인합니다. IoTDeviceInfo, IoTSignals, RetailProducts, RetailSales, RetailStoreDemographics 에 들어온 데이터를 확인합니다.

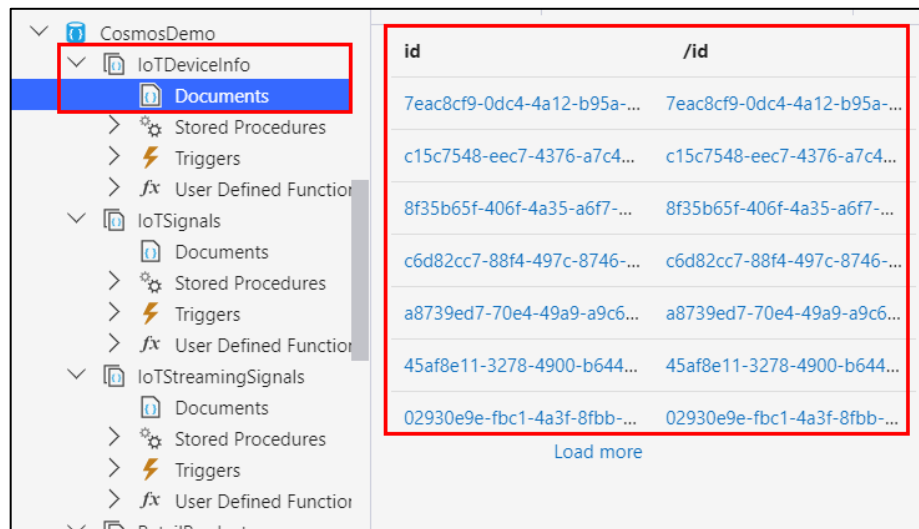
[Azure Portal Cosmos DB 의 데이터 탐색기]



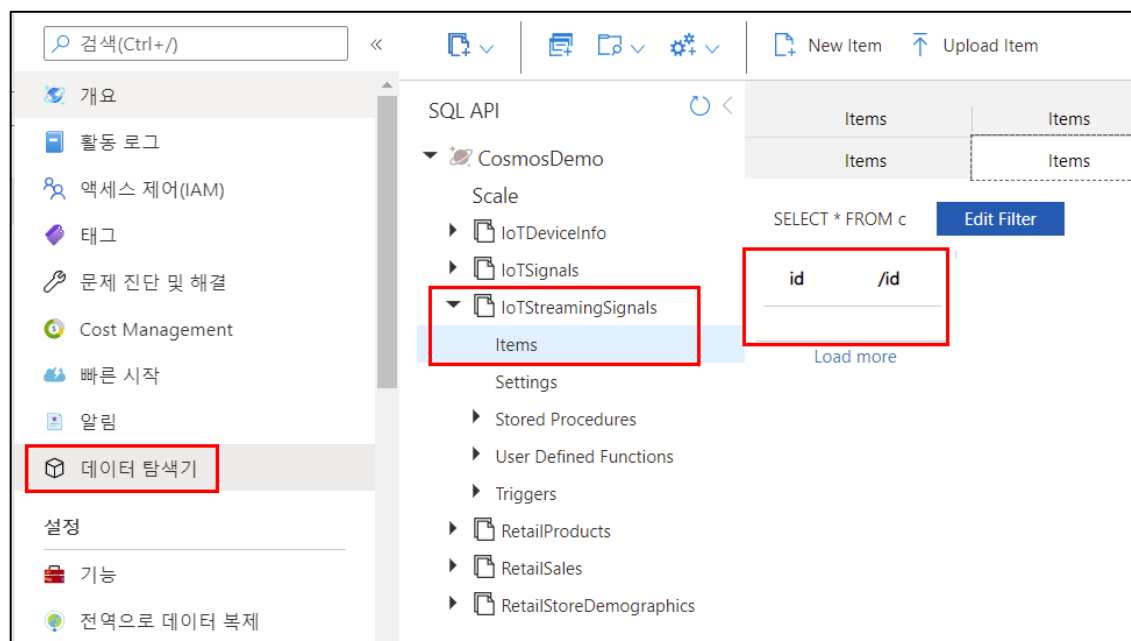
The screenshot shows the Azure Cosmos DB Data Explorer interface. On the left, the 'nuricosmos' database is selected. The 'IoTDeviceInfo' collection is highlighted in the left sidebar. The main pane displays a table of data with columns 'id' and '/id'. The data is as follows:

id	/id
84043e27-751b...	84043e27-751b...
6f0bbe5a-5127-...	6f0bbe5a-5127-...
c30f0f49-8297-4...	c30f0f49-8297-4...
0120fd33-cc99-...	0120fd33-cc99-...
1087122c-5775-...	1087122c-5775-...
f959ee12-d09f-...	f959ee12-d09f-...
1a9d1cd8-1383-...	1a9d1cd8-1383-...
2697f3d5-66f9-...	2697f3d5-66f9-...
0a4d4b3f-fa3f-4...	0a4d4b3f-fa3f-4...
d12bdad0-3fa4-...	d12bdad0-3fa4-...

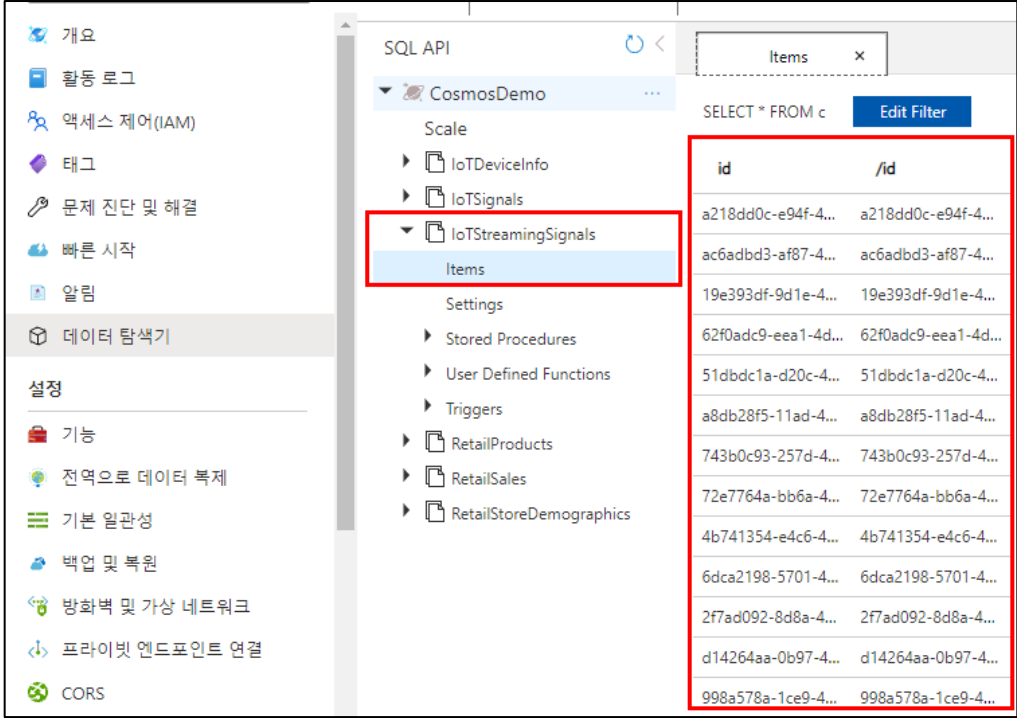
[Storage Explorer]



8. 5, 6 번 셀을 실행하여 Streaming 데이터 Insert 를 준비합니다. Cosmos DB 의 IoTStreamingSignal 컨테이너의 비어있는 데이터를 확인합니다.

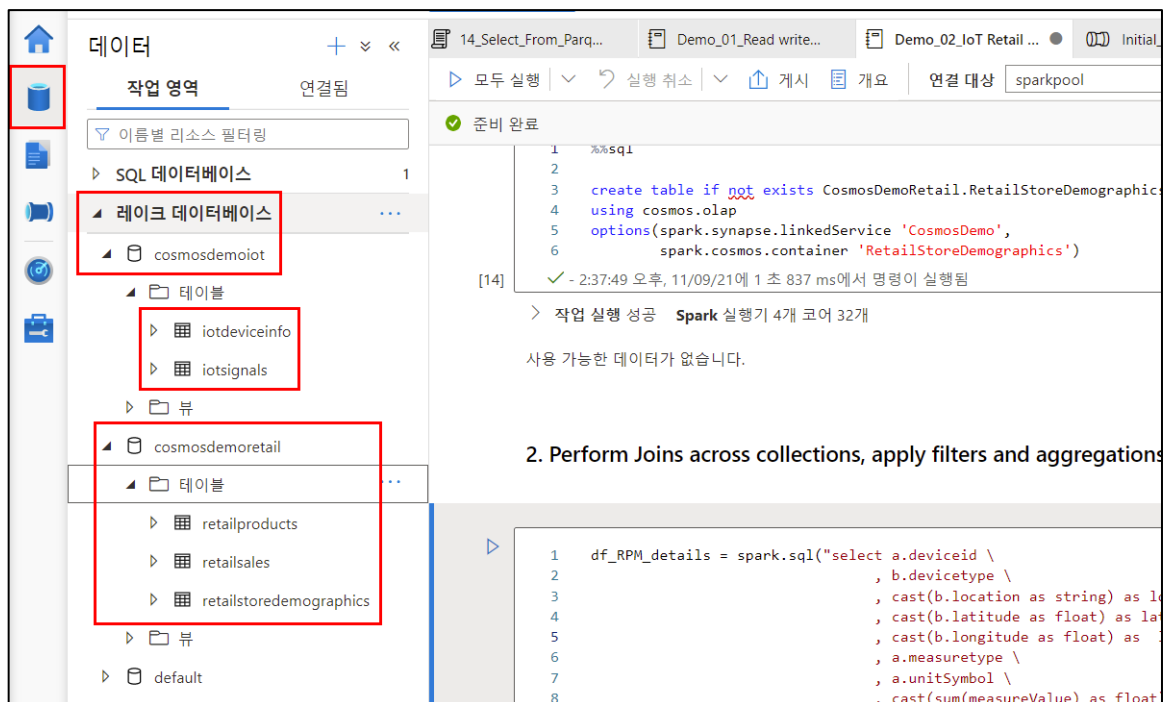


9. 7 번 셀을 실행하고 바로 Cosmos DB 의 데이터를 확인합니다. 초당 10 개씩 데이터가 Insert 됩니다. 약 2 분정도 작업이 실행됩니다.



id	/id
a218dd0c-e94f-4...	a218dd0c-e94f-4...
ac6adb3d-af87-4...	ac6adb3d-af87-4...
19e393df-9d1e-4...	19e393df-9d1e-4...
62f0adc9-eea1-4d...	62f0adc9-eea1-4d...
51dbdc1a-d20c-4...	51dbdc1a-d20c-4...
a8db28f5-11ad-4...	a8db28f5-11ad-4...
743b0c93-257d-4...	743b0c93-257d-4...
72e7764a-bb6a-4...	72e7764a-bb6a-4...
4b741354-e4c6-4...	4b741354-e4c6-4...
6dca2198-5701-4...	6dca2198-5701-4...
2f7ad092-8d8a-4...	2f7ad092-8d8a-4...
d14264aa-0b97-4...	d14264aa-0b97-4...
998a578a-1ce9-4...	998a578a-1ce9-4...

10. 8~14 번 셀을 실행합니다. 데이터베이스와 테이블을 생성합니다. 데이터 허브에서 결과를 확인합니다.



```

1  %%sql
2
3  create table if not exists CosmosDemoRetail.RetailStoreDemographics
4  using cosmos.olap
5  options(spark.synapse.linkedService 'CosmosDemo',
6         spark.cosmos.container 'RetailStoreDemographics')

```

[14] ✓ - 2:37:49 오후, 11/09/21에 1 초 837 ms에서 명령이 실행됨

> 작업 실행 성공 Spark 실행기 4개 코어 32개

사용 가능한 데이터가 없습니다.

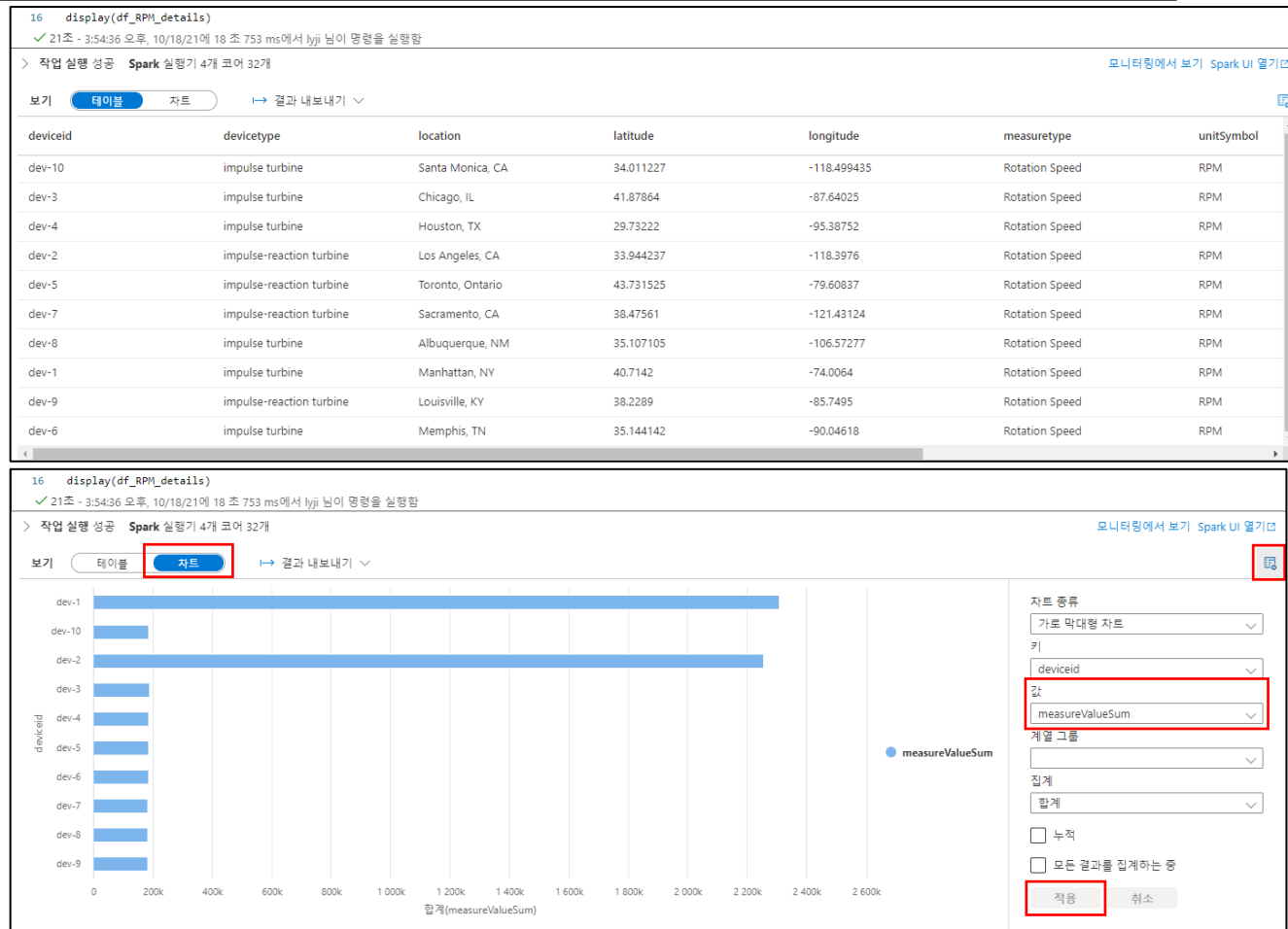
2. Perform Joins across collections, apply filters and aggregations

```

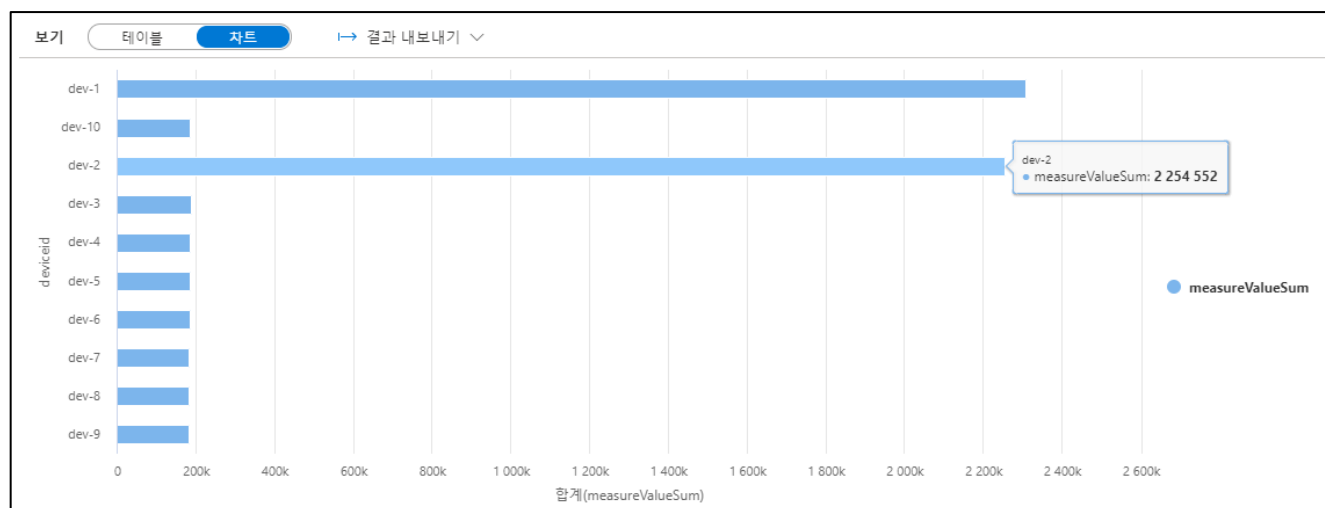
1  df_RPM_details = spark.sql("select a.deviceid \
2                                , b.devicetype \
3                                , cast(b.location as string) as location \
4                                , cast(b.latitude as float) as latitude \
5                                , cast(b.longitude as float) as longitude \
6                                , a.measuretype \
7                                , a.unitSymbol \
8                                , cast(sum(measureValue) as float) as measureValueSum")

```

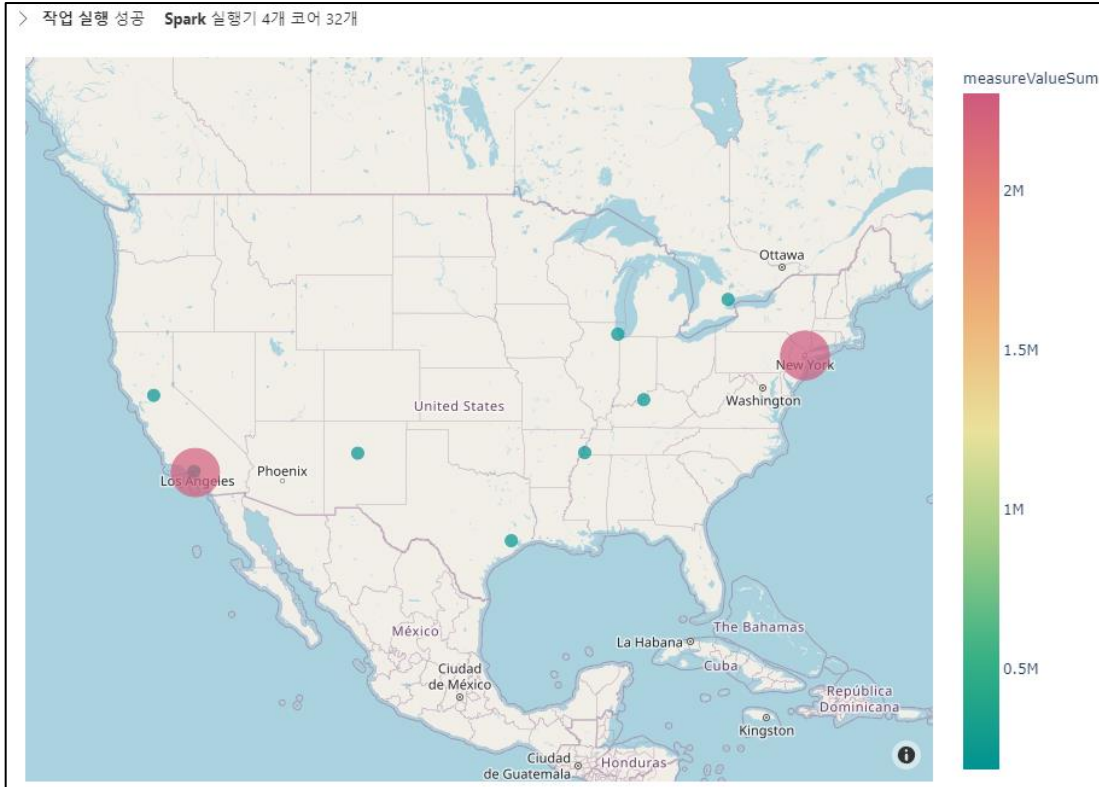
11. 15 번 셀을 실행합니다. 결과를 확인합니다. 보기 옵션을 차트로 변경하고 설정 창에서 값을 **measureValueSum** 으로 변경합니다. 적용을 클릭합니다. 손쉽게 조회 결과를 차트로 변환해볼 수 있습니다.



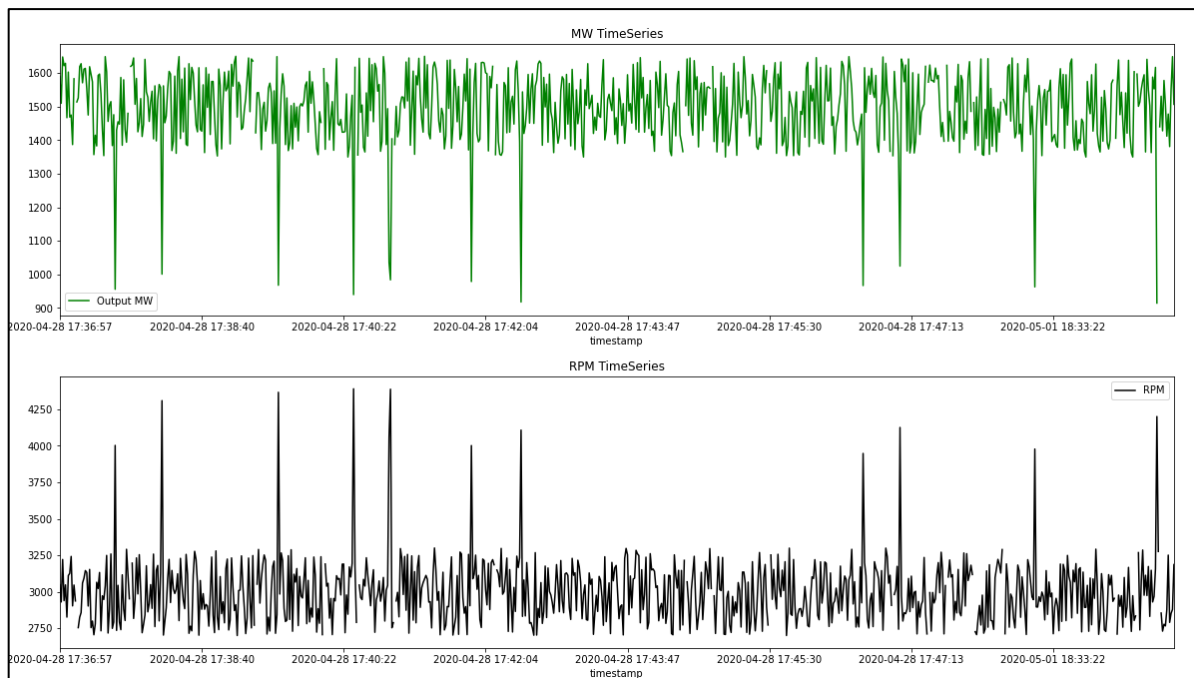
12. 조회결과, **Device ID 1 번과 2 번**의 수치가 높은 것을 볼 수 있습니다.



13. **16 번 셀**을 실행하여 지리적 분포를 확인해봅니다. **뉴욕과 LA**에서 수치가 높게 나타납니다.

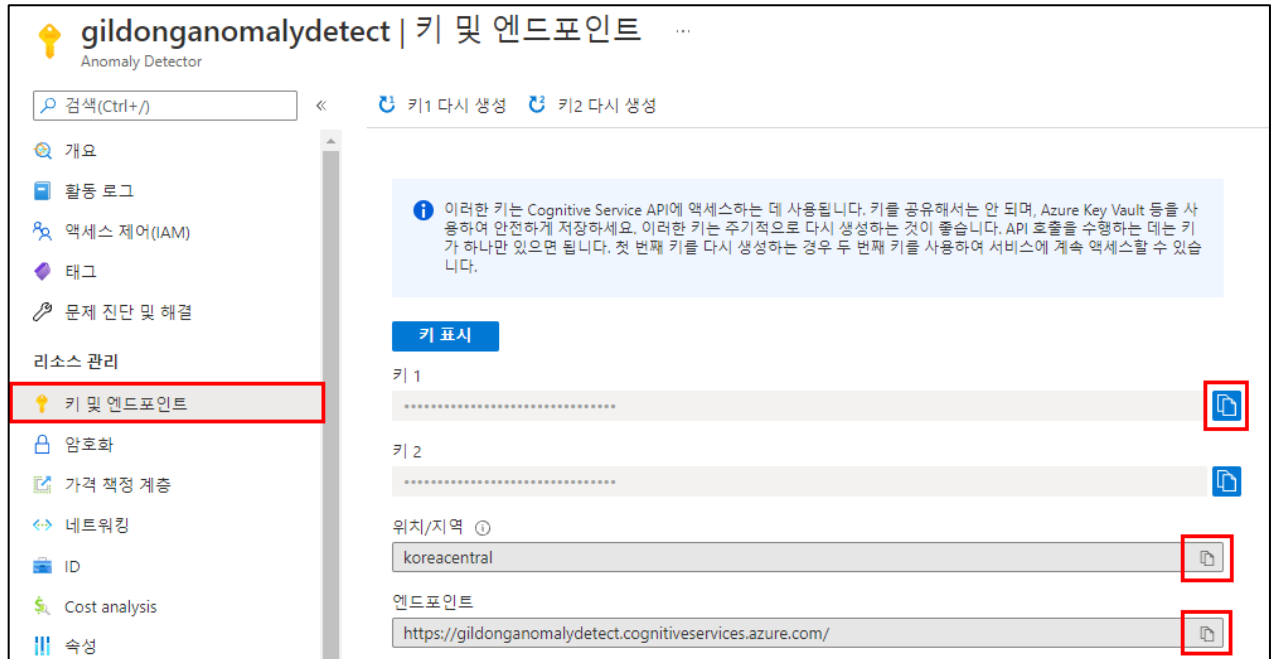


14. 17, 18 번 셀을 실행합니다. 수치가 높게 나온 디바이스 중 하나인 **Dev-1** 에 대해 **unitSymbol** 을 기준으로 **MW** 와 **RPM** 에 **Data** 를 확인합니다.



15. 19 번 셀을 실행하기 전, Anomaly Detector 부분을 적절하게 수정해야 합니다.

Anomaly Detector 로 이동합니다. 키 및 엔드포인트에서 키와 위치/지역, 엔드포인트를 복사해서 아래와 같이 셀을 수정합니다.



6. Perform anomaly detection using Microsoft Machine Learning for Spark (MMLSpark)

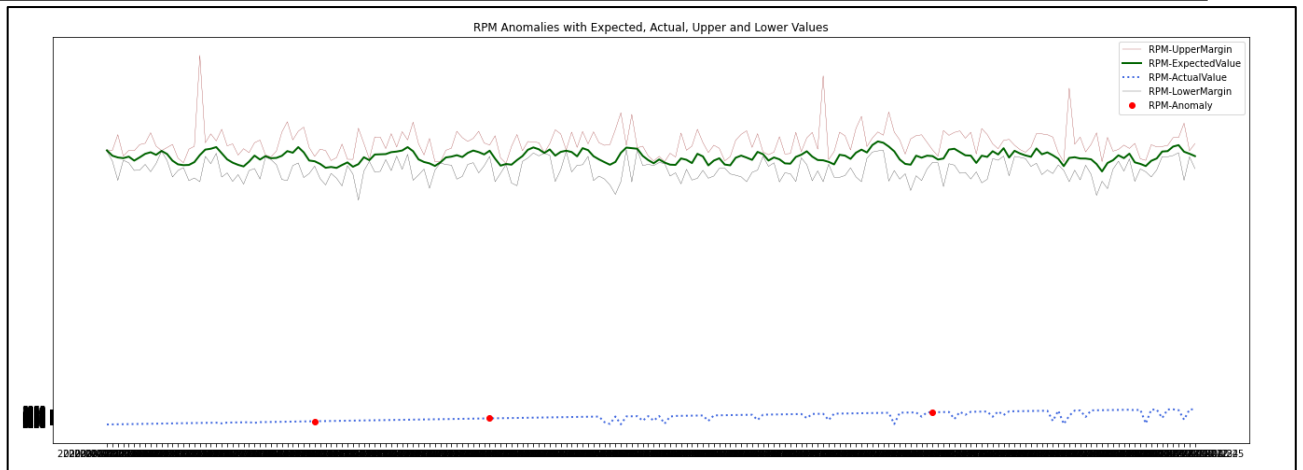
```
1 from pyspark.sql.functions import col
2 from mmlspark.cognitive import SimpleDetectAnomalies
3 from mmlspark.core.spark import FluentAPI
4
5 anomaly_detector = (SimpleDetectAnomalies()
6
7     .setSubscriptionKey("3fb12f31baed41e29cf85be3b5a6b056")
8     .setUrl("https://gildonganomalydetect.cognitiveservices.azure.com/anomalydetector/v1.0/timeseries/entire/detect")
9     .setLocation("koreacentral")
10    .setOutputCol("anomalies")
11    .setGroupbyCol("grouping")
12    .setSensitivity(95)
13    .setGranularity("secondly"))
```

셀 수정에 주의가 필요합니다.

setUrl("<엔드포인트>anomalydetector/v1.0/timeseries/entire/detect")

16. 19, 20 번 셀을 실행합니다. Anomaly Detector 로부터 받은 결과를 조회합니다.

21 번 셀을 실행하여 결과를 시각화합니다. 결과는 아래와 같습니다.



17. 다음은 Retail 데이터를 갖고 Machine Learning 을 해보겠습니다. 22 번 셀을 실행하여 Retail 테이블을 조인합니다. 새로운 데이터셋을 생성합니다.

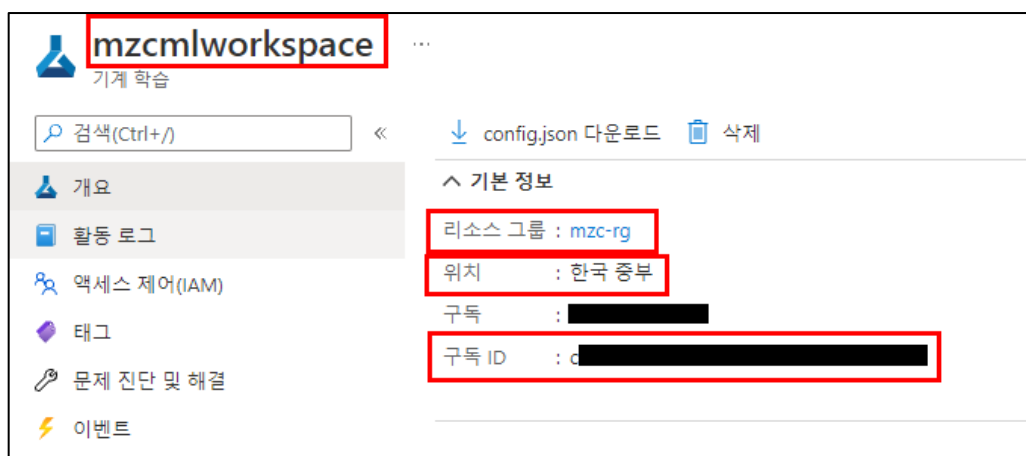
28 display(data)
✓ 21초 - 4:24:19 오후, 10/18/21에 18 초 739 ms에서 lyj 님이 명령을 실행함

> 작업 실행 성공 Spark 실행기 4개 코어 32개

보기 **테이블** 차트 → 결과 내보내기

storeId	productCode	wholeSaleCost	basePrice	ratioAge60	collegeRatio	income
100	surface.go	220.55	499.99	0.136995144	0.049550286	10.0365751
100	surface.laptop3	623.15	1199.99	0.136995144	0.049550286	10.0365751
100	surface.pro7	400.83	899.99	0.136995144	0.049550286	10.0365751
101	surface.go	220.55	499.99	0.225035218	0.174741859	10.65993812
101	surface.laptop3	623.15	1199.99	0.225035218	0.174741859	10.65993812
101	surface.pro7	400.83	899.99	0.225035218	0.174741859	10.65993812
102	surface.go	220.55	499.99	0.216626232	0.120657539	10.49385422
102	surface.laptop3	623.15	1199.99	0.216626232	0.120657539	10.49385422
102	surface.pro7	400.83	899.99	0.216626232	0.120657539	10.49385422
103	surface.go	220.55	499.99	0.058053966	0.194621102	10.58524512
103	surface.laptop3	623.15	1199.99	0.058053966	0.194621102	10.58524512

18. 23 번 셀을 실행하기 전 Azure ML 로 이동합니다. 구독 ID, 리소스 그룹, 워크스페이스 이름, 지역 정보를 보고 셀에 입력합니다.



The screenshot shows the Azure ML workspace configuration page for 'mzcmllworkspace'. The workspace name is highlighted in red. The '기본 정보' (Basic Information) section contains the following details:

- 리소스 그룹 (Resource Group): mzc-rg
- 위치 (Location): 한국 중부 (Central US)
- 구독 (Subscription): [Redacted]
- 구독 ID (Subscription ID): c-[Redacted]

```
import azureml.core
import pandas as pd
import numpy as np
import logging
from azureml.core.workspace import Workspace
from azureml.core import Workspace
from azureml.core.experiment import Experiment
from azureml.train.automl import AutoMLConfig
import os
subscription_id = os.getenv("SUBSCRIPTION_ID", default="")
resource_group = os.getenv("RESOURCE_GROUP", default="mzc-rg")
workspace_name = os.getenv("WORKSPACE_NAME", default="mzcmlworkspace")
workspace_region = os.getenv("WORKSPACE_REGION", default="Korea Central")
```

Resource Group: mzc-rg

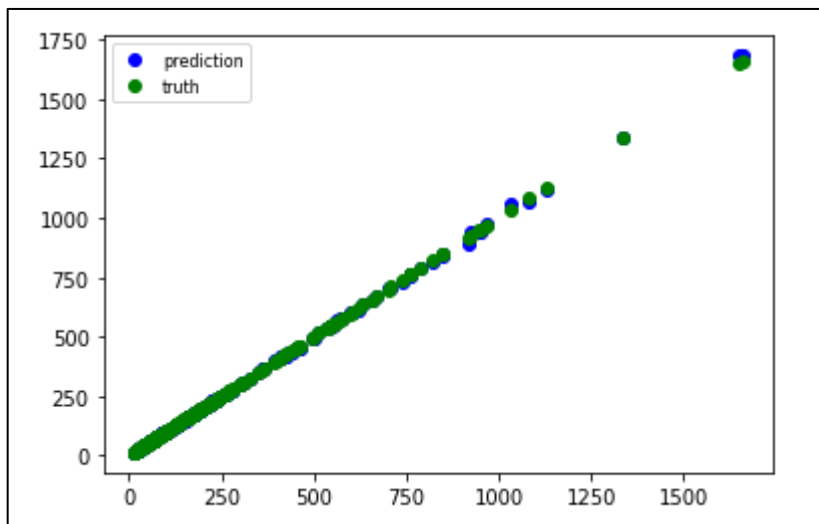
Workspace Name: mzcmlworkspace

Workspace Region: Korea Central

19. 정보를 입력한 후 23 번 셀을 실행합니다. 24 번 셀을 실행하여 Data Prep 을 진행하고 25 번 셀을 실행하여 모델을 빌드합니다. 모델을 만드는 데 시간이 15 분 정도 소요될 수 있습니다.

20. 26 번 셀을 실행하여 Test Data 로 Prediction 을 수행합니다.

21. 27 번 셀을 실행하여 결과를 Plotting 합니다.



22. 28, 29 번 셀을 실행하여 모델을 컨테이너로 배포합니다. 컨테이너가 새로 생성됩니다.

<input type="checkbox"/>	16d22efe9dc942d490d482beeffb4588	컨테이너 레지스트리
<input type="checkbox"/>	gildong-synapse	Synapse 작업 영역
<input type="checkbox"/>	gildongadls	스토리지 계정
<input type="checkbox"/>	gildonganomalydetect	Anomaly Detector
<input type="checkbox"/>	gildongcosmos	Azure Cosmos DB 계정
<input type="checkbox"/>	mzcmllworkspace	기계 학습
<input type="checkbox"/>	mzcmllworkspace2243890366	Application Insights
<input type="checkbox"/>	mzcmllworkspace2409213624	키 자격 증명 모음
<input type="checkbox"/>	mzcmllworkspace2774648598	스토리지 계정
<input type="checkbox"/>	scoring-service--i7SFsmd1EKQ1IK_7-tFiA	Container Instances
<input type="checkbox"/>	sparkpool(gildong-synapse/sparkpool)	Apache Spark 풀


scoring-service--i7SFsmd1EKQ1IK_7-tFiA

Container Instances

시작
다시 시작
중지
삭제
새로 고침

개요

활동 로그

액세스 제어(IAM)

태그

설정

컨테이너

ID

기본 정보

리소스 그룹 (이동) : mzc-rg
상태 : 실행 중
위치 : 한국 중부
구독 (이동) : XXXXXXXXXX
구독 ID : XXXXXXXXXX
태그 (편집) : name : scoring EmittingService : Machine Learning service

23. 나머지 셀들을 실행하여 데이터베이스와 테이블을 정리합니다.

24. 노트북을 닫기 전에 세션 종료 버튼을 눌러 세션을 종료합니다.

연결 대상 sparkpool 언어 PySpark (Python) 변수 미리 보기 가능

님이 명령을 실행함

+ 코드

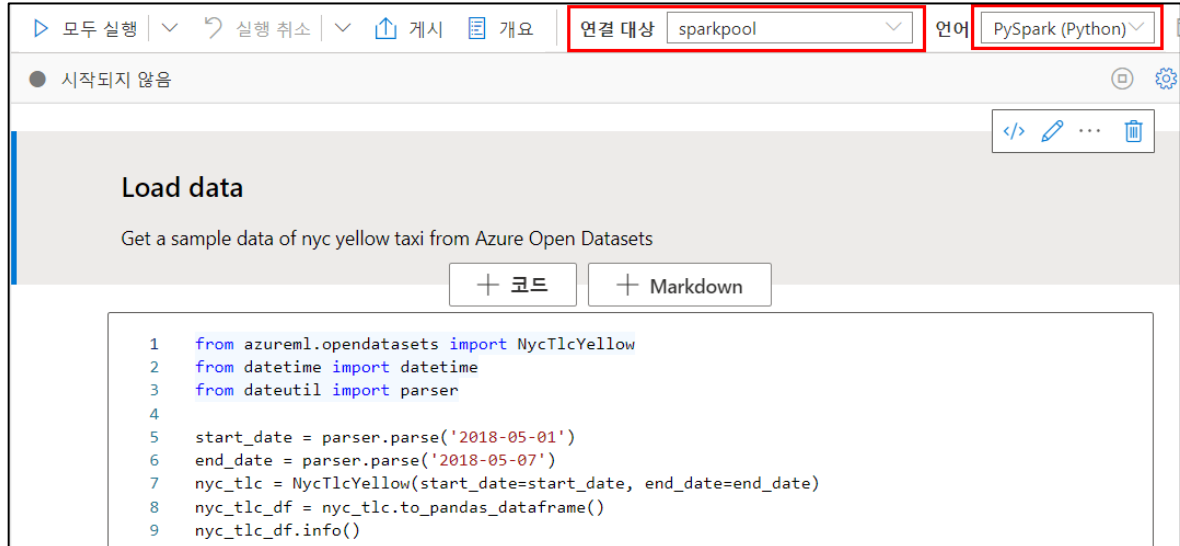
+ Markdown

⊞

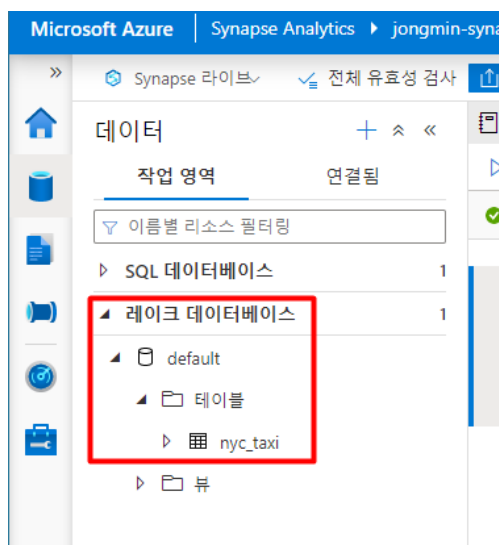
⚙

Task 4 : Demo 03. Create Spark Table with NYC Taxi Data + AutoML

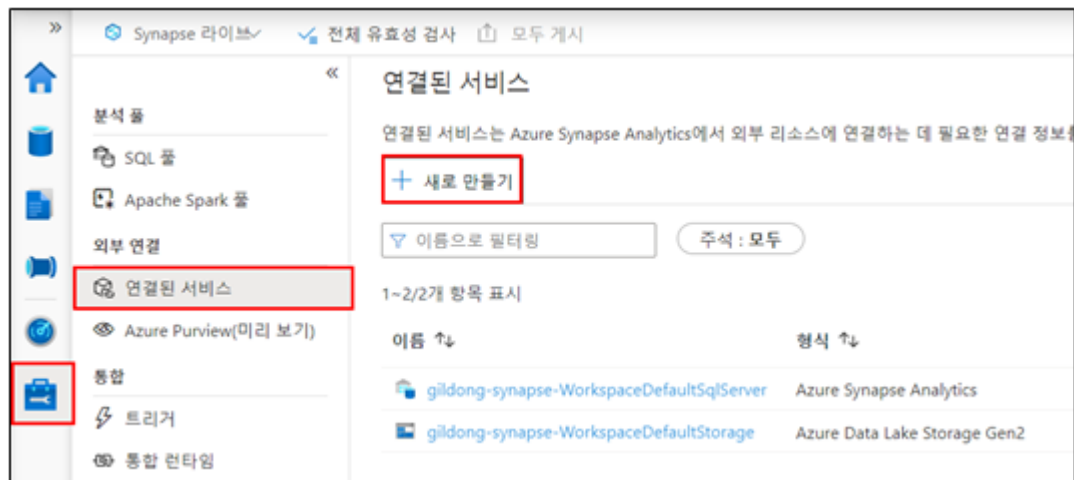
1. **Demo 03** 번 파일도 같은 단계를 거쳐서 실행합니다. 연결대상에 spark pool 을 할당하고 실행하면서 결과를 조회합니다.



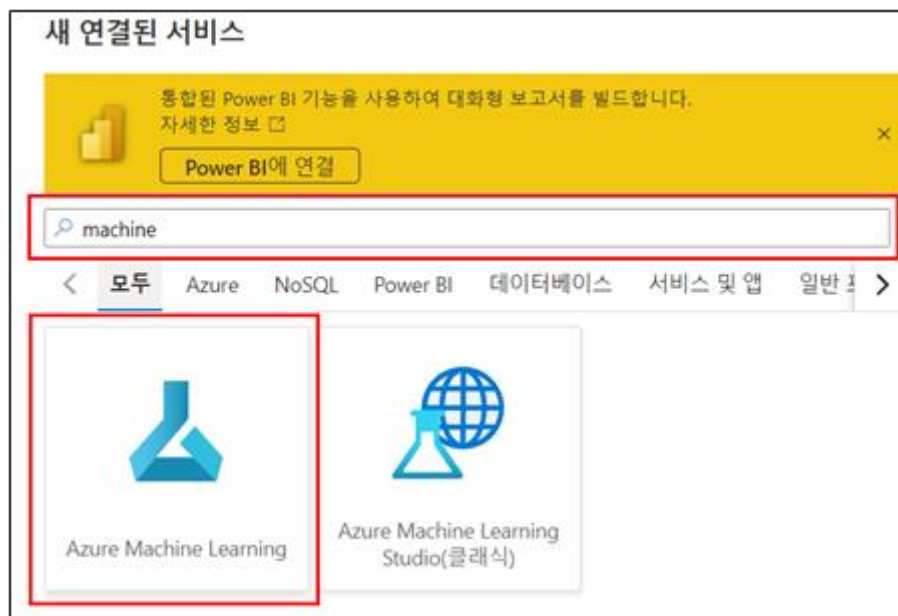
2. 2 번 셀까지 차례대로 실행하면서 데이터를 로드합니다.
3. 4 번 셀까지 차례대로 실행하면서 데이터를 전처리합니다.
4. 5 번 셀을 실행합니다. **Spark table** 로 데이터를 저장합니다. 데이터 허브에서 확인합니다.



5. 관리 허브로 이동합니다. 연결된 서비스로 이동하여 +새로 만들기를 클릭합니다.



6. **Azure Machine Learning** 을 검색하여 선택합니다. **계속**을 눌러 **연결된 서비스로** 등록합니다. **이름**과 **Azure Machine Learning Workspace** 를 설정합니다. **연결테스트**를 눌러 연결이 잘 이루어지는지 테스트합니다.



새 연결된 서비스(Azure Machine Learning)

i 연결된 서비스의 이름을 선택합니다. 이 이름은 나중에 업데이트할 수 없습니다.

이름 *

설명

통합 런타임을 통해 연결 * ⓘ

인증 방법

Azure Machine Learning 작업 영역 선택 방법 ⓘ
☒ Azure 구독에서 ☐ 수동으로 입력

Azure 구독 ⓘ

Azure Machine Learning 작업 영역 이름 *

관리 ID 이름: gildong-synapse
 관리 ID 개체 ID: 052466b2-f38f-4fe4-85c4-1597411518d2
 Azure Machine Learning에 대한 액세스 권한을 작업 영역 서비스 관리 ID에 부여합니다.
[자세한 정보](#)

만들기 **뒤로** **연결 테스트** **취소**

이름: AzureMLService

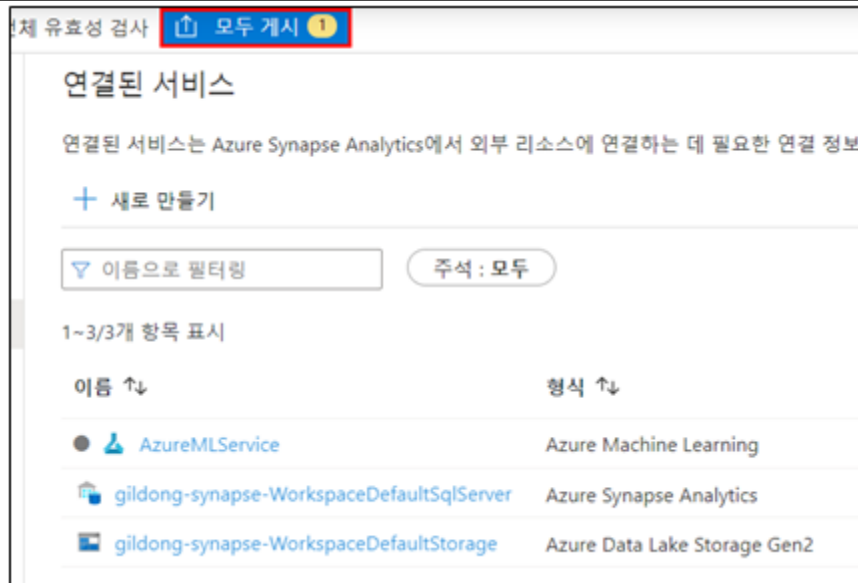
Azure 구독: 구독 선택

Azure Machine Learning 작업 영역 이름: 구독 선택 후 ML 워크스페이스 선택

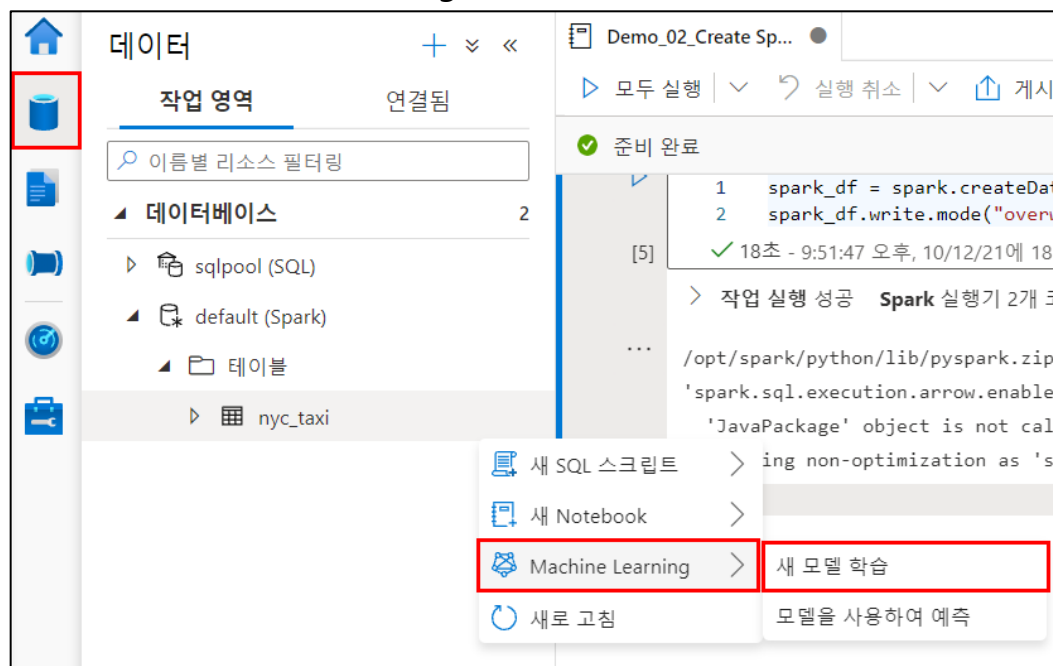
7. 연결 테스트를 성공하면 만들기를 눌러 생성합니다.

만들기 **뒤로** **연결 성공** **연결 테스트** **취소**

8. 모두 게시를 눌러 현재 상태를 저장합니다.



9. 데이터 허브로 이동합니다. 생성된 Spark 테이블 오른쪽 점 세 개(...)를 클릭합니다. **Machine Learning – 새 모델 학습**을 클릭합니다.



10. **Regression(회귀분석, 재발)**을 선택합니다. **계속**을 클릭합니다.

새 모델 학습

nyc_taxi

이 마법사는 **자동화된 기계 학습**(를) 사용하여 기계 학습 모델을 학습시키는 데 도움이 됩니다.

모델 유형 선택

답변하려는 질문에 따라 실험의 기계 학습 모델 유형을 선택합니다. 모델 유형을 선택하고 나면 실험 실행이 생성되기 전에 몇 가지 설정을 묻는 메시지가 표시됩니다. [자세한 정보](#)

분류

특정 결과의 달성 가능성을 확인(이진 분류)하거나 특성이 속한 범주를 식별(다중 클래스 분류)합니다.

예: 고객이 구독을 갱신할지 또는 취소할지를 예측합니다.

재발

입력 변수를 기준으로 숫자 값을 추정합니다.

예: 주택 크기를 기준으로 주택 가격을 예측합니다.

시계열 예측

기록 데이터에 따라 값 및 추세를 예측합니다.

예: 내년 주식 시장 추세를 예측합니다.

계속

취소

11. **Azure Machine Learning** 작업 영역, 실험 이름, 최적 모델 이름을 설정(기본값 유지)합니다. 대상 열은 **fareAmount** 를 선택합니다. **Spark** 풀을 선택하고 **계속**을 클릭합니다.

새 모델 학습 (재발)

nyc_taxi

실험 구성

만들 실험을 구성하고 모델을 학습시키는 데 사용할 Spark 풀을 선택합니다. [자세한 정보](#)

Source data
nyc_taxi

Azure Machine Learning 작업 영역 * ⓘ

mzcmllworkspace (AzureMLService)

실험 이름 * ⓘ

gildong-synapse-nyc_taxi-20211012125721

최적 모델 이름 * ⓘ

gildong-synapse-nyc_taxi-20211012125721-Best

대상 열 * ⓘ

fareAmount (double)

📘 재발 작업 형식에는 숫자 대상 열이 필요합니다. [자세한 정보](#)

Apache Spark 풀 * ⓘ

sparkpool

> Apache Spark 구성 정보

계속

뒤로

취소

대상 열: fareAmount

12. 학습 모델의 구성을 설정합니다. 최대 교육기간(시간)을 최솟값인 0.25(15 분)로 설정합니다. Notebook 에서 열기를 클릭합니다.

새 모델 학습 (재발)

nyc_taxi

재발 모델 구성

이 모델은 입력 변수를 기준으로 숫자 값을 추정합니다. [자세한 정보](#)

주 메트릭

Spearman correlation

최대 교육 기간(시간)

0.25

최대 동시 반복 횟수

2

ONNX 모델 호환성

☐ 사용
 ☒ 사용 안 함

실행 만들기

Notebook에서 열기

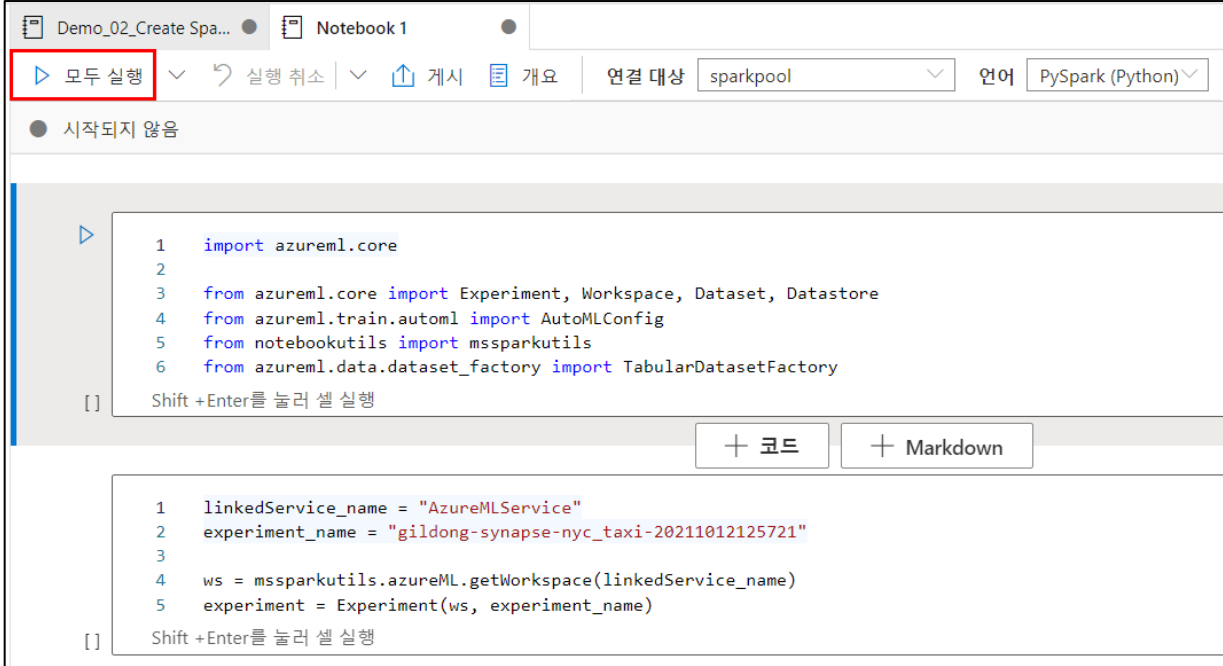
뒤로

취소

최대 교육 기간(시간) : 0.25

***실행 만들기를** 클릭하면 바로 **Machine Learning** 이 시작됩니다. 시작이 시작되고 있다는 **알림이** 표시되며 성공을 나타내는 알림이 표시됩니다.

13. 모델을 만드는 Notebook 이 생성되었습니다. 모두 실행을 클릭하여 모델을 만듭니다. 약 15 분 정도 시간이 소요될 수 있습니다.



Demo_02_Create Spa... Notebook 1

모두 실행 실행 취소 계시 개요 연결 대상 sparkpool 언어 PySpark (Python)

시작되지 않음

```

1 import azureml.core
2
3 from azureml.core import Experiment, Workspace, Dataset, Datastore
4 from azureml.train.automl import AutoMLConfig
5 from notebookutils import mssparkutils
6 from azureml.data.dataset_factory import TabularDatasetFactory

```

[] Shift +Enter를 눌러 셀 실행

+ 코드 + Markdown


```

1 linkedService_name = "AzureMLService"
2 experiment_name = "gildong-synapse-nyc_taxi-20211012125721"
3
4 ws = mssparkutils.azureml.getWorkspace(linkedService_name)
5 experiment = Experiment(ws, experiment_name)

```

[] Shift +Enter를 눌러 셀 실행

14. 6번 셀의 링크를 통해 진행과정을 모니터링할 수 있습니다.



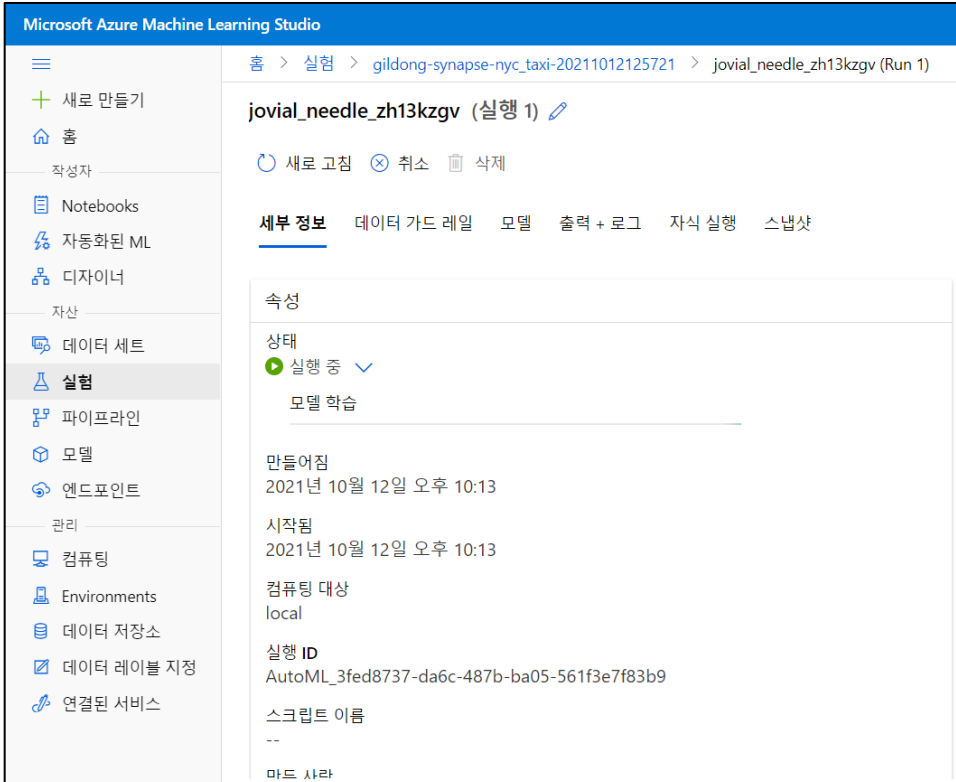
```

1 displayHTML("<a href={} target='_blank'>Your experiment in Azure Machine Learning portal: {}</a>".format(run.get_portal_url(), run.id))

```

[6] <1초 - 10:13:04 오후, 10/12/21에 162 ms에서 lyji 님이 명령을 실행함

Your experiment in Azure Machine Learning portal: AutoML_3fed8737-da6c-487b-ba05-561f3e7f83b9



Microsoft Azure Machine Learning Studio

홈 > 실험 > gildong-synapse-nyc_taxi-20211012125721 > jovial_needle_zh13kzgv (Run 1)

jovial_needle_zh13kzgv (실행 1)

새로 고침 취소 삭제

세부 정보 데이터 가드 레일 모델 출력 + 로그 지식 실행 스냅샷

속성

상태
● 실행 중

모델 학습

만들어짐
2021년 10월 12일 오후 10:13

시작됨
2021년 10월 12일 오후 10:13

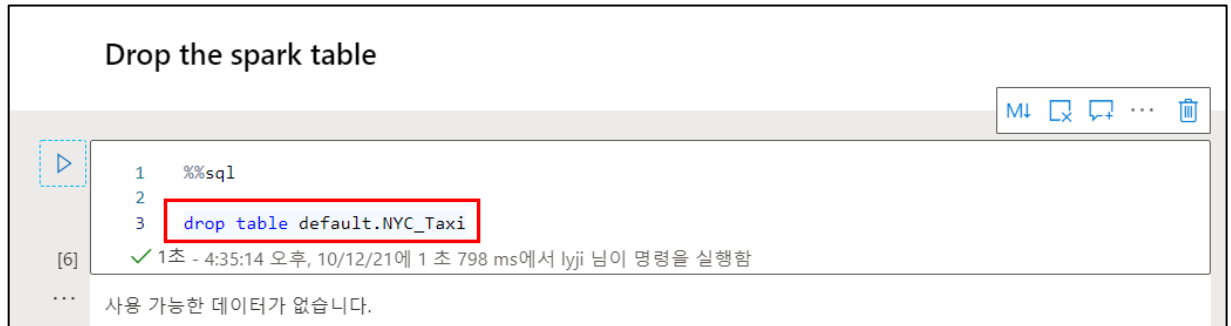
컴퓨팅 대상
local

실행 ID
AutoML_3fed8737-da6c-487b-ba05-561f3e7f83b9

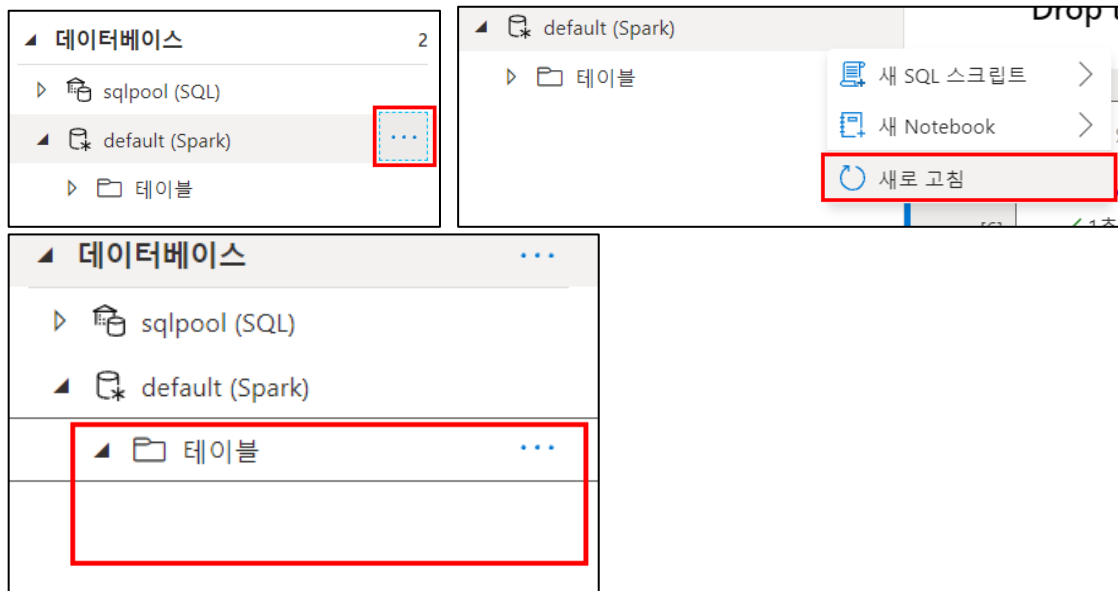
스크립트 이름
--

마트 사라

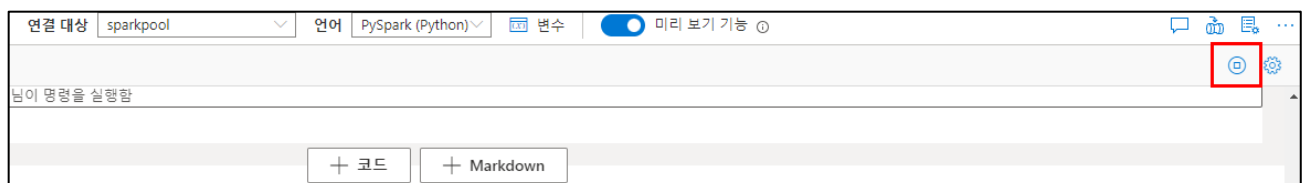
15. 다시 Demo 03 번 파일로 돌아옵니다. 6 번 셀의 주석을 풀고 실행합니다. Spark table 을 Drop 합니다.



16. 새로고침 후 데이터베이스를 확인합니다.

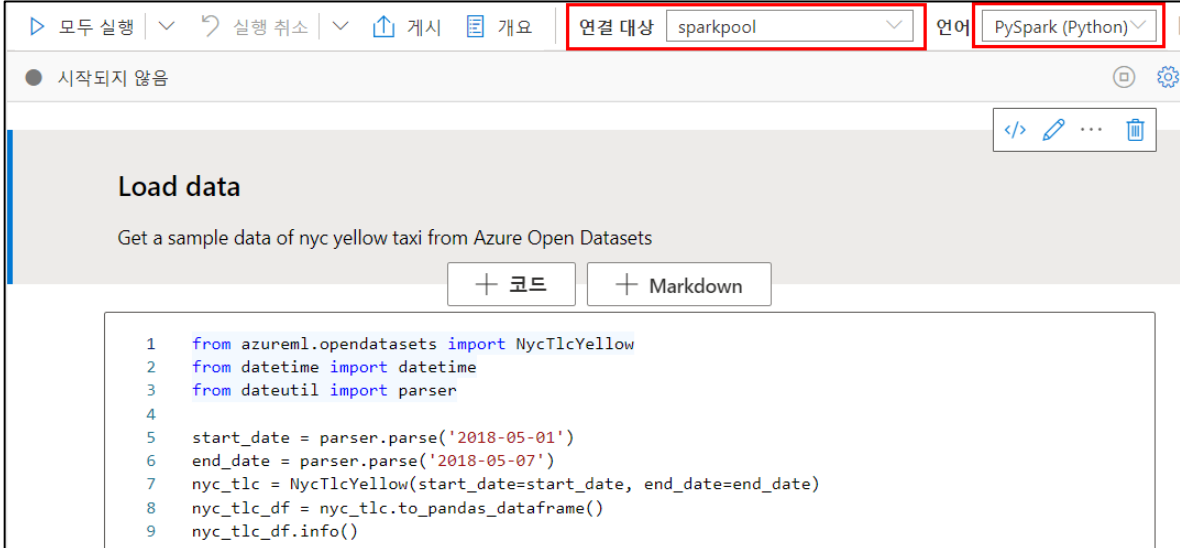


17. 노트북을 닫기 전에 세션 종료 버튼을 눌러 세션을 종료합니다.



Task 5 : Demo 04. NYC Taxi Fare AutoML Notebook from scratch

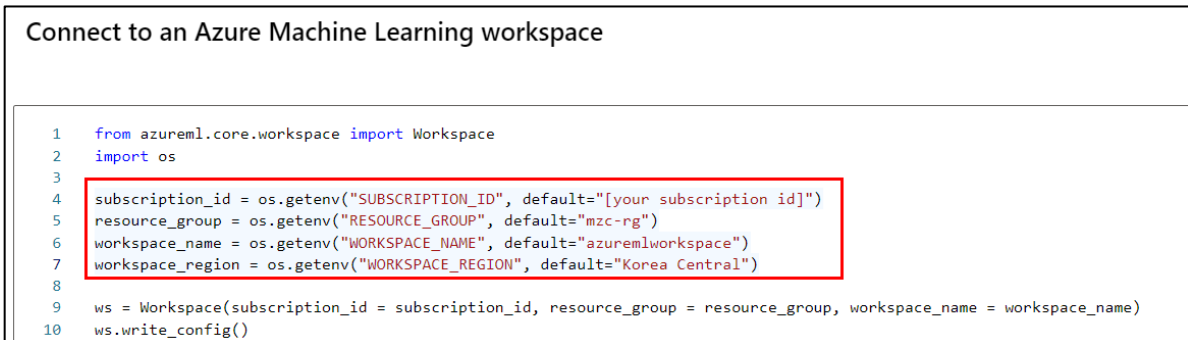
1. **Demo 04 번** 파일도 같은 단계를 거쳐서 실행합니다. 연결대상에 spark pool 을 할당하고 실행하면서 결과를 조회합니다. 7 번 셀까지 실행합니다.



```

1 from azureml.opendatasets import NycTlcYellow
2 from datetime import datetime
3 from dateutil import parser
4
5 start_date = parser.parse('2018-05-01')
6 end_date = parser.parse('2018-05-07')
7 nyc_tlc = NycTlcYellow(start_date=start_date, end_date=end_date)
8 nyc_tlc_df = nyc_tlc.to_pandas_dataframe()
9 nyc_tlc_df.info()
  
```

2. **8 번셀**은 **Azure Machine Learning workspace** 연결하는 부분입니다. 구독 ID, 리소스 그룹, 워크스페이스 이름, 워크스페이스 지역을 알맞게 입력합니다.



```

1 from azureml.core.workspace import Workspace
2 import os
3
4 subscription_id = os.getenv("SUBSCRIPTION_ID", default="[your subscription id]")
5 resource_group = os.getenv("RESOURCE_GROUP", default="mzc-rg")
6 workspace_name = os.getenv("WORKSPACE_NAME", default="azuremlworkspace")
7 workspace_region = os.getenv("WORKSPACE_REGION", default="Korea Central")
8
9 ws = Workspace(subscription_id = subscription_id, resource_group = resource_group, workspace_name = workspace_name)
10 ws.write_config()
  
```

Resource Group: mzc-rg

Workspace Name: mzcmlworkspace

Workspace Region: Korea Central

3. 나머지 셀들을 실행하면서 결과를 조회하고, 노트북을 닫기 전에 **세션 종료** 버튼을 눌러 세션을 종료합니다.

