



Practical Data Preparation and BI Hands-on

Developing, scheduling jobs with Glue ETL & Glue Catalog,
and visualization with Redshift Spectrum & QuickSight

2019. Feb

강성문 (kseongmo@amazon.com), Solutions Architect

© 2018 Amazon Web Services, Inc. or its affiliates. All rights reserved.

This work may not be reproduced or redistributed, in whole or in part, without prior written permission from Amazon Web Services, Inc. Commercial copying, lending, or selling is prohibited.

Corrections or feedback on the lab guide, please email me at: kseongmo@amazon.com

All trademarks are the property of their owners.



Contents

Contents.....	2
1. Overview	3
시나리오.....	3
활용 서비스.....	3
데이터 리소스	3
2. Prerequisite	4
S3 버킷 생성 및 파일 업로드.....	4
S3 버킷 파일 업로드.....	5
IAM Role 생성	7
3. Create Glue Catalog and Crawling	10
Data Catalog databases 생성.....	10
Raw 데이터에 대한 Crawler 및 Glue Catalog 생성.....	11
4. Exploring data with Athena (Optional)	17
GLUE 카탈로그로 업그레이드	17
S3 data query with Athena.....	18
5. Data Transformation & Enrichment.....	21
Transform – 파일포맷 변환	21
Transform – 파일포맷 변환 및 데이터 클린징	25
Transformed data crawler.....	29
Transform – 조인을 통한 새로운 데이터 생성	31
Enriched data crawler	36
6. ETL Job 결과 데이터 확인 (Optional)	38
7. ETL Job Scheduling (Optional).....	40
작업 선후행 걸기	40
Job Scheduling Test.....	43
8. QuickSight dashboard.....	46
QuickSight 초기 설정.....	46
Data set 생성	49
Data set editing	50
Map Dashboard 생성	56
Dashboard share (Optional)	59
9. Closing	64
QuickSight 삭제.....	64
Glue 리소스 삭제	65
S3 버킷 삭제	66

1. Overview

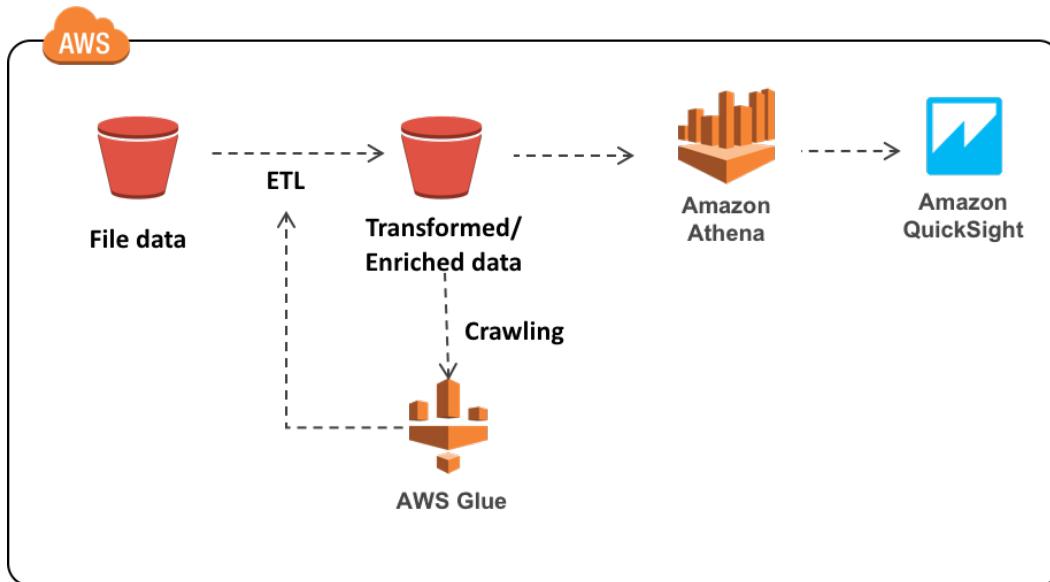
시나리오

파일형태로 수집한 글로벌 거점의 주문정보를 전처리하여 적재하고, 이 데이터를 활용하여 BI Tool 및 타 서비스에서 조회 및 가시화합니다. 실습은 2 시간 가량 소요됩니다.

활용 서비스

본 가이드에서 사용하는 AWS 서비스는 다음과 같습니다.

- S3 : DataLake 저장소로 활용됩니다.
- AWS Glue : 데이터 카탈로그 생성 및 DataLake 버킷간 데이터의 변환, 적재 작업을 수행합니다.
- Athena : SQL을 이용하여 DataLake 데이터를 조회합니다.
- QuickSight : DataLake 의 데이터 가시화를 수행합니다.



데이터 리소스

본 가이드에서 데이터 처리를 위한 리소스는 아래 두 파일을 사용합니다.

- cities.csv : 글로벌 거점의 도시위치(위도, 경도)와 인구정보
- orders.csv : 글로벌 21개국 거점으로부터의 주문정보

2. Prerequisite

실습을 위한 사전 환경 구성은 위해 3 개의 S3 버킷과 1 개의 IAM Role 을 생성할 것입니다. AWS Glue 와 QuickSight 를 모두 사용 가능한 리전을 선택하여 작업합니다. (가이드 작성 시점 기준 버지니아, 오하이오, 오레곤, 아일랜드, 싱가포르, 시드니, 도쿄 선택 가능)

S3 버킷 생성 및 파일 업로드

2-1. S3 콘솔로 이동하여 다음 3 개의 버킷을 생성합니다. [account_id] 부분은 본인의 어카운트 번호를 넣거나 Global Unique 한 임의의 값을 입력합니다.

- dplab-raw-[account_id]
- dplab-transformed-[account_id]
- dplab-enriched-[account_id]

2-2. S3 버킷을 생성하는 방법은 아래를 참고합니다. 이미 방법을 알고 있는 경우 방법대로 진행하시고 2-6 단계로 넘어갑니다.

2-3. S3 콘솔에서 Create bucket 버튼을 클릭합니다.

+ Create bucket

2-4. Bucket name 에 dplab-raw-[account_id] 를 입력하고 Create 버튼을 클릭합니다. 버킷명은 url 에 사용되는 문자를 사용할 수 있으며 글로벌하게 고유한 값이어야 합니다.

Create

2-5. 동일한 방법으로 dplab-transformed-[account_id] 와 dplab-enriched-[account_id] 버킷을 생성합니다.

The screenshot shows the Amazon S3 console interface. At the top, there's a navigation bar with the AWS logo and 'Discover the new AWS' link. Below it is a search bar containing 'dplab'. On the left, there are three buttons: '+ Create bucket', 'Delete bucket', and 'Empty bucket'. On the right, it shows '3 Buckets' and '0 Public' buckets. The main area displays three buckets:

Bucket name	Access	Region
dplab-enriched- 987654321012	Not public *	Asia Pacific (Seoul)
dplab-raw- 987654321012	Not public *	Asia Pacific (Seoul)
dplab-transformed- 987654321012	Not public *	Asia Pacific (Seoul)

S3 버킷 파일 업로드

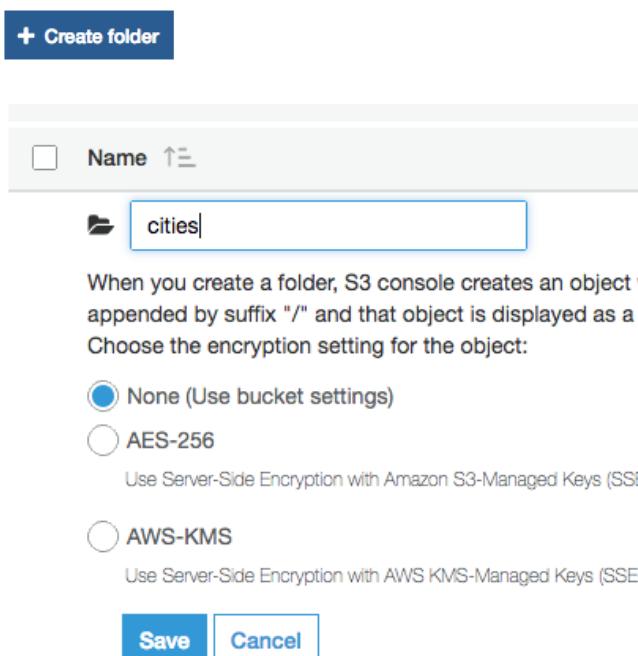
2-6. 이전 단계에서 생성한 버킷에 폴더를 만들고 파일을 업로드합니다.

- dplab-raw-[account] 버킷에 cities 와 orders 라는 폴더를 만들고 cities.csv 파일과 orders.csv 파일을 각 폴더에 업로드합니다.
 - o dplab-raw-[account_id]/cities/cities.csv
 - o dplab-raw-[account_id]/orders/orders.csv
- dplab-transformed-[account_id] 버킷에 cities 와 orders 라는 폴더를 만듭니다.
 - o dplab-transformed-[account_id]/cities/
 - o dplab-transformed-[account_id]/orders/
- dplab-enriched-[account_id] 버킷에 orders_w_city
 - o dplab-enriched-[account_id]/orders_w_city/

2-7. S3 에서 폴더 생성과 파일을 업로드하는 단계별 방법은 아래를 참고합니다. 이미 방법을 알고 있는 경우 해당 방법으로 진행하시고 2-13 단계로 넘어갑니다.

2-8. dplab-raw 로 시작되는 버킷을 클릭합니다.

2-9. Create folder 버튼을 클릭하여 cities 와 orders 라는 폴더를 생성합니다.





<input type="checkbox"/>	Name	↑☰
<input type="checkbox"/>	cities	
<input type="checkbox"/>	orders	

2-10. cities 와 orders 폴더에 각각 cities.csv, orders.csv 파일을 업로드합니다. (업로드 방법은 폴더를 클릭한 후 Upload 버튼을 클릭한 후 팝업 화면에서 Add files 를 통해 .csv 파일을 선택하고 다시 Upload 버튼을 클릭하면 됩니다.)

Amazon S3 > dplab-raw- 987654321012 / cities

Overview	
<input type="text"/> Type a prefix and press Enter to search. Press ESC to clear.	

<input type="checkbox"/>	Name	Last modified	Size	Storage class
<input type="checkbox"/>	cities.csv	Jun 27, 2018 3:24:41 PM GMT+0900	1.2 KB	Standard

Amazon S3 > dplab-raw- 987654321012 / orders

Overview	
<input type="text"/> Type a prefix and press Enter to search. Press ESC to clear.	

<input type="checkbox"/>	Name	Last modified	Size	Storage class
<input type="checkbox"/>	orders.csv	Jun 27, 2018 3:26:04 PM GMT+0900	7.2 MB	Standard

2-11. 마찬가지로 dplab-transformed 로 시작되는 버킷에도 cities 와 orders 폴더를 생성합니다. (파일은 업로드하지 않습니다.)

Amazon S3 > dplab-transformed- 987654321012

Overview	Properties	Permissions	Management
<input type="text"/> Type a prefix and press Enter to search. Press ESC to clear.			
Upload + Create folder More		Asia Pacific (Seoul) Edit	
Viewing 1 to 2			
<input type="checkbox"/> Name ↑	Last modified ↑	Size ↑	Storage class ↑
<input type="checkbox"/> cities	--	--	--
<input type="checkbox"/> orders	--	--	--
Viewing 1 to 2			

2-12. dplab-enriched 로 시작되는 버킷에서는 orders_w_city 라는 폴더를 생성합니다. (파일은 업로드하지 않습니다.)

Amazon S3 > dplab-enriched- 987654321012

Overview	Properties	Permissions	Management
<input type="text"/> Type a prefix and press Enter to search. Press ESC to clear.			
Upload + Create folder More		Asia Pacific (Seoul) Edit	
Viewing 1 to 1			
<input type="checkbox"/> Name ↑	Last modified ↑	Size ↑	Storage class ↑
<input type="checkbox"/> orders_w_city	--	--	--
Viewing 1 to 1			

IAM Role 생성

2-13. Glue 작업에서 사용하게 될 IAM Role 을 dplab-glue-role 이라는 이름으로 생성하겠습니다. 해당 Role 은 Glue Service 와 S3 에 접근할 수 있는 권한을 줄 것입니다.

- Role name: dplab-glue-role
- Trusted entities: Glue (AWS service: glue.amazonaws.com)
- Policies
 - o AWSGlueServiceRole
 - o AmazonS3FullAccess

2-14. Role 을 생성하는 방법은 아래를 참고합니다. 이미 방법을 알고 계신 경우 방법대로 진행하시고 3-1 단계로 넘어갑니다.

2-15. IAM 콘솔의 네비게이션에서 Roles 를 선택한 후 Create role 버튼을 클릭합니다.



2-16. Role 을 consume 하는 서비스로 Glue 를 선택하고 Next: Permissions 버튼을 클릭합니다.

Create role

1 2 3

Select type of trusted entity

AWS service EC2, Lambda and others	Another AWS account Belonging to you or 3rd party	Web identity Cognito or any OpenID provider	SAML 2.0 federation Your corporate directory
---------------------------------------	--	--	---

Allows AWS services to perform actions on your behalf. [Learn more](#)

Choose the service that will use this role

EC2
Allows EC2 instances to call AWS services on your behalf.

Lambda
Allows Lambda functions to call AWS services on your behalf.

API Gateway	Config	Elastic Container Service	Lex	SWF
AppSync	DMS	Elastic Transcoder	Machine Learning	SageMaker
Application Auto Scaling	Data Pipeline	Elastic Load Balancing	MediaConvert	Service Catalog
Auto Scaling	DeepLens	Glue	OpsWorks	Step Functions
Batch	Directory Service	Greengrass	RDS	Storage Gateway
CloudFormation	DynamoDB	GuardDuty	Redshift	
CloudHSM	EC2	Inspector	Rekognition	
CloudWatch Events	EMR	IoT	S3	
CodeBuild	ElastiCache	Kinesis	SMS	
CodeDeploy	Elastic Beanstalk	Lambda	SNS	

Select your use case

Glue
Allows Glue to call AWS services on your behalf.

* Required Cancel **Next: Permissions**

2-17. AmazonS3FullAccess 와 AWSGlueServiceRole 이름을 검색하여 두 Policy 를 추가하고 Next: Review 버튼을 클릭합니다. (각 Policy 를 개별적으로 선택/추가합니다. 아래 화면은 참고용 최종 상태 이미지입니다.)

Permissions Trust relationships Access Advisor Revoke sessions

Attach policy Attached policies: 2

Policy name	Policy type	X
▶ AmazonS3FullAccess	AWS managed policy	X
▶ AWSGlueServiceRole	AWS managed policy	X

[+ Add inline policy](#)

2-18. Role name 을 **dplab-glue-role** 로 입력하고 Create role 버튼을 클릭합니다.

Create role

1 2 3

Review

Provide the required information below and review this role before you create it.

Role name*

dplab-glue-role

Use alphanumeric and '+=, @-_ ' characters. Maximum 64 characters.

Role description

Allows Glue to call AWS services on your behalf.

Maximum 1000 characters. Use alphanumeric and '+=, @-_ ' characters.

Trusted entities AWS service: glue.amazonaws.com

Policies

[AmazonS3FullAccess](#)

[AWSGlueServiceRole](#)

Permissions boundary Permissions boundary is not set

* Required

[Cancel](#)

[Previous](#)

Create role

S3 와 Glue 의 자원을 사용할 수 있는 Role 이 추가되었습니다.

The role **dplab-glue-role** has been created.



3. Create Glue Catalog and Crawling

본 단계에서는 Glue 를 통해 데이터 카탈로그를 생성할 수 있도록 Crawling 작업을 구성합니다.

Data Catalog databases 생성

서비스에서 AWS Glue 를 선택하여 Glue 관리 콘솔로 이동합니다.

이전에 Glue 를 사용한 적이 없는 경우 초기 안내화면에서 Get started 버튼을 클릭합니다.



3-1. 네비게이션에서 Databases 를 선택하고 Add database 버튼을 클릭합니다.

The screenshot shows the AWS Glue Data catalog interface. On the left sidebar, 'Databases' is selected under 'Data catalog'. In the main area, the heading 'Databases' is followed by a subtext: 'A database is a set of associated table definitions, organized into a logical group.' Below this are buttons for 'Add database', 'View tables', and 'Action'. A table header with columns 'Name' and 'Description' is shown, along with a message: 'You don't have any databases defined in your data catalog.' A large blue 'Add database' button is prominently displayed at the bottom right of the table area.

3-2. raw, transformed, enriched 라는 이름으로 세 개의 database 를 생성합니다.

The screenshot shows a modal dialog box titled 'Add database'. Inside, the 'Database name' field contains the value 'transformed'. Below it is a section labeled 'Description and location (optional)'. At the bottom right of the dialog is a blue 'Create' button. The background shows the same AWS Glue Data catalog interface as the previous screenshot, with the 'Databases' section selected.

Databases A database is a set of associated table definitions, organized into a logical group.

Name	Description
enriched	
raw	
transformed	

Raw 데이터에 대한 Crawler 및 Glue Catalog 생성

3-3. Glue 관리콘솔 좌측 네비게이션에서 Crawler 를 선택하고 Add Crawler 버튼을 클릭합니다.

AWS Glue

- Data catalog
- Databases
- Tables
- Connections
- Crawlers**
- Classifiers

ETL

- Jobs

Crawlers

A crawler connects to a data store, progresses through a prioritized list of classifiers to determine the creates metadata tables in your data catalog.

Add crawler Run crawler Action

Name	Schedule	Status	Logs	Last runtime	M

You don't have any crawlers yet.

Add crawler

3-4. Crawler name 을 dplab-raw-crawler 로 입력하고 Next 를 클릭합니다.

Add crawler

Add information about your crawler

Crawler info

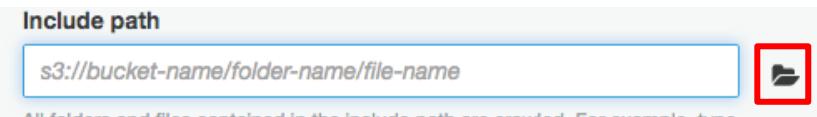
Crawler name

dplab-raw-crawler

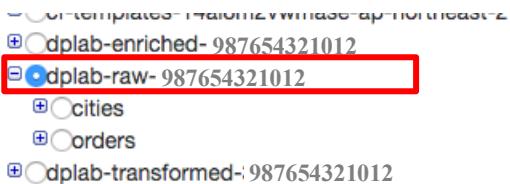
- ▶ Description, security configuration, and classifiers (optional)
- ▶ Grouping behavior for S3 data (optional)

Next

3-5. Data store 설정 화면에서 Include path 오른쪽의 폴더 아이콘을 클릭합니다.



3-6. 팝업 화면에서 dplab-raw-[account] 폴더를 선택하고 Select 버튼을 클릭합니다. (폴더가 아닌 버킷을 선택합니다.)



3-7. 나머지 설정을 Default 로 두고 Next 를 클릭합니다.

Add another data store 화면에서 No 를 선택된 상태로 Next 를 클릭합니다.

Crawler info
dplab-raw-cralwer

Data store
S3: s3://dplab-raw-...

IAM Role

Schedule

Output

Review all steps

Add another data store
 Yes
 No

Back Next

3-8. Choose an existing IAM role 을 선택하고 IAM 입력창에 Prerequisite 단계에서 생성한 dplab-glue-role 을 선택하여 입력하고 Next 버튼을 클릭합니다.

Crawler info
dplab-raw-cralwer

Data store
S3: s3://dplab-raw-...

IAM Role

Schedule

Output

Review all steps

Choose an IAM role
The IAM role allows the crawler to run and access your Amazon S3 data stores. [Learn more](#)

Update a policy in an IAM role
 Choose an existing IAM role
 Create an IAM role

IAM role
dplab-glue-role
dplab-glue-role

AWSGlueServiceRole, plus access to your data stores.
s3://dplab-raw-987654321012

You can also create an IAM role on the [IAM console](#).

Back Next

3-9. Schedule Frequency 는 Run on demand 상태로 두고 Next 를 클릭합니다.

Crawler info
dplab-raw-cralwer

Data store
S3: s3://dplab-raw-...

IAM Role
arn:aws:iam:: 987654321012 :role/dplab-glue-role

Schedule
Run on demand

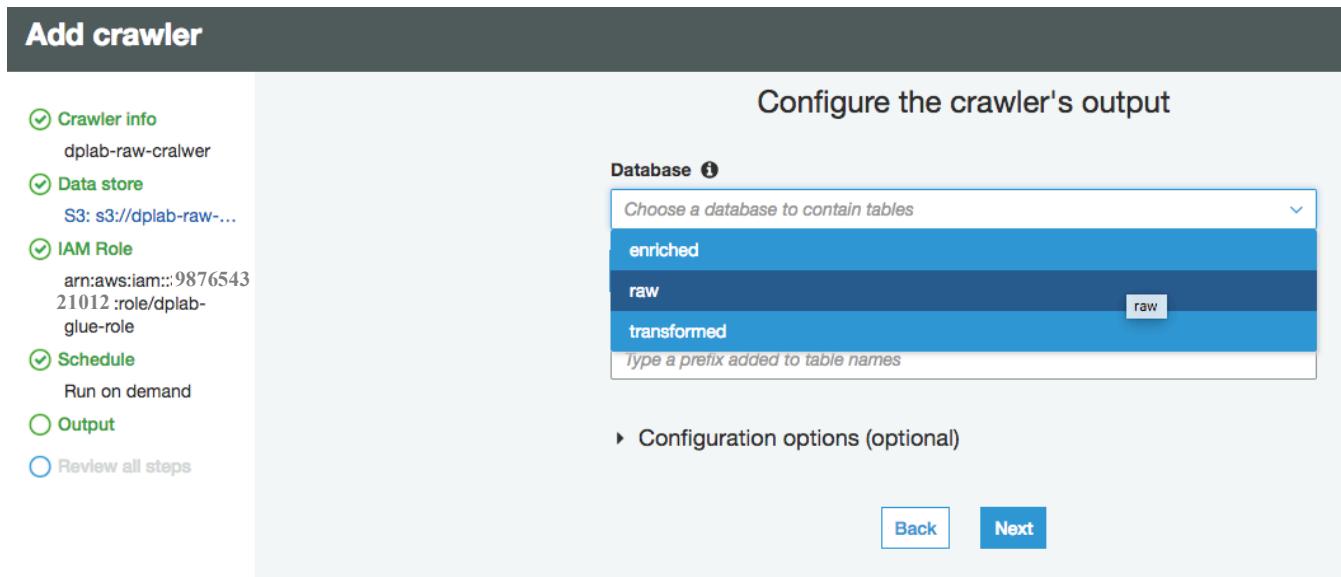
Output

Review all steps

Create a schedule for this crawler
Frequency
Run on demand

Back Next

3-10. Output Database 로 raw 를 선택하고 Next 버튼을 클릭합니다.



3-11. 내용을 Review 한 후 Finish 버튼을 클릭하면 첫번째 Crawler Job 0| 생성됩니다.

	Name	Schedule	Status	Logs	Last runtime	Median runtime	Tables updated	Tables added
<input type="checkbox"/>	dplab-raw-cra...		Ready		0 secs	0 secs	0	0

3-12. Job 체크박스를 선택한 후 Run crawler 를 클릭하여 생성한 Crawler 를 실행합니다. (메시지박스의 Run it now 를 클릭해도 동일 액션을 수행할 수 있습니다.)

ver was created to run on demand. [Run it now?](#)

3-13. Crawler Job 0| Starting 상태로 바뀌었다가 Refresh 를 실행하면 Stopping 을 거쳐 Ready 상태로 종료되고 시트의 맨 오른쪽 컬럼에서 2 개의 Tables 가 생성된 것을 확인할 수 있습니다.

<input type="checkbox"/>	Name	Schedule	Status	Logs	Last runtime	Median runtime	Tables updated	Tables added
<input checked="" type="checkbox"/>	dplab-raw-cra...		Stopping		25 secs	25 secs	0	2

3-14. 네비게이션 바에서 Databases 메뉴로 이동하고 raw database 를 클릭합니다. 화면 아래의 Tables in raw 를 클릭합니다.

Databases > raw

Edit database	Delete database
-------------------------------	---------------------------------

Name raw
 Description
 Location

Tables in raw

3-15. cities 와 orders 테이블이 생성된 것을 확인합니다.

Add tables ▾	Action ▾	<input type="text"/> Database : raw  Filter or search for tables...	Save view ▾	Showing: 1 - 2 < >   
<input type="checkbox"/>	Name	Database	Location	Classification
<input type="checkbox"/>	cities	raw	s3://dplab-raw- 987654321012 /cities/	csv
<input type="checkbox"/>	orders	raw	s3://dplab-raw- 987654321012 /orders/	csv

3-16. S3 버킷의 csv 파일을 파싱하여 테이블 기초 정보와 스키마 정보가 자동으로 생성된 것을 확인할 수 있습니다. 테이블명을 클릭하여 raw 데이터의 파일 건수와 스키마가 잘 생성되었는지 확인합니다.

Tables > orders

Last updated 27 Jun 2018 Table Version (Current version) ▾

[Edit table](#) [Delete table](#)[View properties](#)[Compare versions](#)[Edit schema](#)

Name	orders
Description	
Database	raw
Classification	csv
Location	s3://dplab-raw- 987654321012 /orders/
Connection	
Deprecated	No
Last updated	Wed Jun 27 15:51:20 GMT+900 2018
Input format	org.apache.hadoop.mapred.TextInputFormat
Output format	org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat
Serde serialization lib	org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe
Serde parameters	field.delim , skip.header.line.count 1 sizeKey 7590453 objectCount 1
Table properties	UPDATED_BY_CRAWLER dplab-raw-crawler CrawlerSchemaSerializerVersion 1.0 recordCount 51286 averageRecordSize 148 CrawlerSchemaDeserializerVersion 1.0 compressionType none columnsOrdered true delimiter , typeOfData file

Schema

Showing: 1 - 14 of 14 < >

	Column name	Data type	Partition key	Comment
1	year	bigint		
2	product line	string		
3	product type	string		
4	product	string		
5	order method type	string		
6	retailer country	string		
7	revenue	double		
8	planned revenue	double		
9	product cost	double		
10	quantity	bigint		
11	unit cost	double		
12	unit price	double		
13	gross profit	double		
14	unit sale price	double		

4. Exploring data with Athena (Optional)

Athena 를 이용하여 S3 의 파일 내용을 SQL 문을 통해 조회해 볼 수 있습니다. 4 번 단계는 선택적으로 진행 가능합니다. Lab 시간이 부족하거나 이미 Athena 에 익숙한 경우 5 번 단계로 넘어갑니다.

GLUE 카탈로그로 업그레이드

AWS 콘솔에서 Athena 서비스를 선택합니다. (이전에 Athena 를 실행한 적이 없는 경우 Get Started 버튼을 클릭하고 Tutorial 은 skip 합니다.)

Get Started

4-1. Database 에 Glue 카탈로그가 보이는지 확인합니다. 리전에 따라 Athena 에서 Glue 카탈로그를 이용하기 위해서는 카탈로그 통합 작업이 필요할 수 있습니다. 아래 메시지가 오픈되는 경우 Click here to upgrade 에서 here 링크를 클릭합니다. Database 에 카탈로그가 보이는 경우는 이미 통합이 완료된 것으로 다음 단계로 넘어갑니다.

카탈로그 통합과 관련한 자세한 내용은 다음 링크를 참고할 수 있습니다.

<https://docs.aws.amazon.com/athena/latest/ug/glue-athena.html>

To use the AWS Glue Data Catalog with Amazon Athena and Amazon Redshift Spectrum, you must upgrade your Athena Data Catalog to the AWS Glue Data Catalog. Without the upgrade, tables and partitions created by AWS Glue cannot be queried with Amazon Athena or Redshift Spectrum. Click [here](#) to upgrade.

IAM 정책 업데이트에 대한 안내 페이지가 오픈됩니다. 만약 기존에 Athena 와 Redshift Spectrum 을 사용하면서 사용자 정의 정책을 정의한 경우 카탈로그 업데이트에 따라 권한 추가가 필요하다는 안내입니다. 기존에 Athena 와 Redshift Spectrum 을 사용하지 않았을 경우에는 바로 업그레이드를 실행할 수 있습니다. 파란색 Upgrade 버튼을 클릭합니다.

Upgrade

업그레이드가 완료되면 왼쪽 Database 네비게이션에서 raw, transformed, enriched 데이터베이스와 Crawling 작업을 통해 생성한 두 개의 테이블을 볼 수 있습니다. (Database 리스트에서 raw 를 선택하십시오. 조회되지 않을 경우 리프레시 버튼을 클릭합니다.)

The screenshot shows a database management interface. At the top left is a search bar containing the word "raw". To its right is a refresh icon. Below the search bar is a dropdown menu labeled "Filter tables and views...". The main area is divided into sections: "Tables (2)" and "Views (0)". Under "Tables (2)", there are two entries: "cities" and "orders", each with a three-dot menu icon. Under "Views (0)", there is a "Create view" button. A large text block at the bottom states: "You have not created any views. To create a view, run a query and click 'Create view from query'".

S3 data query with Athena

4-2. 테이블명 옆의 ...을 클릭하고 Preview table 을 클릭하면 샘플 쿼리가 생성되고 실행됩니다. 이제 S3에 저장된 데이터를 sql 문을 통해 조회할 수 있습니다.

New query 1 +

```
1 SELECT * FROM "raw"."cities" limit 10;
```

Run query Save as Format query New query (Run time: 1.49 seconds, Data scanned: 1.15KB)

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

Results

	retail_id	country	latitude	longitude	population	last_updated
1	1	Korea	"37°30'N"	"127°00'E"	51446201	2017-07-01
2	2	United States	"38°55'N"	"77°00'W"	327140000	2018-02-05
3	3	China	"39°55'N"	"116°45'E"	1388021000	2018-02-05
4	4	Brazil	"15°45'S"	"47°57'W"	208594000	2018-02-05
5	5	Canada	"45°25'N"	"75°43'W"	37008800	2018-02-05

New query 1 +

```
1 SELECT * FROM "raw"."orders" limit 10;
```

Run query Save as Format query New query (Run time: 1.61 seconds, Data scanned: 283.29KB)

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

Results

year	product line	product type	product	order method	type	retailer country	revenue	planned revenue	product cost	quantity	unit cost	unit price	gr
1	2014	Camping Equipment	Cooking Gear	TrailChef Water Bag	Telephone	United States	315044.33	437477.15	158371.76	66385	2.55285714	6.59	15
2	2014	Camping Equipment	Cooking Gear	TrailChef Water Bag	Telephone	Canada	13444.68	14313.48	6298.8	2172	2.9	6.59	71
3	2014	Camping Equipment	Cooking Gear	TrailChef Water Bag	Telephone	Mexico							
4	2014	Camping Equipment	Cooking Gear	TrailChef Water Bag	Telephone	Brazil							
5	2014	Camping Equipment	Cooking Gear	TrailChef Water Bag	Telephone	Japan	181120.24	235236.64	89413.06	35696	2.657	6.59	91
6	2014	Camping Equipment	Cooking Gear	TrailChef Water Bag	Telephone	Korea							
7	2014	Camping Equipment	Cooking Gear	TrailChef Water Bag	Telephone	China	69608.15	100200.95	35326.25	15205	2.495	6.59	34

4-3. orders 테이블을 조회해 보면 주문 제품과 국가는 있으나 revenue, quantity(주문량) 등의 정보가 없는 레코드가 다수 포함된 것을 확인할 수 있습니다. 다음 쿼리를 수행하여 클린징이 필요한 레코드를

확인합니다. (해당 레코드는 주문량 집계에 필요 없는 데이터들이므로 다음 단계에서 Data Preparation ETL 작업을 통해 삭제될 것입니다.)

```
SELECT 'Valid', count(*) FROM "raw"."orders" WHERE cast(quantity as varchar) is not null
union
SELECT 'Invalid', count(*) FROM "raw"."orders" WHERE cast(quantity as varchar) is null
union
SELECT 'Total', count(*) FROM "raw"."orders"
```

The screenshot shows the AWS Management Console interface for a Redshift cluster. At the top, there is a code editor window titled 'New query 1' containing the SQL script provided above. Below the code editor is a status bar with the message 'Use Ctrl + Enter to run query, Ctrl + Space to autocomplete'. Underneath the status bar are four buttons: 'Run query' (highlighted in blue), 'Save as', 'Format query', and 'New query'. A note below these buttons states '(Run time: 2.09 seconds, Data scanned: 21.72MB)'. Below the editor is a results pane titled 'Results'. The results are displayed in a table with three columns: '_col0', '_col1', and '_col2'. The data is as follows:

_col0	_col1	_col2
1	Total	84672
2	Invalid	59929
3	Valid	24743

The row for 'Invalid' is highlighted with a red box.

5. Data Transformation & Enrichment

이제 raw 데이터로부터 새로운 데이터를 생성하는 Glue ETL job 을 만들고 실행합니다. 본 Lab 에서는 콘솔에서 직접 Job 의 코드를 수정할 것이지만, 실제 ETL 작업을 개발하고 디버깅하실 때에는 Glue 의 Dev endpoint 기능을 이용하시면 여기에 Zeppelin 노트북을 연결하여 인라인 디버깅을 하실 수 있습니다. 자세한 내용은 다음 개발자 가이드를 참고하십시오.

https://github.com/awsdocs/aws-glue-developer-guide/blob/master/doc_source/index.md

Transform – 파일포맷 변환

raw 데이터베이스의 cities 테이블은 현재 .csv 로 저장되어 있습니다. 해당 테이블을 parquet 로 변환하는 작업을 생성하겠습니다.

5-1. Glue 서비스 콘솔에서 ETL > Jobs 를 선택하고 Add job 버튼을 클릭합니다.

Add job

5-2. Job Name 을 transform-cities 로 지정하고 Role 은 dplab-glue-role 을 선택합니다.

Job properties

Name
transform-cities

IAM role
dplab-glue-role

This job runs
 A proposed script generated by AWS Glue i
 An existing script that you provide
 A new script to be authored by you

ETL language
 Python Scala

Script file name
transform-cities

S3 path where the script is stored
s3://aws-glue-scripts-207287707808.us-east-1.s3...

5-3. 나머지를 디폴트 값으로 두고 Next 를 클릭합니다.



5-4. Data source 는 raw 의 cities 를 선택하고 Next 를 클릭합니다.

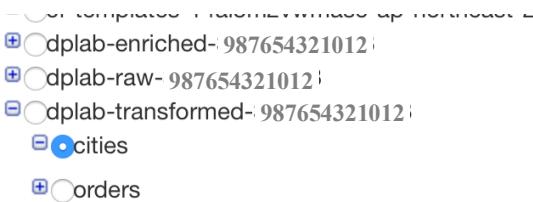
Name	Database	Location	Classification
<input checked="" type="radio"/> cities	raw	s3://dplab-raw-987654321012/cities/	csv
<input type="radio"/> elb_logs	sampledb	s3://athena-examples-ap-northeast-2/...	Unknown
<input type="radio"/> orders	raw	s3://dplab-raw-987654321012/orders/	csv

5-5. Choose your target 페이지에서 Create tables in your data target 라디오버튼을 선택합니다.

5-6. Data store 는 Amazon S3 를 선택합니다.

5-7. Format 은 Parquet 를 선택합니다.

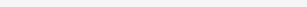
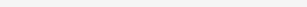
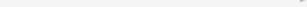
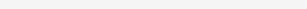
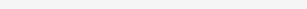
5-8. Target path 입력창 오른쪽의 폴더 버튼을 클릭하고 dplab-transformed-[account] 버킷의 cities 폴더를 선택하고 Next 를 클릭합니다.



5-9. 다음 화면에서 컬럼매핑은 변환하지 않고 Next 를 클릭합니다.

Map the source columns to target columns.

Verify the mappings created by AWS Glue. Change mappings by choosing other columns with **Map to target**. You can **Clear** all mappings and **Reset** to default AWS Glue mappings. AWS Glue generates your script with the defined mappings.

Source			Target		Add column	Clear	Reset
Column name	Data type	Map to target			Column name	Data type	
retail_id	bigint	retail_id 			retail_id	long   	
country	string	country 			country	string   	
latitude	string	latitude 			latitude	string   	
longitude	string	longitude 			longitude	string   	
population	bigint	population 			population	long   	
last_updated	string	last_updated 			last_updated	string   	

5-10. 내용을 검토한 후 Save job and edit script 버튼을 클릭합니다.

5-11. Save 를 클릭하고 오른쪽 상단의 x 버튼을 눌러 편집창을 빠져나옵니다.

The screenshot shows the AWS Glue Job Editor interface. At the top, there's a navigation bar with 'AWS' logo, 'Services', 'Resource Groups', and user information 'Administrator @ leonkang'. Below the navigation is a toolbar with buttons for 'Action', 'Save' (highlighted with a red box), 'Run job', 'Generate diagram', 'Insert template at cursor', 'Source', 'Target', 'Target Location', 'Transform', 'Spigot', and a help icon.

The main area displays a data pipeline. On the left, there's a sidebar with icons for adding databases and tables, and a tree view showing the pipeline structure:

- Database Name raw
Table Name cities
- ↓
- Transform Name ApplyMapping
- ↓
- Transform Name ResolveChoice

On the right, the Python code for the job is shown:

```
1 import sys
2 from awsglue.transforms import *
3 from awsglue.utils import getResolvedOptions
4 from pyspark.context import SparkContext
5 from awsglue.context import GlueContext
6 from awsglue.job import Job
7
8 ## @params: [JOB_NAME]
9 args = getResolvedOptions(sys.argv, ['JOB_NAME'])
10
11 sc = SparkContext()
12 glueContext = GlueContext(sc)
13 spark = glueContext.spark_session
14 job = Job(glueContext)
15 job.init(args['JOB_NAME'], args)
16 ## @type: DataSource
17 ## @name: datasource - "now" - table name = "cities" - transformation rty = "datasource0"
18
```

Below the code editor, there are tabs for 'Logs' and 'Schema'.

5-12. Jobs 목록에서 작업명 옆의 체크박스를 클릭하고 Action에서 Run job을 선택하면 Job을 실행시킵니다.

The screenshot shows the AWS Glue console. A dropdown menu titled 'Action' is open, listing several options: 'Run job' (which is highlighted in blue), 'Stop job run', 'Choose job triggers', 'Delete', 'Edit job', 'Edit script', 'Reset job bookmark', and 'Create development endpoint'. To the left of the dropdown, there are checkboxes for 'Name' and 'transform-cities'.

Parameters (optional)

Review and override parameter values, as needed, before running this job. Changes affect this run only. Edit a job to change default parameter values.

- ▶ Advanced properties
- ▶ Script libraries and job parameters

Only job **enrich-join** is run. Jobs dependent on the completion of job **enrich-join** will not be run. To run a job and trigger dependent jobs, define an on-demand trigger.

Run job

5-13. Job 목록 테이블에서 작업을 선택하면 아래 History에 작업 실행 이력이 표시됩니다. Running 상태인 것을 확인하고 다음 단계로 넘어갑니다. (작업은 3 분~8 분 정도 소요됩니다.)

Jobs A job is your business logic required to perform extract, transform and load (ETL) work. Job runs are initiated by triggers which can be scheduled or driven by events.

Showing: 1 - 1 < > 🔍 ⓘ				
Name	ETL language	Script location	Last modified	Job bookmark
transform-cities	python	s3://aws-glue-scripts-987654.....	8 July 2018 7:40 AM UTC+9	Enable

History											
Run ID	Retry attempt	Run status	Error	Logs	Error logs	Execution time	Timeout	Delay	Triggered by	Start time	End time
j_r_5df7d526d2... -		Running ✘		Logs	Error logs	0 secs	2880 mins			8 July 201...	



Transform – 파일포맷 변환 및 데이터 클린징

다음은 Orders 테이블을 Parquet 형식으로 변환하면서 주문량이 공백인 레코드를 제거하는 Job 을 생성하겠습니다. 이번에는 데이터 IO 등 기본구성은 자동코드로 생성하고 Filter API 를 이용하여 레코드를 제거하는 부분을 추가하겠습니다.

5-14. Cities 와 마찬가지 방법으로 raw 의 orders 테이블을 입력으로 하는 Job 을 생성합니다. Add job 을 실행합니다. 작업명은 transform-orders 로 하겠습니다.

5-15. dplab-glue-role IAM 권한을 선택하고 Next 를 클릭합니다.

Job properties

Name
transform-orders

IAM role [?](#)
dplab-glue-role

Ensure this role has permission to your Amazon S3 sources, targets, temporary directory, scripts, and any libraries used by the job. [Create IAM role](#).

This job runs
 A proposed script generated by AWS Glue [?](#)
 An existing script that you provide
 A new script to be authored by you

ETL language
 Python Scala

Script file name
transform-orders

S3 path where the script is stored
s3://aws-glue-scripts-207627707808-ap-northeast-2/tom

5-16. Raw 데이터베이스의 orders 테이블을 입력으로 설정합니다.

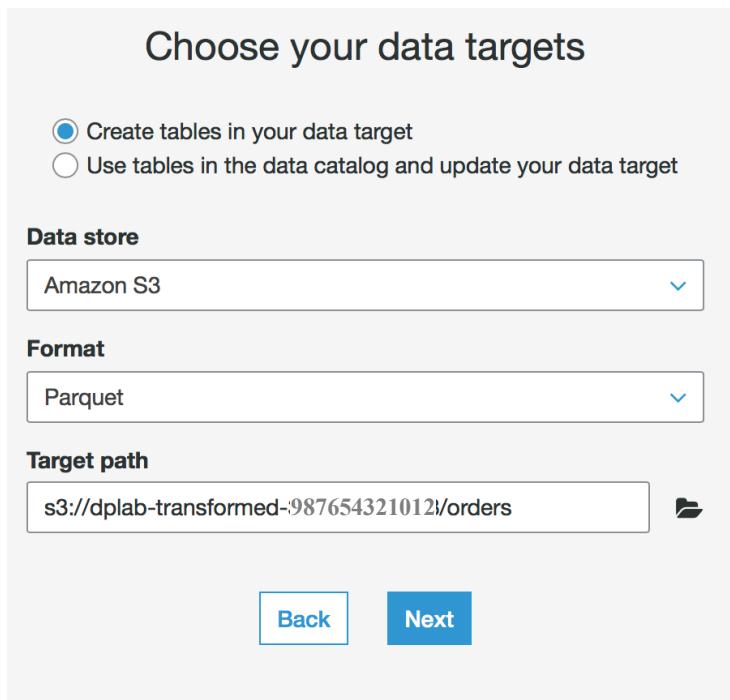
Choose your data sources

Filter by attributes or search by keyword

Showing: 1 - 3 < >

Name	Database	Location	Classification
<input type="radio"/> cities	raw	s3://dplab-raw- 987654321012 /cities/	csv
<input type="radio"/> elb_logs	sampledb	s3://athena-examples-ap-northeast-2/...	Unknown
<input checked="" type="radio"/> orders	raw	s3://dplab-raw- 987654321012 /orders/	csv

5-17. Create tables in your data target 을 선택하고 S3 를 선택하고 Parquet 으로 지정하고 Target path 를 transformed 버킷의 orders 로 설정하고 Next 를 클릭합니다.



5-18. 컬럼매핑에서 Next 를 클릭합니다.

5-19. 내용 리뷰 후 Save job and edit script 화면으로 넘어갑니다.

5-20. 이제 datasource0 와 applymapping1 사이에 filter 명령을 이용하여 주문량이 0 인 건을 제거하는 코드를 추가하겠습니다. 21 번 라인에 아래 코드를 (주석라인을 포함하여) 붙여넣기 합니다.

```
## @type: Filter
## @args: [f = lambda x: x["quantity"] > 0, transformation_ctx = "fitered_ctx"]
## @return: filtered
## @inputs: [frame=datasource0]
filtered = Filter.apply(frame = datasource0, f = lambda x: x["quantity"] > 0, transformation_ctx = "fitered_ctx")
```

glueContext.create_dynamic_frame.from_catalog 를 통해 raw 데이터를 로딩하는 아래 코드의 바로 다음 라인입니다. (작업 시점에 따라 자동 생성된 코드의 라인 번호가 다를 수 있으므로 주의합니다.)

```
datasource0 = glueContext.create_dynamic_frame.from_catalog(database = "raw", table_name = "orders",
transformation_ctx = "datasource0")
```

파란색 글씨에 주목합니다. datasource0 를 입력으로 하여 quantity(주문량) 컬럼의 값이 0 보다 큰 건만 필터링하고 filtered 라는 이름의 DynamicFrame 을 생성하도록 되어있습니다.

5-21. 다음 step 인 ApplyMapping 의 입력 데이터를 datasource0 에서 filtered 로 바꾸어줍니다. 30 라인입니다. 해당 라인을 찾아 직접 수정합니다. (아래 코드에서 파란색 글씨 참조. 다이어그램 생성을 위해 주석부분의 @inputs 항목도 업데이트하였습니다.)

```
## @type: ApplyMapping
## @args: [mapping = [("year", "l...(중략)...transformation_ctx = "applymapping1")]
## @return: applymapping1
## @inputs: [frame = filtered]
applymapping1 = ApplyMapping.apply(frame = filtered, mappings = [("year", ...(중략)", "double")]),
transformation_ctx = "applymapping1")
```

5-22. 수정된 최종 코드는 아래와 같습니다. (마지막 s3 버킷 경로의 account_id 부분을 본인의 버킷으로 수정 후 그대로 전체 붙여넣기 하여도 됩니다. 코드 복사는 별도 제공된 code.txt 파일을 사용하십시오. pdf 파일에서 복사할 경우 라인피드 등이 맞지 않을 수 있습니다.)

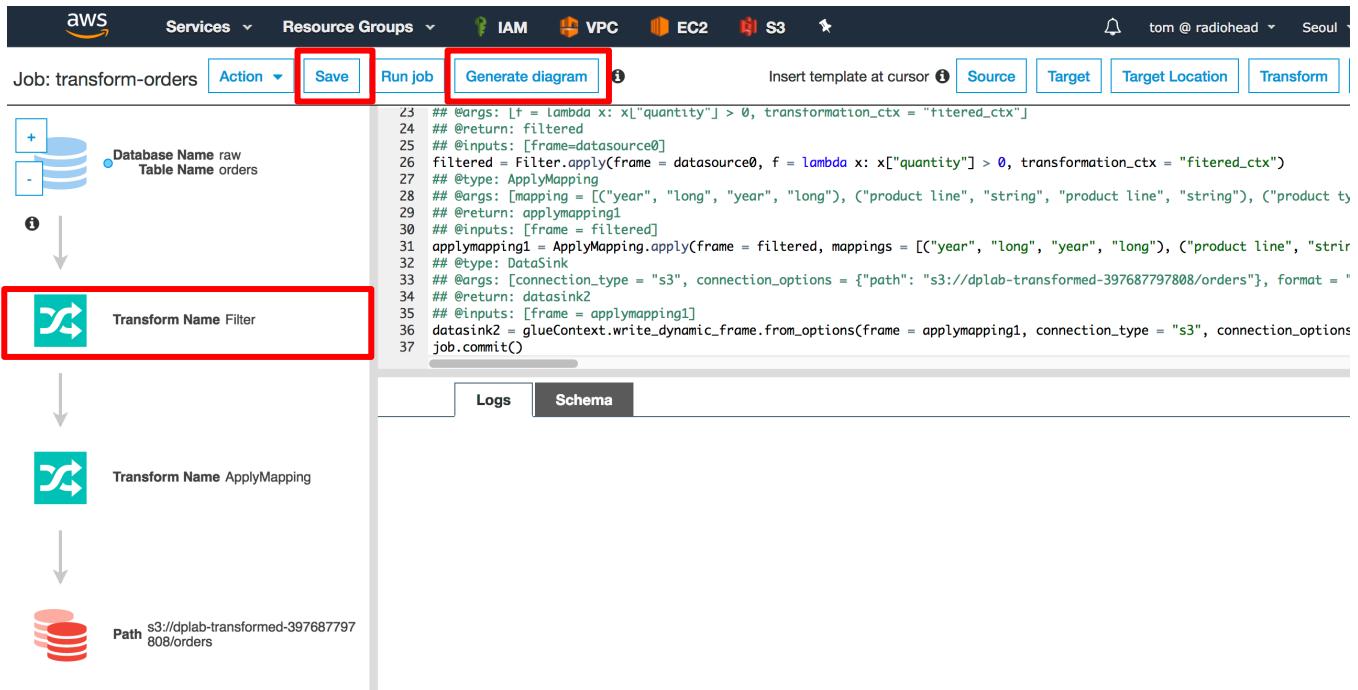
```
import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job

## @params: [JOB_NAME]
args = getResolvedOptions(sys.argv, ['JOB_NAME'])

sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)
job.init(args['JOB_NAME'], args)
## @type: DataSource
## @args: [database = "raw", table_name = "orders", transformation_ctx = "datasource0"]
## @return: datasource0
## @inputs: []
datasource0 = glueContext.create_dynamic_frame.from_catalog(database = "raw", table_name = "orders",
transformation_ctx = "datasource0")
## @type: Filter
## @args: [f = lambda x: x["quantity"] > 0, transformation_ctx = "fitered_ctx"]
## @return: filtered
## @inputs: [frame=datasource0]
filtered = Filter.apply(frame = datasource0, f = lambda x: x["quantity"] > 0, transformation_ctx =
"fitered_ctx")
## @type: ApplyMapping
## @args: [mapping = [("year", "long", "year", "long"), ("product line", "string", "product line",
"string"), ("product type", "string", "product type", "string"), ("product", "string", "product",
"string"), ("order method type", "string", "order method type", "string"), ("retailer country",
"string", "retailer country", "string"), ("revenue", "double", "revenue", "double"), ("planned revenue",
"double", "planned revenue", "double"), ("product cost", "double", "product cost", "double"),
("unit cost", "double", "unit cost", "double"), ("unit price", "double", "unit price", "double"),
("gross profit", "double", "gross profit", "double"), ("unit sale price", "double", "unit sale
price", "double")], transformation_ctx = "applymapping1"]
## @return: applymapping1
## @inputs: [frame = filtered]
applymapping1 = ApplyMapping.apply(frame = filtered, mappings = [("year", "long", "year", "long"),
("product line", "string", "product line", "string"), ("product type", "string", "product type",
"string"), ("product", "string", "product", "string"), ("order method type", "string", "order method
type", "string"), ("retailer country", "string", "retailer country", "string"), ("revenue", "double",
"revenue", "double"), ("planned revenue", "double", "planned revenue", "double"), ("product cost",
"double", "product cost", "double"), ("unit cost", "double", "unit cost", "double"), ("unit price",
"double", "unit price", "double"), ("gross profit", "double", "gross profit", "double"),
("unit sale price", "double", "unit sale price", "double")],
transformation_ctx = "applymapping1")
## @type: ResolveChoice
```

```
## @args: [choice = "make_struct", transformation_ctx = "resolvechoice2"]
## @return: resolvechoice2
## @inputs: [frame = applymapping1]
resolvechoice2 = ResolveChoice.apply(frame = applymapping1, choice = "make_struct", transformation_ctx =
"resolvechoice2")
## @type: DropNullFields
## @args: [transformation_ctx = "dropnullfields3"]
## @return: dropnullfields3
## @inputs: [frame = resolvechoice2]
dropnullfields3 = DropNullFields.apply(frame = resolvechoice2, transformation_ctx = "dropnullfields3")
## @type: DataSink
## @args: [connection_type = "s3", connection_options = {"path": "s3://dplab-transformed-
[account]/orders"}, format = "parquet", transformation_ctx = "datasink4"]
## @return: datasink4
## @inputs: [frame = dropnullfields3]
datasink4 = glueContext.write_dynamic_frame.from_options(frame = dropnullfields3, connection_type =
"s3", connection_options = {"path": "s3://dplab-transformed-[account_id]/orders"}, format = "parquet",
transformation_ctx = "datasink4")
job.commit()
```

5-23. 상단의 Save 버튼과 Generate diagram 버튼을 차례로 누릅니다. 주석정보를 파싱하여 왼쪽의 플로우다이어그램이 업데이트됩니다.



5-24. X 를 눌러 편집창을 빠져나온 후 transform-orders 작업을 선택하고 Action > Run job 버튼을 눌러 작업을 실행합니다

Jobs A job is your business logic required to perform extract, transform and load (ETL) work. Job runs are initiated by triggers which can be scheduled or driven by events.

The screenshot shows the AWS Glue Jobs list and the History tab of a specific job run. The top navigation bar includes 'Add job' and 'Action' dropdown, a search bar, and pagination 'Showing: 1 - 2'. The main table lists two jobs: 'transform-orders' and 'transform-cities'. The 'transform-orders' row is selected, showing details like ETL language (python), script location (s3://aws-glue-scripts-9876543...), last modified (8 July 2018 7:52 AM UTC+9), and job bookmark (Enable). The History tab shows one run with ID 'jr_65fac95743...' in a 'Running' state. The run details table includes columns for Run ID, Retry attempt, Run status, Error, Logs, Error logs, Execution time, Timeout, Delay, Triggered by, Start time, and End time. The 'Run status' column for the running job is highlighted with a red box.

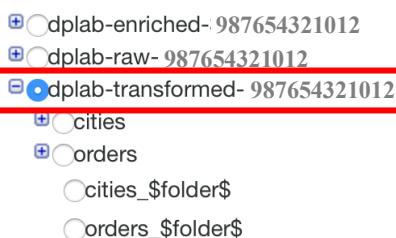
Transformed data crawler

3 번 단계에서 raw 데이터에 대한 crawler 를 생성한 것과 유사한 방법으로 dplab-transformed-[account] 버킷에 대한 crawler 를 생성합니다.

5-25. Crawler 메뉴로 이동하여 Add Crawler 를 클릭합니다.

5-26. Crawler name 을 dplab-transformed-crawler 로 입력하고 Next 를 클릭합니다.

5-27. 데이터소스의 include path 를 dplab-transformed-[account] 로 선택하고 Next 를 클릭합니다.



5-28. Add another store 를 No 로 선택합니다.

5-29 Next 를 클릭합니다.

5-30. Choose an existing IAM role 을 선택하여 dplab-glue-role 을 적용합니다.

5-31. Schedule 은 Run on demand 로 합니다.

5-32. Output Database 를 transformed 로 선택하고 Next 와 Finish 를 눌러 Crawler 를 생성합니다.

5-33. Crawling 작업을 실행하기 전에 Transform ETL 작업이 완료되었는지 확인합니다.

Jobs A job is your business logic required to perform extract, transform and load (ETL) work. Job runs are initiated by triggers which can be scheduled or driven by events.

The screenshot shows the AWS Glue Jobs console. At the top, there are buttons for 'Add job' and 'Action' with a dropdown, and a search bar labeled 'Filter by attributes'. Below this is a table with columns: Name, ETL language, Script location, Last modified, and Job bookmark. Two jobs are listed: 'transform-cities' (python, last modified 8 July 2018 7:40 AM UTC+9, Enabled) and 'transform-orders' (python, last modified 8 July 2018 7:52 AM UTC+9, Enabled). The 'transform-orders' row has a checked checkbox in the first column.

Below the table is a navigation bar with tabs: History (selected), Details, and Script. Under 'History', it says 'Showing: 1 - 1'. A single run is listed in a table with columns: Run ID, Retry attempt, Run status, Error, Logs, Error logs, Execution time, Timeout, Delay, Triggered by, Start time, and End time. The run is identified by 'jr_65fac95743...' and is marked as 'Succeeded'. The 'Logs' link is highlighted in blue. The execution time was 47 secs, and it started and ended on 8 July 2018 at 7:52 AM UTC+9.

5-34. 작업이 완료되었으면 Crawler 작업을 실행합니다. Crawler 화면에서 dplab-transformed-crawler 작업 옆의 체크박스를 클릭하고 상단의 Run crawler 버튼을 클릭합니다.

Crawlers

A crawler connects to a data store, progresses through a prioritized list of classifiers to determine the schema for your data, and then creates metadata tables in your data catalog.

The screenshot shows the AWS Glue Crawlers console. At the top, there are buttons for 'Add crawler' and 'Run crawler' with a dropdown, and a search bar. Below this is a table with columns: Name, Schedule, Status, Logs, Last runtime, Median runtime, Tables updated, and Tables added. Two crawlers are listed: 'dplab-raw-cralwer' (Ready, Logs, 22 secs, 22 secs, 0, 0) and 'dplab-transforme...' (Starting, Logs, 0 secs, 0 secs, 0, 0).

5-35. 작업이 완료되면 Databases 메뉴에서 transformed Database 에 두 테이블이 추가된 것을 확인할 수 있습니다. (Database에서 trnasfomed 클릭 후 Tables in transformed 클릭)

Tables A table is the metadata definition that represents your data, including its schema. A table can be used as a source or target in a job definition.

The screenshot shows the AWS Glue Tables console. At the top, there are buttons for 'Add tables' and 'Action' with a dropdown, and a search bar. Below this is a table with columns: Name, Database, Location, Classification, Last updated, and Deprecated. Two tables are listed: 'cities' (transformed, s3://dplab-transformed-987654321012/cities/, parquet, 8 July 2018 8:53 AM UTC+9...) and 'orders' (transformed, s3://dplab-transformed-987654321012/orders/, parquet, 8 July 2018 8:53 AM UTC+9...).

Transform – 조인을 통한 새로운 데이터 생성

다음은 transformed 버킷의 orders 와 cities 테이블을 조인하는 작업을 생성하겠습니다. (앞 단계의 Crawling 작업이 정상적으로 진행된 후 진행하도록 합니다.) 마찬가지로 자동코드생성 기능을 이용하여 기본 코드를 만들겠습니다. 이번에는 Script Editor 창에서 코드를 추가하는 기능을 활용해 보겠습니다.

5-36. Glue 콘솔에서 ETL > Add job 을 클릭하고 작업명을 enrich-join 으로 입력합니다. IAM Role 은 dplab-glue-role 로 선택합니다.

Job properties

Name
enrich-join

IAM role ⓘ
dplab-glue-role

Ensure this role has permission to your Amazon S3 sources, targets, temporary directory, scripts, and any libraries used by the job. [Create IAM role](#).

This job runs

- A proposed script generated by AWS Glue ⓘ
- An existing script that you provide
- A new script to be authored by you

ETL language

- Python
- Scala

Script file name
enrich-join

S3 path where the script is stored

5-37. 입력 데이터소스는 transformed 의 orders 를 선택합니다. (Database 를 주의합니다.)

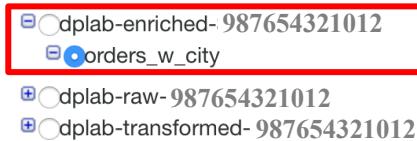
Choose your data sources

Filter by attributes or search by keyword

Showing: 1 - 5 < >

Name	Database	Location	Classification
<input type="radio"/> cities	raw	s3://dplab-raw-987654321012/cities/	csv
<input type="radio"/> cities	transformed	s3://dplab-transformed-987654321012.../	parquet
<input type="radio"/> elb_logs	sampledb	s3://athena-examples-ap-northeast-2/...	Unknown
<input type="radio"/> orders	raw	s3://dplab-raw-987654321012/orders/	csv
<input checked="" type="radio"/> orders	transformed	s3://dplab-transformed-987654321012.../	parquet

5-38. Target 은 Create tables in your data target 을 선택하고 Data store 를 S3 로, Format 은 Parquet 으로, Target path 를 dplab-enriched-[account] 버킷의 orders_w_city 로 선택하고 Next 를 클릭합니다.



5-39. Mapping 화면에서 Next 클릭합니다.

5-40. 내용을 리뷰하고 Save job and edit script 화면으로 이동합니다.

5-41. 21 번라인 맨 앞에 커서를 위치시킨 후 오른쪽 상단의 Source 버튼을 클릭합니다.

Source

5-42. transformed database 의 cities 를 클릭한 후 Add 버튼을 누릅니다.

Add source table

Name	Database	Location	Classification
cities	raw	s3://dplab-raw-9876543....	csv
cities	transformed	s3://dplab-transformed-...	parquet
elb_logs	sampledb	s3://athena-examples-a...	Unknown
orders	raw	s3://dplab-raw-9876543....	csv
orders	transformed	s3://dplab-transformed-...	parquet

Add

5-43. 조인변환을 위해 transformed 데이터베이스의 cities 테이블을 로드하겠습니다. 25 번 라인의 <output>을 datasource1 으로 업데이트합니다. 25 번 라인 맨 오른쪽의 transformation_ctx 파라미터도 datasource1 으로 업데이트합니다. (22, 23 번 라인의 주석부분도 변경해 줍니다.)

```
# @type: DataSource
## @args: [database = "transformed", table_name = "cities", redshift_tmp_dir = args["TempDir"],
transformation_ctx = "datasource1"]
```

```
## @return: datasource1
## @inputs: []
datasource1 = glueContext.create_dynamic_frame.from_catalog(database = "transformed", table_name =
"cities", redshift_tmp_dir = args["TempDir"], transformation_ctx = "datasource1")
```

5-44. 조인작업을 생성할 차례입니다. 커서를 26 번 라인에 위치시킨 후 Transform 버튼을 클릭합니다.

Transform

5-45. 변환함수로 Join 을 선택하고 Create 버튼을 누릅니다.

Add transform

Choose a transform. [Learn more](#)

Name	Description
<input type="radio"/> ApplyMapping	Apply mapping to a DynamicFrame
<input type="radio"/> DropFields	Drop fields from a DynamicFrame
<input type="radio"/> DropNullFields	DynamicFrame without null fields.
<input type="radio"/> Filter	Builds a new DynamicFrame by selecting records from the input frame that satisfy the predicate function
<input checked="" type="radio"/> Join	Join two DynamicFrames
<input type="radio"/> Map	Builds a new DynamicFrame by applying a function to all records in the input DynamicFrame
<input type="radio"/> MapToCollection	Apply a transform to each DynamicFrame in this DynamicFrameCollection
<input type="radio"/> Relationalize	Flatten nested schema and pivot out array columns from the flattened frame
<input type="radio"/> RenameField	Rename a field within a DynamicFrame

Create

5-46. 자동 생성된 코드에서 <output>과 <Transformation_ctx> 을 'joined'로 변경합니다. <frame1>을 datasource0 로, <frame2>를 datasource1 으로 변경하여 조인대상 테이블을 정의합니다. 또, <keys1>은 'retailer country'로 <keys2>는 'country'로 변경하여 조인 조건을 입력합니다. (또는 아래 코드를 26~30 라인에 그대로 붙여넣으면 됩니다.)

```
## @type: Join
## @args: [keys1 = ['retailer_country'], keys2 = ['country']]
## @return: joined
## @inputs: [frame1 = datasource0, frame2 = datasource1]
joined = Join.apply(frame1 = datasource0, frame2 = datasource1, keys1 = ['retailer_country'], keys2 = ['country'], transformation_ctx = "joined")
```

5-47. 다음 단계의 ApplyMapping 입력 데이터를 datasource0 가 아닌 joined로 업데이트하여 앞서 생성한 결과 데이터를 활용하도록 변경합니다. 그리고, 조인을 통해 cities 컬럼이 추가되었으므로, 아래 코드의 파란색 부분과 같이 mappings 파라미터에 cities 컬럼들을 추가합니다. (자동생성 코드의 31 번~35 번 라인을 아래 코드로 대체하면 됩니다.)

```
## @type: ApplyMapping
## @args: [mapping = [("year", "long", "year", "long"), ("product_line", "string", "product_line", "string"), ("product_type", "string", "product_type", "string"), ("product", "string", "product", "string"), ("order_method_type", "string", "order_method_type", "string"), ("retailer_country", "string", "retailer_country", "string"), ("revenue", "double", "revenue", "double"), ("planned_revenue", "double", "planned_revenue", "double"), ("product_cost", "double", "product_cost", "double"), ("quantity", "long", "quantity", "long"), ("unit_cost", "double", "unit_cost", "double"), ("unit_price", "double", "unit_price", "double"), ("gross_profit", "double", "gross_profit", "double"), ("unit_sale_price", "double", "unit_sale_price", "double"), ("retail_id", "long", "retail_id", "long"), ("country", "string", "country", "string"), ("latitude", "string", "latitude", "string"), ("longitude", "string", "longitude", "string"), ("population", "long", "population", "long"), ("last_updated", "string", "last_updated", "string")], transformation_ctx = "applymapping1"]
## @return: applymapping1
## @inputs: [frame = joined]
applymapping1 = ApplyMapping.apply(frame = joined, mappings = [("year", "long", "year", "long"), ("product_line", "string", "product_line", "string"), ("product_type", "string", "product_type", "string"), ("product", "string", "product", "string"), ("order_method_type", "string", "order_method_type", "string"), ("retailer_country", "string", "retailer_country", "string"), ("revenue", "double", "revenue", "double"), ("planned_revenue", "double", "planned_revenue", "double"), ("product_cost", "double", "product_cost", "double"), ("quantity", "long", "quantity", "long"), ("unit_cost", "double", "unit_cost", "double"), ("unit_price", "double", "unit_price", "double"), ("gross_profit", "double", "gross_profit", "double"), ("unit_sale_price", "double", "unit_sale_price", "double"), ("retail_id", "long", "retail_id", "long"), ("country", "string", "country", "string"), ("latitude", "string", "latitude", "string"), ("longitude", "string", "longitude", "string"), ("population", "long", "population", "long"), ("last_updated", "string", "last_updated", "string")], transformation_ctx = "applymapping1")
```

5-48. 수정된 전체 코드는 다음과 같습니다. (마지막 라인의 S3 경로의 account_id 부분을 본인의 버킷으로 수정 후 아래 코드 전체를 붙여넣기해도 됩니다. 코드 복사는 별도 제공된 code.txt 파일을 사용하십시오. pdf 파일에서 복사할 경우 라인피드 등이 맞지 않을 수 있습니다.)

```
import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job

## @params: [JOB_NAME]
args = getResolvedOptions(sys.argv, ['JOB_NAME'])

sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)
job.init(args['JOB_NAME'], args)
## @type: DataSource
## @args: [database = "transformed", table_name = "orders", transformation_ctx = "datasource0"]
## @return: datasource0
## @inputs: []
datasource0 = glueContext.create_dynamic_frame.from_catalog(database = "transformed", table_name = "orders", transformation_ctx = "datasource0")
## @type: DataSource
## @args: [database = "transformed", table_name = "cities", redshift_tmp_dir = args["TempDir"], transformation_ctx = "datasource1"]
## @return: datasource1
```



```

## @inputs: []
datasource1 = glueContext.create_dynamic_frame.from_catalog(database = "transformed", table_name =
"cities", redshift_tmp_dir = args["TempDir"], transformation_ctx = "datasource1")
## @type: Join
## @args: [keys1 = ['retailer_country'], keys2 = ['country']]
## @return: joined
## @inputs: [frame1 = datasource0, frame2 = datasource1]
joined = Join.apply(frame1 = datasource0, frame2 = datasource1, keys1 = ['retailer_country'], keys2 =
['country'], transformation_ctx = "joined")
## @type: ApplyMapping
## @args: [mapping = [("year", "long", "year", "long"), ("product_line", "string", "product_line",
"string"), ("product_type", "string", "product_type", "string"), ("product", "string", "product",
"string"), ("order_method_type", "string", "order_method_type", "string"), ("retailer_country",
"string", "retailer_country", "string"), ("revenue", "double", "revenue", "double"), ("planned_revenue",
"double", "planned_revenue", "double"), ("product_cost", "double", "product_cost", "double"),
("quantity", "long", "quantity", "long"), ("unit_cost", "double", "unit_cost", "double"), ("unit_price",
"double", "unit_price", "double"), ("gross_profit", "double", "gross_profit", "double"),
("unit_sale_price", "double", "unit_sale_price", "double"), ("retail_id", "long", "retail_id", "long"),
("country", "string", "country", "string"), ("latitude", "string", "latitude", "string"), ("longitude",
"string", "longitude", "string"), ("population", "long", "population", "long"), ("last_updated",
"string", "last_updated", "string")], transformation_ctx = "applymapping1"]
## @return: applymapping1
## @inputs: [frame = joined]
applymapping1 = ApplyMapping.apply(frame = joined, mappings = [("year", "long", "year", "long"),
("product_line", "string", "product_line", "string"), ("product_type", "string", "product_type",
"string"), ("product", "string", "product", "string"), ("order_method_type", "string",
"order_method_type", "string"), ("retailer_country", "string", "retailer_country", "string"),
("revenue", "double", "revenue", "double"), ("planned_revenue", "double", "planned_revenue", "double"),
("product_cost", "double", "product_cost", "double"), ("quantity", "long", "quantity", "long"),
("unit_cost", "double", "unit_cost", "double"), ("unit_price", "double", "unit_price", "double"),
("gross_profit", "double", "gross_profit", "double"), ("unit_sale_price", "double", "unit_sale_price",
"double"), ("retail_id", "long", "retail_id", "long"), ("country", "string", "country", "string"),
("latitude", "string", "latitude", "string"), ("longitude", "string", "longitude", "string"),
("population", "long", "population", "long"), ("last_updated", "string", "last_updated", "string")],
transformation_ctx = "applymapping1")
## @type: ResolveChoice
## @args: [choice = "make_struct", transformation_ctx = "resolvechoice2"]
## @return: resolvechoice2
## @inputs: [frame = applymapping1]
resolvechoice2 = ResolveChoice.apply(frame = applymapping1, choice = "make_struct", transformation_ctx =
"resolvechoice2")
## @type: DropNullFields
## @args: [transformation_ctx = "dropnullfields3"]
## @return: dropnullfields3
## @inputs: [frame = resolvechoice2]
dropnullfields3 = DropNullFields.apply(frame = resolvechoice2, transformation_ctx = "dropnullfields3")
## @type: DataSink
## @args: [connection_type = "s3", connection_options = {"path": "s3://dplab-enriched-[account_id]/orders_w_city"}, format = "parquet", transformation_ctx = "datasink4"]
## @return: datasink4
## @inputs: [frame = dropnullfields3]
datasink4 = glueContext.write_dynamic_frame.from_options(frame = dropnullfields3, connection_type =
"s3", connection_options = {"path": "s3://dplab-enriched-[account_id]/orders_w_city"}, format =
"parquet", transformation_ctx = "datasink4")
job.commit()

```

5-49. Save 를 누르고 Run job 을 클릭하여 작업을 실행합니다.



Jobs A job is your business logic required to perform extract, transform and load (ETL) work. Job runs are initiated by triggers which can be scheduled or driven by events.

The screenshot shows two parts of the AWS Glue interface. The top part is a list of jobs with columns: Name, ETL language, Script location, Last modified, and Job bookmark. It includes rows for 'transform-cities' (python, s3://aws-glue-scripts- 9876543..., 8 July 2018 7:40 AM UTC+9, Enable), 'enrich-join' (python, s3://aws-glue-scripts- 9876543..., 8 July 2018 9:03 AM UTC+9, Enable), and 'transform-orders' (python, s3://aws-glue-scripts- 9876543..., 8 July 2018 7:52 AM UTC+9, Enable). The 'enrich-join' row has a checked checkbox. The bottom part shows the 'Details' tab for the 'enrich-join' job, displaying a single run with status 'Running'. The 'Logs' and 'Error logs' buttons are highlighted with a red box.

Name	ETL language	Script location	Last modified	Job bookmark
transform-cities	python	s3://aws-glue-scripts- 9876543...	8 July 2018 7:40 AM UTC+9	Enable
enrich-join	python	s3://aws-glue-scripts- 9876543...	8 July 2018 9:03 AM UTC+9	Enable
transform-orders	python	s3://aws-glue-scripts- 9876543...	8 July 2018 7:52 AM UTC+9	Enable

Run ID	Retry attempt	Run status	Error	Logs	Error logs	Execution time	Timeout	Delay	Triggered by	Start time	End time
jr_c9b9b7b1c...	-	Running		Logs	Error logs	0 secs	2880 mins			8 July 201...	

Enriched data crawler

마지막으로 새롭게 생성된 데이터에 대한 카탈로그를 생성합니다.

5-50. Glue 콘솔에서 Add crawler 를 클릭합니다.

5-51. Crawler 의 이름은 **dplab-enriched-crawler** 로 합니다.

5-52. 데이터소스 선택시에는 이번에는 버킷(dplab-enriched-[account])이 아닌 폴더(orders_w_city)를 선택하도록 합니다

The screenshot shows the AWS Glue Data Catalog interface. A red box highlights the 'orders_w_city' folder under the 'dplab-enriched-' bucket. Other options like 'dplab-raw-' and 'dplab-transformed-' are also listed.

- dplab-enriched- 987654321012
- orders_w_city
- orders_w_city_\$folder\$
- dplab-raw- 987654321012
- dplab-transformed- 987654321012

Glue 가 내부 폴더를 파티션으로 인식하지 않도록 하기 위하여 폴더를 직접 선택하였습니다. 내부 폴더가 하나인 경우에 Glue 는 해당 폴더가 파티션구분인지 테이블 구분인지 알 수 없습니다. 본 실습은 파티션 구성은 하지 않을 예정이어서 명시적으로 소스 폴더를 지정하여 진행하겠습니다. (Crawler 에서 파티션 인식에 대한 내용은 다음 링크를 참조하십시오. https://docs.aws.amazon.com/ko_kr/athena/latest/ug/glue-best-practices.html#schema-crawlers-data-sources)

5-53. Add another data store 를 No 상태로 Next 를 클릭합니다.

5-54. dplab-glue-role 을 선택하여 지정합니다.

5-55. Run on demand 로 진행합니다.

5-56. Output database 를 enriched 로 선택하고 Next 를 클릭합니다.



5-57. Finish 를 클릭한 후 이전 단계의 ETL 작업(enrich-join)이 완료되는 것을 확인합니다.

5-58. ETL 작업이 완료되면 Crawler 작업을 실행합니다.

Jobs A job is your business logic required to perform extract, transform and load (ETL) work. Job runs are initiated by triggers which can be scheduled or driven by events.

Add job	Action	Filter by attributes	Showing: 1 - 3 < > 🔍 🌐		
Name	ETL language	Script location	Last modified	Job bookmark	
transform-orders	python	s3://aws-glue-scripts- 9876543...	8 July 2018 7:52 AM UTC+9	Enable	
<input checked="" type="checkbox"/> enrich-join	python	s3://aws-glue-scripts- 9876543...	8 July 2018 9:03 AM UTC+9	Enable	
transform-cities	python	s3://aws-glue-scripts- 9876543...	8 July 2018 7:40 AM UTC+9	Enable	

History	Details	Script									
Showing: 1 - 1 < > 🔍											
Run ID	Retry attempt	Run status	Error	Logs	Error logs	Execution time	Timeout	Delay	Triggered by	Start time	End time
jr_c9b9b7b1c...	-	Succeeded		Logs		56 secs	2880 mins			8 July 201...	8 July 201...

Crawlers

A crawler connects to a data store, progresses through a prioritized list of classifiers to determine the schema for your data, and then creates metadata tables in your data catalog.

Crawler "dplab-enriched-crawler" is now running.
--

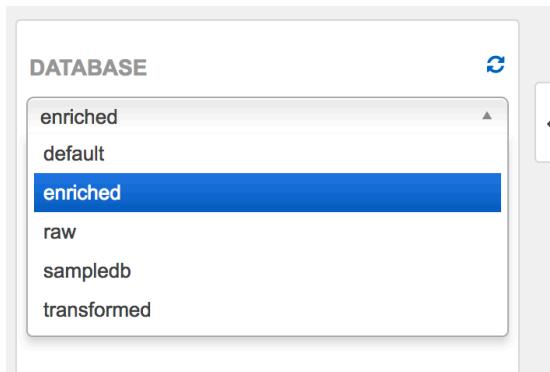
Add crawler	Run crawler	Action	Showing: 1 - 3 < > 🔍 🌐					
Name	Schedule	Status	Logs	Last runtime	Median runtime	Tables updated	Tables added	
dplab-enriched-cr...		⌚ Starting		0 secs	0 secs	0	0	
dplab-raw-cralwer		Ready	Logs	22 secs	22 secs	0	0	
dplab-transforme...		Ready	Logs	21 secs	21 secs	0	2	

6. ETL Job 결과 확인 (Athena, Optional)

4 단계에서 Athena 연결을 진행한 경우 Transform 및 Join 작업결과를 Athena에서 SQL로 쿼리하여 확인할 수 있습니다.

6-1. Athena 서비스 콘솔로 이동합니다.

6-2. DATABASE 오른쪽의 리프레시 버튼을 클릭하면 transformed, enriched 데이터베이스에 테이블이 추가된 것을 확인할 수 있습니다. 최종 작업결과를 보기위해 enriched database 를 선택합니다.



6-3. orders_w_city 테이블 옆의 ...을 클릭하고 Preview table 을 선택합니다. 생성된 쿼리를 실행하면 Join 된 결과를 확인할 수 있습니다. 결과테이블의 오른쪽으로 이동하여 오더 정보에 도시 정보가 조인된 것을 확인합니다.

year	product_line	product_type	product	order_method_type	retailer	country	revenue	planned_revenue	product_cost	quan	
3	3240.0	324	10.0	19.29	2822.04	18.71	21	Austria	48°13'N 16°22'E	8210281	0001-01-01
.05	176220.0	445	396.0	650.89	95839.89	613.6136364	21	Austria	48°13'N 16°22'E	8210281	0001-01-01
1	35040.0	2190	16.0	33.59	36129.69	32.514	21	Austria	48°13'N 16°22'E	8210281	0001-01-01
.45	60492.65	3645	17.08111111	27.81	31927.49	25.92888889	21	Austria	48°13'N 16°22'E	8210281	0001-01-01
.36	9434.48	604	15.62	35.09	10942.98	33.76	21	Austria	48°13'N 16°22'E	8210281	0001-01-01
.55	6279.0	161	39.0	81.55	6173.03	77.47333333	21	Austria	48°13'N 16°22'E	8210281	0001-01-01
3	13193.52	1124	11.73	20.15	9455.08	20.15	21	Austria	48°13'N 16°22'E	8210281	0001-01-01
.0	175016.26	7414	23.55588235	40.5	125250.74	40.5	21	Austria	48°13'N 16°22'E	8210281	0001-01-01
3	3600.09	153	23.53	40.52	2351.61	38.9	21	Austria	48°13'N 16°22'E	8210281	0001-01-01
.4	110999.63	629	176.47	359.6	99356.84	334.43	21	Austria	48°13'N 16°22'E	8210281	0001-01-01

6-4. 다음 쿼리를 수행하여 레코드 건수를 비교합니다.

```
SELECT 'raw.cities', count(*) FROM "raw"."cities" UNION
SELECT 'raw.orders', count(*) FROM "raw"."orders" UNION
SELECT 'transformed.cities', count(*) FROM "transformed"."cities" UNION
SELECT 'transfomred.orders', count(*) FROM "transformed"."orders" UNION
SELECT 'enriched.orders_w_city', count(*) FROM "enriched"."orders_w_city"
```

The screenshot shows the AWS Lambda interface for running queries. The query editor contains the following SQL code:

```
1 SELECT 'raw.cities', count(*) FROM "raw"."cities" UNION
2 SELECT 'raw.orders', count(*) FROM "raw"."orders" UNION
3 SELECT 'transformed.cities', count(*) FROM "transformed"."cities" UNION
4 SELECT 'transfomred.orders', count(*) FROM "transformed"."orders" UNION
5 SELECT 'enriched.orders_w_city', count(*) FROM "enriched"."orders_w_city" |
```

Below the code, a message says "Use Ctrl + Enter to run query, Ctrl + Space to auto-complete". The toolbar includes buttons for "Run query" (which is highlighted in blue), "Save as", "Format query", and "New query". A status message indicates "(Run time: 1.59 seconds, Data scanned: 7.24MB)".

The results section shows the following table:

_col0	_col1
1 raw.cities	21
2 raw.orders	84672
3 transformed.cities	21
4 transfomred.orders	24743
5 enriched.orders_w_city	24743

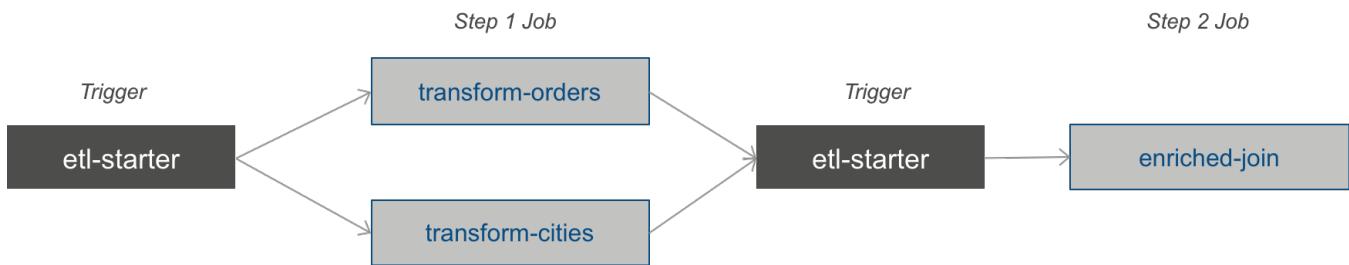
주문량이 Null 인 데이터를 클린징하고 조인하는 작업의 생성 및 수행이 완료되었습니다.

7. ETL Job Scheduling (Optional)

이전 단계까지 총 3 개의 ETL 작업을 생성하고 실행하였습니다. 이번 단계에서는 이 세 작업들에 선후행 조건과 Trigger 조건을 추가해 보겠습니다.

작업 선후행 걸기

작업의 순서는 transform-orders 와 transform-cities 작업이 독립적으로 실행된 후 enriched-join 작업이 후속으로 실행되도록 하겠습니다. 아래 그림을 참고하십시오. 그림에서 회색 박스는 6 단계에서 생성하신 ‘작업’이고 검은색 박스는 본 단계에서 생성할 ‘트리거’입니다.



7-1. ETL 의 Trigger 메뉴로 이동합니다.

7-2. Add Trigger 버튼을 클릭합니다. Name 을 etl-starter 로 입력하고 Trigger type 을 On-demand 로 설정합니다. Next 버튼을 클릭합니다.

Set up your trigger's properties

Name

Trigger type

Schedule Job events On-demand

Choose: Schedule to fire the trigger on a timer, Job events to fire the trigger when job events match your watched list, On-demand to fire the trigger immediately when started.

Next

본 실습가이드에서는 작업 플로우를 수동으로 스타트시킬 것입니다. 만약 시간 기준 스케줄링을 하고 싶다면 Trigger type 을 Schedule 로 선택하고 Frequency 와 Time 설정을 통해 작업 주기와 빈도를 정의할 수 있습니다. (아래 화면 참조)

Set up your trigger's properties

Name
etl-starter

Trigger type
 Schedule Job events On-demand
 Choose: Schedule to fire the trigger on a timer, Job events to fire the trigger when job events match your watched list, On-demand to fire the trigger immediately when started.

Frequency
Daily

Time
05:00 UTC

Next

7-3. Choose jobs to trigger 에서 transform-orders 와 transform-cities 옆의 Add 버튼을 각각 클릭합니다.

Choose jobs to trigger

Choose jobs to start when this trigger fires.

All Jobs	Showing: 1 - 3 < >	Jobs to start	Showing: 1 - 2 < >
transform-orders	Add	transform-cities	✖
enrich-join	Add	transform-orders	✖
transform-cities	Add		

7-4. 나머지 설정은 변경하지 않고 Next 버튼을 클릭합니다.

7-5. 내용 리뷰 후 Finish 버튼을 클릭합니다.

Review

Trigger properties

Name etl-starter
Trigger type On-demand

Jobs to start

Jobs transform-cities, transform-orders

[Back](#)

[Finish](#)

최초 스타트용 Trigger 가 생성되었습니다. 이번에는 후행 enrich-join 작업을 등록하기 위한 Trigger 를 생성하겠습니다.

7-6. Add trigger 버튼을 클릭합니다.

7-7. Name 을 ready-to-join 으로 입력하고 Trigger type 을 Job events 로 선택합니다.

7-8. Choose a job event 를 Succeeded 를 선택한 상태로 두고 Before firing this trigger match 도 All watched job events 로 둡니다.

7-9. 왼쪽 아래 Job 리스트에서 transform-cities 와 transform-orders 옆의 Add 버튼을 각각 클릭하여 선행 조건으로 추가하고 Next 버튼을 클릭합니다.

All Jobs		Watched job events	
Job		Job	Event
transform-cities	Add	transform-cities	Job succeeded
enrich-join	Add	transform-orders	Job succeeded
transform-orders	Add		

7-10. Choose jobs to trigger 화면에서 enrich-join 옆의 Add 를 클릭하고 Next 버튼을 클릭합니다.

All Jobs		Jobs to start	
Job		Job	
transform-orders	Add	enrich-join	X
enrich-join	Add		
transform-cities	Add		

7-11. 리뷰 화면에서 Enable trigger on creation 을 체크하고 Finish 를 클릭합니다.

Review

Trigger properties

Name	ready-to-join
Trigger type	Job events
Trigger matches	All watched job events
Watched job events	transform-cities - Job succeeded, transform-orders - Job succeeded

Jobs to start

Jobs	enrich-join
-------------	-------------

Enable trigger on creation

[Back](#)
[Finish](#)

Job Scheduling Test

7-12. Trigger 메뉴에 2 개의 Trigger 가 생성된 것을 확인합니다. etl-starter trigger 는 CREATED, ready-to-join trigger 는 ACTIVATED 상태일 것입니다.

7-13. etl-starter 옆의 체크박스를 클릭하고 Action 메뉴에서 Start trigger 를 선택합니다.

Triggers A trigger starts a job when it fires.

Trigger "ready-to-join" was created and enabled. ×

Add trigger	Action ▾	Showing: 1 - 2 < > ↻ ?												
<input type="checkbox"/> Trigger name <input checked="" type="checkbox"/> etl-starter <input type="checkbox"/> ready-to-join	Edit trigger Disable trigger Enable trigger Choose jobs to trigger Delete Start trigger	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 25%;">Trigger type</th> <th style="width: 25%;">Trigger status</th> <th style="width: 25%;">Trigger parameters</th> <th style="width: 25%;">Jobs to trigger</th> </tr> </thead> <tbody> <tr> <td>In-demand</td> <td>CREATED</td> <td></td> <td>transform-cities, transform-orders</td> </tr> <tr> <td>Job events</td> <td>ACTIVATED</td> <td>Job events: transform-cities, tran...</td> <td>enrich-join</td> </tr> </tbody> </table>	Trigger type	Trigger status	Trigger parameters	Jobs to trigger	In-demand	CREATED		transform-cities, transform-orders	Job events	ACTIVATED	Job events: transform-cities, tran...	enrich-join
Trigger type	Trigger status	Trigger parameters	Jobs to trigger											
In-demand	CREATED		transform-cities, transform-orders											
Job events	ACTIVATED	Job events: transform-cities, tran...	enrich-join											

만약 램다 등을 통해 외부에서 Job 을 스타트하려는 경우, On-demand 로 Trigger 를 생성한 후 외부에서 API 를 호출을 통해 Start trigger 이벤트를 발생시킬 수 있습니다. API 를 통해 Trigger 를 시작시키는 방법은 다음 링크를 참조하십시오.

<https://docs.aws.amazon.com/glue/latest/dg/aws-glue-api-jobs-trigger.html#aws-glue-api-jobs-trigger-StartTrigger>

https://docs.aws.amazon.com/glue/latest/webapi/API_StartTrigger.html

7-14. ETL > Jobs 메뉴로 이동합니다. transform-cities 또는 transform-orders 작업을 선택하고 아래의 History 탭을 보면 작업이 Running 상태로 바뀐 것을 확인할 수 있습니다.

Jobs A job is your business logic required to perform extract, transform and load (ETL) work. Job runs are initiated by triggers which can be scheduled or driven by events.

The screenshot shows the AWS Glue Jobs list and the History tab for a selected job.

Jobs List:

Name	ETL language	Script location	Last modified	Job bookmark
transform-orders	python	s3://aws-glue-scripts- 9876543...	8 July 2018 7:52 AM UTC+9	Enable
enrich-join	python	s3://aws-glue-scripts- 9876543...	8 July 2018 9:03 AM UTC+9	Enable
<input checked="" type="checkbox"/> transform-cities	python	s3://aws-glue-scripts- 9876543...	8 July 2018 7:40 AM UTC+9	Enable

History Tab:

Run ID	Retry attempt	Run status	Error	Logs	Error logs	Execution time	Timeout	Delay	Triggered by	Start time	End time
jr_6083233e8...	-	Running ✖		Logs	Error logs	0 secs	2880 mins		etl-starter	8 July 201...	
jr_5df7d526d2...	-	Succeeded		Logs		39 secs	2880 mins			8 July 201...	8 July 201...

In the History table, the first row is highlighted with a red box around the 'Run status' column, which shows 'Running'. Another red box highlights the 'Triggered by' column, which shows 'etl-starter'.

7-15. Enrich-join 작업을 선택하고 화면 아래에서 Details 탭을 클릭합니다. Ready-to-join trigger 가 조건으로 설정된 것을 확인합니다.

Jobs A job is your business logic required to perform extract, transform and load (ETL) work. Job runs are initiated by triggers which can be scheduled or driven by events.

Add job	Action	Filter by attributes	Showing: 1 - 3 < >	
<input type="checkbox"/>	Name	ETL language	Script location	Last modified
<input type="checkbox"/>	transform-orders	python	s3://aws-glue-scripts-9876543...	8 July 2018 7:52 AM UTC+9
<input checked="" type="checkbox"/>	enrich-join	python	s3://aws-glue-scripts-9876543...	8 July 2018 9:03 AM UTC+9
<input type="checkbox"/>	transform-cities	python	s3://aws-glue-scripts-9876543...	8 July 2018 7:40 AM UTC+9

History	Details	Script																																
	<table border="1"> <tr><td>Name</td><td>enrich-join</td><td>Python lib path</td><td>-</td></tr> <tr><td>IAM role</td><td>dplab-glue-role</td><td>Jar lib path</td><td>-</td></tr> <tr><td>ETL language</td><td>python</td><td>Other lib path</td><td>-</td></tr> <tr><td>Script location</td><td>s3://aws-glue-scripts-987654321012-ap-northeast-2/tom/enrich-join</td><td>Parameters</td><td>-</td></tr> <tr><td>Temporary directory</td><td>s3://aws-glue-temporary-987654321012-ap-northeast-2/tom</td><td>Connections</td><td>-</td></tr> <tr><td>Job bookmark</td><td>Enable</td><td>DPU</td><td>10</td></tr> <tr><td>Server-side encryption</td><td>Disabled</td><td>Job timeout (minutes)</td><td>2880</td></tr> <tr><td></td><td></td><td>Delay notification threshold (minutes)</td><td>-</td></tr> </table>	Name	enrich-join	Python lib path	-	IAM role	dplab-glue-role	Jar lib path	-	ETL language	python	Other lib path	-	Script location	s3://aws-glue-scripts-987654321012-ap-northeast-2/tom/enrich-join	Parameters	-	Temporary directory	s3://aws-glue-temporary-987654321012-ap-northeast-2/tom	Connections	-	Job bookmark	Enable	DPU	10	Server-side encryption	Disabled	Job timeout (minutes)	2880			Delay notification threshold (minutes)	-	
Name	enrich-join	Python lib path	-																															
IAM role	dplab-glue-role	Jar lib path	-																															
ETL language	python	Other lib path	-																															
Script location	s3://aws-glue-scripts-987654321012-ap-northeast-2/tom/enrich-join	Parameters	-																															
Temporary directory	s3://aws-glue-temporary-987654321012-ap-northeast-2/tom	Connections	-																															
Job bookmark	Enable	DPU	10																															
Server-side encryption	Disabled	Job timeout (minutes)	2880																															
		Delay notification threshold (minutes)	-																															

Automatically run this job if any of the following triggers fire:

Trigger name	Trigger type	Trigger status	Trigger parameters	Jobs to trigger
ready-to-join	Job events	ACTIVATED	Job events: transform-orders, tr...	enrich-join

7-16. transform-cities 와 transform-orders 작업이 완료되면 자동으로 enrich-join 작업이 실행됩니다.
리프레시를 눌러 작업의 진행상태를 확인합니다. (작업이 완료되기를 기다리는 동안 8 번 단계를 미리
진행할 수 있습니다.)

7-17. 수행작업이 완료되면 아래처럼 후속 작업이 자동으로 Triggering 됩니다.

Jobs A job is your business logic required to perform extract, transform and load (ETL) work. Job runs are initiated by triggers which can be scheduled or driven by events.

Add job	Action	Filter by attributes	Showing: 1 - 3 < >	
<input type="checkbox"/>	Name	ETL language	Script location	Last modified
<input type="checkbox"/>	transform-cities	python	s3://aws-glue-scripts-9876543...	8 July 2018 7:40 AM UTC+9
<input type="checkbox"/>	enrich-join	python	s3://aws-glue-scripts-9876543...	8 July 2018 9:03 AM UTC+9
<input checked="" type="checkbox"/>	transform-orders	python	s3://aws-glue-scripts-9876543...	8 July 2018 7:52 AM UTC+9

History	Details	Script																																				
Showing: 1 - 2 < >																																						
	<table border="1"> <thead> <tr> <th>Run ID</th> <th>Retry attempt</th> <th>Run status</th> <th>Error</th> <th>Logs</th> <th>Error logs</th> <th>Execution time</th> <th>Timeout</th> <th>Delay</th> <th>Triggered by</th> <th>Start time</th> <th>End time</th> </tr> </thead> <tbody> <tr> <td>jr_af958386cf...</td> <td>-</td> <td>Succeeded</td> <td></td> <td>Logs</td> <td></td> <td>48 secs</td> <td>2880 mins</td> <td></td> <td>etl-starter</td> <td>8 July 201...</td> <td>8 July 201...</td> </tr> <tr> <td>jr_65fac95743...</td> <td>-</td> <td>Succeeded</td> <td></td> <td>Logs</td> <td></td> <td>47 secs</td> <td>2880 mins</td> <td></td> <td></td> <td>8 July 201...</td> <td>8 July 201...</td> </tr> </tbody> </table>	Run ID	Retry attempt	Run status	Error	Logs	Error logs	Execution time	Timeout	Delay	Triggered by	Start time	End time	jr_af958386cf...	-	Succeeded		Logs		48 secs	2880 mins		etl-starter	8 July 201...	8 July 201...	jr_65fac95743...	-	Succeeded		Logs		47 secs	2880 mins			8 July 201...	8 July 201...	
Run ID	Retry attempt	Run status	Error	Logs	Error logs	Execution time	Timeout	Delay	Triggered by	Start time	End time																											
jr_af958386cf...	-	Succeeded		Logs		48 secs	2880 mins		etl-starter	8 July 201...	8 July 201...																											
jr_65fac95743...	-	Succeeded		Logs		47 secs	2880 mins			8 July 201...	8 July 201...																											



Jobs A job is your business logic required to perform extract, transform and load (ETL) work. Job runs are initiated by triggers which can be scheduled or driven by events.

The screenshot shows the AWS Glue Jobs list and the History tab of a specific job's details. In the Jobs list, there are three entries: 'transform-cities' (python script, last modified 8 July 2018 7:40 AM UTC+9, Enabled), 'enrich-join' (python script, last modified 8 July 2018 9:03 AM UTC+9, Enabled, highlighted with a red box), and 'transform-orders' (python script, last modified 8 July 2018 7:52 AM UTC+9, Enabled). In the History tab, two runs are listed: 'jr_9becb6527...' (Running, triggered by 'ready-to-join', start time 8 July 2018 9:03 AM UTC+9) and 'jr_c9b9b7b1c...' (Succeeded, start time 8 July 2018 9:03 AM UTC+9, end time 8 July 2018 9:03 AM UTC+9). The 'Triggered by' column is also highlighted with a red box.

8. QuickSight dashboard

앞서 ETL 작업을 통해 주문정보와 국가별 retailer 의 위도 경도 정보를 Join 한 orders_w_city 라는 테이블을 생성하였습니다. 이제 이 데이터셋을 바탕으로 지도 위에 주문량을 표시하는 대시보드를 생성해 보겠습니다.

QuickSight 초기 설정

8-1. 콘솔에서 QuickSight 로 이동합니다. 최초 실행시 아래와 같은 화면이 표시됩니다. Sign up for QuickSight 버튼을 클릭합니다.



Your AWS Account is not signed up for QuickSight. Would you like to sign up now?

AWS Account

987654321012

[Sign up for QuickSight](#)

To access QuickSight with a different account, [log in](#) again.



8-2. Standard Edition 으로 진행합니다.

Create your QuickSight account

Standard Enterprise

Edition	Standard	Enterprise
First author with 1GB SPICE	FREE	FREE
Team trial for 60 days (4 authors)*	FREE	FREE
Additional author per month (yearly)**	\$9	\$18
Additional author per month (monthly)**	\$12	\$24
Additional readers (Pay-per-Session)	N/A	\$0.30/session (max \$5/reader/month) ****
Additional SPICE per month	\$0.25 per GB	\$0.38 per GB
Single Sign On with SAML or OpenID Connect	✓	✓
Connect to spreadsheets, databases & business apps	✓	✓
Access data in Private VPCs		✓
Row-level security for dashboards		✓
Hourly refresh of SPICE data		✓
Secure data encryption at rest		✓
Connect to your Active Directory		✓
Use Active Directory Groups ***		✓

* Trial authors are auto-converted to month-to-month subscription upon trial expiry

** Each additional author includes 10GB of SPICE capacity

*** Active Directory groups are available in accounts connected to Active Directory

**** Sessions of 30-minute duration. Total charges for each reader are capped at \$5 per month. [Conditions](#) apply

[Continue](#)

8-3. account name 을 적당한 이름으로 입력합니다. (본인의 어카운트 alias 가 디폴트로 표시됩니다.)

8-4. Notification email address 에 본인의 메일주소를 입력합니다.

8-5. QuickSight capacity region 은 US East(N. Virginia)로 진행합니다.

8-6. S3 데이터 접근을 위해 AWS S3 항목을 체크합니다.

Create your QuickSight account

Edition Standard

QuickSight account name
 ?
 You will need this for you and others to sign in.

Notification email address
 ?
 For QuickSight to send important notifications.

QuickSight capacity region
 ▼ ?
 Select a region.

> Enable autodiscovery of data and users in your Amazon Redshift, Amazon RDS and AWS IAM services.

Amazon Athena
 Enables QuickSight access to Amazon Athena databases

Please ensure the right Amazon S3 buckets are also enabled for QuickSight.

Amazon S3 Choose S3 buckets
 Enables QuickSight to auto-discover your Amazon S3 buckets

Amazon S3 Storage Analytics
 Enables QuickSight to visualize your S3 Storage Analytics data

Amazon IoT Analytics
 Enable QuickSight to visualize your IoT Analytics data

Finish

8-7. 팝업화면에서 접근대상 버킷으로 dplab-enriched-[account]를 선택하고 Select buckets 를 클릭합니다.

dplab-enriched-987654321012

Select buckets

8-8. Go to Amazon QuickSight 버튼을 클릭합니다.

Congratulations! You are signed up for Amazon QuickSight!

Access Quicksight with the following information

Account name: radiohead

Username: tom

Go to Amazon QuickSight

지금 진행한 계정초기설정은 이후 QuickSight 우상단 Manage QuickSight 메뉴를 통해 변경할 수 있습니다.



Community

Send feedback

Help

Sign out

위 링크를 통해 QuickSight에 대한 별도 사용자관리, SPICE 용량관리 등을 할 수 있는 화면으로 연결할 수 있습니다. 사용자관리와 SPICE 관리는 다음 링크를 참고하십시오.

<https://docs.aws.amazon.com/quicksight/latest/user/access-and-authentication.html>

<https://docs.aws.amazon.com/quicksight/latest/user/managing-spice-capacity.html>

Data set 생성

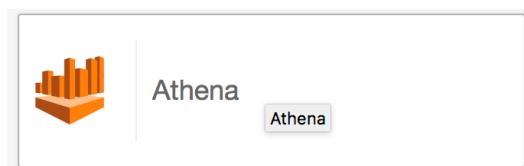
8-9. 오른쪽 위의 Manage data 버튼을 클릭합니다

해당 버튼이 보이지 않는다면 QuickSight 로고를 클릭하여 메인화면으로 돌아온 후 재시도합니다.



8-10. 왼쪽 위의 New data set 버튼을 클릭합니다.

8-11. Athena를 선택합니다.



8-12. 오픈되는 팝업에서 Data source name 을 dplap 으로 입력합니다.

8-13. Validate connection 버튼을 클릭하여 초록색으로 Validate 체크가 완료되는지 확인합니다.





8-14. Create data source 버튼을 클릭합니다.

8-15. Choose your table 팝업에서 enriched 의 orders_w_city 테이블을 선택하고 Select 버튼을 클릭합니다.

Choose your table ×

dplab

Database: contain sets of tables.

enriched ▼

Tables: contain the data you can visualize.

orders_w_city

Edit/Preview data Use custom SQL Select

8-16. Finish data set creation 화면에서 import SPICE for quicker analytics 를 체크하고 Edit/Preview data 버튼을 클릭합니다.

Finish data set creation ×

Table: orders_w_city
 Data source: dplab
 Schema: enriched

Import to SPICE for quicker analytics ✓ 11GB available SPICE

Directly query your data

Edit/Preview data Visualize

Data set editing

본 실습에서는 대시보드 생성시 지도 위에 주문량을 표시하려고 합니다. 지도 위에 위치 좌표를 표시하기 위해서는 위도/경도 정보가 필요하며 orders_w_city 테이블은 앞서 Glue ETL 작업을 통해 주문 거점의 위도/경도 정보가 조인을 통해 생성되어 있습니다. 이번 단계에서는 해당 위도/경도 정보를 위치정보로 페어링시키는 작업을 하게 됩니다. (미국내 도시인 경우에는 위도/경도 정보를 지정하지 않아도

QuickSight에서 자동으로 위치정보를 해석하여 표시할 수 있습니다. 하지만 보다 정교한 위치정보 제어가 필요하거나 본 실습에서와 같이 미국이 아닌 다른 도시의 위치를 지정하기 위해서는 명시적으로 위치정보를 지정해야 합니다.)

8-17. 앞 단계에서 Data source editing 화면으로 이동하였을 것입니다. (다른 화면에 있다면 QuickSight 로고를 클릭한 후 우 상단의 Manage data 버튼을 클릭한 후 orders_w_city 아이콘을 선택하여 아래 화면으로 이동할 수 있습니다.)

year	prod...	prod...	prod...	ord...	retail...	reve...	plan...	prod...	quan...	unit...
2014	Camping Eq...	Tents	Star Gazer 2	Web	China	2453674.5	2638362.6	1778342.1	4530	392.57
2014	Camping Eq...	Tents	Star Gazer 6	Telephone	China	47192.65	50744.68	29890.0	61	490.0
2014	Camping Eq...	Packs	Canyon Mul...	Telephone	China	113152.84	121669.38	83107.5	1583	52.5
2014	Camping Eq...	Packs	Canyon Mul...	Web	China	489320.88	526113.0	294766.44	7158	41.18
2014	Camping Eq...	Lanterns	Firefly Map...	Telephone	China	15530.52	19433.97	8947.5	1193	7.5
2014	Personal Ac...	Knives	Bear Edge	Special	China	20014.8	21070.4	12235.6	520	23.53
2014	Personal Ac...	Navigation	Glacier GPS ...	Web	China	897610.12	965166.4	473645.48	2684	176.47
2014	Outdoor Pro...	Sunscreen	Sun Shelter ...	Web	China	122077.35	129700.0	50842.4	25940	1.96
2014	Outdoor Pro...	First Aid	Insect Bite R...	Web	China	44488.8	45780.0	21058.8	7630	2.76
2014	OutDoor Envi...	Woods	Leather Utilities	Special	China	400580.5	400580.5	400580.5	400580.5	400.58

8-18. 미리보기로 조회된 레코드 테이블에서 오른쪽 끝으로 이동하여 latitude 와 longitude 값이 DMS 형식(도,분,초 방식)으로 표시된 것을 확인합니다.

unit...	retail...	coun...	latitude	longitude	popu...	last...
# Decimal	# Int	Country	Latitude	Longitude	# Int	String
541.65	3	China	39°55'N	116°45'E	1388021000	2018-02-05
773.65	3	China	39°55'N	116°45'E	1388021000	2018-02-05
71.48	3	China	39°55'N	116°45'E	1388021000	2018-02-05
68.36	3	China	39°55'N	116°45'E	1388021000	2018-02-05
7.82	3	China	39°55'N	116°45'E	1388021000	2018-02-05
38.49	3	China	39°55'N	116°45'E	1388021000	2018-02-05
334.43	3	China	39°55'N	116°45'E	1388021000	2018-02-05
4.7083	3	China	39°55'N	116°45'E	1388021000	2018-02-05
5.86	3	China	39°55'N	116°45'E	1388021000	2018-02-05

8-19. 왼쪽 Fields 아래의 New field 버튼을 클릭합니다.

8-20. New calculated field 팝업이 오픈되면 calculated field name 은 latitude_n 으로 입력하고 아래 Formula에 다음 코드를 붙여넣고 Apply changes 를 클릭합니다. (코드 복사는 별도 제공된 code.txt 파일을 사용하십시오. pdf 파일에서 복사할 경우 라인피드 등이 맞지 않을 수 있습니다.)

```
/*latitude*/
ifelse(
    right(latitude, 1) = "N",
    (parseInt(split(latitude, '.', 1)) +
     parseDecimal(substring(latitude, (locate(latitude, '.', 3)+1), 2)) / 60) ,
    (parseInt(split(latitude, '.', 1)) +
     parseDecimal(substring(latitude, (locate(latitude, '.', 3)+1), 2)) / 60) * -1
)
```

New calculated field

Function list	Field list	Calculated field name	Formula
ceil coalesce concat dateDiff decimalToInt epochDate extract floor formatDate ifelse intToDecimal isNotNull isNull left locate	# gross_profit # last_updated # order_method_type # planned_revenue # population # product # product_cost # product_line # product_type # quantity # retail_id # revenue # unit_cost # unit_price # .. .	latitude_n	<pre>/*latitude*/ ifelse(right(latitude, 1) = "N", (parseInt(split(latitude, '.', 1)) + parseDecimal(substring(latitude, (locate(latitude, '.', 3)+1), 2)) / 60) , (parseInt(split(latitude, '.', 1)) + parseDecimal(substring(latitude, (locate(latitude, '.', 3)+1), 2)) / 60) * -1)</pre>

formatDate formats a date using a specified format.
Syntax: formatDate(date, [format], [time_zone])

Create

코드의 내용은 latitude 필드를 시/분/초 구분자 문자를 이용하여 파싱하고 단위변환을 하는 로직입니다. 변환 문법에 대한 설명은 다음 링크를 참조하십시오.

https://docs.aws.amazon.com/ko_kr/quicksight/latest/user/calculated-field-reference.html

8-21. 마찬가지 방법으로 아래 코드를 이용하여 `longitude_n` 컬럼을 추가합니다. (코드 복사는 복사시 별도 제공된 code.txt 파일을 사용하십시오. pdf 파일에서 복사할 경우 라인피드 등이 맞지 않을 수 있습니다.)

```
/*longitude*/
ifelse(
    right(longitude, 1) = "E",
    (parseInt(split(longitude, '.', 1)) +
     parseDecimal(substring(longitude, (locate(longitude, '.',3)+1), 2) ) / 60) ,
    (parseInt(split(longitude, '.', 1)) +
     parseDecimal(substring(longitude, (locate(longitude, '.',3)+1), 2) ) / 60) * -1
)
```

New field

New calculated field

x

Calculated field name: longitude_n

Formula:

```
/*longitude*/
ifelse(
    right(longitude, 1) = "E",
    (parseInt(split(longitude, '.', 1)) +
     parseDecimal(substring(longitude, (locate(longitude, '.',3)+1), 2) ) / 60) ,
    (parseInt(split(longitude, '.', 1)) +
     parseDecimal(substring(longitude, (locate(longitude, '.',3)+1), 2) ) / 60) * -1
)
```

Function list

- ceil
- coalesce
- concat**
- dateDiff
- decimalToInt
- epochDate
- extract
- floor
- formatDate
- ifelse
- intToDecimal
- isNotNull
- isNull
- left
- locate

Field list

- # gross_profit
- last_updated
- order_method_type
- # planned_revenue
- # population
- product
- # product_cost
- product_line
- product_type
- # quantity
- # retail_id
- # revenue
- # unit_cost
- # unit_price
- ...

Calculated field name: longitude_n

Formula:

```
/*longitude*/
ifelse(
    right(longitude, 1) = "E",
    (parseInt(split(longitude, '.', 1)) +
     parseDecimal(substring(longitude, (locate(longitude, '.',3)+1), 2) ) / 60) ,
    (parseInt(split(longitude, '.', 1)) +
     parseDecimal(substring(longitude, (locate(longitude, '.',3)+1), 2) ) / 60) * -1
)
```

concat concatenates multiple strings.

Syntax: concat(expression, expression, expression...)

Create

8-22. 왼쪽 Calculated fields에 `latitude_n`과 `longitude_n`이 추가되었습니다.

Data source		Refresh now
<input checked="" type="radio"/> SPICE	<input type="radio"/> Query	
Tables 1 table selected		▼
Fields All fields selected		▲
New field		
<input type="text" value="Search fields"/> 		
Calculated fields		
 latitude_n Latitude		
 longitude_n Longitude		▼
Select All None		
 year	<input checked="" type="checkbox"/>	
 product_line	<input checked="" type="checkbox"/>	
 product_type	<input checked="" type="checkbox"/>	
 product	<input checked="" type="checkbox"/>	
 order_method_type	<input checked="" type="checkbox"/>	
 retailer_country Country	<input checked="" type="checkbox"/>	

8-23. 미리보기창의 맨 오른쪽으로 이동하여 값이 숫자 형식으로 잘 변환되었는지 확인합니다.

8-24. 다음은 새롭게 생성한 필드를 지오코딩 필드로 지정하겠습니다. 새롭게 생성된 필드의 오른쪽에 마우스커서를 올리고 아래쪽 화살표를 클릭하고 Add to coordinates 를 선택합니다.

Calculated fields

- latitude_n
Latitude
- longitude_n
Longitude

Select All | None

year

Add to coordinates

Edit "latitude_n"

Delete "latitude_n"

Change data type >

8-25. Create new geospatial coordinates 를 선택하고 Add 버튼을 클릭합니다.

Add field to coordinates ×

Create new geospatial coordinates

Add to existing geospatial coordinates

Cancel Add

8-26. Create coordinates 화면에서 Name 을 retailer_position 으로 입력하고, longitude 는 longitude_n 필드를 선택하고 Create coordinates 버튼을 클릭합니다.

Create coordinates ×

Select fields that correspond to latitude and longitude

Name your coordinates

Field to use for latitude

latitude_n

Field to use for longitude

longitude_n

Cancel Create coordinates

8-27. 왼쪽 Fields 리스트에 retailer_position 이라는 새로운 컬럼이 생성된 것을 확인합니다.

Calculated fields

Select All | None

retailer_position

- 📍 latitude_n
Latitude
- 📍 longitude_n
Longitude

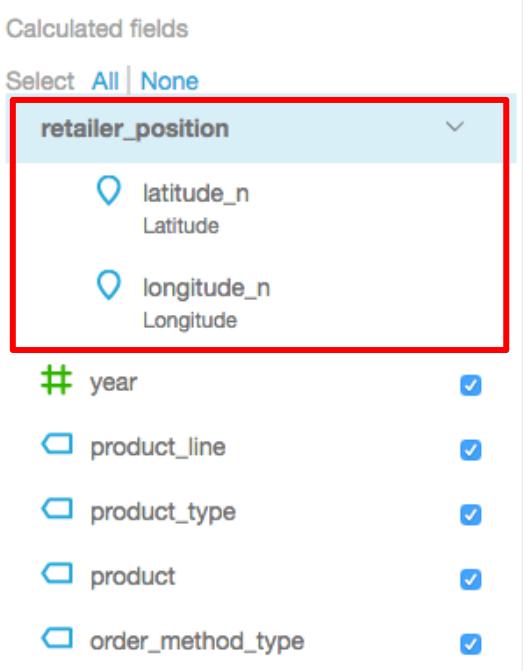
year

product_line

product_type

product

order_method_type



8-28. 이제 Data set 의 준비가 완료되었습니다. 위쪽의 Save & Visualize 버튼을 클릭합니다.



Map Dashboard 생성

다음은 세계 21 개국의 retailer 로으로부터의 주문정보를 지도 위에 가시화 해 보겠습니다..

8-29. 앞 단계에서 대시보드를 편집할 수 있는 Analysis 화면으로 이동하였을 것입니다. 좌측 아래 Visual Type에서 지구 모양의 Points on map 아이콘을 클릭합니다.

You need to 1 geosp:

points on map:
At least
1 geospatial field in Geospatial or
1 latitude and 1 longitude field in Geospatial

8-30. Field List에서 retailer_position 컬럼을 선택한 후 오른쪽 에디터 상단의 Geospatial 항목에 드래그 앤 드롭 합니다.

Geospatial

retailer_position

latitude_n

longitude_n

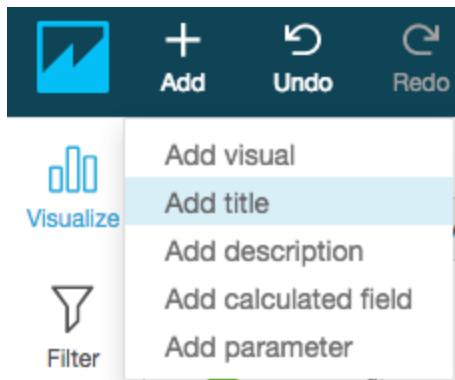
8-31. Field List에서 quantity 컬럼을 선택한 후 오른쪽 에디터 상단의 Size 항목에 드래그 앤 드롭 합니다.



8-32. Field List에서 product_type 컬럼을 선택한 후 오른쪽 에디터 상단의 Color 항목에 드래그앤 드롭합니다.

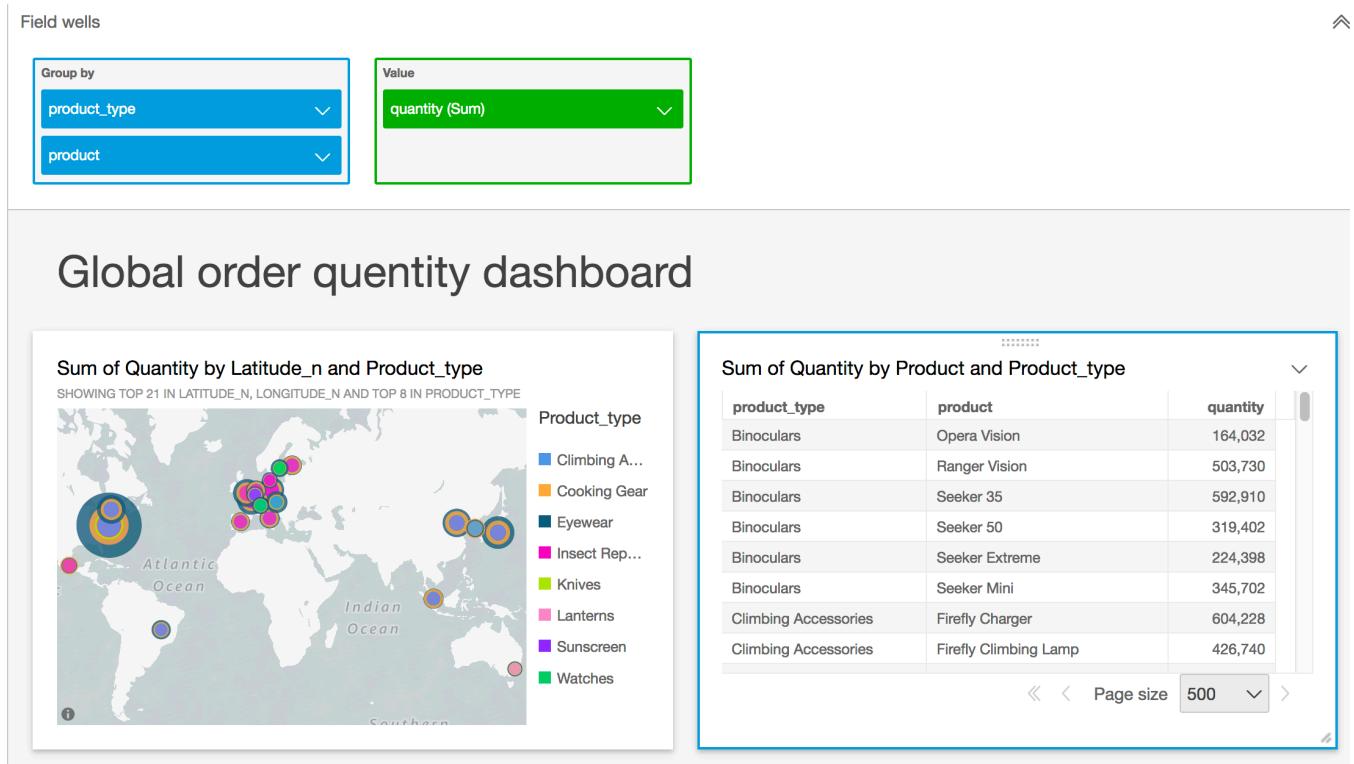


8-33. 왼쪽 위 메뉴에서 십자가 모양의 Add 버튼을 클릭하고 Add title 을 선택합니다.



8-34. Title 을 Global order quantity dashboard 라고 입력합니다.

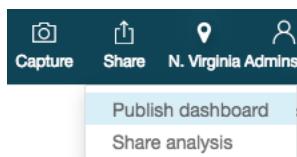
8-35. 유사한 방법으로 Add description 을 통해 설명을 추가하거나 Add visual 을 통해 그래프 오브젝트를 추가할 수 있습니다. 다양한 방법으로 막대그래프, 파이그래프, 피봇테이블 등을 생성해 봅니다. (Visual 오브젝트의 오른쪽 하단을 드래그하여 크기를 조정할 수 있습니다.)



Dashboard share (Optional)

다음은 생성한 가시화 그래프를 대시보드로 배포하는 부분입니다. Lab 시간에 여유가 있다면 진행해 보실 것을 권장드립니다.

8-36. 대시보드의 배포를 위해 오른쪽 상단의 Share 아이콘을 클릭하고 Publish dashboard 를 선택합니다.



8-37. Dashboard 이름을 dplab-dashboard 로 입력하고 Publish dashboard 를 클릭합니다.

Publish a dashboard

Publish new dashboard as

dplab-dashboard

Replace an existing dashboard

8-38. Share with all users in this account 를 클릭합니다. (다른 사용자가 있는 경우 아래 창에 표시됩니다.)

Share dashboard with users ×

Select users in this account.

Share with all users in this account

Name	Email	Permission	Role

[Manage dashboard access](#) [Share](#)

Share with all users in this account? ×

This will provide all users in the Quicksight account reader access to this dashboard

[Back](#)

[Confirm](#)

8-39. x 를 눌러 창을 닫습니다.

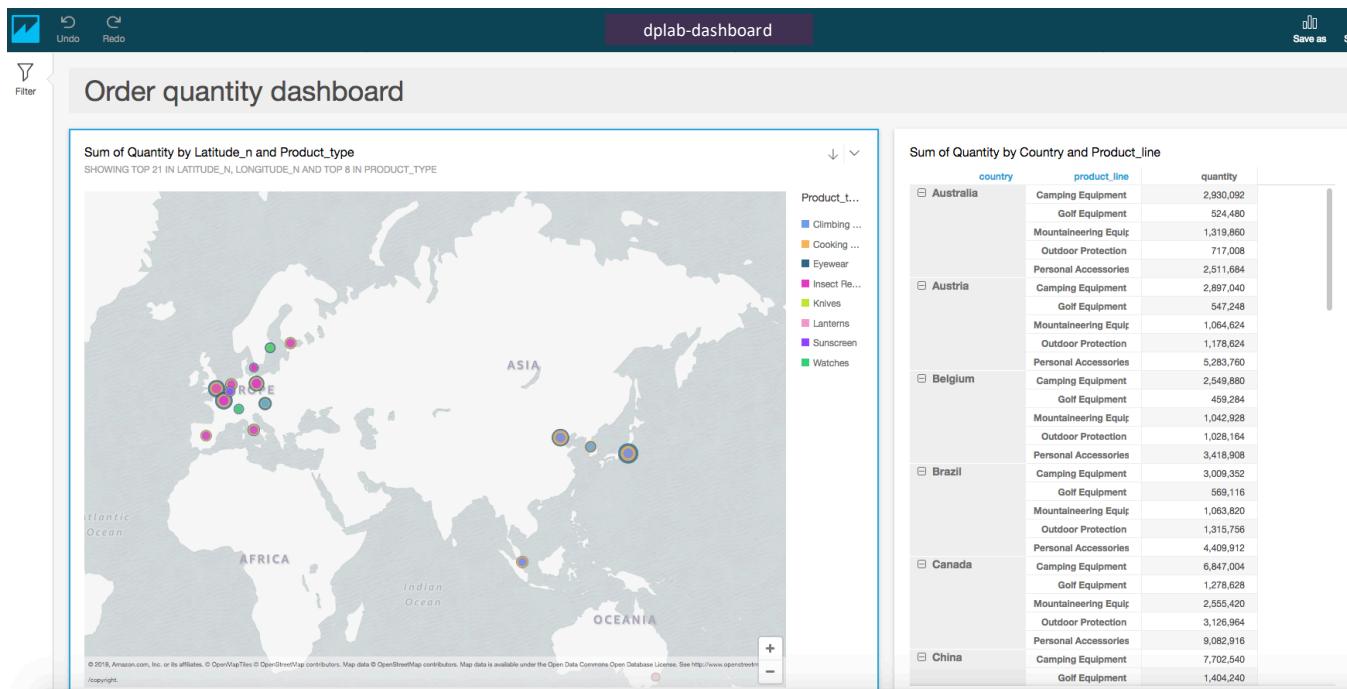
Manage dashboard sharing ×

Manage dashboard permissions, enable save as or revoke access.

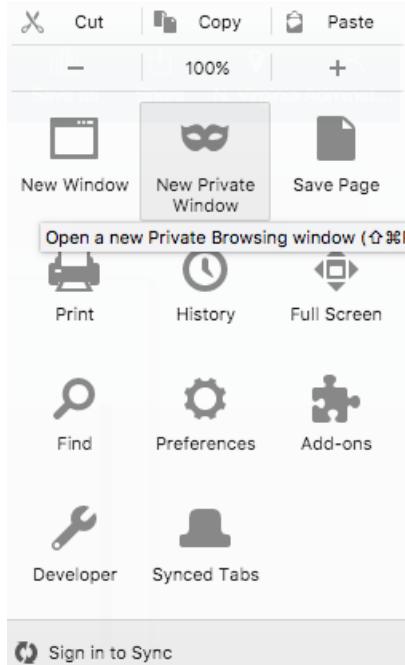
[Add users](#) Share with all users in this account 🔍

Name	Email	Permission	Role	Save as	Action
tom	mullue@gmail.com	Owner	ADMIN		

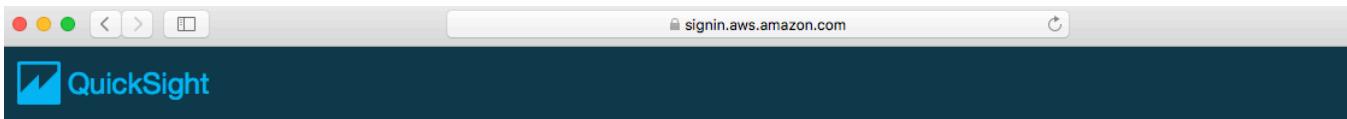
8-40. dplab-dashboard 가 생성되었습니다. 웹브라우저의 URL 값을 복사합니다.



8-41. 다른 브라우저(크롬, 사파리, 파이어폭스 등)를 열거나 웹브라우저의 Incognito mode 또는 Private mode 를 이용하여 새로운 세션을 오픈합니다.



8-42. 복사한 URL 을 붙여넣고 접속을 시도하면 아래와 같이 account 를 입력하라고 합니다. 8-3 단계에서 생성한 계정정보를 입력하고 로그인하면 대시보드를 볼 수 있습니다.



QuickSight account name (i)

Enter your QuickSight account name ▼

Continue

Not signed up for QuickSight yet? [Sign up here](#)

Does your organization use an identity provider?

[Learn more](#)

Sign in to QuickSight

QuickSight account name

radiohead

Email address or username

mullue@gmail.com

Password

.....

I have an [MFA token](#)

Sign in

[Forgot Password?](#) | [Identity Provider Sign-in](#)

Not signed up for QuickSight yet? [Sign up here](#)

Welcome to QuickSight

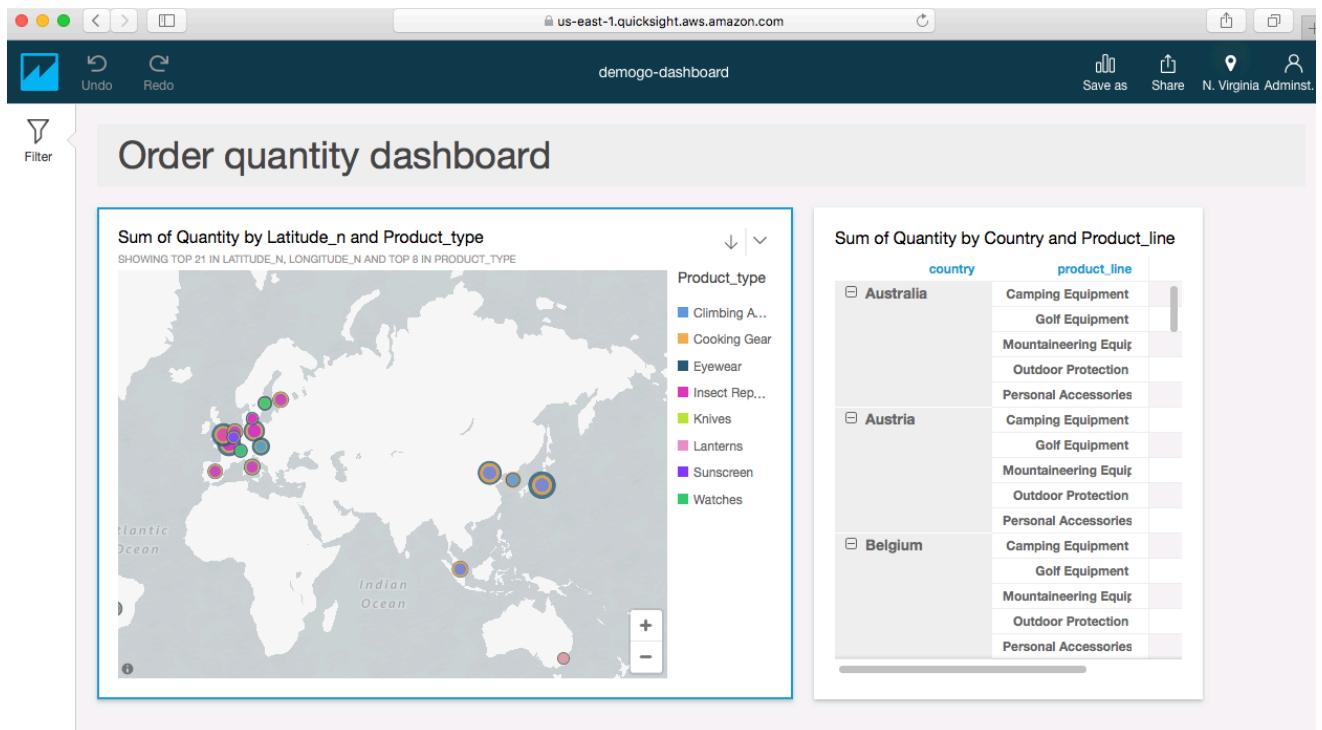
You are about to access QuickSight in AWS Account 987654321012 .

Email address

mullue@gmail.com

By choosing to continue, you will be provisioned as a user in the QuickSight account. Monthly charges for QuickSight usage will apply until you or an administrator revokes access privileges to this account.

Continue

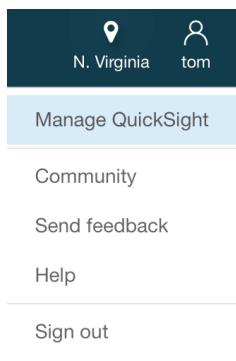


9. Closing

실습이 완료되었습니다. 추가 과금을 막기 위해 아래 리소스별 삭제방법을 참고하여 리소스를 삭제합니다. (생성 역순으로 진행합니다.)

QuickSight 삭제

9-1. QuickSight 화면 우상단에서 Manage QuickSight 를 클릭합니다.



9-2. 계정관리 네비게이션에서 Account Setting 을 클릭하고 Unsubscribe 를 클릭합니다. (QuickSight 계정을 삭제하게 됩니다.)

A screenshot of the QuickSight 'Account settings' page. At the top left is the QuickSight logo and the account name 'radiohead'. Below it, the 'Upgrade now' button is visible. The main area contains several sections: 'Manage users', 'Your subscriptions', 'SPICE capacity', and 'Account settings' (which is highlighted with a red box). The 'Account settings' section includes fields for 'Notification email address' (set to 'mullue@gmail.com') and a checkbox for 'Enable IAM user access requests to this account'. Below this is the 'Account permissions' section, which includes a 'Set QuickSight permissions on AWS resources' link and a 'Edit AWS permissions' button. At the bottom is the 'Close this QuickSight account' section, which contains a list of items that will be deleted upon unsubscribing and a prominent red-outlined 'Unsubscribe' button.

By unsubscribing you will be deleting all content related to this account including:

- Data sources
- Data sets
- Analyses
- Published dashboards

[Unsubscribe](#)

Unsubscribe from QuickSight

QuickSight account name radiohead

By unsubscribing you will be deleting all content related to this account including:

- Data sources
- Data sets
- Analyses
- Published dashboards

[Cancel](#)[Unsubscribe](#)

Unsubscribe successful

Most of your content has been deleted, but there were a few items we were unable to remove. Address the items below to ensure no further costs are accrued by this QuickSight account.

! Unless you are using them in other services, use the AWS IAM console to remove [\[i\]](#) the following roles and policies:

```
arn:aws:iam::987654321012:policy/service-role/AWSQuickSightIAMPolicy
arn:aws:iam::987654321012:policy/service-role/AWSQuickSightRDSPolicy
arn:aws:iam::987654321012:policy/service-role/AWSQuickSightRedshiftPolicy
arn:aws:iam::aws:policy/service-role/AWSQuicksightAthenaAccess
arn:aws:iam::987654321012:role/service-role/aws-quicksight-service-role-v0
```

[Go to AWS console](#)

9-3. 마지막 화면 안내를 참고하여 생성된 IAM Role 을 삭제할 수 있습니다. 단, IAM 은 과금대상 서비스는 아닙니다.

Glue 리소스 삭제

9-4. Glue 콘솔에서 Jobs 리스트화면으로 이동하여 Action 에서 Delete 를 선택하고 팝업화면에서 Delete 버튼을 클릭합니다. (세 개의 작업에 대하여 동일하게 진행합니다.)

Jobs A job is your business logic required to perform extract, transform and load (ETL) work. Job runs are initiated by triggers.

Name	ETL language	Sc
python	s3	
python	s3	
python	s3	



Delete Job

Delete the following permanently? This action cannot be undone.

- enrich-join

Delete

9-5. 동일한 방법으로 Trigger 를 삭제합니다. (네비게이션을 통해 Triggers 리스트 화면으로 이동한 후 Trigger 를 선택하고 Action 에서 Delete 를 선택하고 팝업에서 Delete 버튼을 클릭합니다.)

Triggers A trigger starts a job when it fires.

The screenshot shows the AWS Lambda Triggers list. On the left, there's a sidebar with 'Add trigger' and a dropdown labeled 'Action'. Below that is a list of triggers: 'Trigger name' (unchecked), 'ready-to-join' (checked), and 'etl-starter' (unchecked). A context menu is open over the 'ready-to-join' trigger, listing options: 'Edit trigger', 'Disable trigger', 'Enable trigger', 'Choose jobs to trigger', 'Delete' (which is highlighted in blue), and 'Start trigger'.

Delete Triggers

Delete the following permanently? This action cannot be undone.

- ready-to-join

Delete

9-6. 동일한 방법으로 Crawler 를 삭제합니다. (네비게이션을 통해 Crawlers 리스트 화면으로 이동한 후 Crawler 를 선택하고 Action 에서 Delete crawler 를 선택하고 팝업에서 Delete 버튼을 클릭합니다.)

9-7. 동일한 방법으로 Databases 를 삭제합니다. Database 를 삭제하면 Tables 도 함께 삭제됩니다. (네비게이션을 통해 Databases 리스트 화면으로 이동한 후 Database 를 선택하고 Action 에서 Delete database 를 선택하고 팝업에서 Delete 버튼을 클릭합니다.)

S3 버킷 삭제

9-8. dplab 으로 시작하는 S3 버킷을 삭제합니다. S3 콘솔에서 버킷을 선택한 후 Delete bucket 버튼을 누르고 버킷명을 한번 더 넣은 후 Confirm 을 클릭합니다.



