

Analytics in a Day

---

# Hands on Lab Guide

# Contents

1. 사전 준비 .....	3
1.1. 시나리오 개요 .....	3
1.2. 학습 목표 .....	3
1.3. 솔루션 아키텍처 .....	3
1.4. 실습 환경 설정 .....	3
2. Exercise 0: Configure Azure Services.....	4
2.1. Task 1: Create Resources .....	4
2.2. Task 2: Upload the data used in the lab.....	13
2.3. Task 3: Configure the SQL on-demand pool and SQLPool01 .....	15
2.4. Task 4: Create Linked service and datasets .....	18
2.5. Task 5: Create Azure Synapse Pipelines & Run .....	33
2.6. Task 6: Populate Primarystorage with data .....	42
3. Exercise 1: Explore the Data Lake with Azure Synapse serverless SQL pool and Azure Synapse Spark.....	44
3.1. Task 1: Explore the Data Lake with Azure Synapse serverless SQL pool .....	44
3.2. Task 2: Explore the Data Lake with Azure Synapse Spark .....	47
4. Exercise 2: Build a Modern Data Warehouse with Azure Synapse Pipelines .....	49
4.1. Task 1: Explore, modify, and run a Pipeline containing a Data Flow .....	49
5. Exercise 3: Power BI Integration .....	70
5.1. Task 1: Create a Power BI dataset in Synapse .....	70
5.2. Task 2: Create a Power BI report in Synapse .....	73
5.3. Task 3: View the SQL query.....	78
6. Exercise 4: High-Performance Analysis with Azure Synapse Dedicated SQL Pools .....	79
6.1. Task 1: Use a dedicated SQL pool query to understand a dataset.....	79
6.2. Task 2: Investigate query performance and table design.....	80
7. Exercise 5 - Data Science with Azure Synapse Spark.....	83

7.1. Task 1: Making predictions with a trained model .....	84
7.2. Task 2: Examining the model training and registration process .....	87

## 1. 사전 준비

### 1.1. 시나리오 개요

Wide World Importers (WWI)는 샌프란시스코 베이 지역에서 운영되는 도매 상품 수입 및 유통하는 가상의 업체입니다. 도매업자로서 WWI의 고객은 대부분 개인에게 재 판매하는 회사입니다. WWI는 전문점, 슈퍼마켓, 컴퓨터 상점, 관광 명소 상점 및 일부 개인을 포함하여 미국 전역의 소매 고객에게 판매합니다. WWI의 모든 고객은 현재 미국에 있지만 회사는 다른 국가로 확장 할 계획입니다.

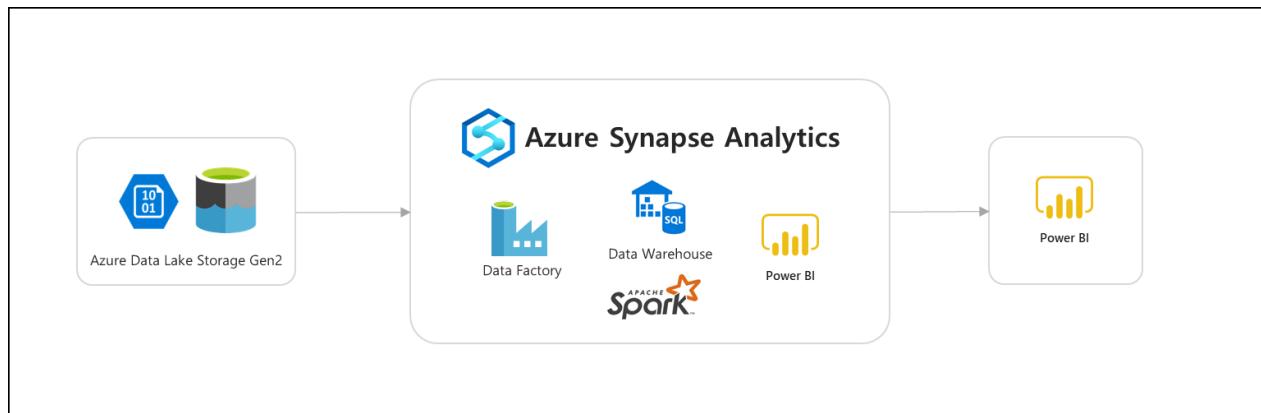
WWI는 빅데이터 분석을 위한 통합 데이터 분석 플랫폼 구축을 위한 PoC를 수행하고자 합니다. 그들의 목표는 사일로화 된 팀을 단일 플랫폼에서 함께 작업하도록 하는 것입니다.

### 1.2. 학습 목표

이 실습에서는 데이터 엔지니어, 비즈니스 분석가, 데이터 과학자등의 역할을 수행 하여 데이터 탐색, 전처리, 적재, 시각화 보고서 생성, 머신러닝 모델을 통한 예측 수행 방법을 연습합니다.

이 실습이 끝나면 빅 데이터 분석을 단일 플랫폼으로 전체 작업을 수행하게 됩니다.

### 1.3. 솔루션 아키텍처



### 1.4. 실습 환경 설정

이 연습에서는 이 실습을 위한 소스 환경을 배포합니다.

1. Azure Portal (<https://portal.azure.com>)에서 이 랩에 사용할 구독으로 로그인 했는지 확인합니다.

## 2. Exercise 0: Configure Azure Services

이 실습에서는 Azure Storage Account와 Azure Synapse Analytics Studio내에서 자원들을 생성하고 구성합니다. 이러한 서비스를 사용하여 통합 분석 플랫폼 환경을 구축하고 분석을 수행 합니다.

### 2.1. Task 1: Create Resources

1. Azure 포탈로 이동하여 <https://portal.azure.com/>에서 로그인합니다.
2. 홈 화면 상단의 리소스 만들기를 선택하여, **storage account**를 검색하고 스토리지 계정을 선택합니다.

The screenshot shows the Azure Marketplace search results for 'storage account'. The search bar at the top contains 'storage account'. On the left, there is a sidebar with categories like '프라이빗 Marketplace(미리 보기)', '내 저장된 목록', '최근에 만들어짐', '서비스 공급자', '범주', '시작' (which is highlighted), 'AI + 기계 학습', '분석', '블록체인', and '컴퓨팅'. The main area displays two cards: 'Flexify.IO - Amazon S3 API for Azure Blob Storage' by Flexify.IO and '스토리지 계정' by Microsoft. Both cards have a blue heart icon below them.

3. 저장소 계정 블레이드에서 **만들기**를 클릭합니다. 다음 정보를 입력합니다.
  - ✓ 구독: 본인의 구독 선택
  - ✓ 리소스 그룹: 새로 만들기 클릭하여 생성 -> **ASA\_EduXXXX**
  - ✓ 스토리지 계정 이름: **asablobstorageXXXX**
  - ✓ 위치: (아시아 태평양)한국 중부
  - ✓ 성능: 표준
  - ✓ 계정 종류: **StorageV2 (범용 v2)**
  - ✓ Replication: **LRS(로컬 중복 스토리지)**

## 스토리지 계정 만들기

기본 사항 네트워킹 데이터 보호 고급 태그 검토 + 만들기

Azure Storage는 가용성, 보안, 내구성, 확장성 및 중복성이 뛰어난 클라우드 스토리지를 제공하는 Microsoft 관리 서비스입니다. Azure Storage는 Azure Blob(객체), Azure Data Lake Storage Gen2, Azure Files, Azure 큐 및 Azure 테이블을 포함합니다. 스토리지 계정의 비용은 사용량 및 아래에서 선택한 옵션에 따라 다릅니다. [Azure Storage 계정에 대한 자세한 정보](#)

### 프로젝트 정보

배포된 리소스와 비용을 관리할 구독을 선택합니다. 풀더 같은 리소스 그룹을 사용하여 모든 리소스를 정리 및 관리합니다.

구독 *	<input type="button" value="사용자 구독 선택"/>
리소스 그룹 *	<input type="button" value="(신규) ASA_Edu0000"/>
	<a href="#">새로 만들기</a>

### 인스턴스 정보

기본 배포 모델은 최신 Azure 기능을 지원하는 Resource Manager입니다. 대신 클래식 배포 모델을 사용하여 배포하도록 선택할 수 있습니다. [클래식 배포 모델 선택](#)

스토리지 계정 이름 *	<input type="text" value="asablobstorage0000"/>
위치 *	<input type="button" value="(Asia Pacific) 한국 중부"/>
성능	<input checked="" type="radio"/> 표준 <input type="radio"/> 프리미엄
계정 종류	<input type="button" value="StorageV2(범용 v2)"/>
복제	<input type="button" value="LRS(로컬 중복 스토리지)"/>

[검토 + 만들기](#)

< 이전

다음: 네트워킹 >

- 상단의 고급을 클릭하고 Data Lake Storage Gen2를 사용으로 변경합니다.

홈 > 새로 만들기 >

## 스토리지 계정 만들기

기본 사항 네트워킹 데이터 보호 **고급** 태그 검토 + 만들기

**보안**

보안 전송 필요  사용 안 함  사용

최소 TLS 버전

인프라 암호화  사용 안 함  사용

현재 인프라 암호화를 사용하도록 설정하려면 구독별 등록이 필요합니다.  
[인프라 암호화 등록](#)

**Blob Storage**

Blob 공용 액세스 허용  사용 안 함  사용

Blob 액세스 계층(기본값)  끌  핫

NFS v3  사용 안 함  사용

현재 NFS v3 기능을 사용하려면 구독 별 등록이 필요합니다. [NFS v3 등록](#)

**Data Lake Storage Gen2**

계층 구조 네임스페이스  사용 안 함  사용

**Azure Files**

대용량 파일 공유  사용 안 함  사용

**테이블 및 큐**

고객 관리형 키 지원  사용 안 함  사용

현재 구독 단위로 테이블 및 큐에 대한 고객 관리형 키 지원을 사용하도록 설정  
하려면 가입해야 합니다. [CMK 지원에 등록](#)

**검토 + 만들기** < 이전 다음: 태그 >

5. **검토+만들기**를 클릭하고 구성 선택 사항을 확인한 후 **만들기**를 선택합니다.
6. 새 스토리지 계정이 완성되면 리소스로 이동하여 **컨테이너**를 클릭합니다.

홈 > Microsoft.StorageAccount-20210111100247 >

## asblobstorage0000

스토리지 계정

검색(Ctrl+ /) | 템색기에서 열기 | 이동 | 새로 고침 | 삭제 | 사용자 의견

개요 | 활동 로그 | 태그 | 문제 진단 및 해결 | 액세스 제어(IAM) | 데이터 전송 | 이벤트 | Storage Explorer(미리 보기)

기본 정보

리소스 그룹 (변경) : ASA\_Edu0000  
상태 : 기본: 사용 가능  
위치 : 한국 중부  
구독 (변경) : Visual Studio Enterprise 구독 - MPN  
구독 ID : 3000d766-ee13-46de-a8f3-828411ceb65b  
태그 (변경) : 태그를 추가하려면 여기를 클릭

컨테이너 대규모로 확장 가능한 데이터 레이크 스토리지 | 자세한 정보

파일 공유 서비스 SMB 및 NFS 파일 공유 | 자세한 정보

7. +컨테이너를 클릭한 후 staging를 입력하고 만들기 버튼을 클릭합니다.

홈 > Microsoft.StorageAccount-20210111100247 > asblobstorage0000 | 컨테이너

## 새 컨테이너

+ 컨테이너 | 검색 | 컨테이너 복원 | 새로 고침 | 삭제

접두사로 컨테이너 검색

이름 마지막으로 수정한 날짜 공유 액세스 수준

컨테이너가 없습니다. 시작하려면 [+ 컨테이너]를 클릭하세요.

이름\* : staging

공유 액세스 수준 : 프라이빗(익명 액세스 없음)

고급

만들기 취소

8. 위와 같은 방식으로 +컨테이너를 클릭하여 models를 입력하고 만들기 버튼을 클릭합니다.

이름	마지막으로 수정한 날짜	공용 액세스 수준	임대 단계
models	2021. 1. 11. 오전 11:04:13	프라이빗	사용 가능
staging	2021. 1. 11. 오전 11:04:01	프라이빗	사용 가능

9. 2~5번과 동일한 작업을 통해 **asaprimarystorageXXXX**을 추가 합니다.

- ✓ 구독: 본인의 구독 선택
- ✓ 리소스 그룹: 이전에 생성한 리소스 선택 -> **ASA\_EduXXXX** 선택
- ✓ 스토리지 계정 이름: **asaprimarystorageXXXX**
- ✓ 위치: (아시아 태평양)한국 중부
- ✓ 성능: 표준
- ✓ 계정 종류: **StorageV2 (범용 v2)**
- ✓ Replication: **LRS(로컬 중복 스토리지)**

10. 6~7과 동일한 작업을 통해 **dev, staging, wwi** Container를 추가 합니다.

이름	마지막으로 수정한 날짜	공용 액세스 수준	임대 단계
dev	2021. 1. 11. 오후 1:23:43	프라이빗	사용 가능
staging	2021. 1. 11. 오후 1:25:24	프라이빗	사용 가능
wwi	2021. 1. 11. 오후 1:25:32	프라이빗	사용 가능

11. 왼쪽 메뉴의 리소스 만들기를 선택 하여 **Azure Synapse Analytics**를 검색하여 선택 후 **Azure Synapse Analytics** 블레이드에서 만들기를 클릭하고 아래 내용을 입력 합니다.

- ✓ 구독: 본인의 구독 선택.
- ✓ 리소스 그룹: 이전에 생성한 리소스 선택 -> **ASA\_EduXXXX** 선택
- ✓ 작업 영역 이름: **asaworkspaceXXXX**
- ✓ 지역: (아시아 태평양)한국 중부
- ✓ Data Lake Storage Gen2 선택: 구독에서 선택
- ✓ 계정 이름: 이전에 생성한 리소스 선택 -> **asaprimarystorageXXXX**
- ✓ 파일 시스템 이름: 새로 만들기 클릭하여 생성 -> **asaworkspace**
- ✓ “나에게 Data Lake Storage Gen2 계정 ‘asaprimarystorageXXXX’에 대한 Storage Blob 데이터 기여자 역할을 할당합니다.”에 **Check**

## Synapse 작업 영역 만들기

\*기본 사항 \*보안 네트워킹 태그 요약

Synapse 작업 영역을 만들어 몇 번의 클릭만으로 엔터프라이즈 분석 솔루션을 개발할 수 있습니다.

### 프로젝트 정보

배포된 리소스와 비용을 관리할 구독을 선택합니다. 풀더 같은 리소스 그룹을 사용하여 모든 리소스를 정리 및 관리합니다.

구독 \*

사용자 구독 선택

리소스 그룹 \*

ASA\_Edu0000

[새로 만들기](#)

### 작업 영역 정보

작업 영역의 이름을 지정하고, 위치를 선택하고, 로그 및 작업 출력의 기본 위치로 사용할 기본 Data Lake Storage Gen2 파일 시스템을 선택합니다.

작업 영역 이름 \*

asaworkspace0000

지역 \*

한국 중부

Data Lake Storage Gen2 선택 \*

구독에서  URL을 통해 수동으로

계정 이름 \*

asaprimarystorage0000

[새로 만들기](#)

파일 시스템 이름 \*

(새로 만들기) asaworkspace

[새로 만들기](#)

나에게 Data Lake Storage Gen2 계정 'asaprimarystorage0000'에 대한 Storage Blob 데이터 기여자 역할을 할당합니다.

**i** Storage Blob 데이터 Contributor 역할을 사용하여 지정된 Data Lake Storage Gen2 계정에 작업 영역 ID 데이터 액세스 권한을 자동으로 부여합니다. 작업 영역을 만든 후 다른 사용자가 이 스토리지 계정을 사용할 수 있도록 하려면 다음 작업을 수행하세요.

- 다른 사용자를 작업 영역의 참가자 역할에 할당
- Synapse Studio를 사용하여 다른 사용자를 작업 영역, SQL 또는 Spark 관리자 역할에 할당
- 자신과 다른 사용자를 스토리지 계정의 Storage Blob 데이터 참가자 역할에 할당

[검토 + 만들기](#)

< 이전

다음: 보안 >

12. 상단의 **보안**을 클릭하고 SQL 관리자 자격 증명을 아래와 같이 입력합니다.

- ✓ 관리자 사용자 이름: **sqladminuser**
- ✓ 암호: **Demo@pass123** <- 암호 확인 부분에 동일하게 입력

## Synapse 작업 영역 만들기

\* 기본 사항 \* 보안 네트워킹 태그 요약

작업 영역의 보안 옵션을 구성합니다.

### SQL 관리자 자격 증명

작업 영역의 SQL 풀에 대한 관리자 액세스에 사용할 수 있는 자격 증명을 제공합니다. 암호를 제공하지 않으면 자동으로 생성됩니다. 나중에 암호를 변경할 수 있습니다.

관리자 사용자 이름 \*

sqladminuser

암호

\*\*\*\*\*



암호 확인

\*\*\*\*\*



### 작업 영역 암호화

⚠️ 작업 영역을 만들 때 고객 관리형 키 사용에 옵트인한 후에는 이중 암호화 구성을 변경할 수 없습니다.

사용자가 관리하는 키(고객 관리형 키)를 사용하여 작업 영역의 미사용 데이터를 모두 암호화하도록 선택합니다. 그러면 플랫폼 관리형 키를 사용하는 인프라 계층에서 암호화를 사용한 이중 암호화가 제공됩니다. [자세한 정보](#)

고객 관리형 키를 사용한 이중 암호화 사용

### 시스템이 할당한 관리 ID

파이프라인 통합을 위해 SQL 풀에 작업 영역의 시스템이 할당한 관리 ID CONTROL 권한을 할당할지를 선택합니다.  
[자세한 정보](#)

파이프라인(작업 영역의 시스템 할당 ID로 실행됨)에서 SQL 풀에 액세스할 수 있도록 허용합니다.

검토 + 만들기

< 이전

다음: 네트워킹 >

13. 검토+만들기를 클릭하고 구성 선택 사항을 확인한 후 만들기를 선택합니다

14. 새 Synapse Workspace가 만들어 지면 리소스로 이동하여 **Synapse Studio 열기(개시)**를 클릭합니다

15. 새로운 창(Tab)으로 Workspace가 열리면 왼쪽의 **Manage** 메뉴를 클릭 합니다.

16. Manage 화면에서 SQL Pools의 +New를 선택하고 아래 내용을 입력 합니다.

- ✓ Dedicated SQL pool name: **SQLPool01**

The screenshot shows the Azure Synapse Analytics workspace interface. On the left, there's a sidebar with various service links like Analytics pools, External connections, Integration, Security, etc. The main area is titled 'SQL pools' and displays a table with one item: 'Built-in' (Type: Serverless). On the right, a modal window titled 'Create dedicated SQL pool' is open. It has tabs for 'Basics', 'Additional settings', 'Tags', and 'Review + create'. Under 'Basics', the 'Dedicated SQL pool name' field is set to 'SQLPool01' (highlighted with a red box), and the 'Performance level' is set to 'DW1000c'. Below these, there's an 'Estimated price' section showing 'Est. cost per hour 13495.80 KRW' and a 'View pricing details' link. At the bottom of the modal are buttons for 'Review + create' and 'Cancel'.

17. 검토+만들기(Review+create)를 클릭하고 구성 선택 사항을 확인한 후 만들기(Create)를 선택합니다

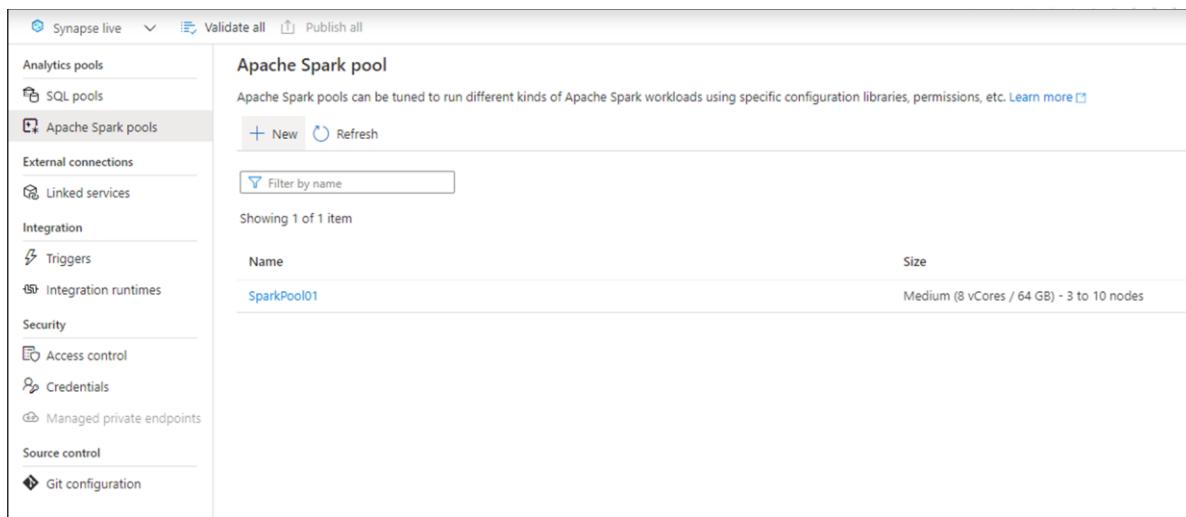
This screenshot shows the same workspace interface after the creation of the dedicated SQL pool. The 'SQL pools' table now contains two rows: 'Built-in' (Serverless) and 'SQLPool01' (Dedicated). Both entries are marked as 'Online' and have an 'Auto' size. The rest of the workspace interface remains the same.

18. Apache Spark Pool관리 화면에서 +New 를 선택 하고 아래 내용을 입력합니다.

- ✓ Apache Spark pool name: **SparkPool01**

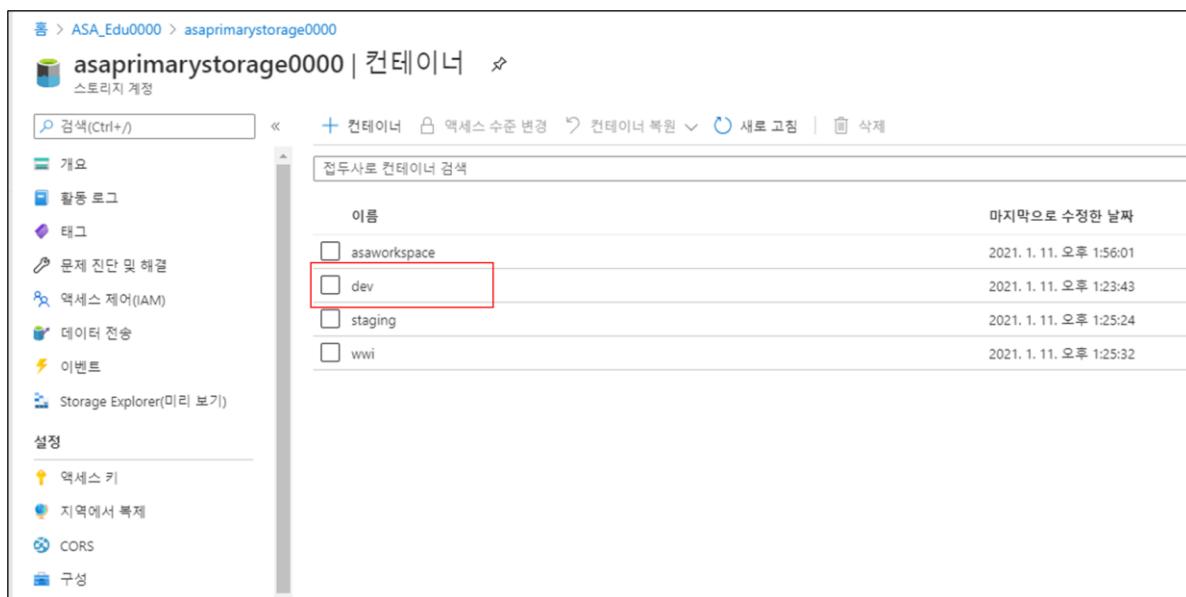
This screenshot shows the workspace interface with the 'Apache Spark pools' section selected. A modal window titled 'Create Apache Spark pool' is open. In the 'Basics' tab, the 'Apache Spark pool name' is set to 'SparkPool01' (highlighted with a red box). Other settings include 'Node size family' (MemoryOptimized), 'Node size' (Medium (8 vCores / 64 GB)), 'Auto-scale' (Enabled), and 'Number of nodes' (set to 10). An 'Estimated price' section shows 'Est. cost per hour 4588.57 to 15295.24 KRW' with a 'View pricing details' link. Buttons at the bottom include 'Review + create' and 'Cancel'.

19. 검토+만들기(Review+create)를 클릭하고 구성 선택 사항을 확인한 후 만들기(Create)를 선택합니다.

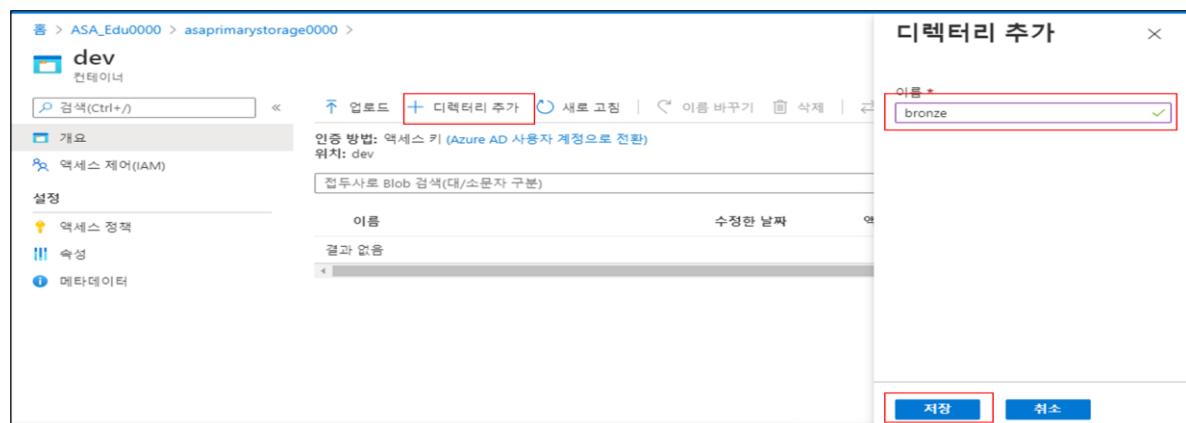


## 2.2. Task 2: Upload the data used in the lab

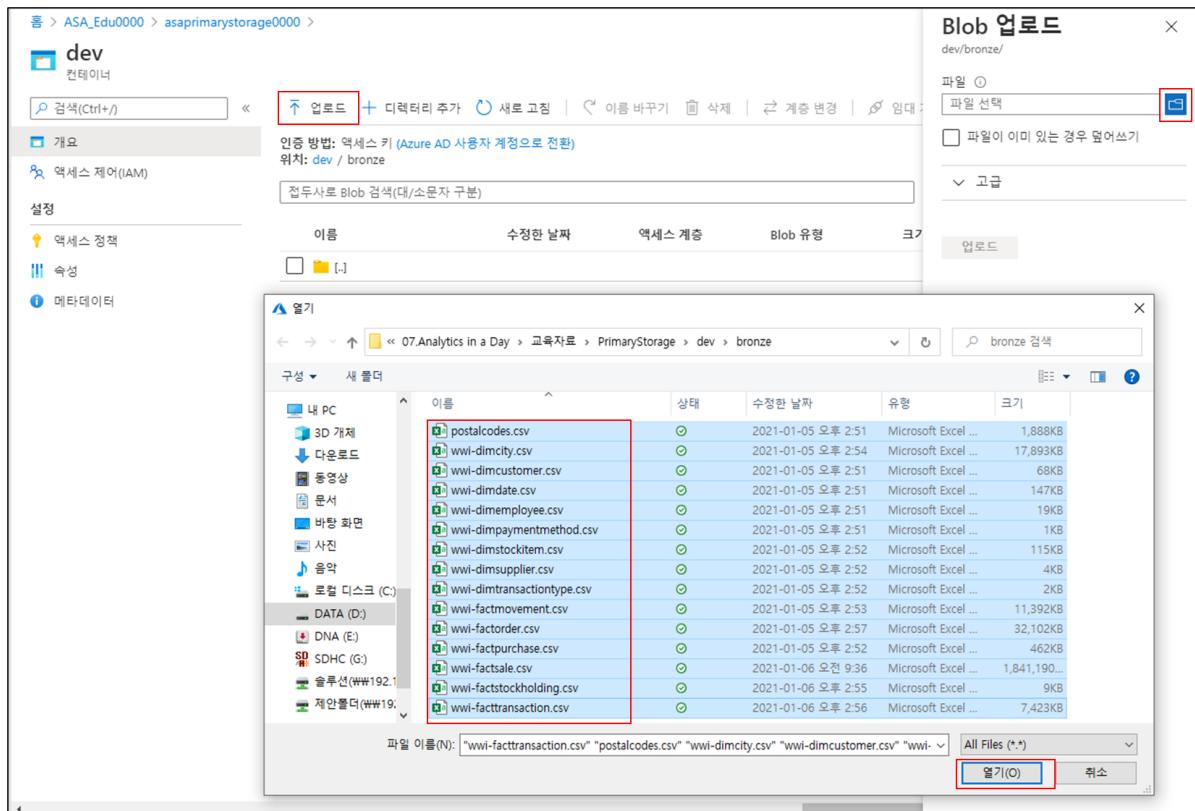
1. Azure Portal로 돌아가서 Task1에서 생성한 **asaprimarystorageXXXX** 스토리지 계정 리소스로 이동하여 컨테이너를 클릭하고 **dev**를 선택 합니다.



2. 디렉터리 추가를 클릭하고 오른쪽 블레이드에 **bronze**를 입력, 저장 하여 폴더를 추가합니다.



3. 추가된 **bronze** 폴더를 선택, 업로드를 클릭하여 오른쪽 블레이드에 **파일선택**을 클릭합니다.  
로컬 PC에서 업로드할 파일들(../PrimaryStorage/dev/bronze/ 폴더안 파일 전체)을 모두 선택하고 **열기** 버튼을 클릭합니다.



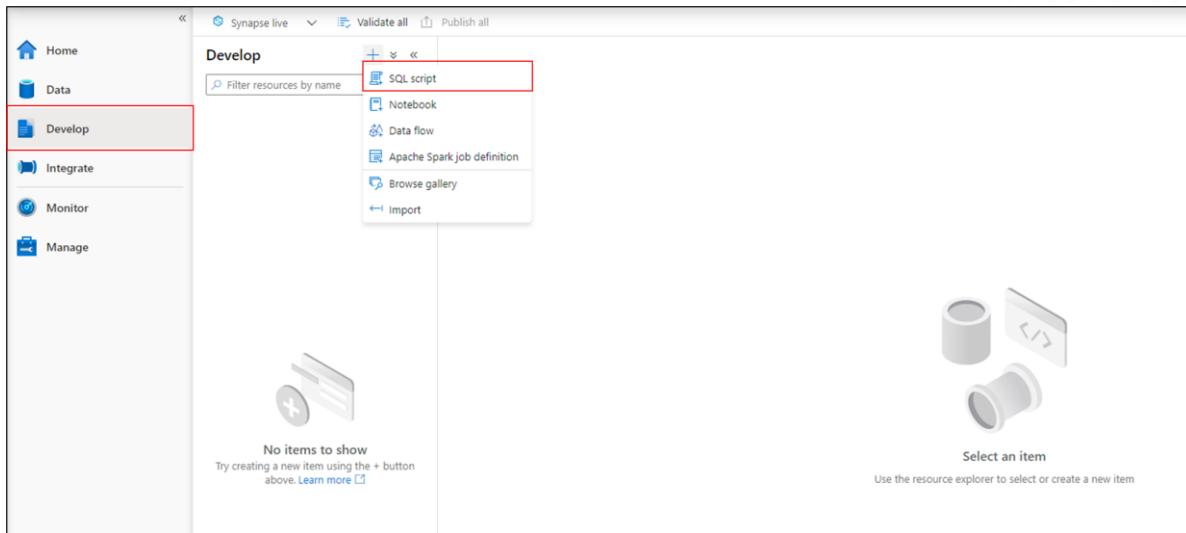
4. **업로드** 버튼을 클릭하여 위에서 선택한 파일들을 업로드합니다.



5. 네트워크 전송 속도에 따라 차이가 있으나 약 10분정도 소요 됩니다.

### 2.3. Task 3: Configure the SQL on-demand pool and SQLPool01

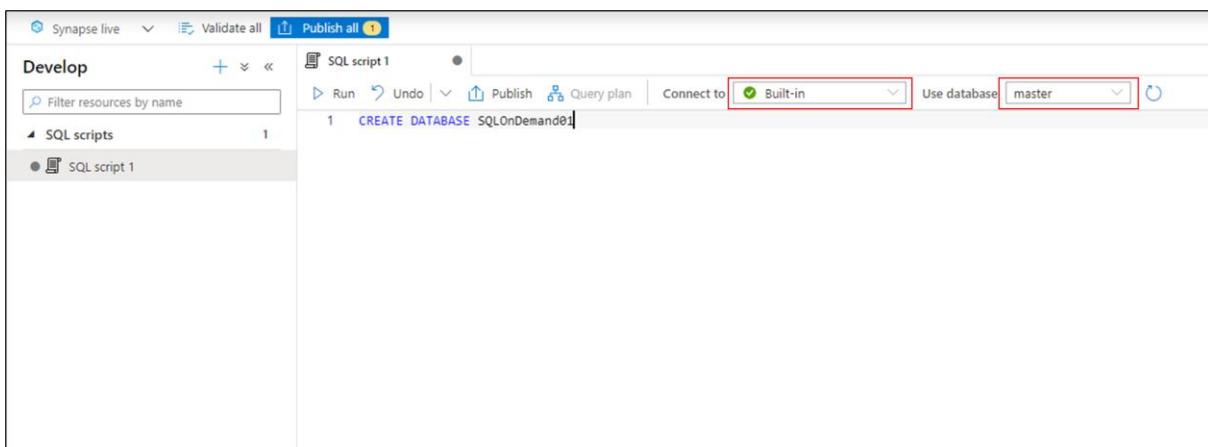
1. Synapse Studio의 왼쪽 메뉴에서 Develop을 선택 후 +를 클릭 하여 SQL script를 선택 합니다.



2. SQL on-demand pool의 master database에서 아래 스크립트를 수행 하여 SQL on-demand database를 생성 합니다.

<Script>

```
CREATE DATABASE SQLOnDemand01
```



3. 새로 생성 한 SQLOnDemand01 Database에서 아래 스크립트를 수행 하여 asaprimarystorageXXXX 인증 정보를 생성 합니다. <primary\_storage>는 이전에 만든 Primary Storage명 asaprimarystorageXXXX로 변경 합니다.

<Script>

```
CREATE CREDENTIAL [https://<primary_storage>.dfs.core.windows.net]
WITH IDENTITY='User Identity';
```

The screenshot shows the Azure Data Studio interface. In the top navigation bar, there are tabs for 'Synapse live', 'Validate all', and 'Publish all'. Below the tabs, the 'Develop' section is selected, showing a list of 'SQL scripts'. One script, 'SQL script 1', is currently open. The code in the editor is:

```

1 CREATE CREDENTIAL [https://asaprimarystorage0000.dfs.core.windows.net]
2 WITH IDENTITY='User Identity';
3
4

```

The results pane below the editor shows a magnifying glass icon and the message 'No results to show' followed by 'Your query yielded no displayable results'. At the bottom of the results pane, a green status bar indicates '00:00:00 Query executed successfully.'

- 새로운 SQL Scripts를 추가 후 Connect to에서 SQLPool01을 선택, Use database SQLPool01을 확인하고 아래 파일의 내용을 수행 하여 SQLPool01 스키마를 생성 합니다.

<Script> : [WWI-Reset-SQLPool-Schema.sql](#) 파일 참고

The screenshot shows the Azure Data Studio interface. In the top navigation bar, there are tabs for 'Synapse live', 'Validate all', and 'Publish all'. Below the tabs, the 'Develop' section is selected, showing a list of 'SQL scripts'. Two scripts are listed: 'SQL script 1' and 'SQL script 2'. 'SQL script 2' is currently open. The code in the editor is:

```

1 create schema wwi
2 go
3 create schema wwi_perf
4 go
5 create schema wwi_staging
6 go
7 create schema wwi_ml
8 go
9
10 IF OBJECT_ID('wwi.DimCity', 'U') IS NOT NULL
11 | DROP TABLE wwi.DimCity
12 GO
13 |
14 --Location was eliminated (data type geography not supported)
15
16 CREATE TABLE [wwi].[DimCity]
17 (
18     [CityKey] [int] NOT NULL,
19     [WWICityID] [int] NOT NULL,
20     [City] [nvarchar](50) NOT NULL,
21     [StateProvince] [nvarchar](50) NOT NULL,
22     [Country] [nvarchar](60) NOT NULL,

```

The results pane below the editor shows a magnifying glass icon and the message 'No results to show' followed by 'Your query yielded no displayable results'. At the bottom of the results pane, a green status bar indicates '00:00:25 Query executed successfully.'

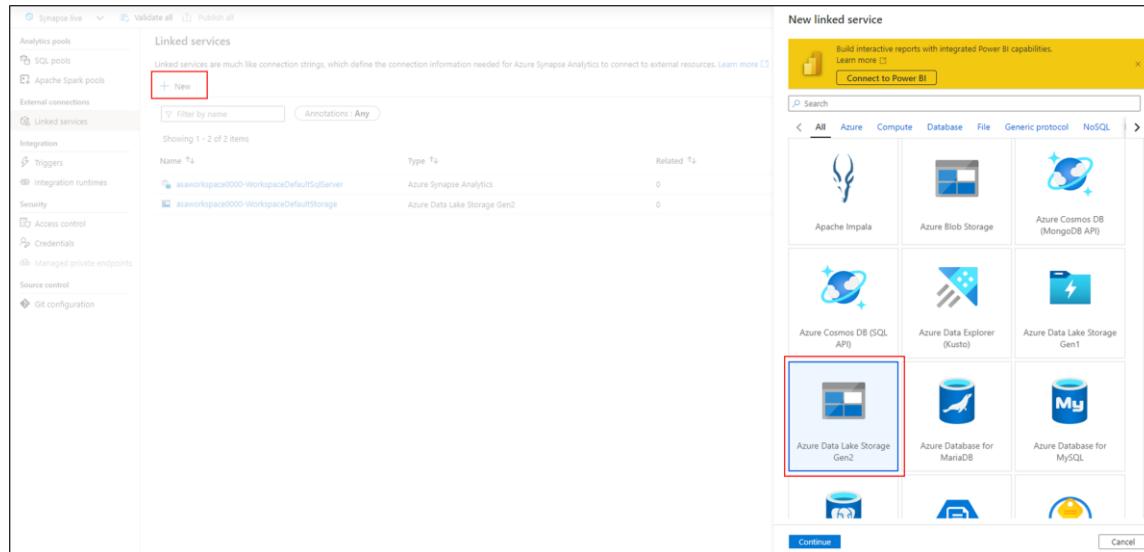
- 왼쪽 메뉴에서 Data를 클릭 하여 Data영역으로 이동 후 **SQLPool01(SQL)**의 **Tables**를 선택 하여 생성 된 테이블을 확인 합니다.

The screenshot shows the Microsoft Synapse Studio interface. On the left, there's a navigation sidebar with icons for Home, Data (which is selected), Develop, Integrate, Monitor, and Manage. The main area is titled "Data" and shows a "Workspace" tab selected. A search bar at the top says "Filter resources by name". Below it, there's a section for "Databases" which lists "SQLPool01 (SQL)" with a count of 2. Underneath, the database structure is detailed:

- Tables
  - wwi.DimCity
  - wwi.DimCustomer
  - wwi.DimDate
  - wwi.DimEmployee
  - wwi.DimPaymentMethod
  - wwi.DimStockItem
  - wwi.DimSupplier
  - wwi.DimTransactionType
  - wwi.FactMovement
  - wwi.FactOrder
  - wwi.FactPurchase
  - wwi.FactSale
  - wwi.FactStockHolding
  - wwi.FactTransaction
  - wwi\_perf.FactSale\_Fast
  - wwi\_perf.FactSale\_Slow
- External tables
- External resources

## 2.4. Task 4: Create Linked service and datasets

1. Synapse Studio의 Manage 화면에서 **Linked services**를 선택 합니다.
2. +New를 클릭 후 **Azure Data Lake Storage Gen2**를 선택 후 **Continue** 버튼을 클릭합니다.



3. New linked service 블레이드에서 아래 내용을 입력 후 **Test Connection**을 통해 연결 성공을 확인하고 **만들기(Create)** 버튼을 클릭합니다.
- ✓ Name: **asadatalake01**
  - ✓ Connect via integration runtime: **AutoResolveIntegrationRuntime** 선택
  - ✓ Authentication method: **Account Key** 선택
  - ✓ Account selection method: **From Azure subscription** 선택
  - ✓ Azure Subscripton: **본인의 구독** 선택
  - ✓ Storage account name: 이전에 생성한 **asaprimarystorageXXXX** 선택

New linked service (Azure Data Lake Storage Gen2)

Choose a name for your linked service. This name cannot be updated later.

Name \* asadatalake01

Description

Connect via integration runtime \* ⓘ AutoResolveIntegrationRuntime

Authentication method Account key

Account selection method ⓘ  
 From Azure subscription  Enter manually

Azure subscription ⓘ 사용자 구독 선택

Storage account name \* asaprimarystorage0000

Test connection ⓘ  
 To linked service  To file path

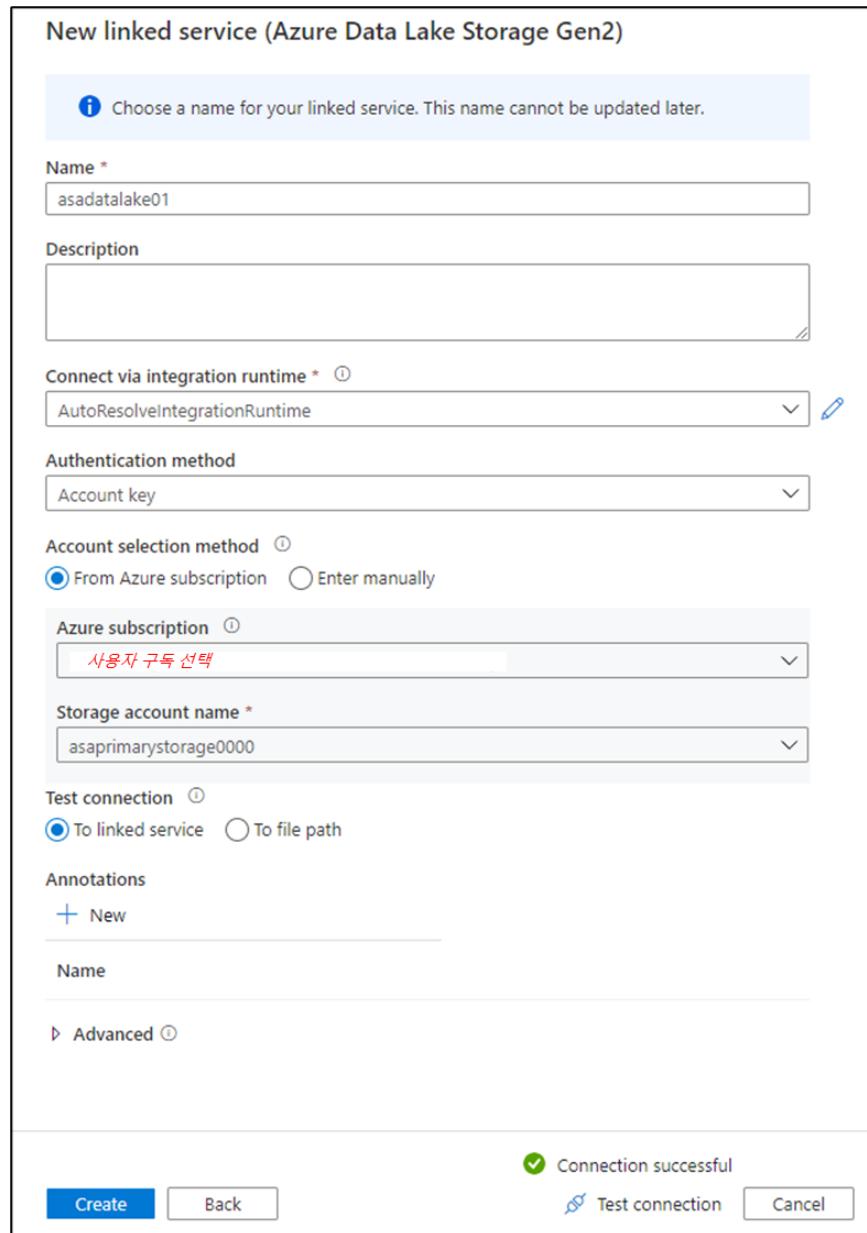
Annotations + New

Name

Advanced ⓘ

✓ Connection successful

Create Back ⚡ Test connection Cancel



4. Data영역으로 이동 하여 **Linked**를 클릭, 방금 생성한 **asadatalake01** → **dev**를 선택 합니다.  
파일 **postalcodedes.csv**를 선택 마우스 우 클릭 하여 **New Integration dataset**을 선택 합니다.

The screenshot shows the Azure Synapse Studio interface. On the left, the 'Data' blade displays a 'Linked' workspace. In the center, a list of files in the 'dev' folder of the 'bronze' container is shown. A context menu is open over the file 'postalcode.csv', with the 'New integration dataset' option highlighted by a red box.

5. Create integration dataset 블레이드에 아래 내용을 입력하고 **Create** 버튼을 클릭합니다.

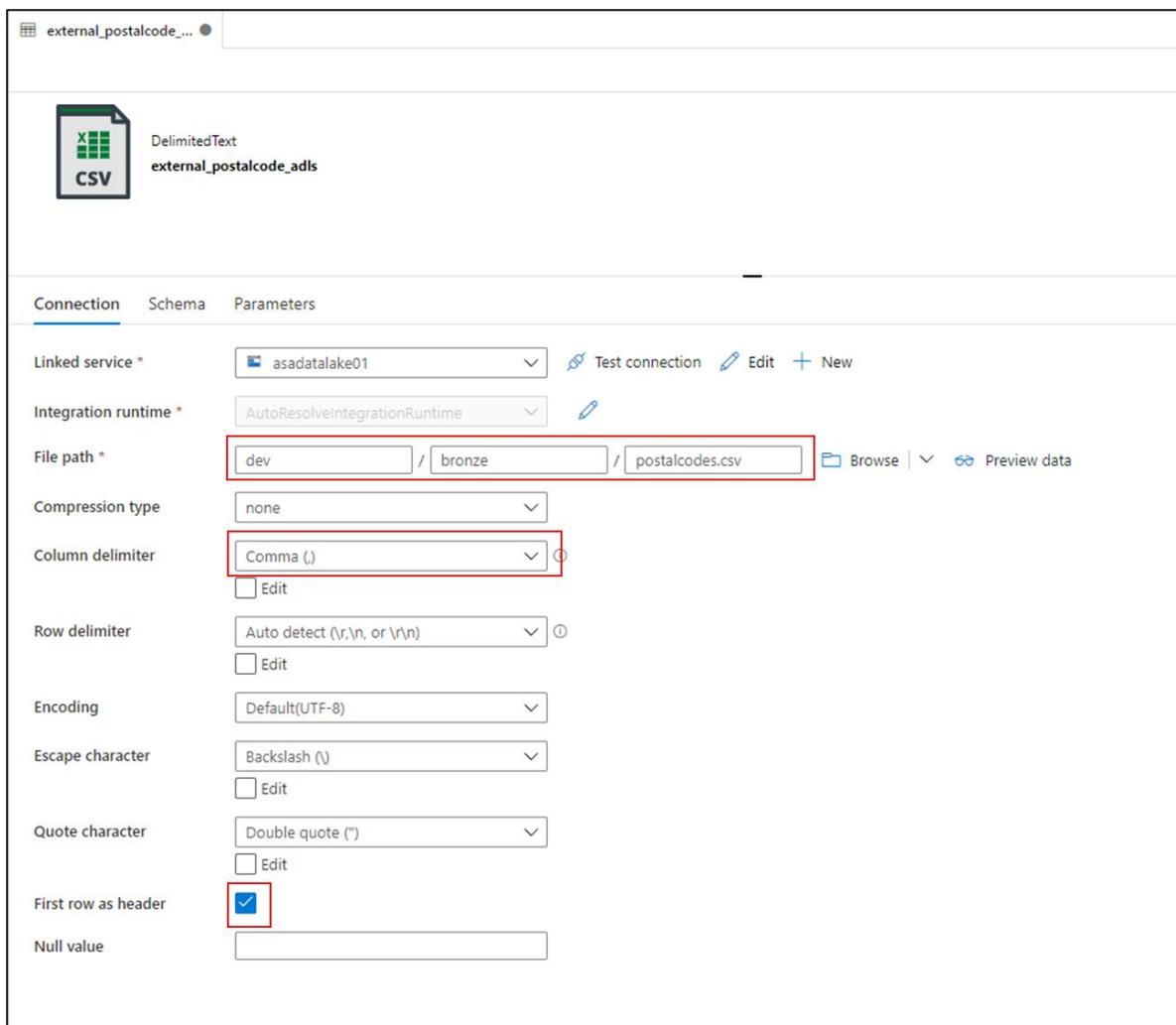
- ✓ Integration dataset name: **external\_postalcode\_adls**
- ✓ Format: **DelimitedText** 선택
- ✓ Import schema: **None** 선택

The screenshot shows the 'Create integration dataset' blade. It includes fields for 'Integration dataset name' (set to 'external\_postalcode\_adls'), 'Format' (set to 'DelimitedText'), and 'Import schema' (set to 'None'). The 'Create' button is visible at the bottom right of the blade.

6. 생성된 external\_postalcode\_adls에 아래 내용을 입력하여 Connection 설정을 합니다.

- ✓ File Path: **dev/bronze/postalcodes.csv**
- ✓ Compression type : **None**
- ✓ Column delimiter: **Comma (,)**
- ✓ Row delimiter: **Auto detect (\r,\n, or \r\n)**
- ✓ Encoding: **Default (UTF-8)**
- ✓ Escape character: **Backslash (\n)**
- ✓ Quote character: **Double quote ("")**
- ✓ First row as header: **Check**

\*First row as header 만 Check 해주세요 (나머지는 Default로 위의 내용으로 설정 됩니다.)



7. Schema를 선택하고 Import schema 버튼을 클릭, From connection/store를 선택 합니다..

파일의 첫 번째 Row를 컬럼명으로 자동으로 Schema 값이 채워 집니다.

external\_postalcode\_adls

DelimitedText  
external\_postalcode\_adls

Connection Schema Parameters

Import schema Clear

| Column name | Type   |
|-------------|--------|
| City        | String |
| State       | String |
| Zip         | String |
| Latitude    | String |
| Longitude   | String |
| Timezone    | String |
| DST         | String |

8. 상단의 **Publishing**을 클릭하여 Publish all 블레이드에서 목록을 확인하고 **Publish** 버튼을 클릭하여 변경 내용을 저장 합니다.

Synapse live ▾ Validate a Publishing 1

Data Workspace Linked

Filter resources by name

- Azure Data Lake Storage Gen2 2
- aseworkspace0000 (Primary - asa...)
- asadatalake01 (asaprimarystorage...)
  - asworkspace
  - dev
  - staging
  - wwi
- Integration datasets 1
  - external\_postalcode\_adls

external\_postalcode\_adls

DelimitedText  
external\_postalcode\_adls

Connection Schema Parameters

Import schema Clear

| NAME                     | CHANGE | EXISTING |
|--------------------------|--------|----------|
| external_postalcode_adls | (New)  | -        |

Publish all

You are about to publish all pending changes to the live environment. [Learn more](#)

Pending changes (1)

NAME CHANGE EXISTING

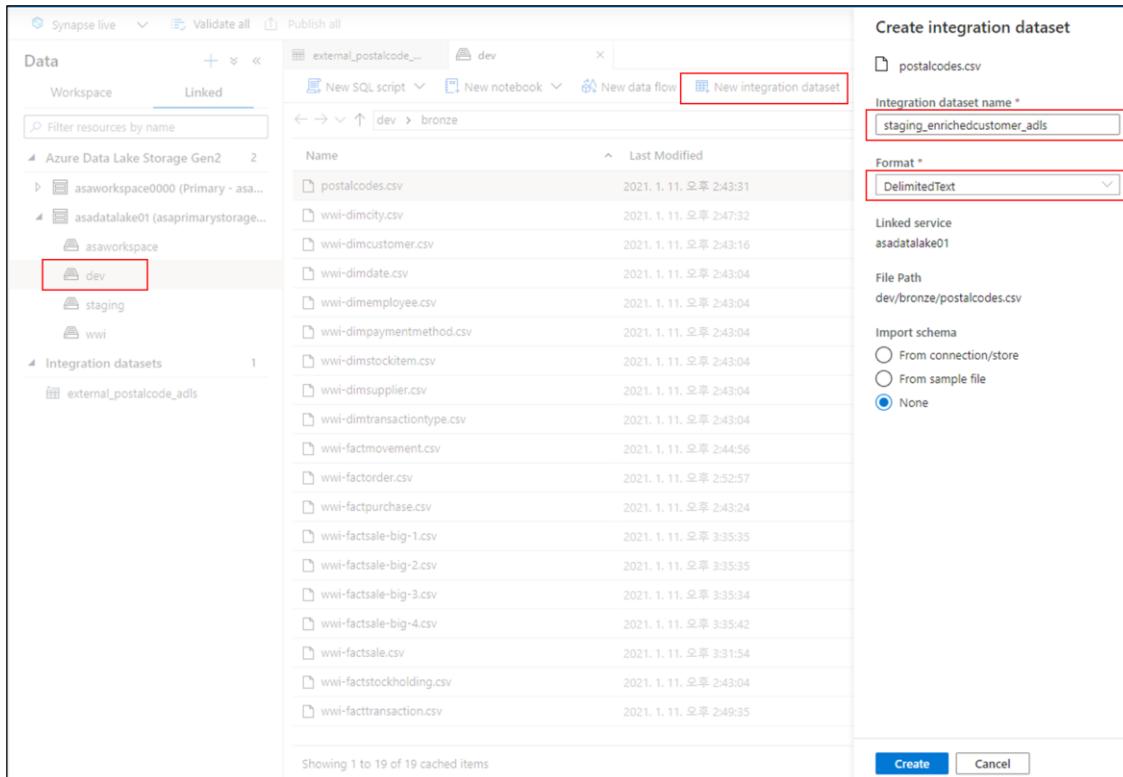
external\_postalcode\_adls (New) -

Publish Cancel

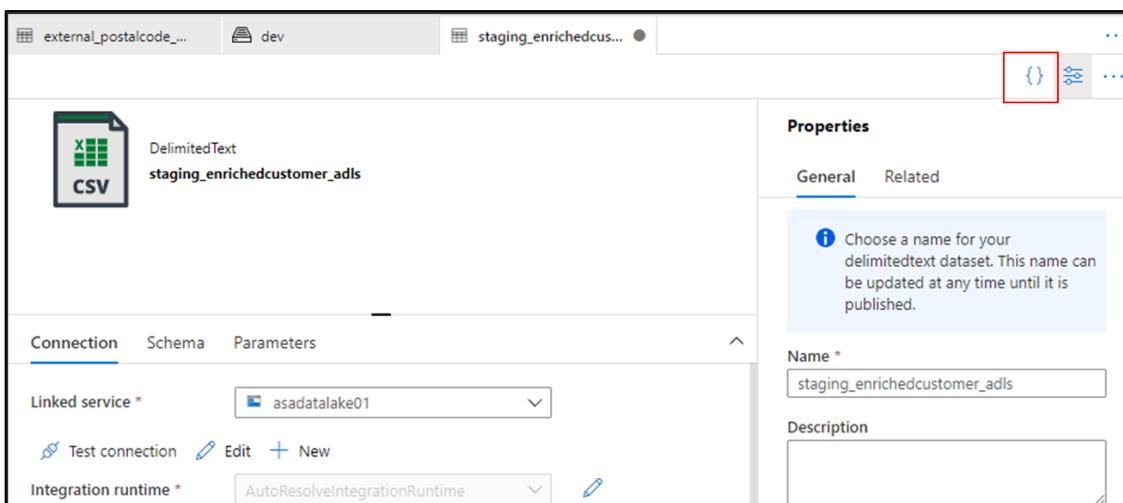
9. 다시 **asadatalake01** → **dev**로 이동하여 상단의 **New integration dataset**을 클릭합니다.  
Create integration dataset 블레이드에서 아래 내용을 입력 후 **Create** 버튼을 클릭합니다.

- ✓ Integration dataset name: **staging\_enrichedcustomer\_adls**
- ✓ Format: **DelimitedText** 선택
- ✓ Import schema: **None** 선택

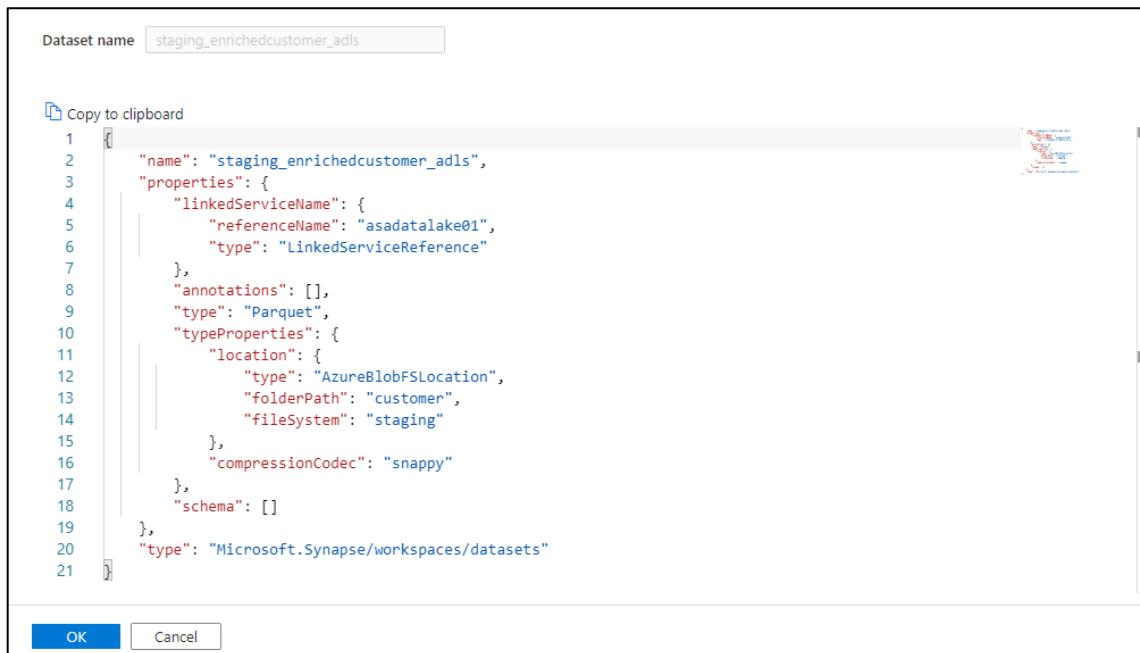
\*새로 Json코드를 입력하여 내용을 작성하여 dataset을 생성할 예정으로 File Path는 무시 합니다.



10. 오른쪽 상단의 {} 를 클릭하여 Code Editor 창으로 이동 하여 아래 코드를 추가하고 **OK** 버튼을 클릭합니다. dataset 정보가 CSV에서 Parquet으로 설정 내용도 함께 변경 된 것을 확인합니다.



```
{
    "name": "staging_enrichedcustomer_adls",
    "properties": {
        "linkedServiceName": {
            "referenceName": "asadatalake01",
            "type": "LinkedServiceReference"
        },
        "annotations": [],
        "type": "Parquet",
        "typeProperties": {
            "location": {
                "type": "AzureBlobFSLocation",
                "folderPath": "customer",
                "fileSystem": "staging"
            },
            "compressionCodec": "snappy"
        },
        "schema": []
    },
    "type": "Microsoft.Synapse/workspaces/datasets"
}
```



Data

Workspace Linked

Filter resources by name

- Azure Data Lake Storage Gen2 2
- asaworkspace0000 (Primary - asa...)
- asadatalake01 (asaprimarystorage...)
- asaworkspace
- dev
- staging
- wwi

Integration datasets 2

- external\_postalcode\_adls
- staging\_enrichedcustomer\_adls

staging\_enrichedcustomer\_adls

Parquet

Connection Schema Parameters

Linked service \* asadatalake01 Test connection Edit + New

Integration runtime \* AutoResolveIntegrationRuntime

File path \* staging / customer / File Browse Preview data

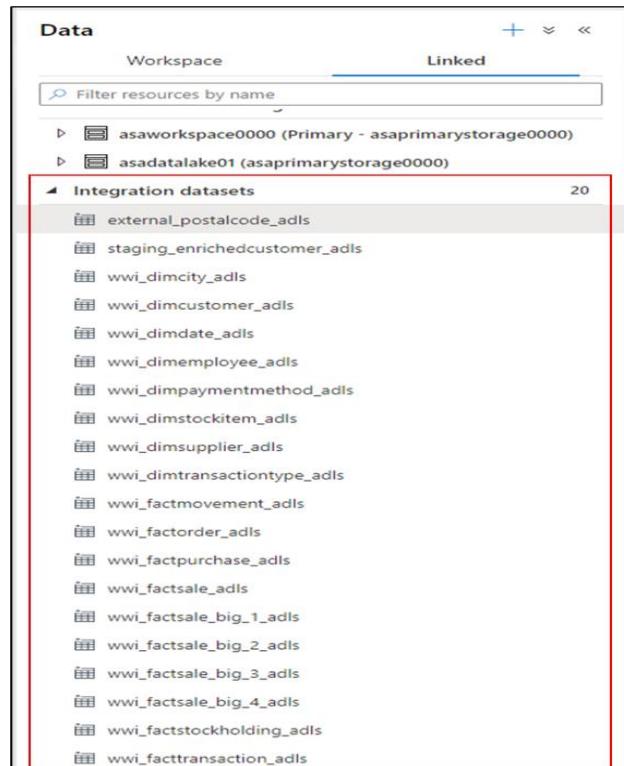
Compression type snappy

11. 위의 방법으로 Json 코드를 이용하여 wwi 파일들에 대한 dataset 생성 작업을 수행합니다.

Dataset 이름과 관련 Json 코드 파일은 아래 표를 참고 하시면 됩니다.

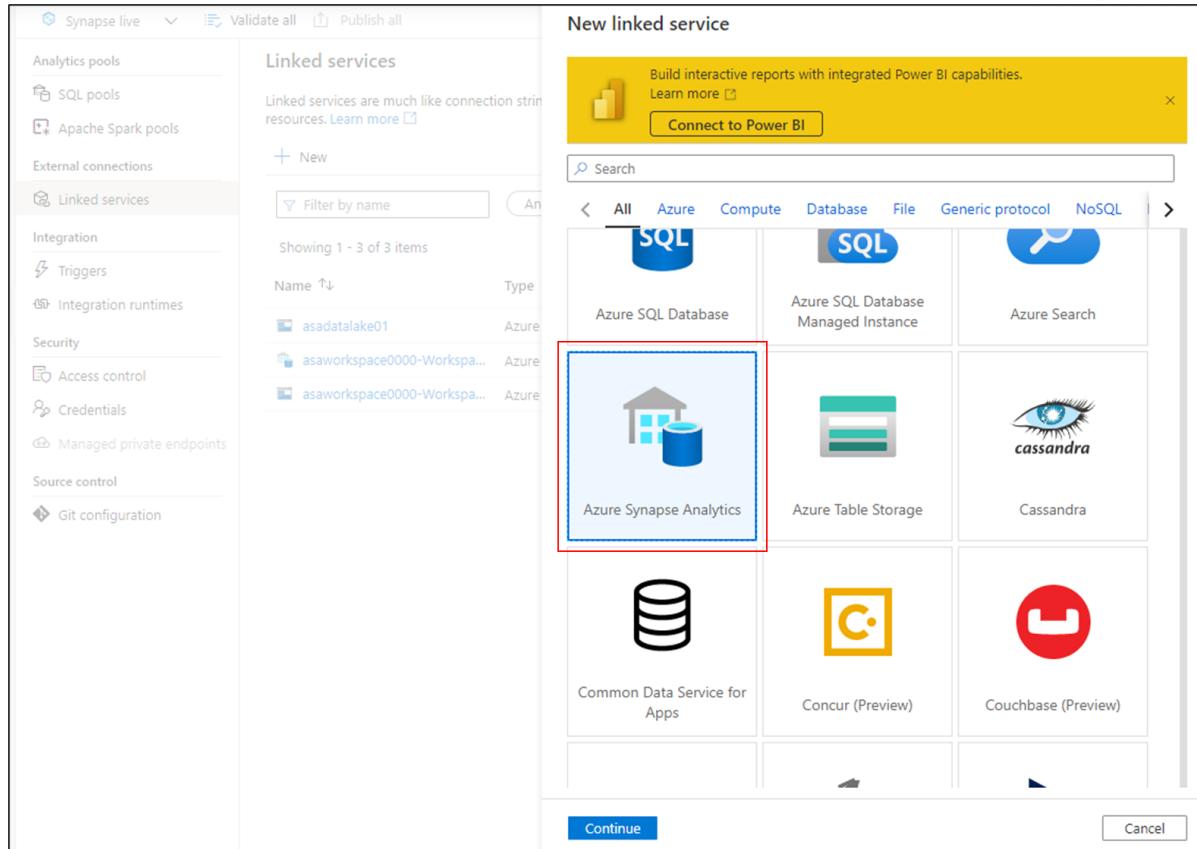
전부 작성이 완료 되면 상단의 **Publish all** 버튼을 클릭하여 저장 합니다.

| Dataset name                | source code file (Json)          |
|-----------------------------|----------------------------------|
| wwi_dimcity_adls            | wwi_dimcity_adls.json            |
| wwi_dimcustomer_adls        | wwi_dimcustomer_adls.json        |
| wwi_dimdate_adls            | wwi_dimdate_adls.json            |
| wwi_dimemployee_adls        | wwi_dimemployee_adls.json        |
| wwi_dimpaymentmethod_adls   | wwi_dimpaymentmethod_adls.json   |
| wwi_dimstockitem_adls       | wwi_dimstockitem_adls.json       |
| wwi_dimsupplier_adls        | wwi_dimsupplier_adls.json        |
| wwi_dimtransactiontype_adls | wwi_dimtransactiontype_adls.json |
| wwi_factmovement_adls       | wwi_factmovement_adls.json       |
| wwi_factorder_adls          | wwi_factorder_adls.json          |
| wwi_factpurchase_adls       | wwi_factpurchase_adls.json       |
| wwi_factsale_adls           | wwi_factsale_adls.json           |
| wwi_factsale_big_1_adls     | wwi_factsale_big_1_adls.json     |
| wwi_factsale_big_2_adls     | wwi_factsale_big_2_adls.json     |
| wwi_factsale_big_3_adls     | wwi_factsale_big_3_adls.json     |
| wwi_factsale_big_4_adls     | wwi_factsale_big_4_adls.json     |
| wwi_factstockholding_adls   | wwi_factstockholding_adls.json   |
| wwi_facttransaction_adls    | wwi_facttransaction_adls.json    |



12. Synapse Studio의 Manage 화면에서 **Linked services**를 선택 합니다

+**New**를 클릭 후 **Azure Synapse Analytics**를 선택 후 **Continue** 버튼을 클릭합니다



13. New linked service 블레이드에서 아래 내용을 입력 후 **Test Connection**을 통해 연결 성공을 확인

하고 **만들기(Create)** 버튼을 클릭합니다.

- ✓ Name: **sqlpool01**
- ✓ Connect via integration runtime: **AutoResolveIntegrationRuntime** 선택
- ✓ Account selection method: **From Azure subscription** 선택
- ✓ Azure Subscripton: **본인의 구독** 선택
- ✓ Server name: **asaworkspaceXXXX** 선택
- ✓ Database name: **SQLPool01**
- ✓ Authentication type: **SQL authentication** 선택
- ✓ User name: **sqladminuser**
- ✓ Password: **Demo@pass123**

New linked service (Azure Synapse Analytics)

Choose a name for your linked service. This name cannot be updated later.

Name \*  
sqlpool01

Description

Connect via integration runtime \* ⓘ  
AutoResolveIntegrationRuntime

Connection string Azure Key Vault

Account selection method ⓘ  
 From Azure subscription  Enter manually

Azure subscription  
사용자 구독 선택

Server name \*  
asaworkspace0000

Database name \*  
SQLPool01

Authentication type \*  
SQL authentication

User name \*  
sqladminuser

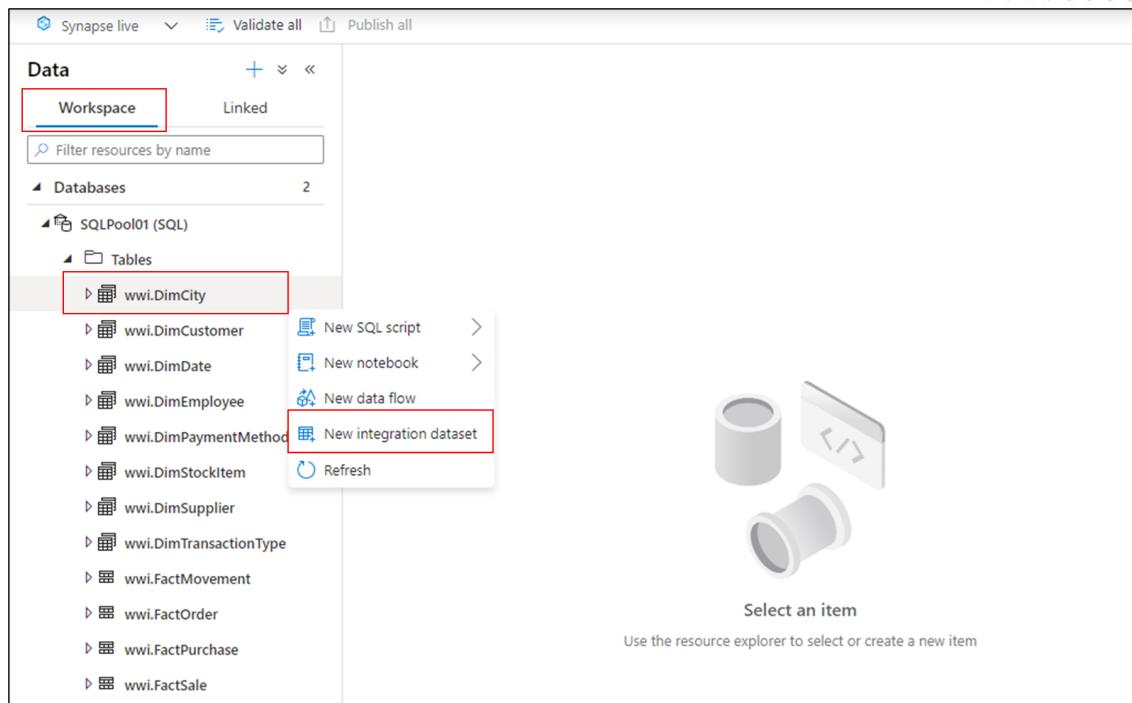
Password Azure Key Vault

Password \*  
\*\*\*\*\*

Connection successful

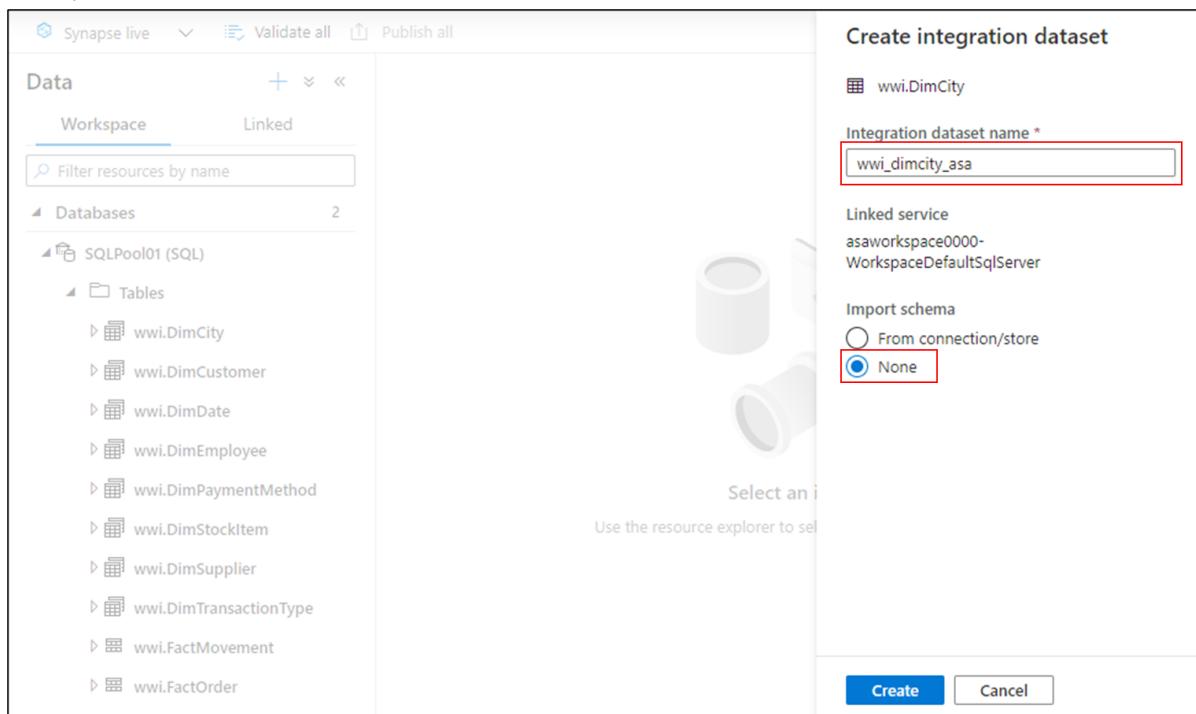
Create Back Test connection Cancel

14. Data영역으로 이동 하여 **Workspace**를 클릭, **SQLPool01** → **Tables** → **www.DimCity** 테이블을 선택 하고, 마우스 우 클릭 하여 **New Integration dataset**을 선택 합니다



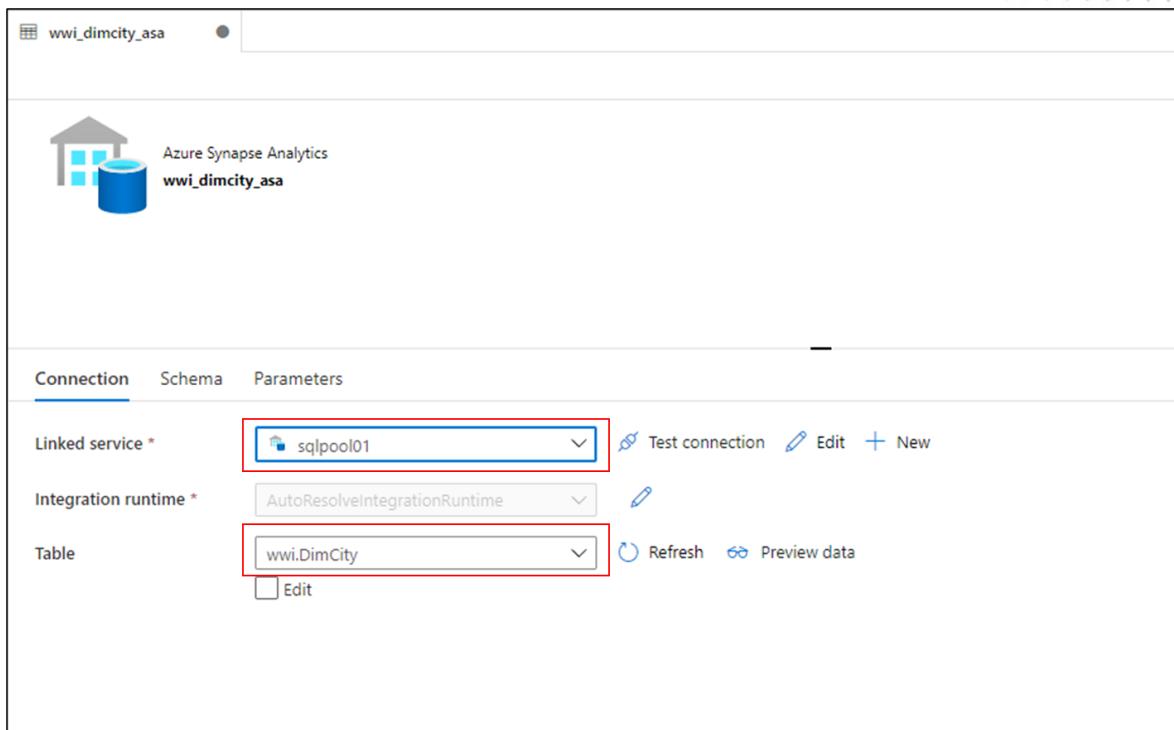
15. Create integration dataset 블레이드에 아래 내용을 입력하고 **Create** 버튼을 클릭합니다.

- ✓ Integration dataset name: **wwi\_dimcity\_asa**
- ✓ Import schema: **None** 선택



16. 생성된 **wwi\_dimcity\_asa**에 아래 내용을 입력하여 Connection 설정을 합니다.

- ✓ Linked service: **sqlpool01**
- ✓ Table : **wwi.DimCity**

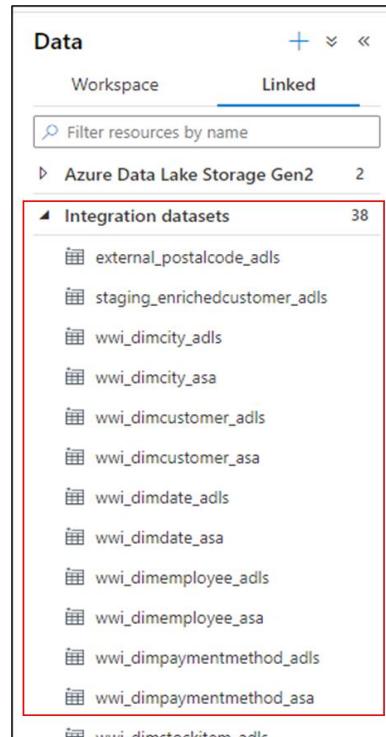


17. Schema를 선택하고 Import schema 버튼을 클릭 합니다.

| Column name              | Type      |
|--------------------------|-----------|
| CityKey                  | int       |
| WWICityID                | int       |
| City                     | nvarchar  |
| StateProvince            | nvarchar  |
| Country                  | nvarchar  |
| Continent                | nvarchar  |
| SalesTerritory           | nvarchar  |
| Region                   | nvarchar  |
| Subregion                | nvarchar  |
| LatestRecordedPopulation | bigint    |
| ValidFrom                | datetime2 |
| ValidTo                  | datetime2 |
| LineageKey               | int       |

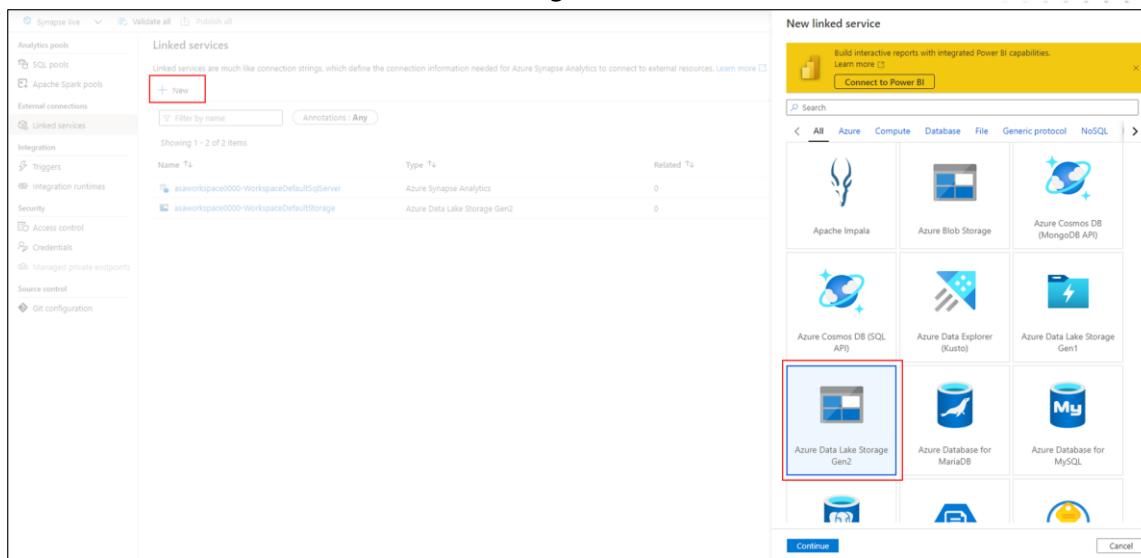
18. 상단의 Publish all 버튼을 클릭하여 변경 내용을 저장 합니다.
19. SQLPool의 테이블에 대한 Dataset도 파일의 Dataset과 동일하게 Json 코드를 이용하여 생성 할 수 있습니다. 아래 표를 참고 하여 나머지 테이블에 대한 Dataset을 Json코드를 이용하여 생성합니다. 전부 작성이 완료 되면 상단의 **Publish all** 버튼을 클릭하여 저장 합니다.

| Dataset name                     | source code file (Json)               |
|----------------------------------|---------------------------------------|
| wwi_dimcity_asa                  | wwi_dimcity_asa.json                  |
| wwi_dimcustomer_asa              | wwi_dimcustomer_asa.json              |
| wwi_dimdate_asa                  | wwi_dimdate_asa.json                  |
| wwi_dimemployee_asa              | wwi_dimemployee_asa.json              |
| wwi_dimpaymentmethod_asa         | wwi_dimpaymentmethod_asa.json         |
| wwi_dimstockitem_asa             | wwi_dimstockitem_asa.json             |
| wwi_dimsupplier_asa              | wwi_dimsupplier_asa.json              |
| wwi_dimtransactiontype_asa       | wwi_dimtransactiontype_asa.json       |
| wwi_factmovement_asa             | wwi_factmovement_asa.json             |
| wwi_factorder_asa                | wwi_factorder_asa.json                |
| wwi_factpurchase_asa             | wwi_factpurchase_asa.json             |
| wwi_factsale_asa                 | wwi_factsale_asa.json                 |
| wwi_factstockholding_asa         | wwi_factstockholding_asa.json         |
| wwi_facttransaction_asa          | wwi_facttransaction_asa.json          |
| wwi_perf_factsale_fast_asa       | wwi_perf_factsale_fast_asa.json       |
| wwi_perf_factsale_slow_asa       | wwi_perf_factsale_slow_asa.json       |
| wwi_staging_dimcustomer_asa      | wwi_staging_dimcustomer_asa.json      |
| wwi_staging_enrichedcustomer_asa | wwi_staging_enrichedcustomer_asa.json |



20. Synapse Studio의 Manage 화면에서 **Linked services**를 선택 합니다.

21. +New를 클릭 후 **Azure Data Lake Storage Gen2**를 선택 후 **Continue** 버튼을 클릭합니다.



22. New linked service 블레이드에서 아래 내용을 입력 후 **Test Connection**을 통해 연결 성공을 확인하고 **만들기(Create)** 버튼을 클릭합니다.

- ✓ Name: **asastore01**
- ✓ Connect via integration runtime: **AutoResolveIntegrationRuntime** 선택
- ✓ Authentication method: **Account Key** 선택
- ✓ Account selection method: **From Azure subscription** 선택
- ✓ Azure Subscripton: **본인의 구독** 선택
- ✓ Storage account name: 이전에 생성한 **asablobstorageXXXX** 선택

## New linked service (Azure Data Lake Storage Gen2)

*Choose a name for your linked service. This name cannot be updated later.*

Name \*

asastore01

Description

Connect via integration runtime \* ⓘ

AutoResolveIntegrationRuntime



Authentication method

Account key



Account selection method ⓘ

From Azure subscription  Enter manually

Azure subscription ⓘ

사용자 구독 선택

Storage account name \*

asablobstorage0000



Test connection ⓘ

To linked service  To file path

Annotations

+ New

Name

Advanced ⓘ

✓ Connection successful

Create

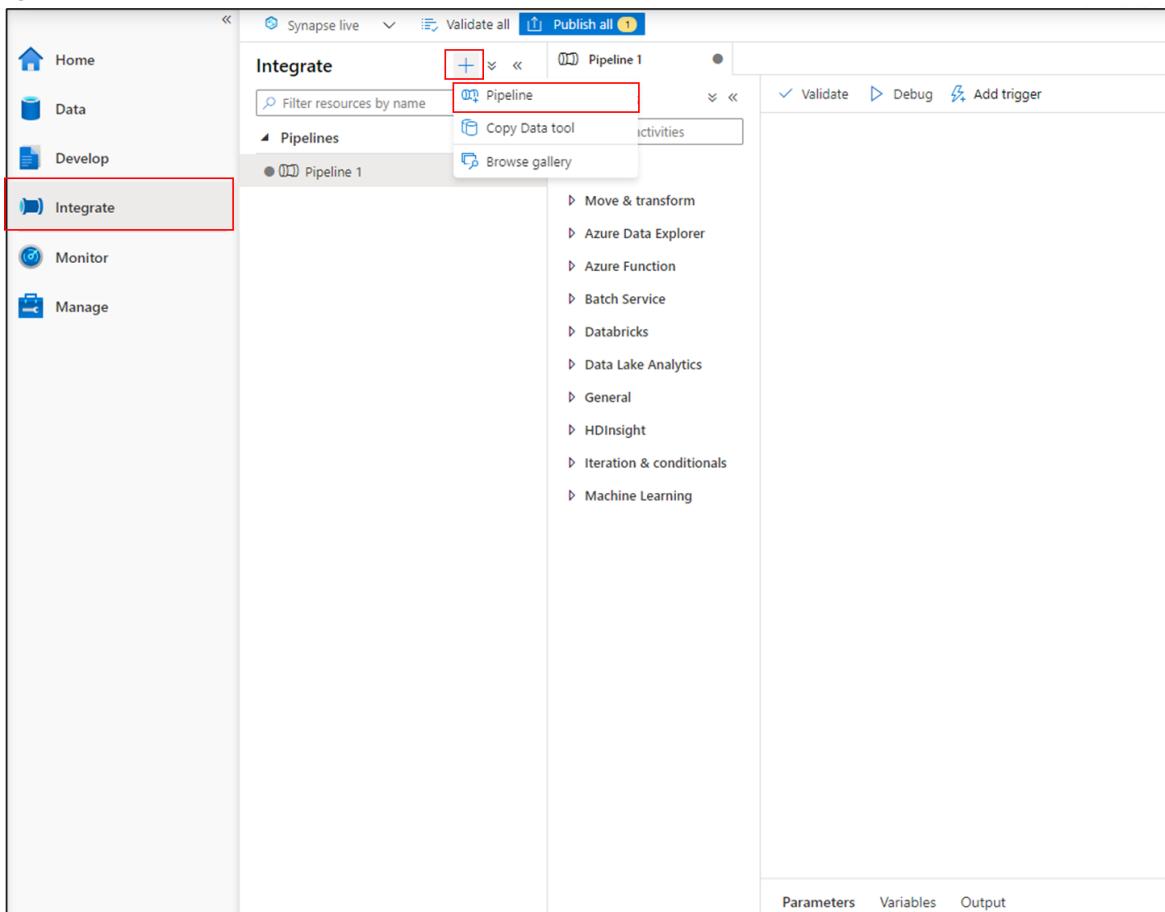
Back

🔗 Test connection

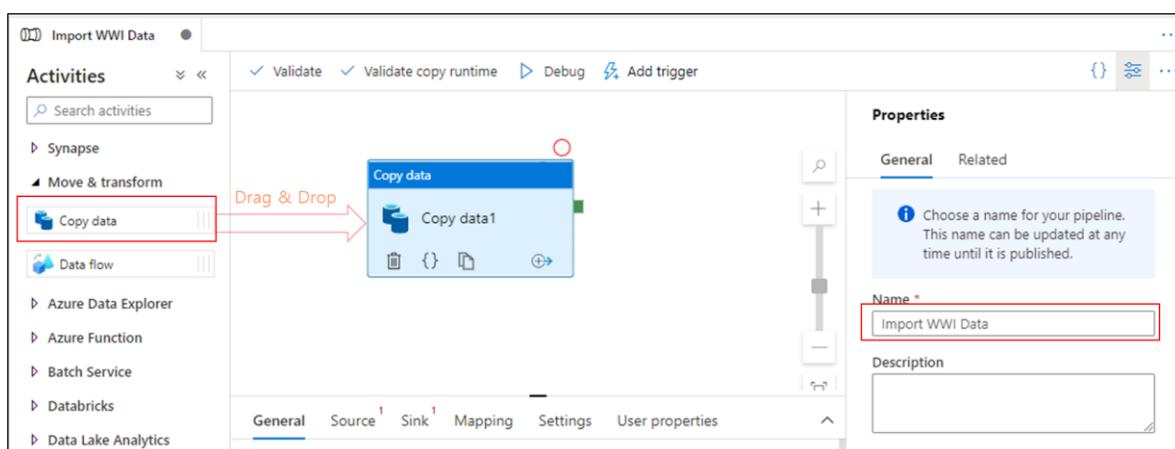
Cancel

## 2.5. Task 5: Create Azure Synapse Pipelines & Run

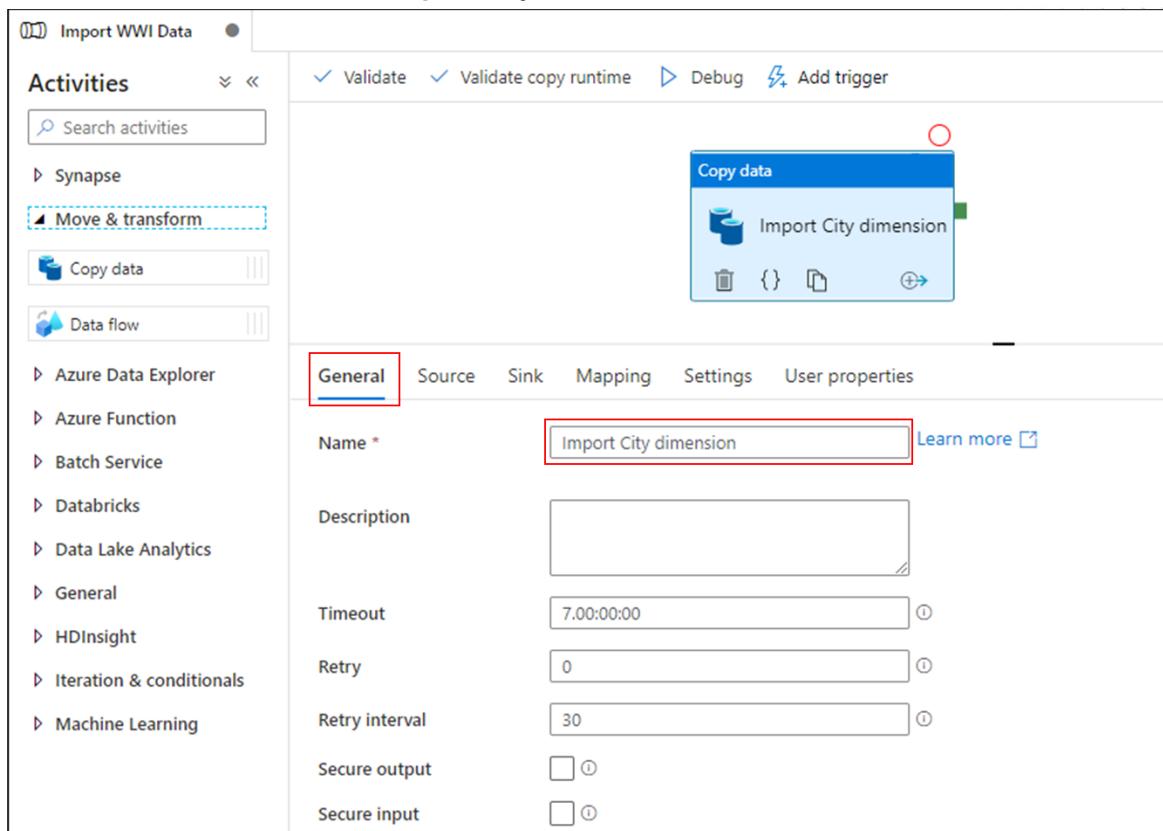
1. Synapse Studio의 왼쪽 메뉴에서 **Integrate** 클릭하여 Integrate 영역으로 이동하고, **+**를 클릭 **Pipeline**을 선택 합니다.



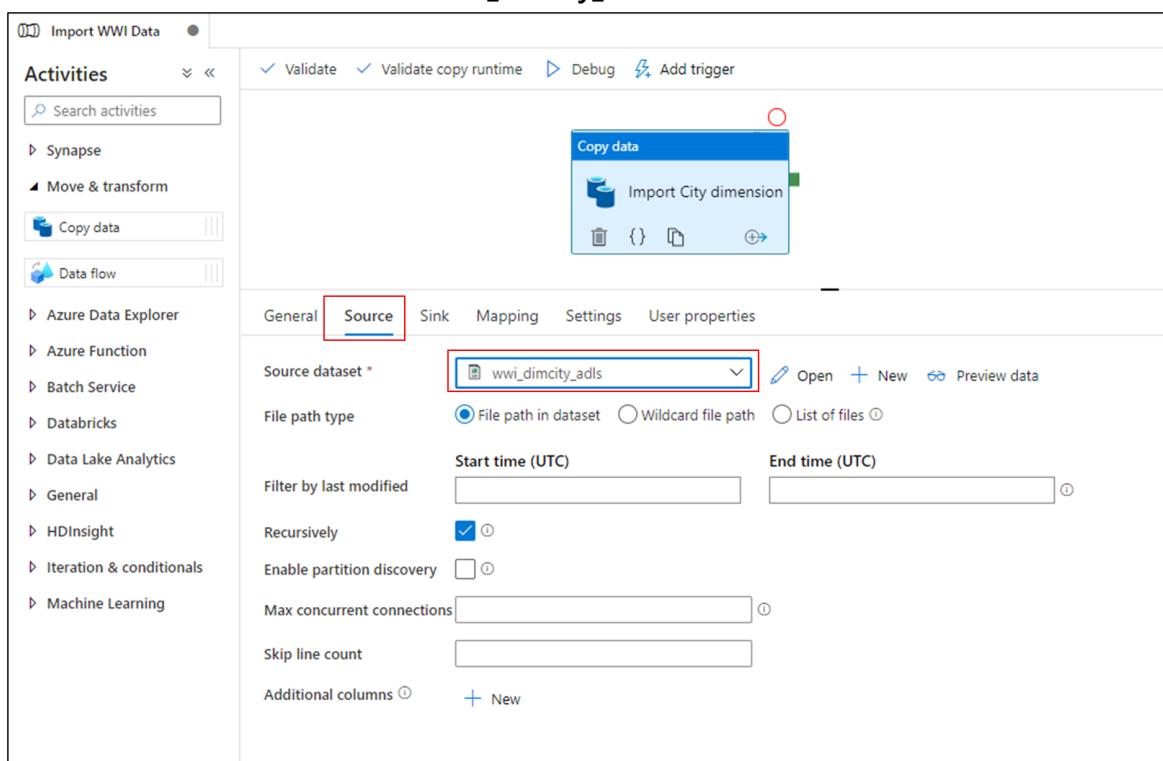
2. Properties 블레이드의 Name란에 **Import WWI Data**를 입력하고, Activities의 **Move&transform** → **Copy data**를 Drag&Drop으로 Canvas에 가져다 놓습니다.



3. General Tab에서 Name 항목에 Import City dimension을 입력 합니다.

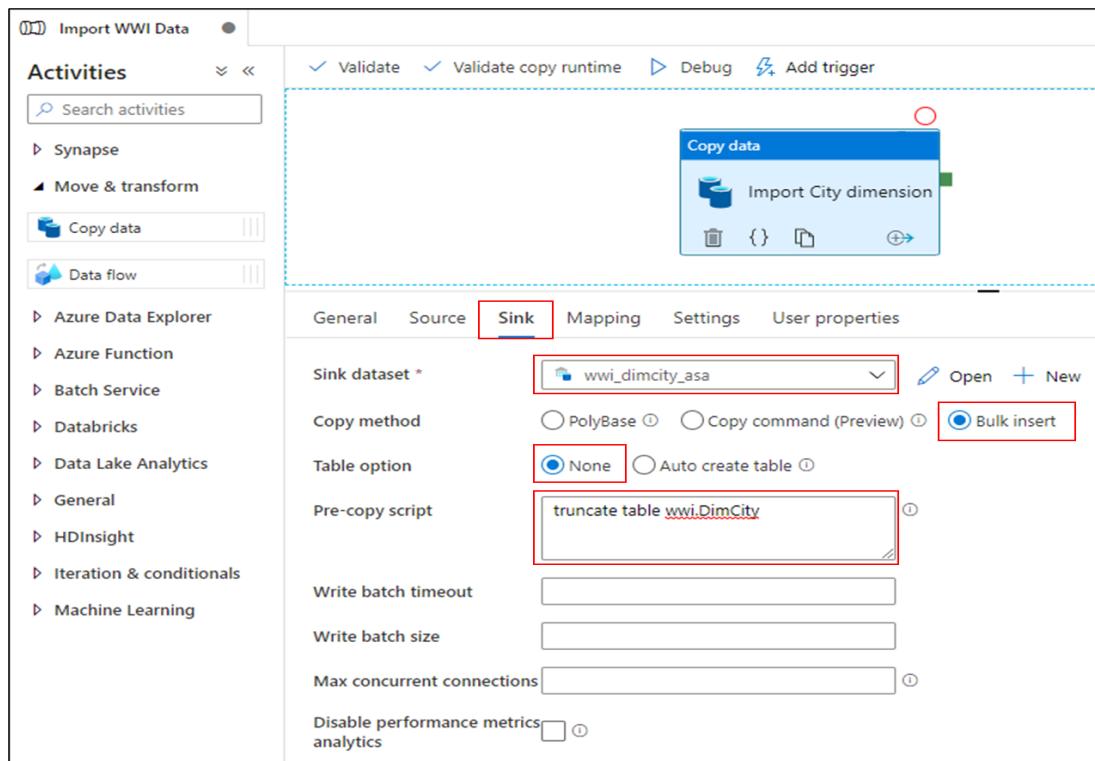


4. Source Tab에서 Source dataset을 www\_dimcity\_adls를 선택합니다.

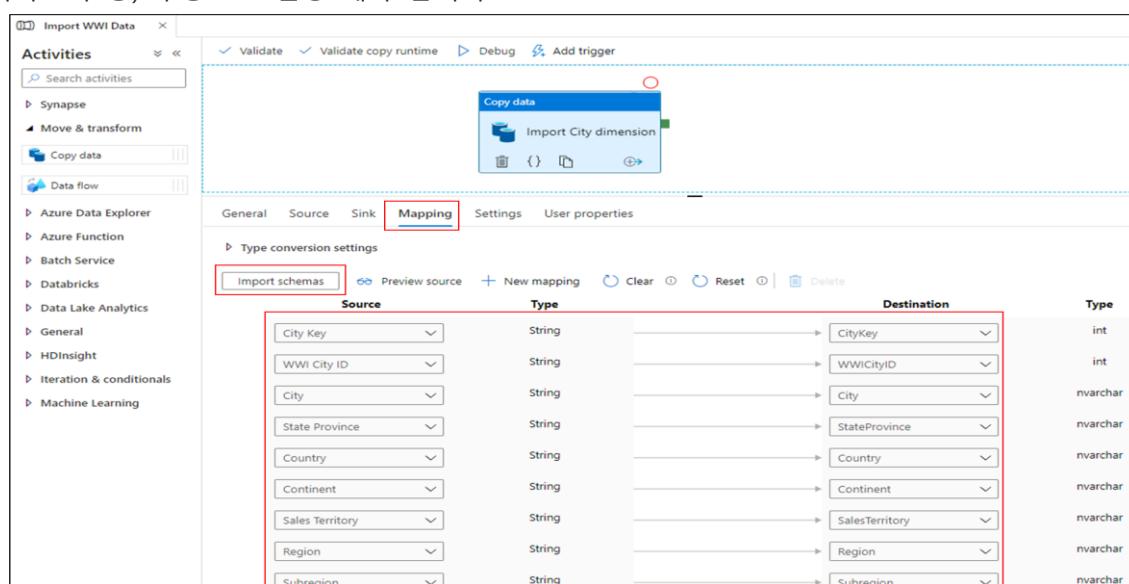


5. Sink Tab에서 아래 내용으로 설정 합니다.

- ✓ Sink dataset: **wwi\_dimcity\_asa** 선택
- ✓ Copy method: **Bulk insert** 선택
- ✓ Table option: **None** 선택
- ✓ Pre-copy script: **truncate table wwi.DimCity** 입력



6. Mapping Tab에서 Import schemas 버튼을 클릭하여 Source 컬럼과 Destination 컬럼을 일치시킵니다. 이 때 Source와 Destination의 컬럼명은 자동으로 Mapping 되나 컬럼명이 다를 경우(띄어쓰기 등) 수동으로 설정 해야 합니다.



7. Source와 Destination의 Type을 Json 코드를 수정하여 변경 합니다.

City Key, WWI City ID 등의 Type 불일치 컬럼을 코드 편집기를 열고 수정 후 OK 버튼을 클릭합니다.

```

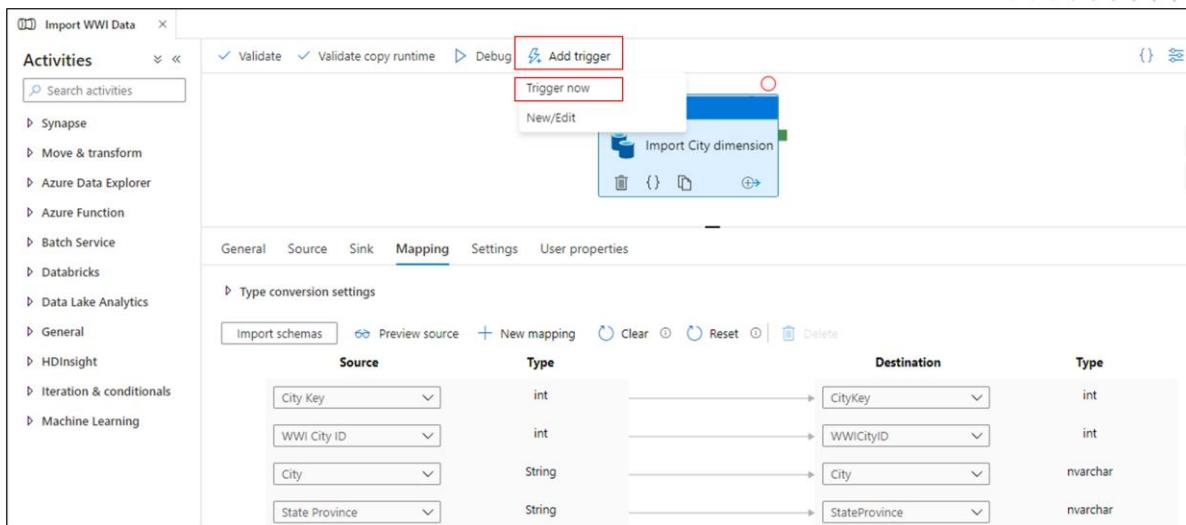
mappings : [
    {
        "source": {
            "name": "City Key",
            "type": "Int32",
            "physicalType": "int"
        },
        "sink": {
            "name": "CityKey",
            "type": "Int32",
            "physicalType": "int"
        }
    },
    {
        "source": {
            "name": "WWI City ID",
            "type": "Int32",
            "physicalType": "int"
        },
        "sink": {
            "name": "WWICityID",
            "type": "Int32",
            "physicalType": "int"
        }
    }
],
{
    "source": {
        "name": "Latest Recorded Population",
        "type": "Int32",
        "physicalType": "int"
    },
    "sink": {
        "name": "LatestRecordedPopulation",
        "type": "Int64",
        "physicalType": "bigint"
    }
},
{
    "source": {
        "name": "Valid From",
        "type": "DateTime",
        "physicalType": "DateTime"
    },
    "sink": {
        "name": "ValidFrom",
        "type": "DateTime",
        "physicalType": "datetime2"
    }
},
{
    "source": {
        "name": "Valid To",
        "type": "DateTime",
        "physicalType": "DateTime"
    },
    "sink": {
        "name": "ValidTo",
        "type": "DateTime",
        "physicalType": "datetime2"
    }
},
{
    "source": {
        "name": "Lineage Key",
        "type": "Int32",
        "physicalType": "int"
    },
    "sink": {
        "name": "LineageKey",
        "type": "Int32",
        "physicalType": "int"
    }
}
]
}

```

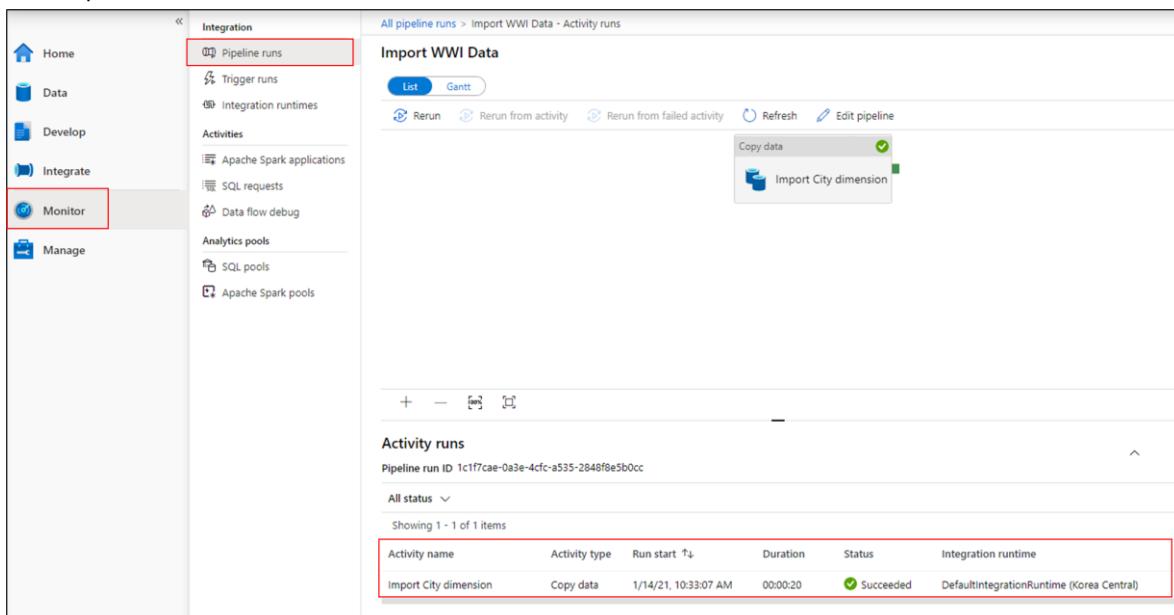
8. 코드 변경 후 수정 된 내용을 확인 합니다.

| Source                  | Type   | Destination             | Type     |
|-------------------------|--------|-------------------------|----------|
| City Key                | int    | CityKey                 | int      |
| WWI City ID             | int    | WWICityID               | int      |
| City                    | String | City                    | nvarchar |
| State Province          | String | StateProvince           | nvarchar |
| Country                 | String | Country                 | nvarchar |
| Continent               | String | Continent               | nvarchar |
| Sales Territory         | String | SalesTerritory          | nvarchar |
| Region                  | String | Region                  | nvarchar |
| Subregion               | String | Subregion               | nvarchar |
| Latest Recorded Popu... | int    | LatestRecordedPopula... | bigint   |

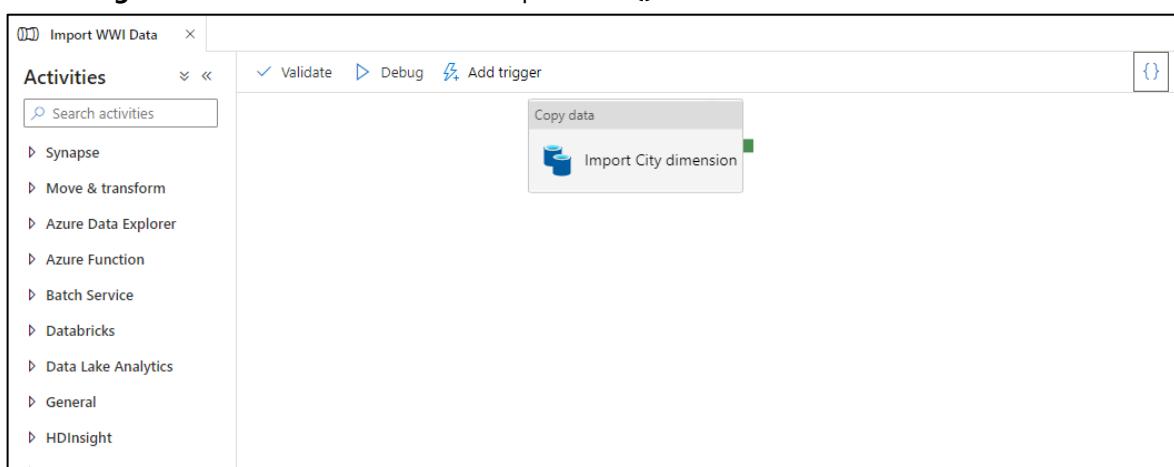
9. Add trigger → Trigger now를 선택 하여 Pipeline을 실행 합니다.



10. 왼쪽 메뉴의 **Monitor** → **Pipeline runs**를 선택하시면 Pipeline 수행목록이 나오고 Pipeline을 선택하면 Pipeline의 상세 모니터링을 확인 할 수 있습니다.



11. 다시 **Integrate**로 돌아와서 방금 수행한 Pipeline의 {} 를 클릭하여 코드 편집 창을 띄웁니다.



12. 코드 편집 창에 **Import WWI Data.json** 파일의 내용을 복사하여 붙여 넣고 **OK** 버튼을 클릭합니다.

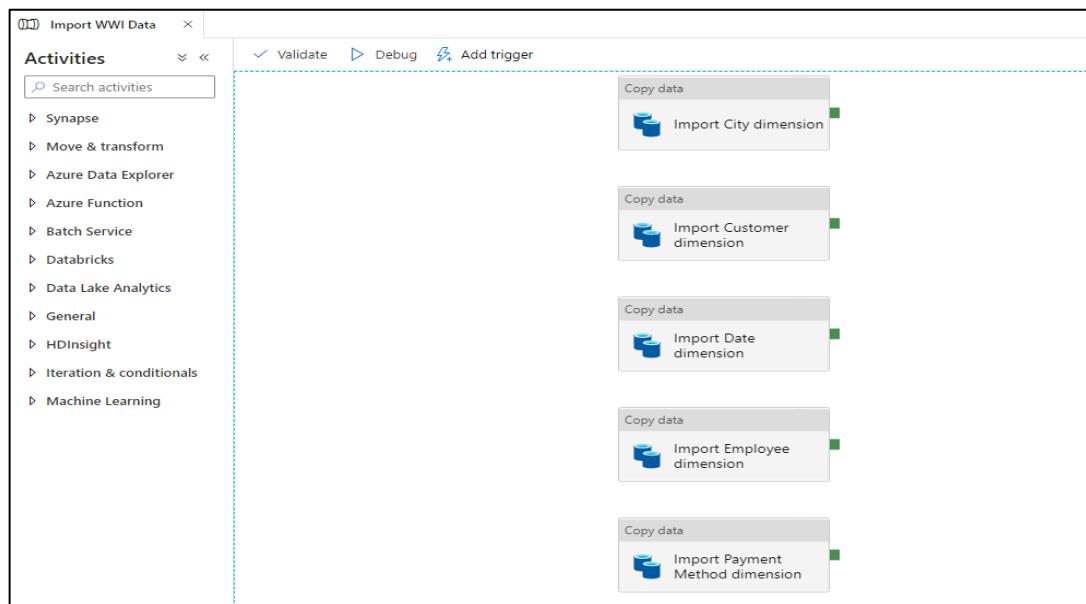
```
Pipeline name Import WWI Data

Copy to clipboard

2110     "name": "IsFinalized",
2111     "type": "Boolean"
2112   },
2113   {
2114     "source": {
2115       "name": "Lineage Key",
2116       "type": "Int32"
2117     },
2118     "sink": {
2119       "name": "LineageKey",
2120       "type": "Int32"
2121     }
2122   }
2123 ],
2124 ],
2125 },
2126 "inputs": [
2127   {
2128     "referenceName": "wwi_facttransaction_adls",
2129     "type": "DatasetReference"
2130   }
2131 ],
2132 "outputs": [
2133   {
2134     "referenceName": "wwi_facttransaction_asa",
2135     "type": "DatasetReference"
2136   }
2137 ],
2138 ],
2139 ],
2140 ],
2141 "annotations": []
2142 },
2143 "type": "Microsoft.Synapse/workspaces/pipelines"
2144

OK Cancel
```

13. 다른 테이블들에 대해서도 Copy data Activity가 생성 되어진 것을 확인 할 수 있습니다.



14. **Publish all**을 클릭 하여 수정된 Pipeline을 저장 후 **Add trigger → Trigger now**를 클릭하여 Pipeline을 실행합니다. 파이프라인 실행이 완료 되면 Data영역으로 이동하여 테이블에 데이터가 적재 되었는지 확인합니다.

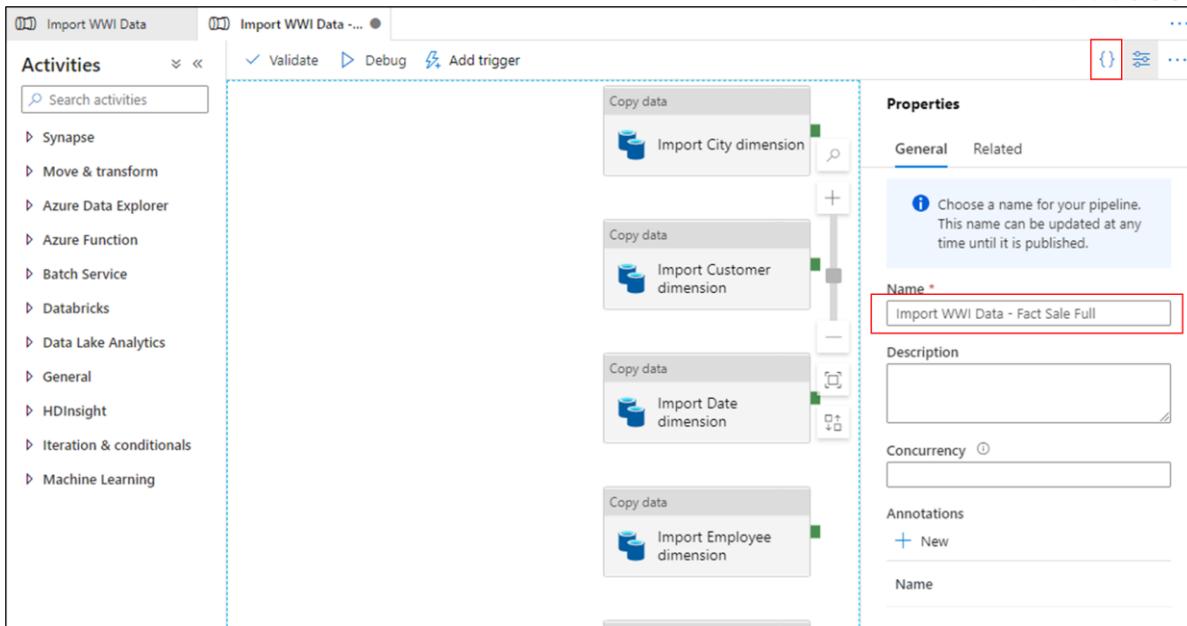
The screenshot shows the Azure Data Studio interface with the 'Data' workspace selected. On the left, the 'Tables' section of the 'SQLPool01 (SQL)' database is visible, with 'wwi.DimCustomer' selected. A context menu is open over this table, with two options highlighted by a red box: 'New SQL script' and 'Select TOP 100 rows'. The main pane displays the results of a query against the 'wwi.DimCustomer' table, showing columns such as CustomerKey, WWICustomerID, Customer, BillToCustomer, Category, BuyingGroup, PrimaryContact, PostalCode, and ValidFrom. The results show several rows of data from the Tailspin Toys dataset.

| CustomerKey | WWICustomerID | Customer             | BillToCustomer      | Category     | BuyingGroup   | PrimaryContact | PostalCode | ValidFrom     |
|-------------|---------------|----------------------|---------------------|--------------|---------------|----------------|------------|---------------|
| 0           | 0             | Unknown              | N/A                 | N/A          | N/A           | N/A            | N/A        | 2013-01-01T00 |
| 135         | 135           | Tailspin Toys (Te... | Tailspin Toys (H... | Novelty Shop | Tailspin Toys | Gaurav Sikdar  | 90145      | 2013-01-01T00 |
| 271         | 470           | Wingtip Toys (U...   | Wingtip Toys (L...  | Novelty Shop | Wingtip Toys  | Libuse Srbova  | 90500      | 2013-01-01T00 |
| 1           | 1             | Tailspin Toys (H...  | Tailspin Toys (H... | Novelty Shop | Tailspin Toys | Waldemar Fisar | 90410      | 2013-01-01T00 |

15. Import WWI Data의 오른쪽 ...를 클릭하여 확장하고 Clone을 클릭 Pipeline을 추가 합니다.  
Pipeline을 복제하지 않고 Integrate 의 +를 클릭하여 Pipeline을 추가하여도 됩니다.

The screenshot shows the Azure Data Studio interface with the 'Integrate' workspace selected. On the left, the 'Pipelines' section shows a single pipeline named 'Import WWI Data'. A context menu is open over this pipeline, with the 'Clone' option highlighted by a red box. The right side of the screen shows the 'Activities' view, which includes sections for 'Copy data' (Import City dimension, Import Customer dimension, Import Date dimension) and 'General' (HDInsight, Iteration & conditionals).

16. 추가된 Pipeline의 Name을 Import WWI Data – Fact Sale Full로 수정하고 {} 를 클릭하여 코드 편집창을 엽니다.



17. 코드 편집 창에 Import WWI Data - Fact Sale Full.json 파일의 내용을 복사하여 붙여 넣고 OK 버튼을 클릭합니다

Pipeline name: Import WWI Data - Fact Sale Full

```

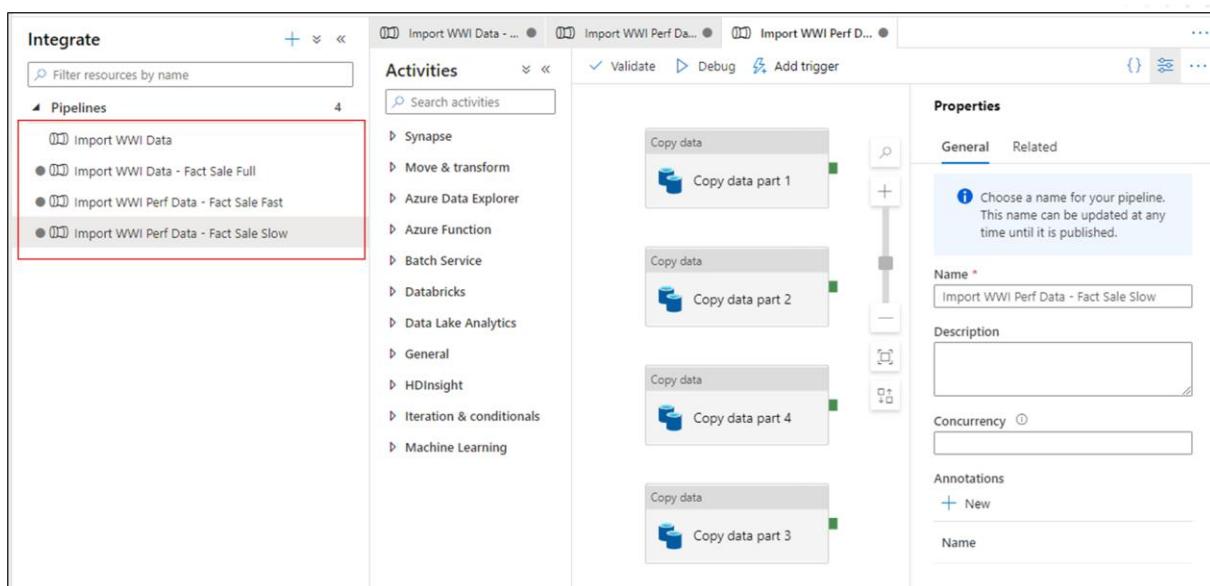
Copy to clipboard
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
    },
    "sink": {
        "name": "TotalChillerItems",
        "type": "Int32"
    }
},
{
    "source": {
        "name": "Lineage Key",
        "type": "Int32"
    },
    "sink": {
        "name": "LineageKey",
        "type": "Int32"
    }
}
],
"inputs": [
{
    "referenceName": "wwi_factsale_adls",
    "type": "DatasetReference"
}
],
"outputs": [
{
    "referenceName": "wwi_factsale_asa",
    "type": "DatasetReference"
}
]
},
"annotations": []
},
"type": "Microsoft.Synapse/workspaces/pipelines"

```

**OK**   **Cancel**

## 18. 동일한 방법으로 Import WWI Perf Data - Fact Sale Fast.json , Import WWI Perf Data - Fact Sale Slow.json 파일을 이용하여 WWI Perf Data - Fact Sale Fast, WWI Perf Data - Fact Sale Slow Pipeline을 생성합니다.

아래 그림과 같이 총 4개의 Pipeline이 생성 되어야 하며 완료 후 Publish all 을 클릭하여 저장합니다.

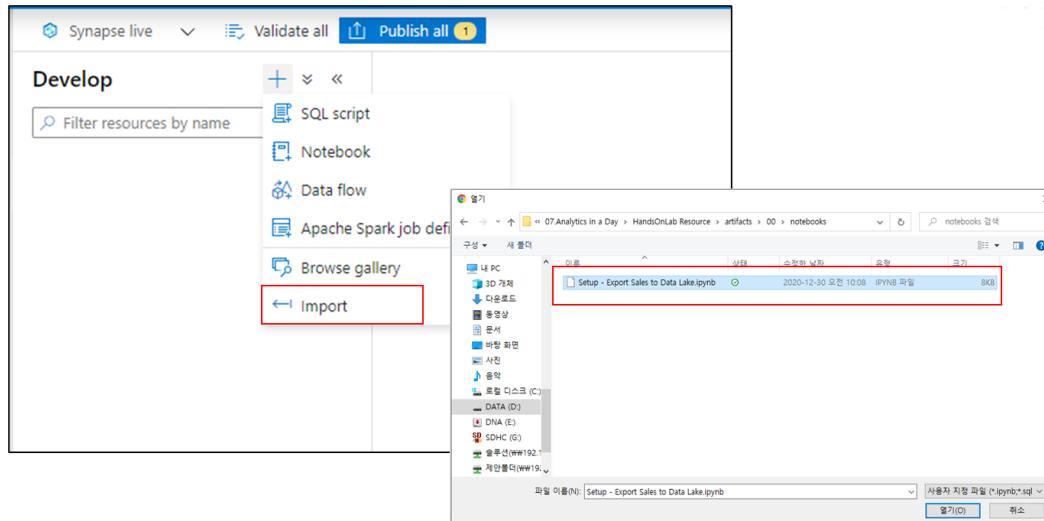


19. 추가로 생성 된 3개의 Pipeline 각각에서 Add Trigger → Trigger now를 클릭하여 Pipeline을 실행하고 실행 결과를 확인 합니다. 추가로 생성 된 3개의 Pipeline은 Polybase 기능을 이용하여 적재하였습니다.

The screenshot shows the 'Pipeline runs' page in the Azure Data Factory portal. The left sidebar has sections for 'Integration', 'Activities', and 'Analytics pools'. The main area displays a table of pipeline runs. The table has columns for 'Pipeline name', 'Run start', 'Run end', 'Duration', 'Triggered by', and 'Status'. Three rows are highlighted with a red box: 'Import WWI Perf Data - Fact Sale Slow' (Run start: 1/14/21, 2:01:19 PM; Run end: 1/14/21, 2:10:05 PM; Status: Succeeded), 'Import WWI Perf Data - Fact Sale Fast' (Run start: 1/14/21, 2:01:14 PM; Run end: 1/14/21, 2:19:05 PM; Status: Succeeded), and 'Import WWI Data - Fact Sale Full' (Run start: 1/14/21, 2:01:07 PM; Run end: 1/14/21, 2:08:34 PM; Status: Succeeded). The 'Triggered by' column shows 'Manual trigger' for all runs.

## 2.6. Task 6: Populate Primarystorage with data

1. **Develop** 영역으로 이동하고, **+**를 클릭 **Import**를 선택하여 **Setup - Export Sales to Data Lake.ipynb** 파일을 불러 옵니다.



2. Python 코드에서 <primary\_storage>를 **asapprimarystorageXXXX** (primary storage 이름)으로 변경 합니다. (Cell1 , Cell4, Cell6에 있습니다.)

The screenshot shows the Microsoft Azure Synapse Studio Notebook interface. The notebook is titled 'Setup - Export Sales to Data Lake'. The code in Cell 1 is as follows:

```

1 %pyspark
2 wwi_sales = spark \
3     .read \
4     .option("sep", ",") \
5     .load("abfs://dev/asapprimarystorage0001.dfs.core.windows.net/bronze/wwi-factsale.csv", format="csv", header=True)
6
7 wwi_sales = wwi_sales \
8     .withColumnRenamed('Sale Key', 'SaleKey') \
9     .withColumnRenamed('City Key', 'CityKey') \
10    .withColumnRenamed('Customer Key', 'CustomerKey') \
11    .withColumnRenamed('Bill To Customer Key', 'BillToCustomerKey') \
12    .withColumnRenamed('Stock Item Key', 'StockItemKey') \
13    .withColumnRenamed('Invoice Date Key', 'InvoiceDateKey') \
14    .withColumnRenamed('Delivery Date Key', 'DeliveryDateKey') \
15    .withColumnRenamed('Salesperson Key', 'SalespersonKey') \
16    .withColumnRenamed('WMT Invoice ID', 'WMTInvoiceID') \
17    .withColumnRenamed('Description', 'Description') \
18    .withColumnRenamed('Package', 'Package') \
19    .withColumnRenamed('Quantity', 'Quantity') \
20    .withColumnRenamed('Unit Price', 'UnitPrice') \
21    .withColumnRenamed('Tax Rate', 'TaxRate') \
22    .withColumnRenamed('Total Excluding Tax', 'TotalExcludingTax') \
23    .withColumnRenamed('Tax Amount', 'TaxAmount') \
24    .withColumnRenamed('Profit', 'Profit') \
25    .withColumnRenamed('Total Including Tax', 'TotalIncludingTax') \
26    .withColumnRenamed('Total Dry Items', 'TotalDryItems') \
27    .withColumnRenamed('Total Chiller Items', 'TotalChillerItems') \
28    .withColumnRenamed('Lineage Key', 'LineageKey')

```

Cell 2 contains the command: `import numpy as np`.

3. 각 Cell의 내용을 확인 후 **Ctrl+Enter**를 이용하여 실행 시킵니다.

처음 SparkPool이 기동 될 때 약간 시간이 소요 됩니다.

코드는 Primary Storage 의 **dev/bronze** 안에 **wwi-factsale.csv** 파일을 읽어 년도/분기/일별로 폴더를 만들고 파일을 생성합니다.

파일은 parquet 파일과 csv 파일로 생성 됩니다.

Setup - Export Sales ...

Cell ▾ Run all Undo Publish Attach to SparkPool01 Language PySpark (Python) Preview Features

Ready

```
7 end_date = start_date + relativedelta(months=3) + relativedelta(days=-1)
8
9 print(f'Exporting data for {start_date.year} Q{quarter_number} ({start_date:%Y-%m-%d} : {end_date:%Y-%m-%d}) ...')
10
11 storage_path_csv = f'abfss://wwi@asapprimarystorage0000.dfs.core.windows.net/factsale-csv/{start_date.year}/Q{quarter_number}'
12
13 wwi_sales \
14     .where((wwi_sales['InvoiceDateKey'] >= f'{start_date:%Y-%m-%d}') & (wwi_sales['InvoiceDateKey'] <= f'{end_date:%Y-%m-%d}')) \
15     .write \
16     .partitionBy('InvoiceDateKey') \
17     .mode("overwrite") \
18     .option("quote", "\u0000") \
19     .option("sep", ",") \
20     .csv(storage_path_csv, header=True)
21
22 start_date = end_date + relativedelta(days=1)
```

Command executed in 6mins 34s 536ms by knightdoo on 01-14-2021 15:32:58.091 +09:00

> Job execution Succeeded Spark 2 executors 16 cores

View in monitoring Open Spark UI

```
Exporting data for 2012 Q1 (2012-01-01 : 2012-03-31) ...
Exporting data for 2012 Q2 (2012-04-01 : 2012-06-30) ...
Exporting data for 2012 Q3 (2012-07-01 : 2012-09-30) ...
Exporting data for 2012 Q4 (2012-10-01 : 2012-12-31) ...
Exporting data for 2013 Q1 (2013-01-01 : 2013-03-31) ...
Exporting data for 2013 Q2 (2013-04-01 : 2013-06-30) ...
Exporting data for 2013 Q3 (2013-07-01 : 2013-09-30) ...
Exporting data for 2013 Q4 (2013-10-01 : 2013-12-31) ...
Exporting data for 2014 Q1 (2014-01-01 : 2014-03-31) ...
Exporting data for 2014 Q2 (2014-04-01 : 2014-06-30) ...
Exporting data for 2014 Q3 (2014-07-01 : 2014-09-30) ...
Exporting data for 2014 Q4 (2014-10-01 : 2014-12-31) ...
Exporting data for 2015 Q1 (2015-01-01 : 2015-03-31) ...
Exporting data for 2015 Q2 (2015-04-01 : 2015-06-30) ...
Exporting data for 2015 Q3 (2015-07-01 : 2015-09-30) ...
Exporting data for 2015 Q4 (2015-10-01 : 2015-12-31) ...
Exporting data for 2016 Q1 (2016-01-01 : 2016-03-31) ...
Exporting data for 2016 Q2 (2016-04-01 : 2016-06-30) ...
Exporting data for 2016 Q3 (2016-07-01 : 2016-09-30) ...
Exporting data for 2016 Q4 (2016-10-01 : 2016-12-31) ...
```

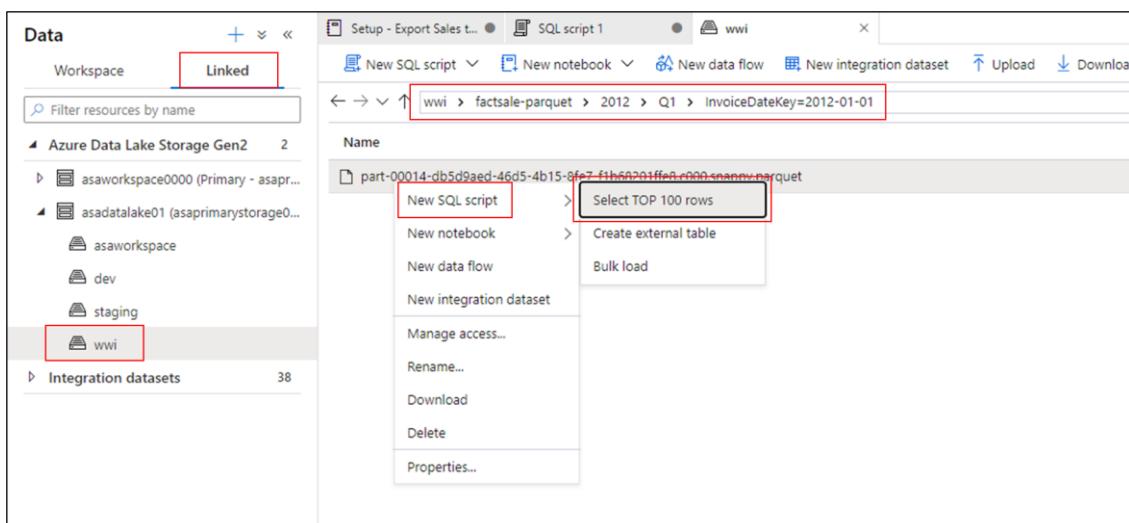
### 3. Exercise 1: Explore the Data Lake with Azure Synapse serverless SQL pool and Azure Synapse Spark

이 연습에서는 SQL, Spark 엔진을 통해 데이터를 탐색합니다. 데이터 탐색을 통한 데이터 이해는 오늘날 데이터 엔지니어와 데이터 과학자가 직면 한 핵심 과제 중 하나입니다.

데이터의 기본 구조와 탐색 프로세스의 특정 요구 사항에 따라 다양한 데이터 처리 엔진이 다양한 수준의 성능, 복잡성 및 유연성을 제공합니다.

#### 3.1. Task 1: Explore the Data Lake with Azure Synapse serverless SQL pool

1. Data 영역으로 이동 하여 Linked를 선택한 다음 Primary Storage의 **wwi/factsale-parquet/2012/Q1/InvoiceDateKey=2012-01-01** 아래 parquet 파일을 마우스 우클릭 하고 **New SQL script** > **Select TOP 100 rows**를 클릭합니다.



2. Run 버튼을 클릭하여 SQL Script를 실행 하면 Query를 통해서 Data Lake Storage에 있는 parquet 파일의 내용을 조회 할 수 있습니다.

The screenshot shows the Azure Synapse Studio SQL script editor. The 'Run' button is highlighted with a red box. Below it, the SQL script is displayed:

```
1 SELECT
2     TOP 100 *
3     FROM
4         OPENROWSET(
5             BULK 'https://asaprimarystorage0000.dfs.core.windows.net/wwi/factsale-parquet/2012/Q1/InvoiceDateKey=2012-01-01/part-00014-db5d9aed-46d5-4b15-8fe7-f1b50201ffea.0000.snappy.parquet'
6             FORMAT='PARQUET'
7         ) AS [result]
```

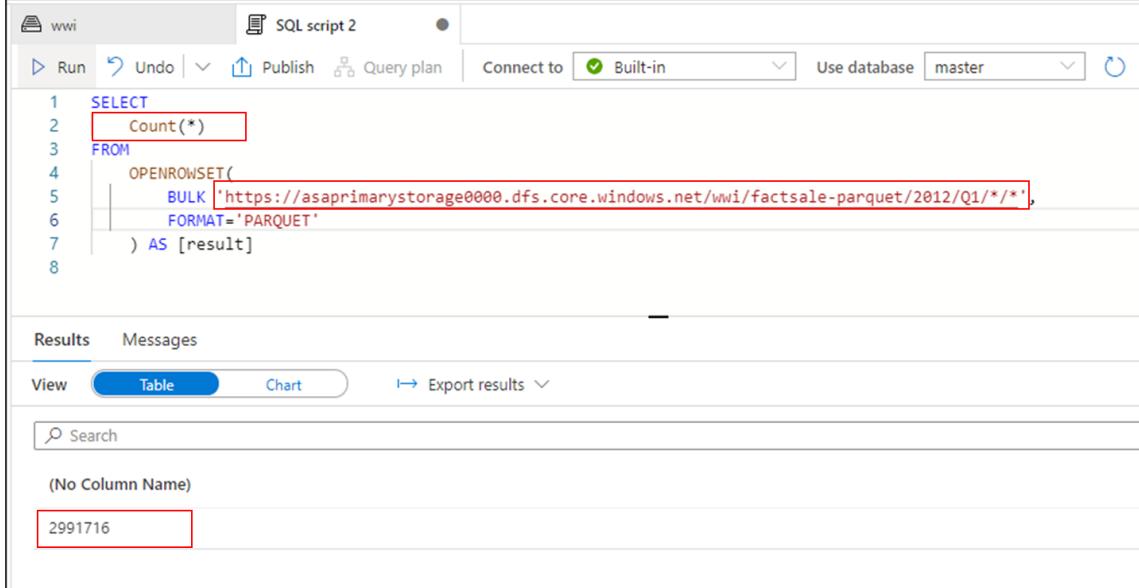
The results pane shows a table with the following data:

| SaleKey | CityKey | CustomerKey | BillToCustomer... | StockItemKey | DeliveryDateKey | SalespersonKey | WW   |
|---------|---------|-------------|-------------------|--------------|-----------------|----------------|------|
| 228266  | 77447   | 0           | 0                 | 194          | 2012-01-02      | 19             | 5961 |
| 228267  | 77447   | 0           | 0                 | 198          | 2012-01-02      | 19             | 5961 |
| 228268  | 75493   | 0           | 0                 | 144          | 2012-01-02      | 143            | 5961 |
| 228269  | 75493   | 0           | 0                 | 19           | 2012-01-02      | 143            | 5961 |
| 228270  | 75493   | 0           | 0                 | 210          | 2012-01-02      | 143            | 5961 |
| 228271  | 92554   | 366         | 202               | 72           | 2012-01-02      | 19             | 5961 |
| 228272  | 92554   | 366         | 202               | 80           | 2012-01-02      | 19             | 5961 |
| 228273  | 49036   | 332         | 202               | 38           | 2012-01-02      | 154            | 5961 |
| 228274  | 10036   | 332         | 202               | 132          | 2012-01-02      | 154            | 5961 |

At the bottom, a message indicates the query was executed successfully.

3. Query 내용을 아래 내용으로 수정하고 실행합니다. Q1 폴더안의 모든 파일들의 Row 건수가 출력 됩니다.

- ✓ 2 Line: **Top 100\*** 을 **Count(\*)** 로 수정
- ✓ 5 Line: .../2012/Q1/InvoiceDateKey=2012-01-01/part.... 내용을 .../2012/Q1/\*/\*로 수정



```

wwi SQL script 2
Run Undo Publish Query plan Connect to Built-in Use database master
1 SELECT
2 Count(*)
3 FROM
4 OPENROWSET(
5 BULK 'https://asaprimarystorage0000.dfs.core.windows.net/wwi/factsale-parquet/2012/Q1/*/*',
6 FORMAT='PARQUET'
7 ) AS [result]
8

```

Results Messages

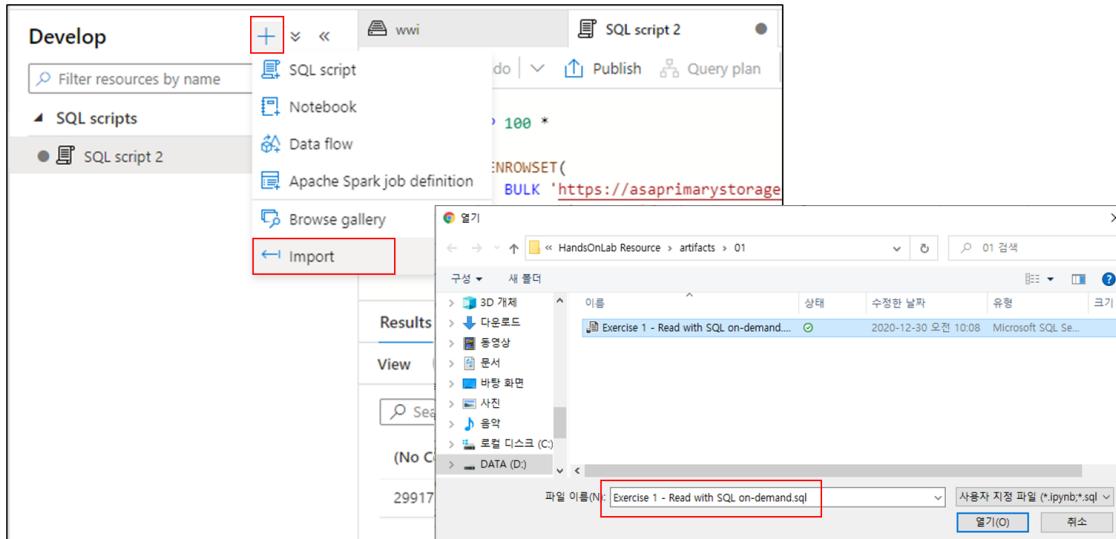
View Table Chart Export results

Search

(No Column Name)

2991716

4. Develop 영역으로 이동하여 + → Import를 클릭하고 **Exercise 1 - Read with SQL on-demand.sql** 파일을 선택하여 Import 합니다.



5. <primary\_storage>를 asaprimarystorageXXXX (Primary Storage 이름)으로 수정하고 Query를 실행합니다. 위의 parquet파일들의 Row 수 조회와 동일한 기능이지만 CSV 파일을 읽어 Row 수를 Count 합니다.

```
wwi SQL script 2 Exercise 1 - Read with... Run Undo Publish Query plan Connect to Built-in Use database SQLOnDemand01
```

```
1 SELECT
2     COUNT(*)
3 FROM
4     OPENROWSET(
5         BULK 'https://asaprimarystorage0000.dfs.core.windows.net/wwi/factsale-csv/2012/Q1/*/*',
6         FORMAT = 'CSV',
7         FIELDTERMINATOR = '|',
8         FIELDQUOTE = '',
9         FIRSTROW = 2
10    )
11 WITH
12     [SalesKey] BIGINT,
13     CityKey INT,
14     CustomerKey INT,
15     BillToCustomerKey INT,
16     StockItemKey INT,
17     DeliveryDateKey DATE
```

Results Messages

View Table Chart Export results

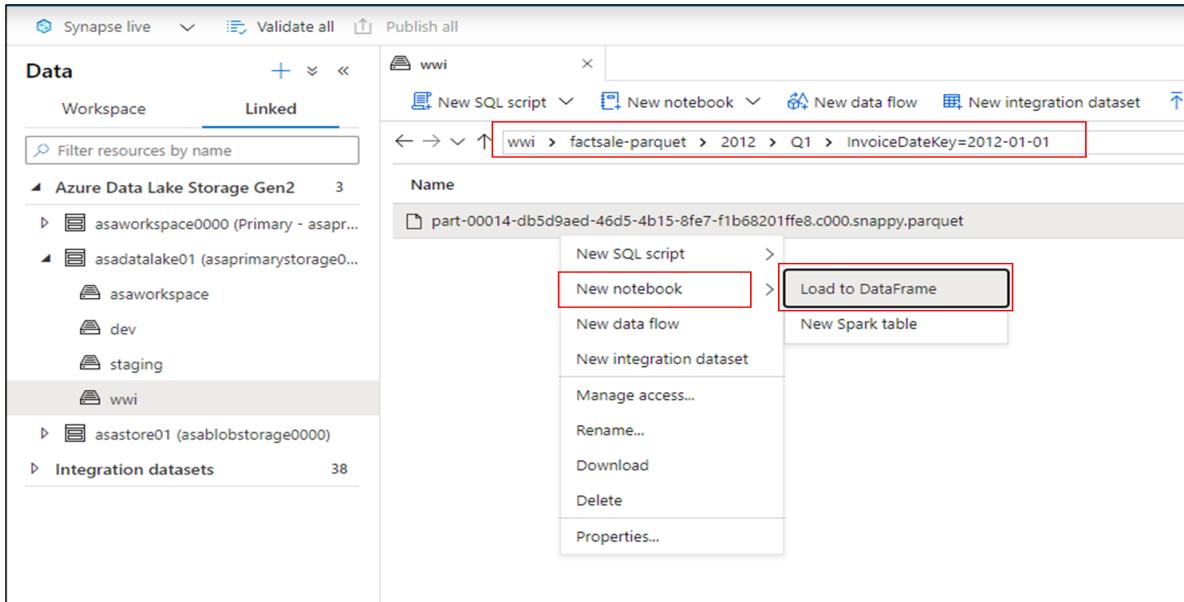
Search

(No Column Name)

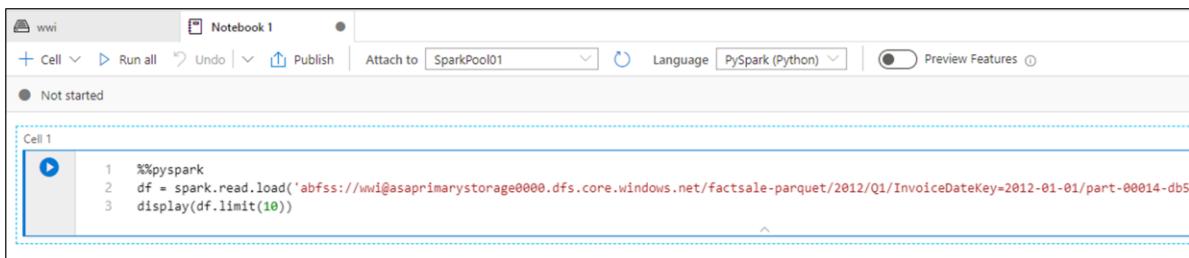
|         |
|---------|
| 2991716 |
|---------|

### 3.2. Task 2: Explore the Data Lake with Azure Synapse Spark

1. Data 영역으로 이동하여 Linked를 선택한 다음 Primary Storage의 **wwi/factsale-parquet/2012/Q1/InvoiceDateKey=2012-01-01** 아래 parquet 파일을 마우스 우클릭하고 **New notebook>Load to DataFrame**을 선택합니다..



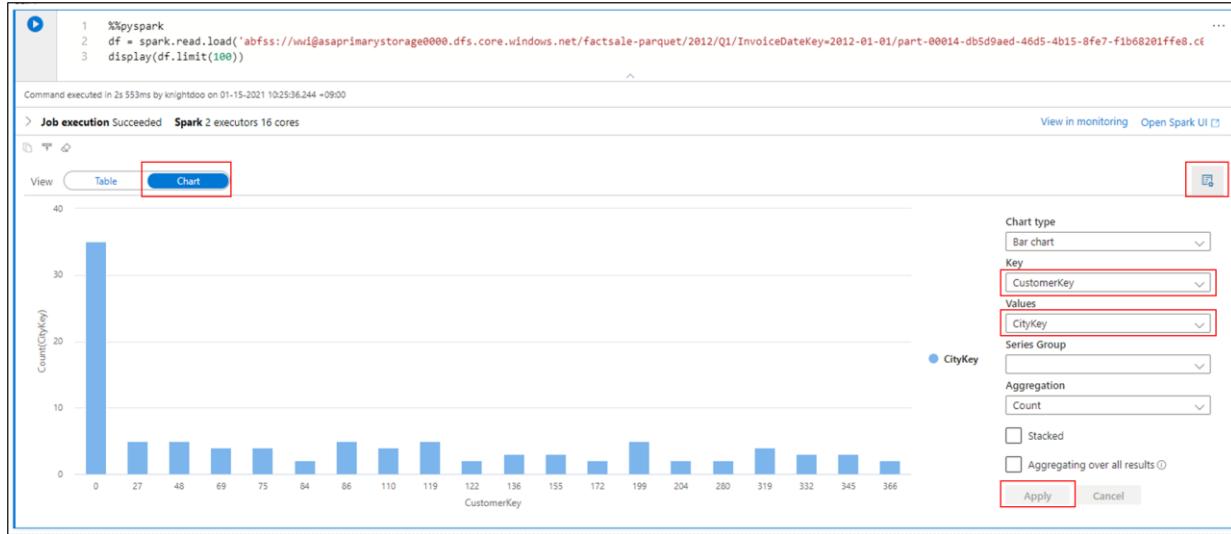
2. Parquet 파일을 읽어 10 Row를 출력하는 PySpark 코드가 자동으로 생성됩니다.



3. `display(df.limit(10))` 구문을 `display(df.limit(100))` 으로 수정하고 상단의 **Run all** 버튼을 클릭하여 코드를 실행 합니다. 100건의 결과가 Table 형태로 출력 되었습니다.

| SaleKey | CityKey | CustomerKey | BillToCustomerK... | StockItemKey | DeliveryDateKey | SalespersonKey | WWIInvoiceID | Description          | Package | Quantity | UnitPric |
|---------|---------|-------------|--------------------|--------------|-----------------|----------------|--------------|----------------------|---------|----------|----------|
| 228266  | 77447   | 0           | 0                  | 194          | 2012-01-02      | 19             | 59611        | DBA joke mug - ...   | Each    | 8        | 13.00    |
| 228267  | 77447   | 0           | 0                  | 198          | 2012-01-02      | 19             | 59611        | DBA joke mug - ...   | Each    | 8        | 13.00    |
| 228268  | 75493   | 0           | 0                  | 144          | 2012-01-02      | 143            | 59612        | "The Gu" red shir... | Each    | 60       | 18.00    |
| 228269  | 75493   | 0           | 0                  | 19           | 2012-01-02      | 143            | 59612        | Red and white ur...  | Each    | 72       | 3.70     |
| 228270  | 75493   | 0           | 0                  | 210          | 2012-01-02      | 143            | 59612        | USB food flash dr... | Each    | 6        | 32.00    |
| 228271  | 92554   | 366         | 202                | 72           | 2012-01-02      | 19             | 59613        | Halloween skull ...  | Each    | 48       | 18.00    |
| 228272  | 92554   | 366         | 202                | 80           | 2012-01-02      | 19             | 59613        | Furry animal sock... | Pair    | 84       | 5.00     |
| 228273  | 49036   | 332         | 202                | 38           | 2012-01-02      | 154            | 59614        | Shipping carton (... | Each    | 200      | 1.28     |
| 228274  | 49036   | 332         | 202                | 132          | 2012-01-02      | 154            | 59614        | "The Gu" red shir... | Each    | 12       | 18.00    |
| 228275  | 49036   | 332         | 202                | 171          | 2012-01-02      | 154            | 59614        | Developer joke ...   | Each    | 3        | 13.00    |
| 228276  | 77605   | 0           | 0                  | 188          | 2012-01-02      | 155            | 59615        | Developer joke ...   | Each    | 1        | 12.00    |

4. 데이터를 시각적으로 탐색할 수 있는 Chart 기능이 포함되어 있습니다. **Chart** 버튼을 클릭하여 Chart로 변환하고 오른쪽 **보기옵션**을 선택하여 Key를 Customerkey로 변경, Value를 Citykey로 변경 합니다. Apply 버튼을 클릭하여 변경사항을 적용 합니다.



5. {} Add code버튼을 클릭하여 아래에 Cell을 추가하고 아래 코드를 복사하여 붙여 넣고 **asaprimarystorageXXXX** 를 primary storage 이름으로 수정합니다.

```
data_path = spark.read.load(
    'abfss://wwi@asaprimarystorageXXXX.dfs.core.windows.net/factsale-csv/2012/Q1/*/*',
    format='csv',
    sep="|",
    header=True)

display(data_path.limit(100))
```



6. Cell 왼쪽의 **Run Cell** 버튼을 클릭하여 코드를 수행 합니다. 이는 parquet파일이 아닌 CSV 파일을 읽어 결과를 출력 하는 코드 입니다.

| SaleKey | CityKey | CustomerKey | BillToCustomerK... | StockItemKey | DeliveryDateKey | SalespersonKey | WWInvoiceID | Description           | Package | Quantity | UnitPrice |
|---------|---------|-------------|--------------------|--------------|-----------------|----------------|-------------|-----------------------|---------|----------|-----------|
| 1444678 | 81238   | 17          | 1                  | 159          | 2012-02-08      | 155            | 54186       | RC toy sedan car ...  | Each    | 8        | 25.00     |
| 1444679 | 92521   | 0           | 0                  | 179          | 2012-02-08      | 150            | 54187       | Developer joke ...    | Each    | 4        | 13.00     |
| 1444680 | 92521   | 0           | 0                  | 56           | 2012-02-08      | 150            | 54187       | 32 mm Double si...    | Each    | 70       | 112.00    |
| 1444681 | 92521   | 0           | 0                  | 61           | 2012-02-08      | 150            | 54187       | 20 mm Double si...    | Each    | 30       | 18.00     |
| 1444682 | 92521   | 0           | 0                  | 205          | 2012-02-08      | 150            | 54187       | USB food flash dr...  | Pocket  | 3        | 240.00    |
| 1444683 | 109262  | 0           | 0                  | 72           | 2012-02-08      | 154            | 54188       | Halloween skull ...   | Each    | 48       | 18.00     |
| 1444684 | 109262  | 0           | 0                  | 120          | 2012-02-08      | 154            | 54188       | "The Gru" red shir... | Each    | 60       | 18.00     |
| 1444685 | 72808   | 19          | 1                  | 65           | 2012-02-08      | 19             | 54189       | Large sized bubb...   | Each    | 100      | 24.00     |
| 1444686 | 72808   | 19          | 1                  | 216          | 2012-02-08      | 19             | 54189       | USB food flash dr...  | Each    | 4        | 32.00     |
| 1444687 | 72808   | 19          | 1                  | 46           | 2012-02-08      | 19             | 54189       | Bubblewrap disp...    | Each    | 9        | 240.00    |
| 1444688 | 72808   | 19          | 1                  | 96           | 2012-02-08      | 19             | 54189       | Ogre battery-po...    | Each    | 10       | 32.00     |

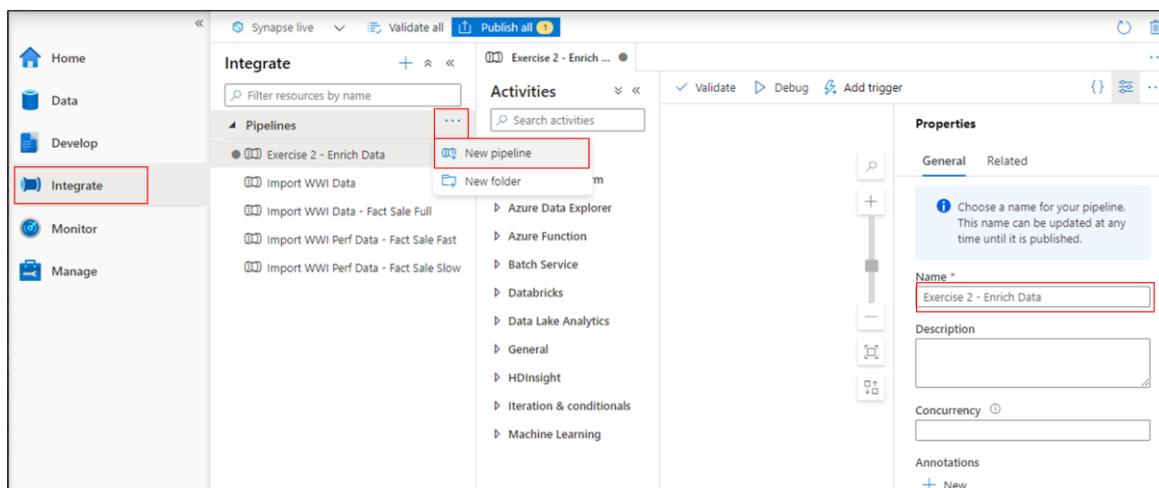
#### 4. Exercise 2: Build a Modern Data Warehouse with Azure Synapse Pipelines

이 연습에서는 파이프 라인을 사용하여 데이터를 Data Lake로 가져 와서 변환 한 다음 Azure Synapse SQL Pool에 로드합니다. 또한 관련 작업의 진행 상황을 모니터링합니다.

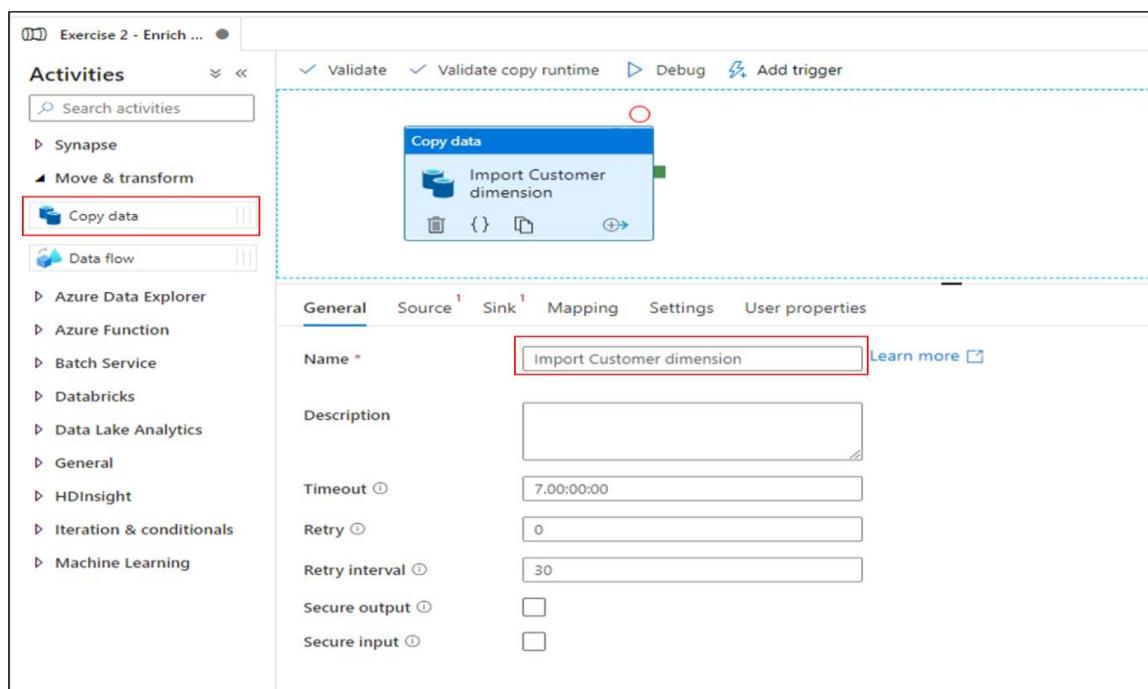
데이터가 제대로 이해되고 해석되면 처리 단계가 발생하는 다양한 대상으로 데이터를 이동하는 것이 다음 큰 작업입니다. 모든 최신 데이터 플랫폼은 추출, 구문 분석, 결합, 표준화, 증강, 정리, 통합 및 필터링과 같은 모든 일반적인 데이터 랭글링 작업에 대한 원활한 경험을 제공해야합니다..

##### 4.1. Task 1: Explore, modify, and run a Pipeline containing a Data Flow

1. 왼쪽 메뉴에서 **Integrate**를 선택하신 후 Pipelines 오른쪽에 ... 버튼(Action)을 클릭하고 **New pipeline**을 선택하여 Pipeline을 추가합니다. 추가된 Pipeline의 Name에 **Exercise 2 - Enrich Data**를 입력 합니다.



2. **Copy data Activity**를 Drag&Drop으로 Canvas 가져다 놓고 **Name**을 **Import Customer dimension**으로 변경 합니다..



3. Source Tab으로 이동 하여 Source dataset에서 **wwi\_dimcustomer\_adls**를 선택합니다.

The screenshot shows the 'Copy data' pipeline configuration. The 'Source' tab is selected. The 'Source dataset' dropdown is set to 'wwi\_dimcustomer\_adls'. Other settings include 'File path type' (File path in dataset), 'Start time (UTC)', 'End time (UTC)', 'Recursively' checked, and 'Additional columns'.

4. Sink Tab으로 이동 하여 아래 내용을 설정 합니다.

- ✓ Sink dataset: **wwi\_staging\_dimcustomer\_asa** 선택
- ✓ Uniqueid: **@substring(pipeline().RunId,0,8)** 입력
- ✓ Copy method: **PolyBase** 선택
- ✓ Table option: **Auto create table** 선택

The screenshot shows the 'Copy data' pipeline configuration. The 'Sink' tab is selected. The 'Sink dataset' dropdown is set to 'wwi\_staging\_dimcustomer\_asa'. The 'Dataset properties' table shows a row for 'Uniqeid' with value '@substring(pipeline().RunId,0,8)'. The 'Copy method' section has 'PolyBase' selected. The 'Table option' section has 'Auto create table' selected.

5. Mapping Tab으로 이동 하여 Import Schemas 버튼을 클릭하여 Mapping 정보를 생성합니다.

| Source           | Type     | Destination      | Type   |
|------------------|----------|------------------|--------|
| Customer Key     | Int32    | Customer Key     | String |
| WWI Customer ID  | Int32    | WWI Customer ID  | String |
| Customer         | String   | Customer         | String |
| Bill To Customer | String   | Bill To Customer | String |
| Category         | String   | Category         | String |
| Buying Group     | String   | Buying Group     | String |
| Primary Contact  | String   | Primary Contact  | String |
| Postal Code      | String   | Postal Code      | String |
| Valid From       | DateTime | Valid From       | String |
| Valid To         | DateTime | Valid To         | String |
| Lineage Key      | Int32    | Lineage Key      | String |

6. Sink dataset Table은 현재 존재하지 않는 Table이므로 컬럼명이 Source dataset 파일과 동일하게 성 됩니다. Code 편집을 통해 컬럼명과 Type을 변경하도록 하겠습니다.  
코드편집기 창을 열고 mapping 값의 내용을 아래 내용으로 변경 한 후 OK 버튼을 클릭하여 변경 내용을 적용합니다..

```
{
  "source": {
    "name": "Customer Key",
    "type": "Int32"
  },
  "sink": {
    "name": "CustomerKey",
    "type": "Int32"
  }
},
{
  "source": {
    "name": "WWI Customer ID",
    "type": "Int32"
  },
  "sink": {
    "name": "WWICustomerID",
    "type": "Int32"
  }
},
```

```
"source": {
    "name": "Customer",
    "type": "String"
},
"sink": {
    "name": "Customer",
    "type": "String"
}
},
{
    "source": {
        "name": "Bill To Customer",
        "type": "String"
    },
    "sink": {
        "name": "BillToCustomer",
        "type": "String"
    }
},
{
    "source": {
        "name": "Category",
        "type": "String"
    },
    "sink": {
        "name": "Category",
        "type": "String"
    }
},
{
    "source": {
        "name": "Buying Group",
        "type": "String"
    },
    "sink": {
        "name": "BuyingGroup",
        "type": "String"
    }
},
{
    "source": {
        "name": "Primary Contact",
        "type": "String"
    },
    "sink": {
        "name": "PrimaryContact",
        "type": "String"
    }
},
{
    "source": {
        "name": "Postal Code",
        "type": "String"
    },
    "sink": {
        "name": "PostalCode",
        "type": "String"
    }
}
```

```

        }
    },
    {
        "source": {
            "name": "Valid From",
            "type": "DateTime"
        },
        "sink": {
            "name": "ValidFrom",
            "type": "DateTime"
        }
    },
    {
        "source": {
            "name": "Valid To",
            "type": "DateTime"
        },
        "sink": {
            "name": "ValidTo",
            "type": "DateTime"
        }
    },
    {
        "source": {
            "name": "Lineage Key",
            "type": "Int32"
        },
        "sink": {
            "name": "LineageKey",
            "type": "Int32"
        }
    }
}

```

Pipeline name: Exercise 2 - Enrich Data

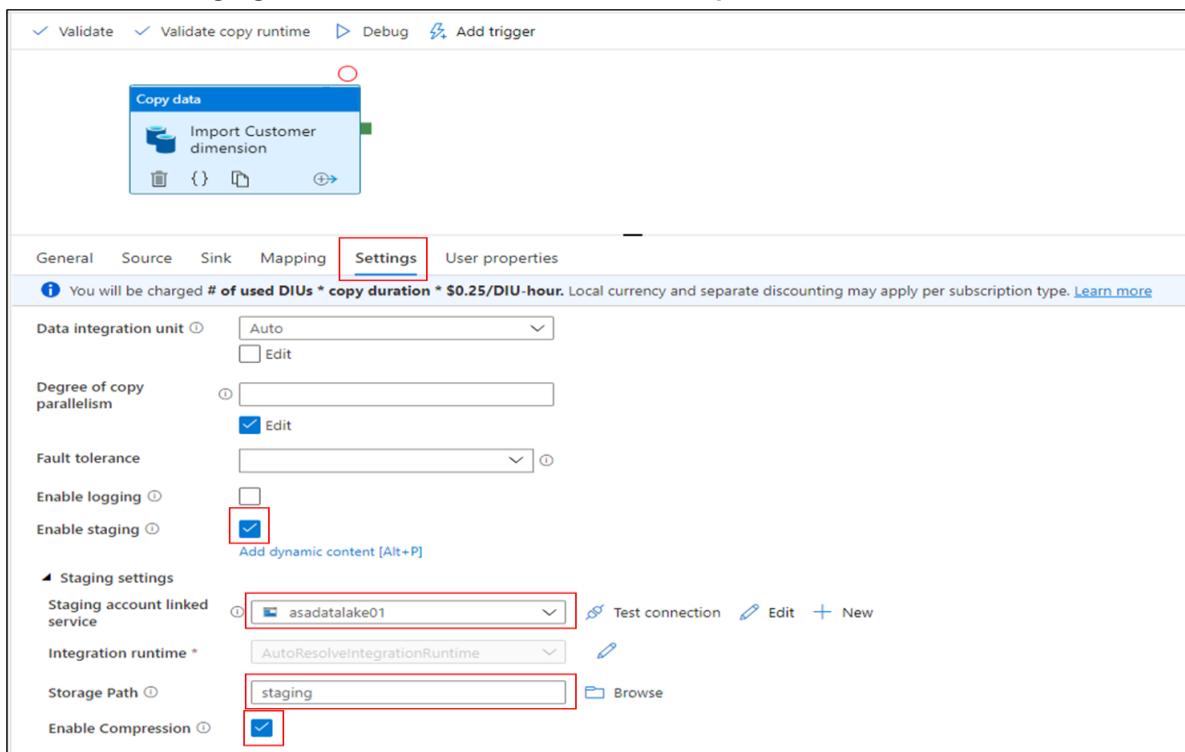
Copy to clipboard

```

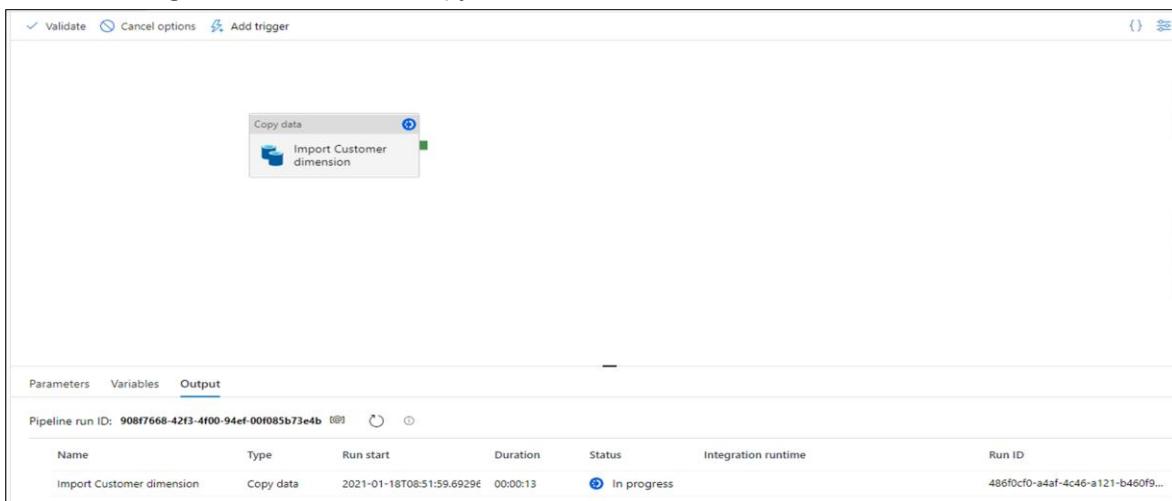
48     "translator": {
49         "type": "TabularTranslator",
50         "mappings": [
51             {
52                 "source": {
53                     "name": "Customer Key",
54                     "type": "Int32"
55                 },
56                 "sink": {
57                     "name": "CustomerKey",
58                     "type": "Int32"
59                 }
60             },
61             {
62                 "source": {
63                     "name": "WWI Customer ID",
64                     "type": "Int32"
65                 },
66                 "sink": {
67                     "name": "WWICustomerID",
68                     "type": "Int32"
69                 }
70             },
71             {
72                 "source": {
73                     "name": "Customer",
74                     "type": "String"
75                 },
76                 "sink": {
77                     "name": "Customer",
78                     "type": "String"
79                 }
80             },
81             {
82                 "source": {
83                     "name": "Bill To Customer",
84                     "type": "String"
85                 }
86             }
87         ]
88     }
89 
```

OK Cancel

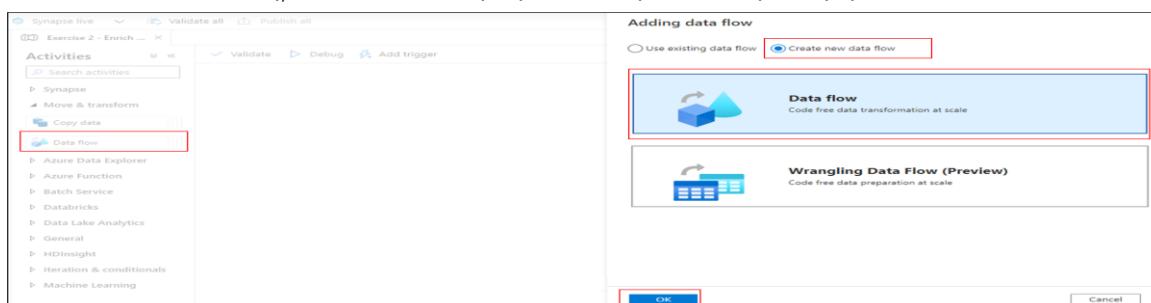
7. Settings Tab으로 이동하여 **Enable staging** 항목을 Check하고 **Staging account linked service** 에서 **asadatalake01**을 선택 합니다. **Storage Path** 항목에 **staging**을 입력 하거나 옆에 **Browse** 버튼을 클릭하여 **staging** 디렉토리를 선택 하고, **Enable Compression**항목을 **Check** 합니다.



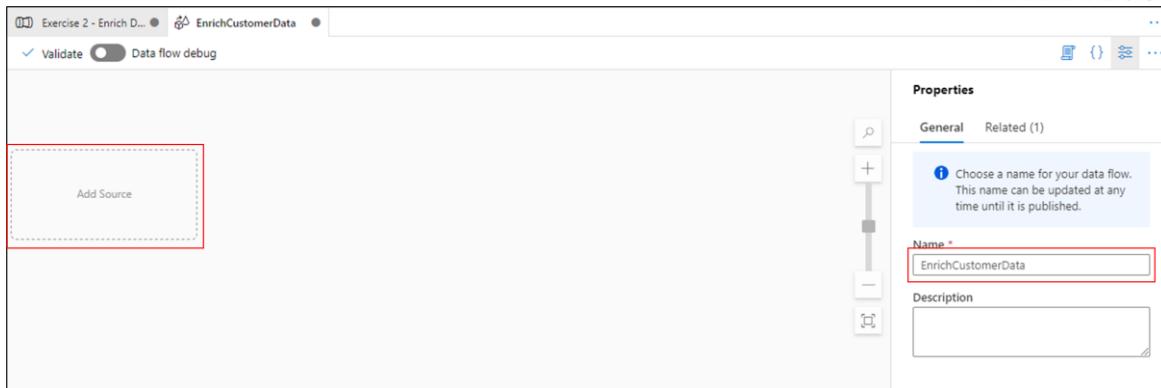
8. 상단의 **Debug** 버튼을 클릭하여 Copy data를 수행 합니다.



9. **Data flow Activity**를 Drag&Drop으로 Canvas 가져다 놓고 Adding data flow 블레이드에서 **Create new data flow**를 선택, **Data flow** 선택하고 **OK** 버튼을 클릭 합니다..

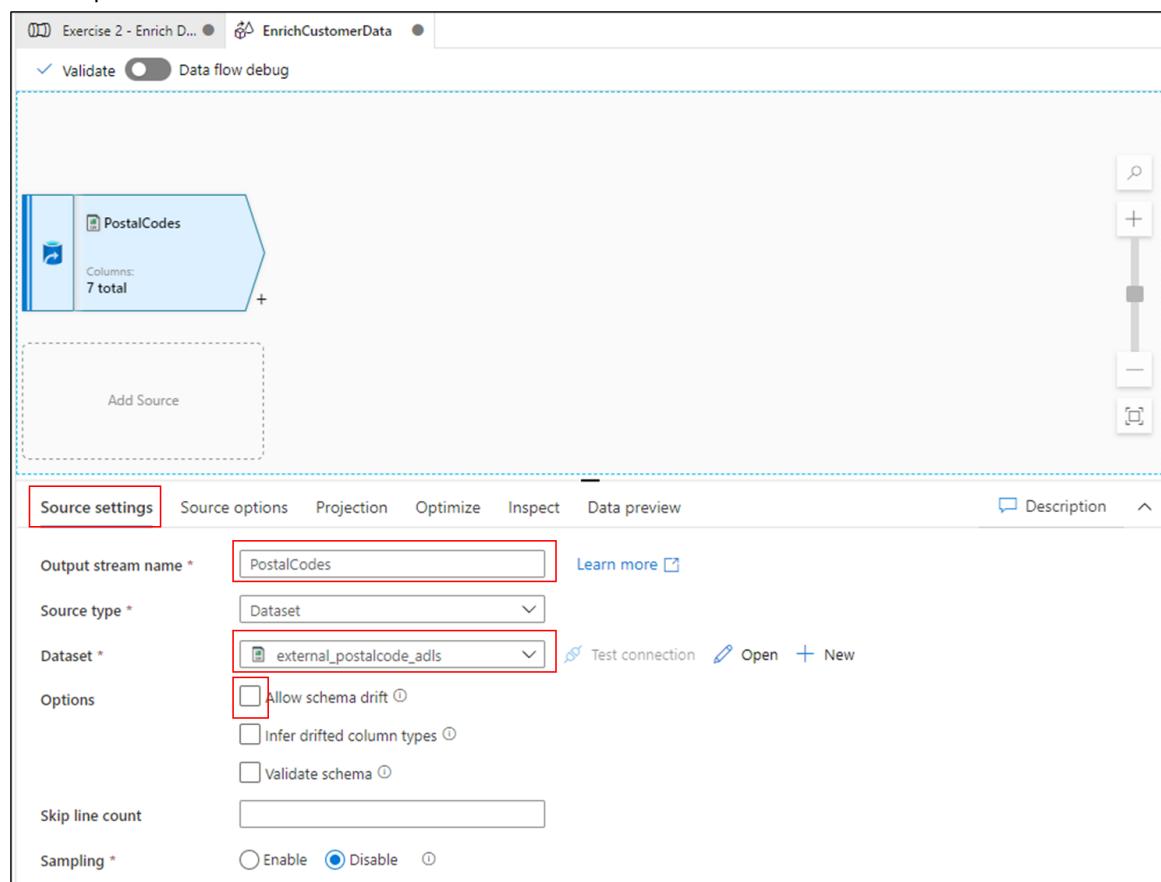


10. 새로 만들어진 Data flow의 Properties 블레이드에서 Name에 **EnrichCustomerData**를 입력하고 왼쪽 Canvas의 **Add Source**를 클릭합니다.

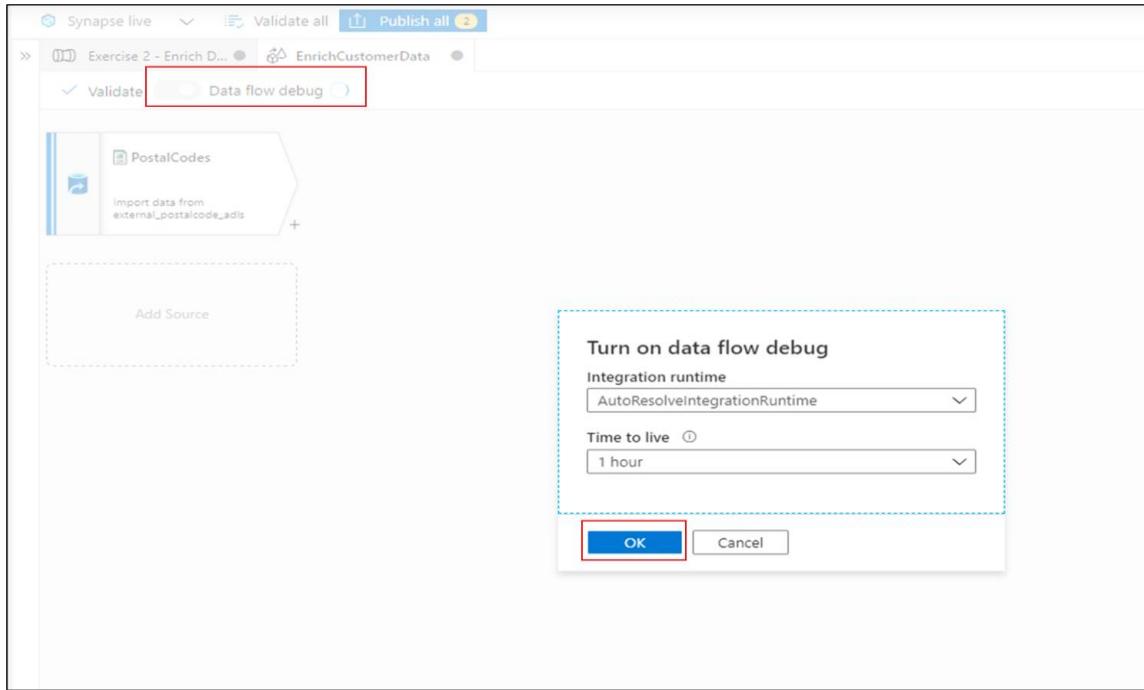


11. **Source settings** Tab을 선택하고 아래 내용으로 입력합니다.

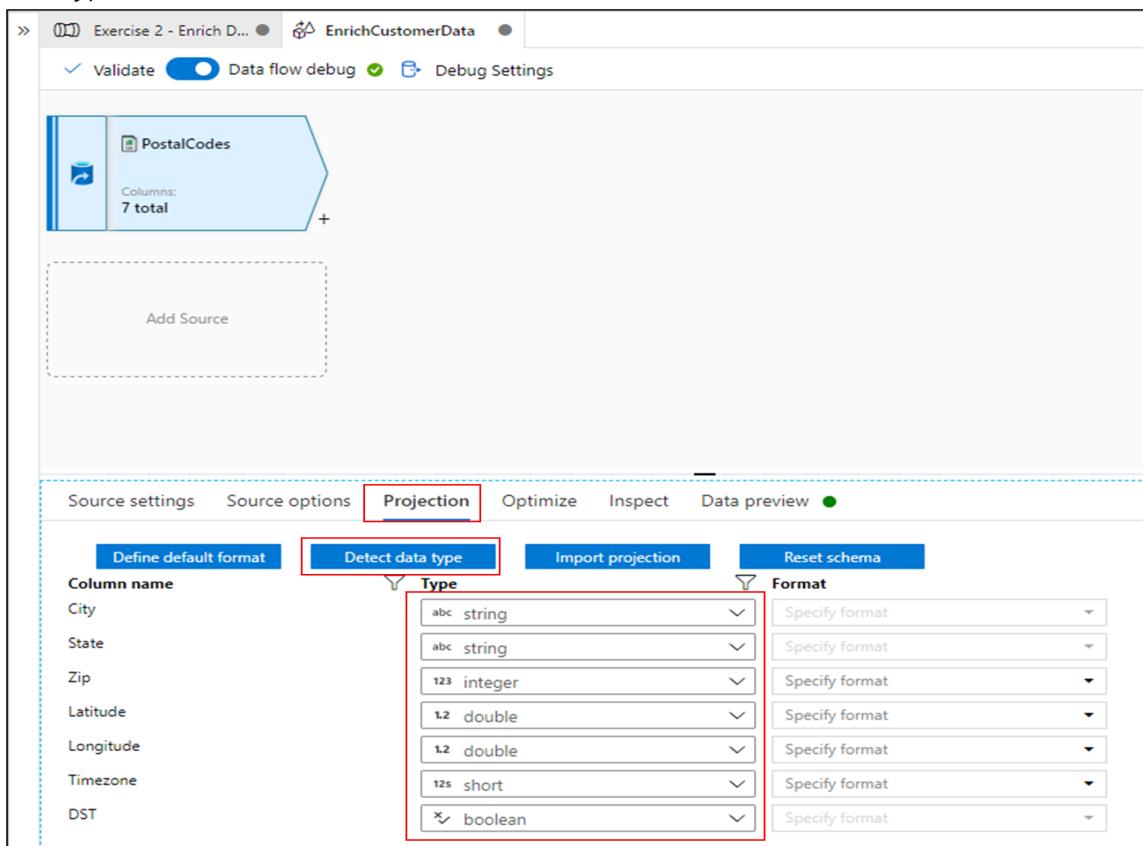
- ✓ Output stream name: **PostalCodes** 입력
- ✓ Dataset: **external\_postalcode\_adls** 선택
- ✓ Options: **Allow schema drift** 항목 UnCheck



12. 위 상단의 **Data flow debugging**을 클릭하고 Turn on data flow debug 창에서 **OK** 버튼을 클릭하여 Debugging을 활성화 합니다. Debugging이 활성화 되어있어야 Data Preview가 가능합니다.



13. **Projection Tab**을 선택하고 **Detect data type** 버튼을 클릭하여 Source 데이터를 기준으로 컬럼 Type을 설정합니다.



14. Data preview Tab으로 이동하여 Refresh 버튼을 클릭하여 Source Data를 조회합니다.

The screenshot shows the Data preview tab of a data source configuration interface. At the top, there are tabs for Source settings, Source options, Projection, Optimize, Inspect, and Data preview (which is selected). Below the tabs, there are counters for INSERT (100), UPDATE (0), DELETE (0), UPSERT (0), LOOKUP (0), and TOTAL (1000). A red box highlights the 'Refresh' button. The main area displays a table of data with columns: City, State, Zip, Latitude, and Longitude. The 'Zip' column is highlighted with a red box. The data includes rows for various cities like Cove, Edgemont, Sherburn, Lamont, Richland, Cannelton, Zeeland, and Covington, along with their corresponding state abbreviations and coordinates.

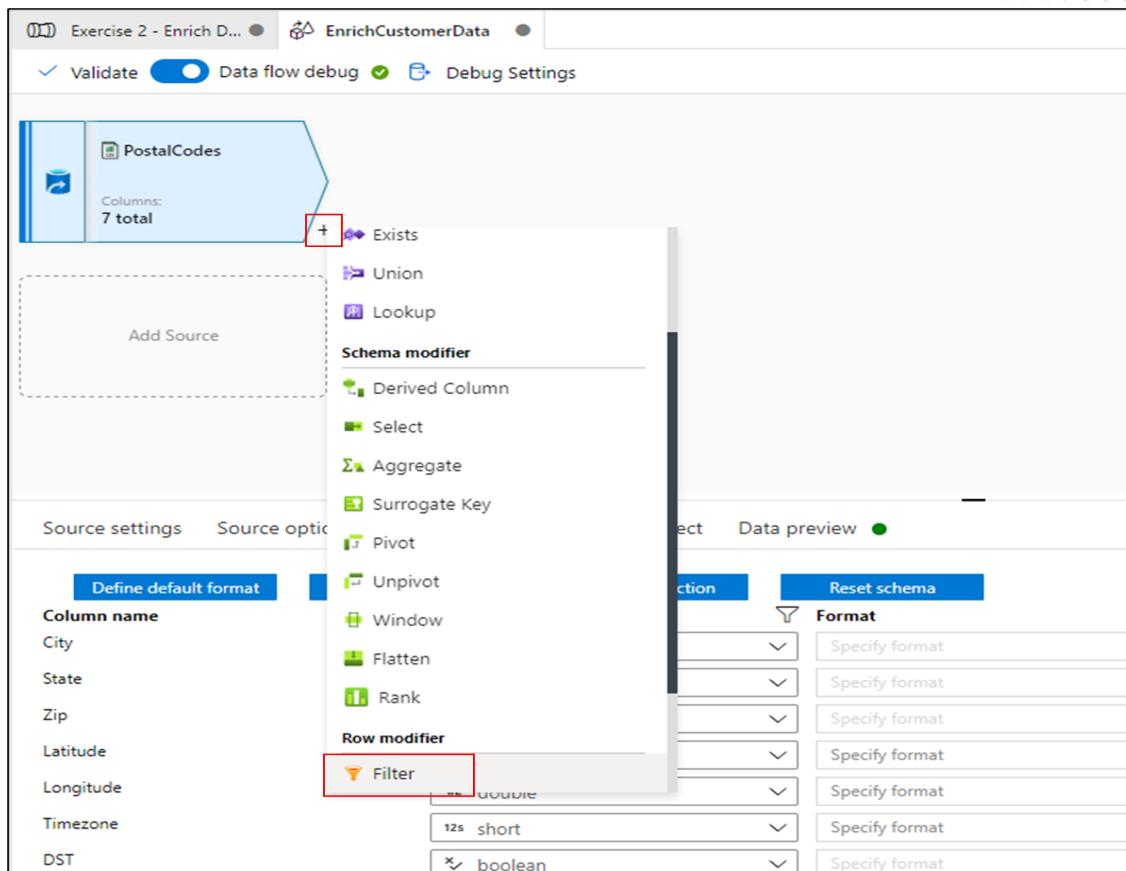
15. Projection Tab으로 이동하여 Zip 컬럼의 Type을 string으로 변경 합니다

Zip 컬럼의 데이터를 숫자형으로 처리 시 앞에 0은 제거 되므로 제거되지 않도록 String으로 변경 하였습니다.

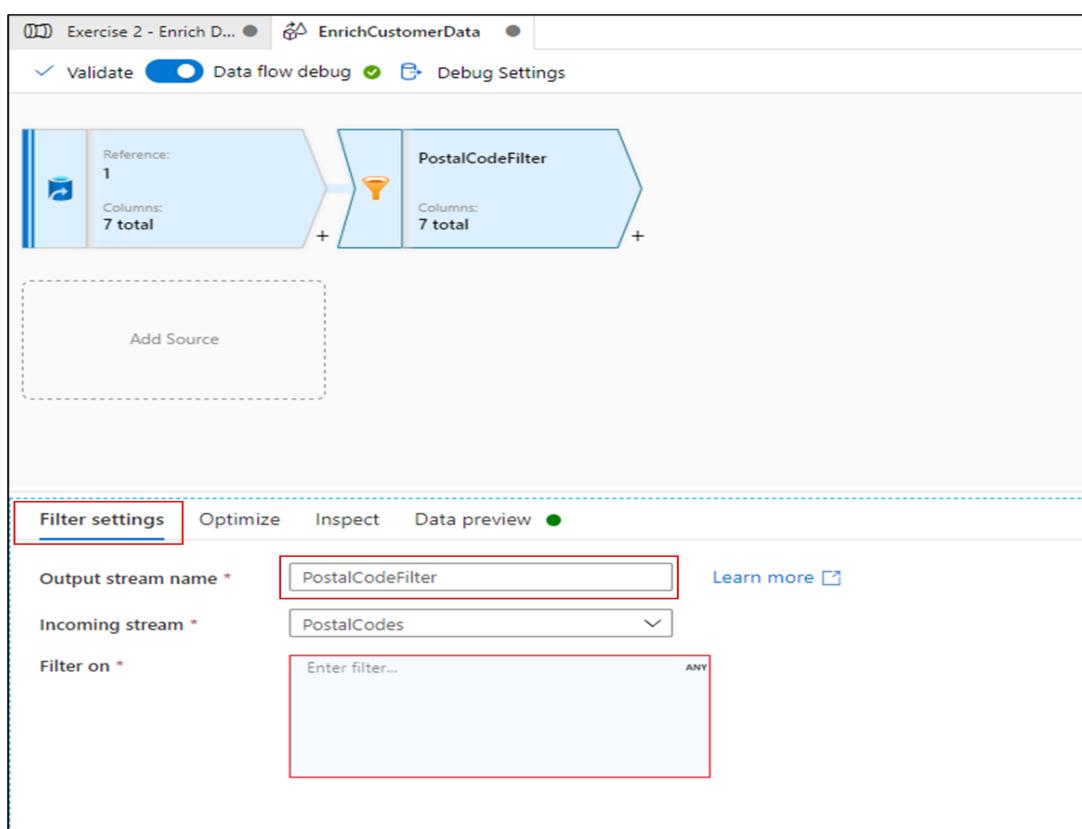
The screenshot shows the Projection tab of a data source configuration interface. At the top, there are tabs for Source settings, Source options, Projection (which is selected), Optimize, Inspect, and Data preview. Below the tabs are buttons for Define default format, Detect data type, Import projection, and Reset schema. The main area shows a schema mapping for several columns: City, State, Zip, Latitude, Longitude, Timezone, and DST. The 'Zip' column is highlighted with a red box. The 'Type' column for Zip is set to 'string'. To the right of each column, there is a 'Format' section with a 'Specify format' dropdown. The 'Format' dropdown for the Zip column is also highlighted with a red box.

16. PostalCodes Data Source의 오른쪽 +를 클릭하여 Filter를 선택합니다.

Multiple inputs/outputs, Schema modifier, Row modifier 등 다양한 Transform을 위한 조건을 선택 할 수 있습니다.

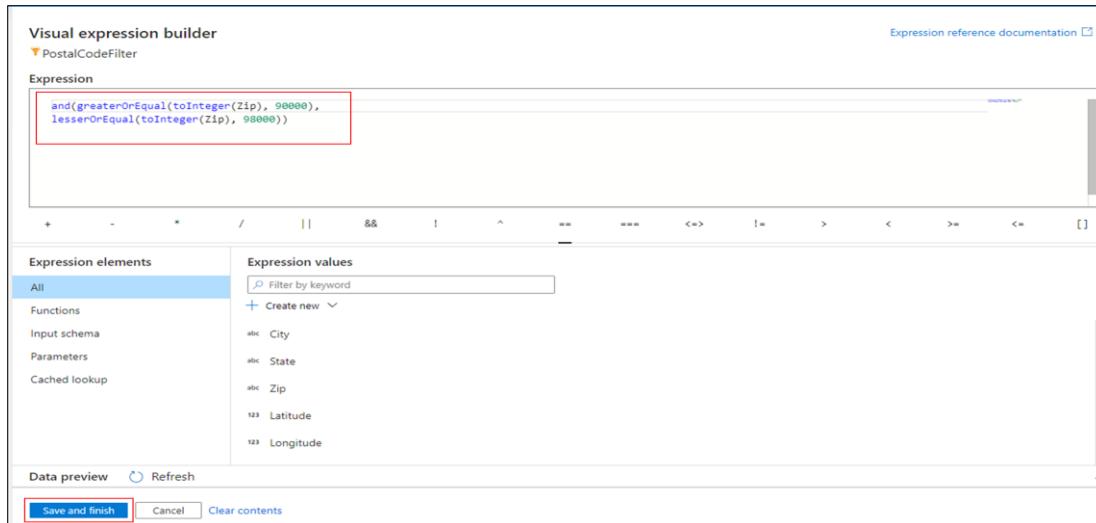


17. Filter settings Tab에서 Output stream name에 PostalCodeFilter를 입력하고 Filter on Box를 클릭하여 Visual expression builder를 호출합니다.

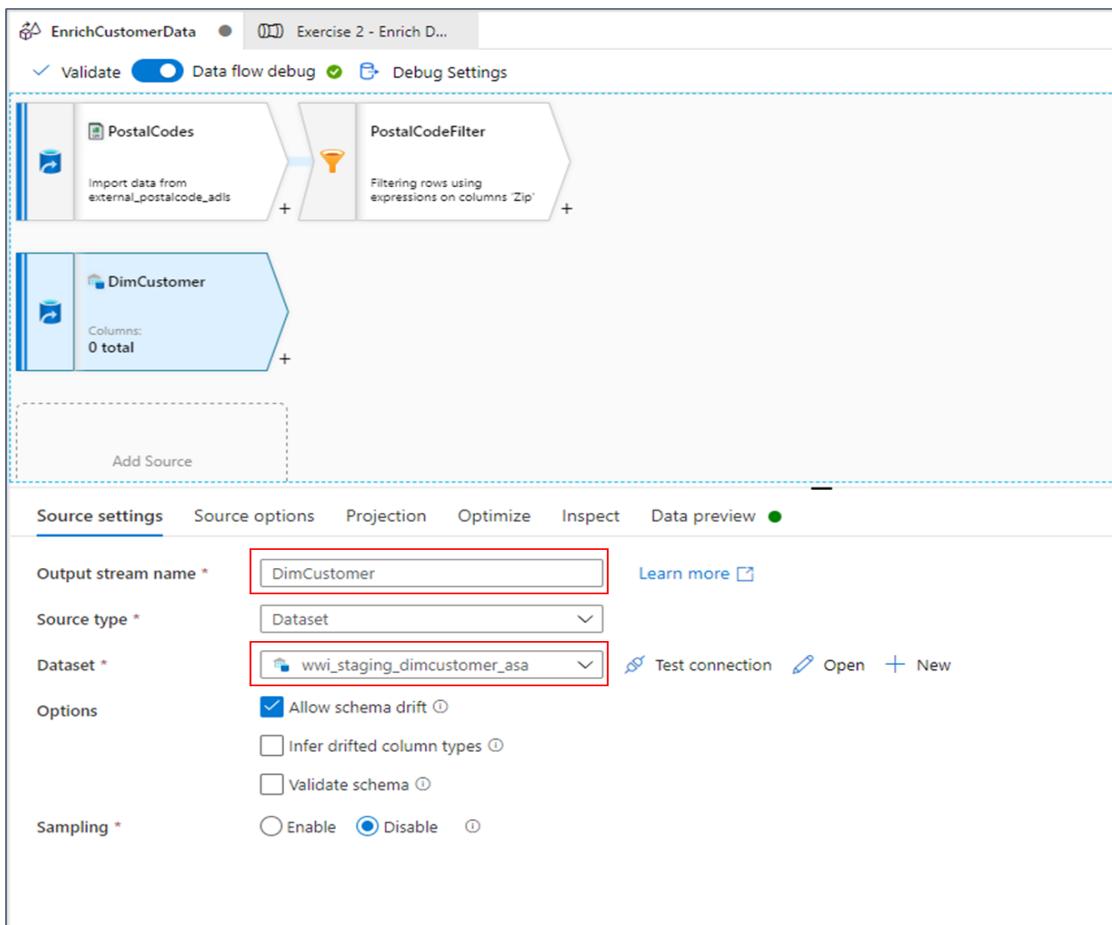


18. Visual expression builder에서 아래 내용을 입력하고 **Save and Finish** 버튼을 클릭합니다.

```
and(greaterOrEqual(toInteger(Zip), 90000),
lesserOrEqual(toInteger(Zip), 98000))
```



19. **Add Source**를 클릭하여 Data Source를 추가하고 **Output Stream name**에 **DimCustomer** 입력, **Dataset**은 **wwi\_staging\_dimcustomer\_asa**를 선택 합니다.



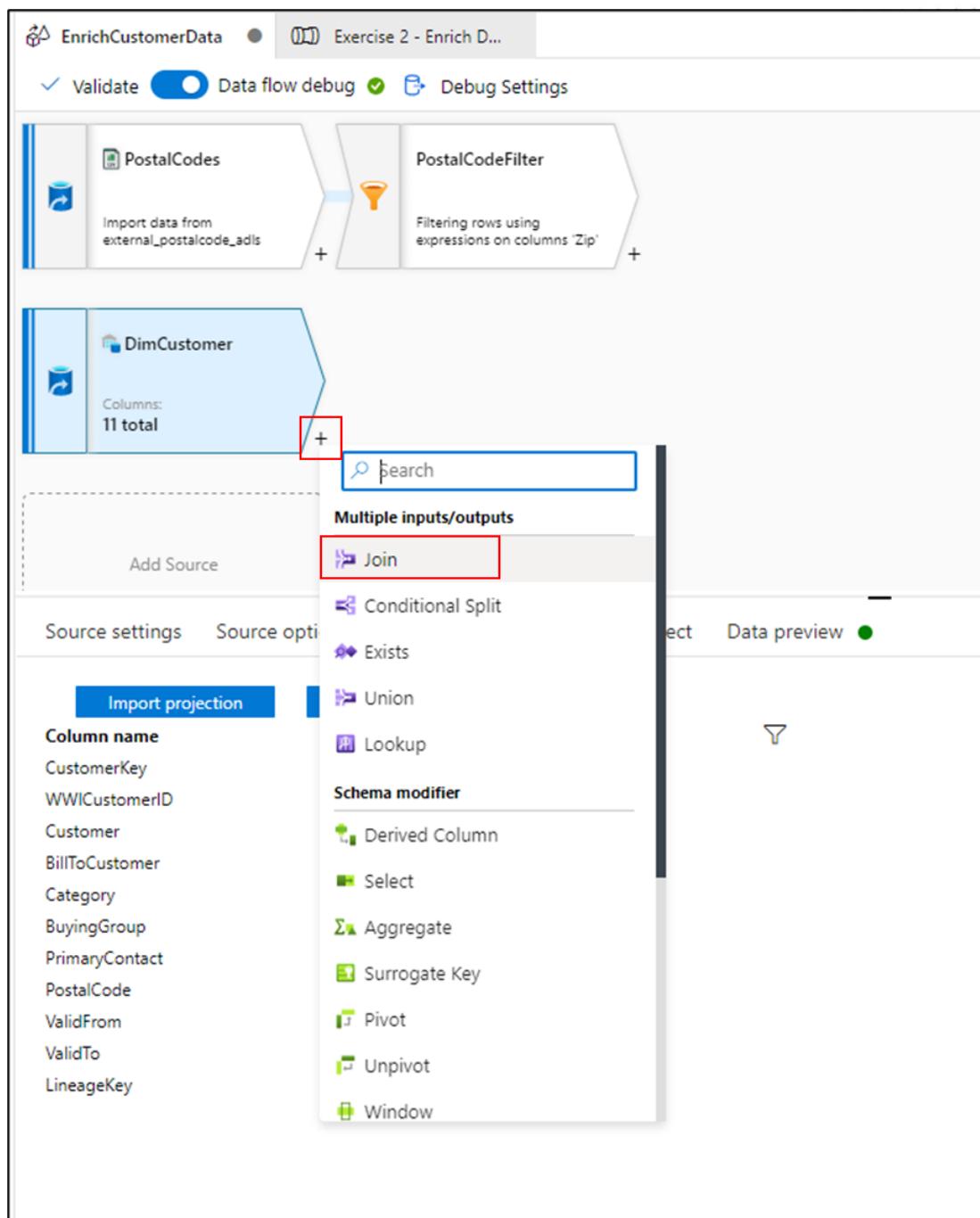
20. Projection Tab으로 이동 한 후 왼쪽 메뉴의 **Data→SQLPool01→Tables**를 확장 하여 **wwi\_staging.DimCustomer\_XXXXXXX** 테이블을 확인합니다. 만약 보이지 않으면 Tables 옆 …를 클릭하고 **Refresh**를 클릭합니다. 테이블명 뒤의 Pipeline RunID 앞8자리인 난수를 확인 하고 **Import projection** 버튼을 클릭하고 **UniqueId Value**에 XXXXXXXXX을 입력 후 **Save** 버튼을 클릭합니다.

The screenshot shows the 'Projection' tab for the 'DimCustomer' table. The 'UniqueId' parameter is highlighted with a red box, showing its value as '4ae82417'. The 'Save' button at the bottom right is also highlighted with a red box.

21. Column 명과 Type을 확인합니다.

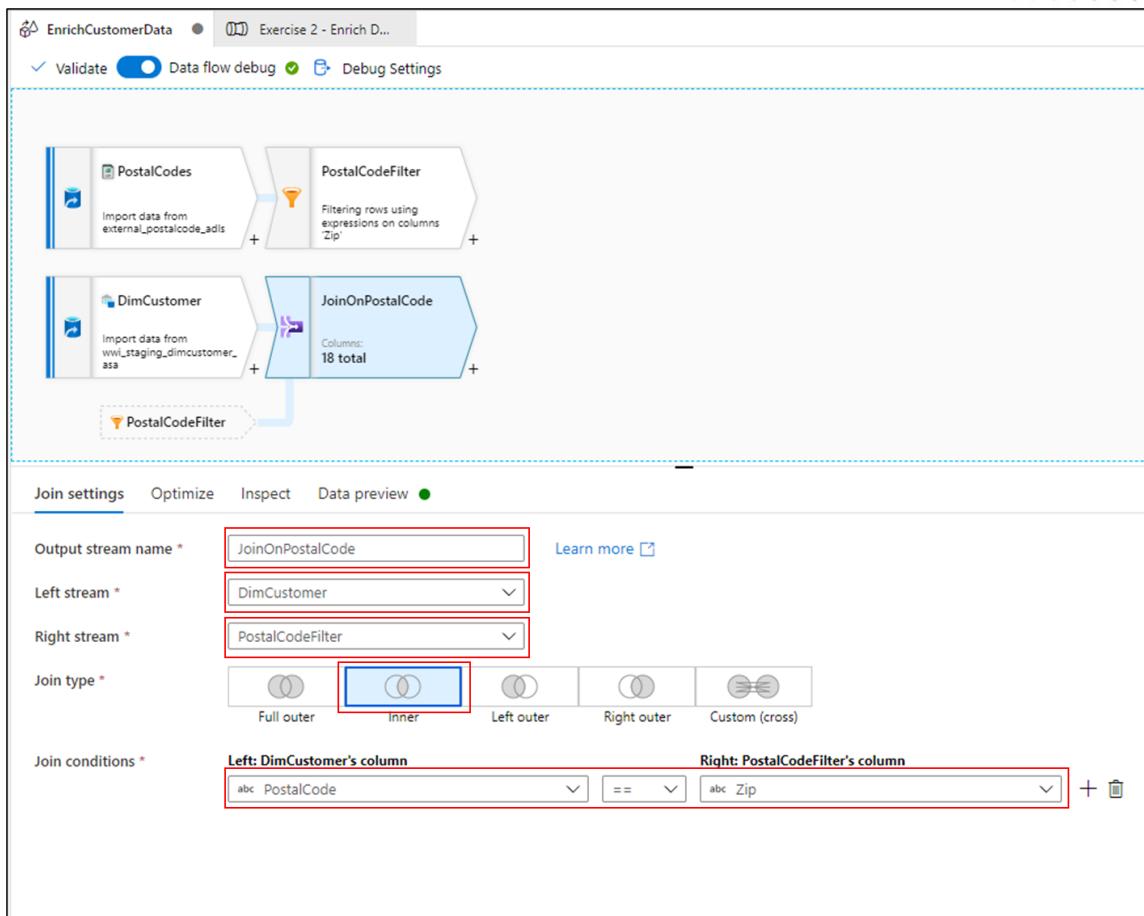
| Column name    | Type      |
|----------------|-----------|
| CustomerKey    | integer   |
| WWICustomerID  | integer   |
| Customer       | string    |
| BillToCustomer | string    |
| Category       | string    |
| BuyingGroup    | string    |
| PrimaryContact | string    |
| PostalCode     | string    |
| ValidFrom      | timestamp |
| ValidTo        | timestamp |
| LineageKey     | integer   |

22. **DimCustomer** 옆의 +버튼을 클릭하고 **Join**을 선택합니다.

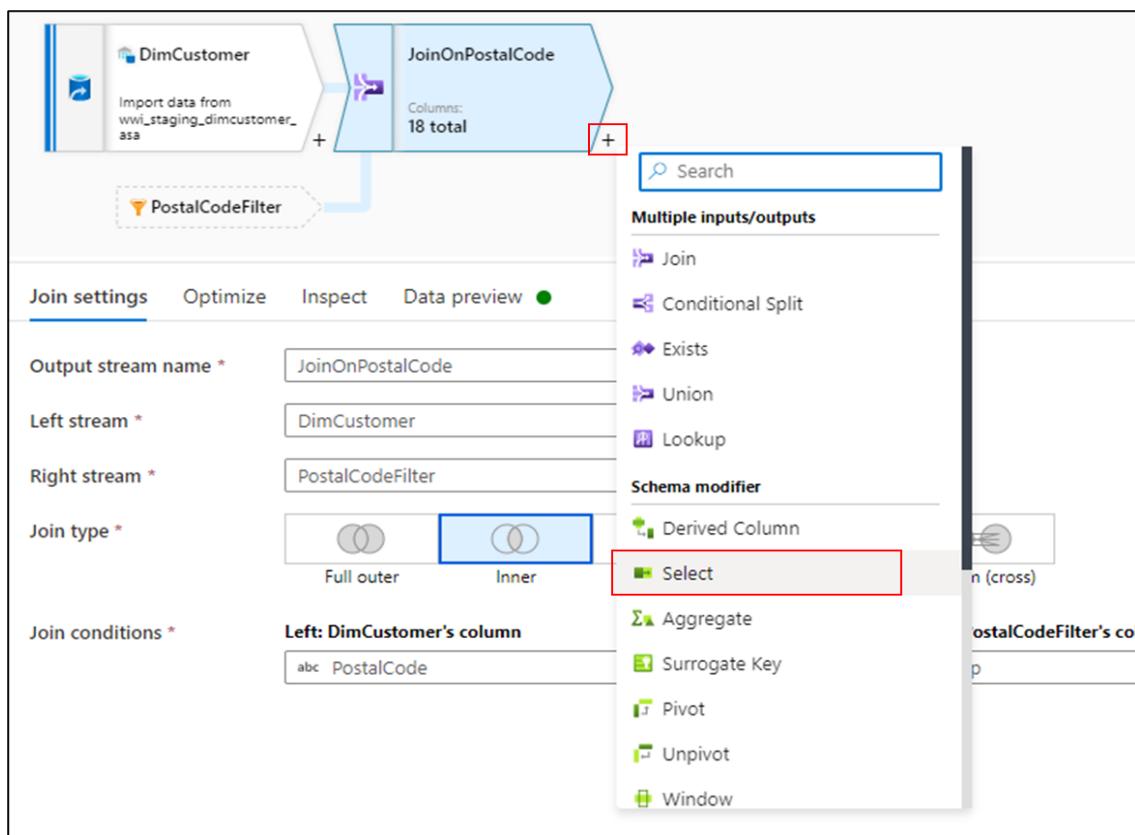


23. Join settings Tab에 아래 내용으로 설정합니다.

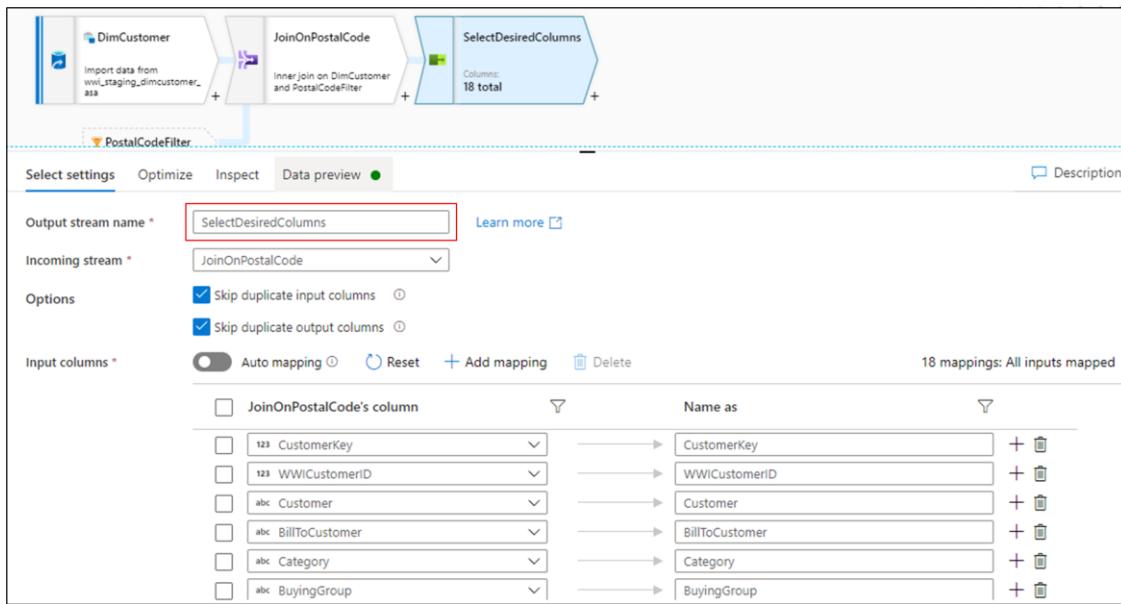
- ✓ Output stream name: **JoinOnPostalCode** 입력
- ✓ Left stream: **DimCustomer** 선택
- ✓ Right stream: **PostalCodeFilter** 선택
- ✓ Join type: **inner** 선택
- ✓ Join conditions: **PostalCode** 컬럼, **Zip** 컬럼 선택



24. **JoinOnPostalCode** 오른쪽 +를 클릭하여 **Select**를 선택합니다.

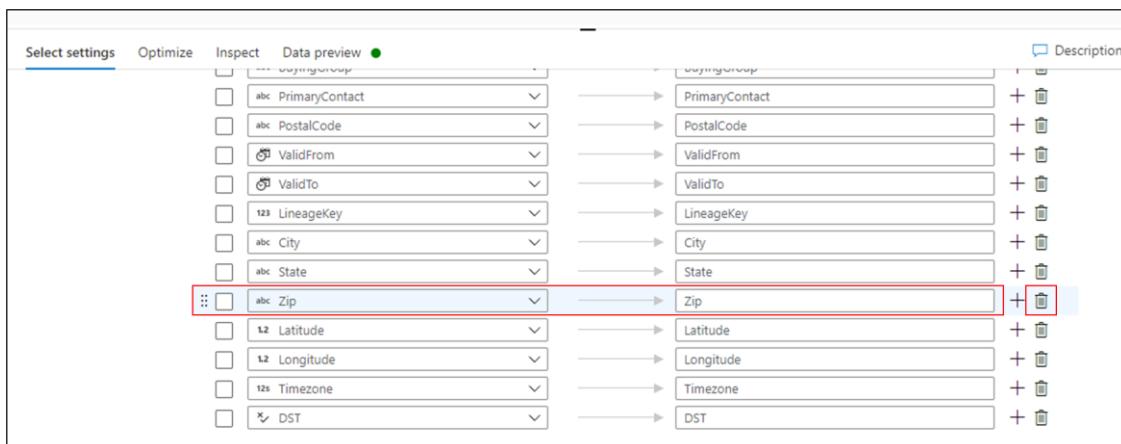


25. Select settings Tab에서 Output stream name에 SelectDesiredColumns를 입력 합니다.

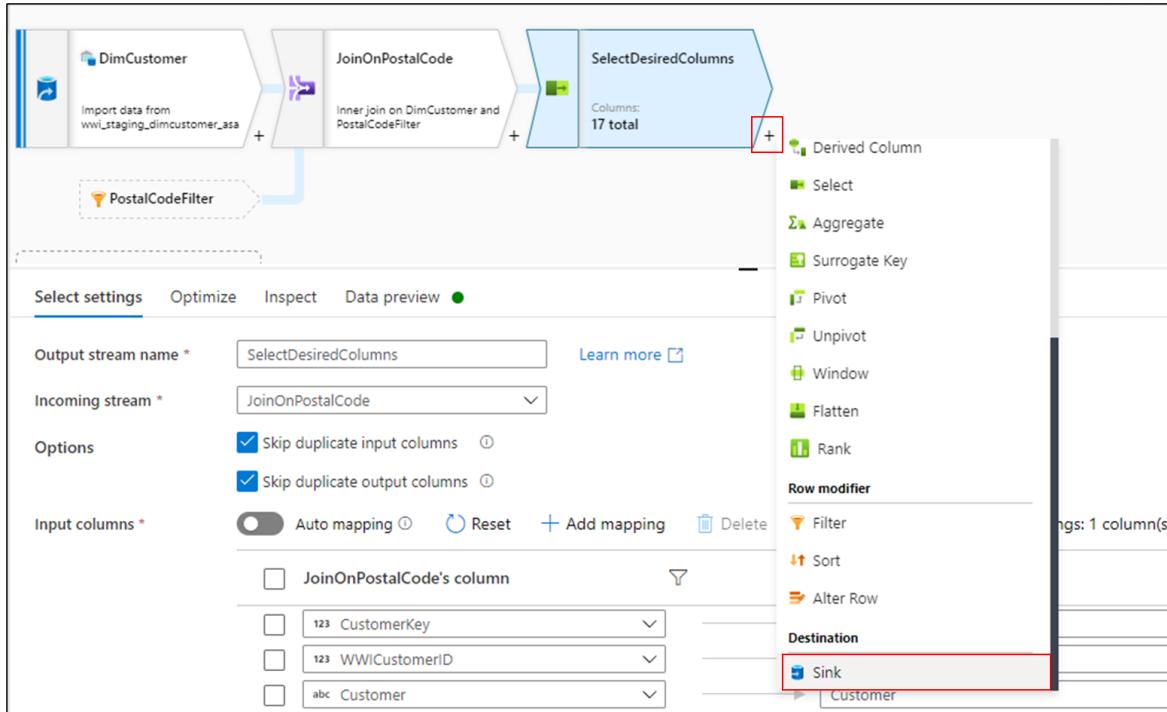


26. 아래 Input columns에서 사용한 컬럼을 선택하고 이름을 변경할 수 있습니다.

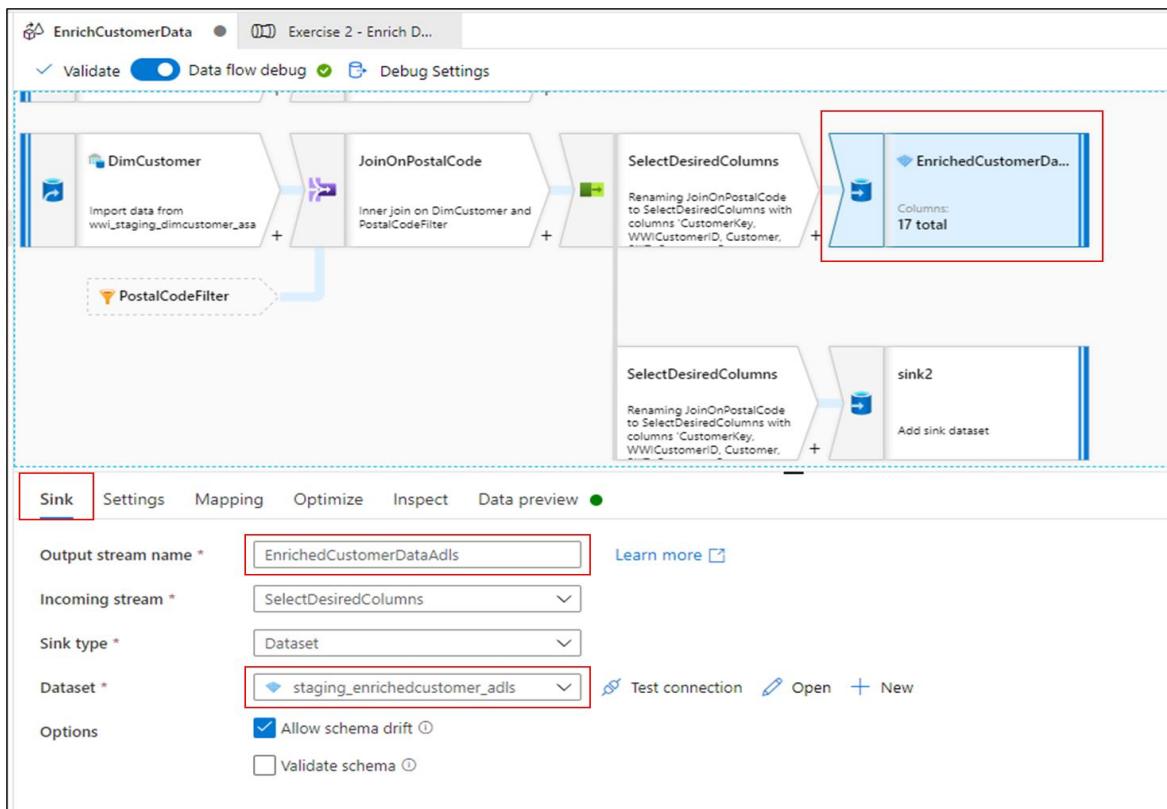
Zip 컬럼은 PostalCode와 중복 되므로 오른쪽 휴지통 버튼을 클릭하여 삭제 합니다.



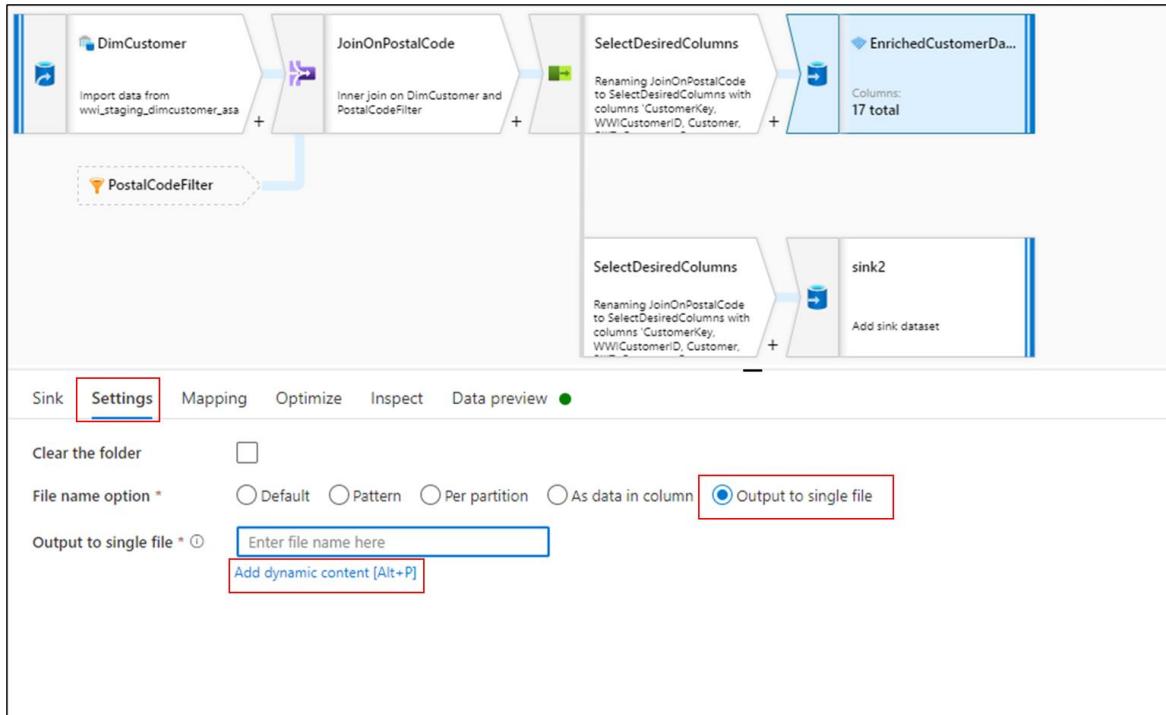
27. **SelectDesiredColumns** 오른쪽 +를 클릭하여 맨아래 **Sink**를 선택합니다. 한번 더 반복해서 총 2개의 **Sink**를 추가합니다. 이는 **SelectDesiredColumns**의 Output 결과를 두개의 Destination (Azure Synapse Analytics, Azure Data Lake Storage Gen2)에 전송하기 위해서입니다.



28. 먼저 첫번째 Sink를 선택하고 **Sink Tab**에서 **Output stream name**에 **EnrichedCustomerDataAdls**를 입력하고 **Dataset** 항목에 **staging\_enrichedcustomer\_adls**를 선택합니다.



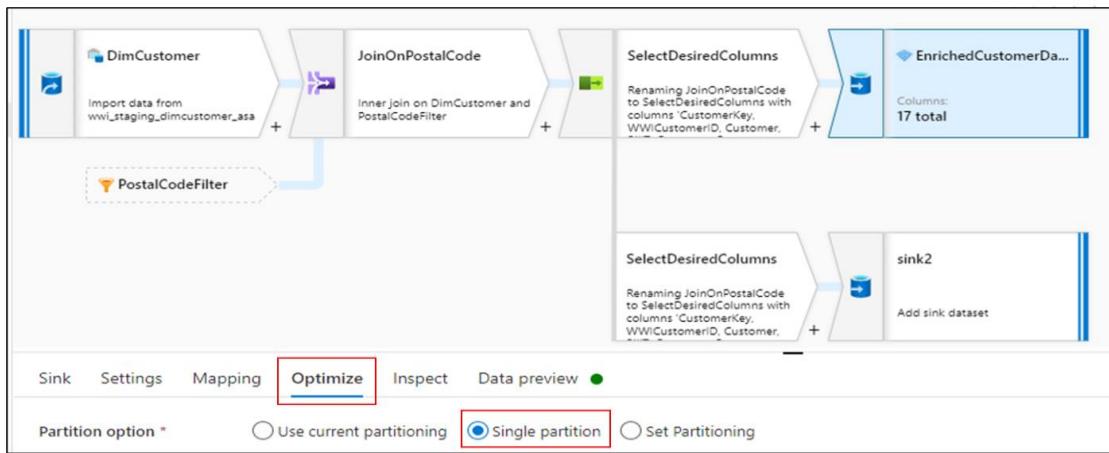
29. Settings Tab에서 File name option을 Output to single file을 선택하고 Output to single file 항목에서 Add dynamic contents(Alt+P)를 클릭 합니다.



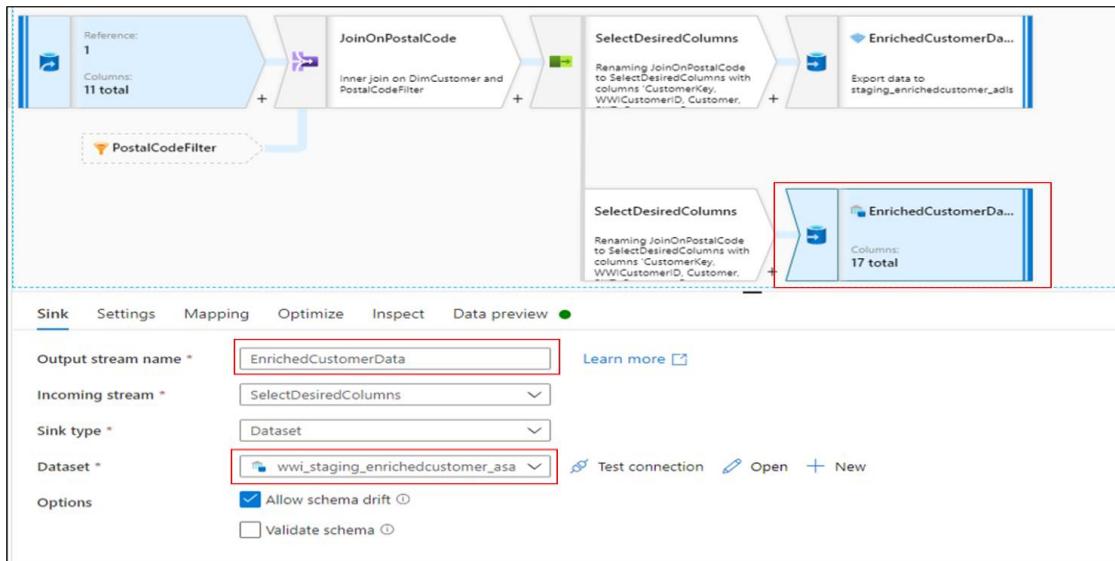
30. Visual expression builder화면의 Expression Editor에 아래 내용을 복사해서 붙여넣고 Save and finish 버튼을 클릭합니다.

```
concat('enrichedcustomer_', $UniqueId, '.parquet')
```

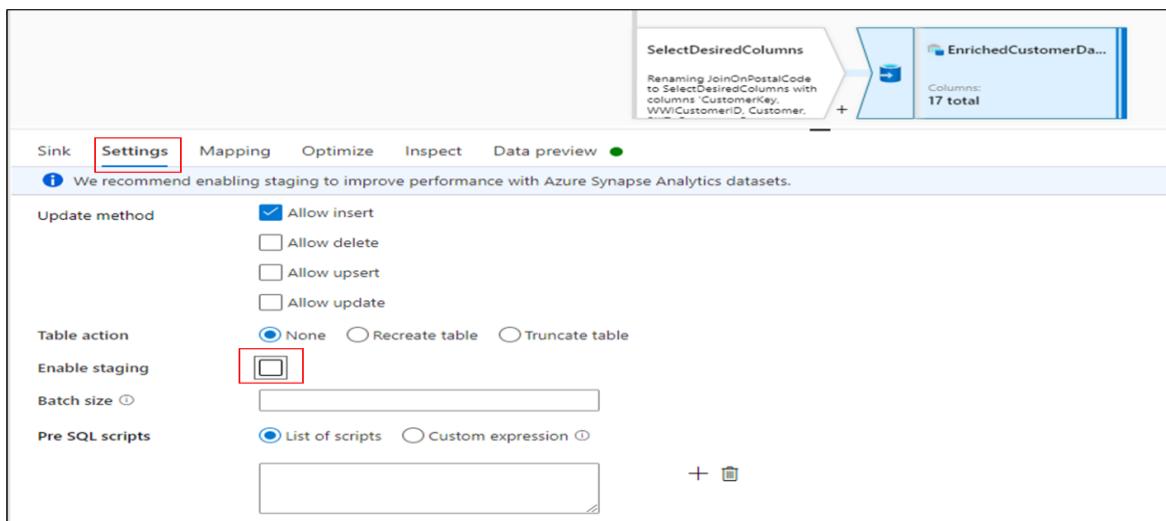
31. Optimize Tab으로 이동하여 Partition option을 Single partition을 선택합니다.



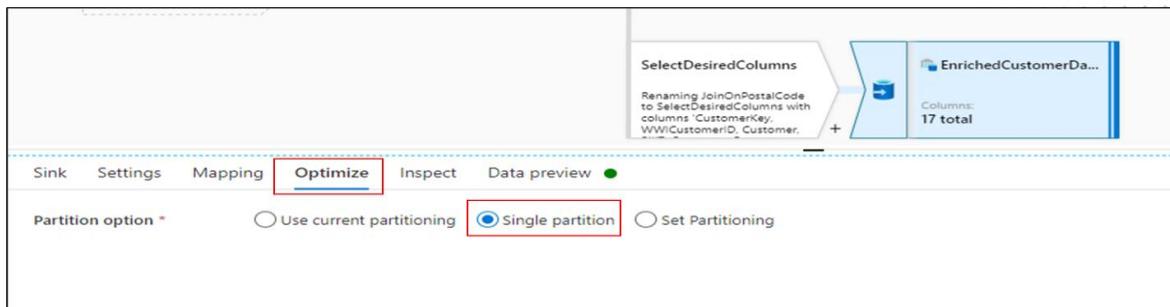
32. 두 번째 Sink를 선택하고 Output stream name에 EnrichedCustomerData를 입력, Dataset에 wwi\_staging.enrichedcustomer\_asa를 선택합니다.



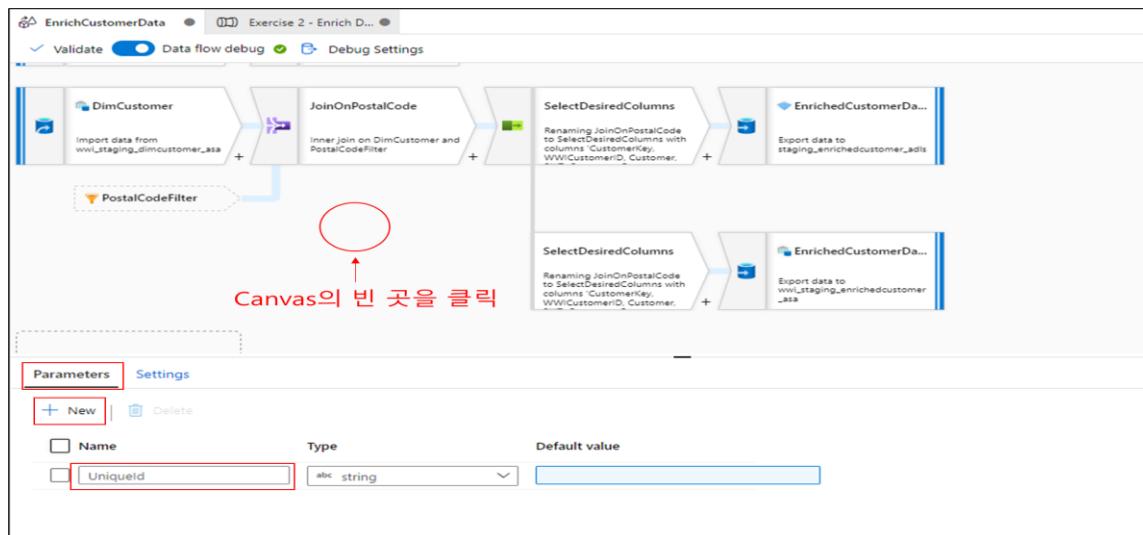
33. Settings Tab에서 Enable staging을 UnCheck 합니다.



34. Optimize Tab에서 Partition option을 Single partition을 선택합니다.

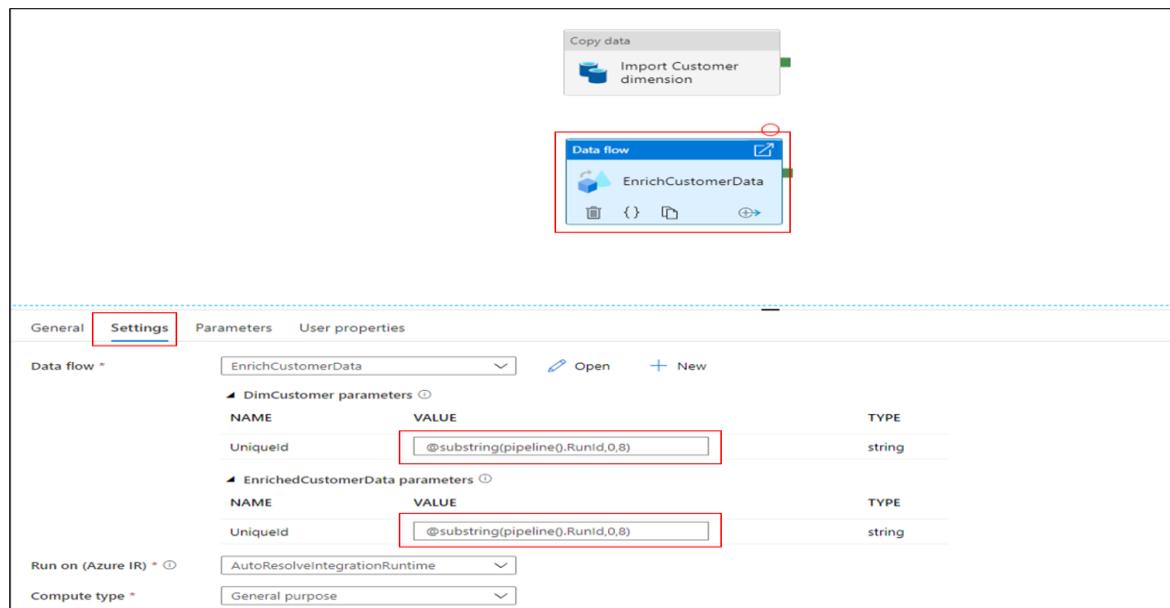


35. Canvas의 빈 곳을 클릭 후 Parameters Tab에서 +New 버튼을 클릭하고 UniqueId를 입력 합니다.

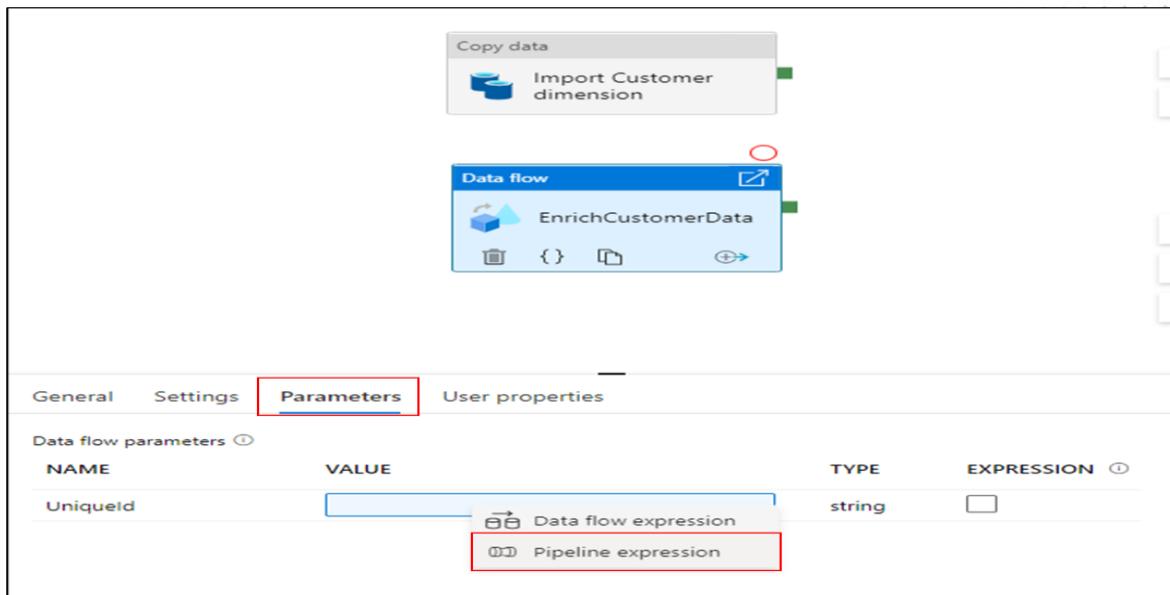


36. Data flow의 Settings Tab에서 UniqueID Value를 입력하는 두 곳에 아래 내용을 복사 하여 붙여넣습니다.

```
@substring(pipeline().RunId,0,8)
```

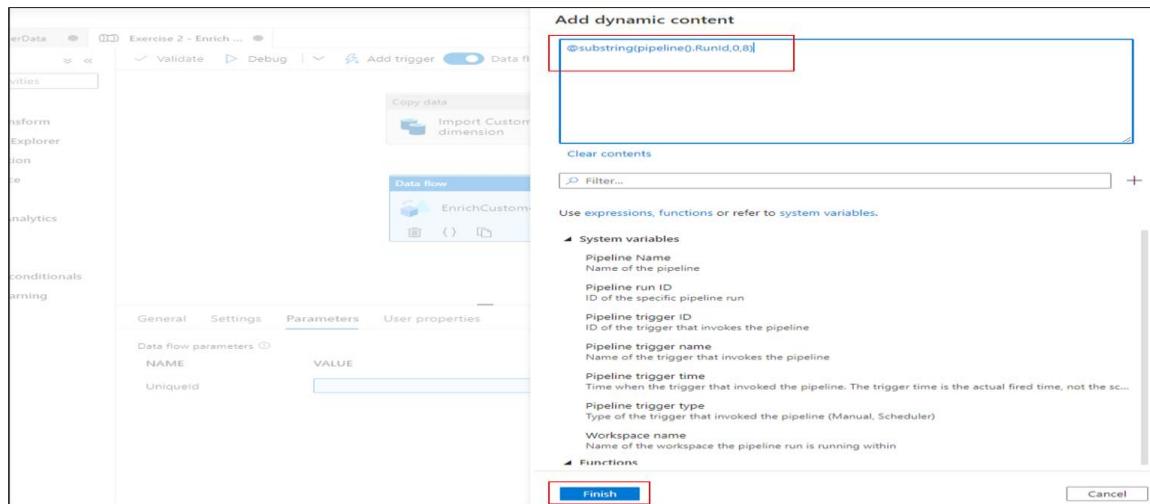


37. Parameters Tab으로 이동 후 Value 항목의 Text Box를 클릭하고 Pipeline expression을 선택 합니다.

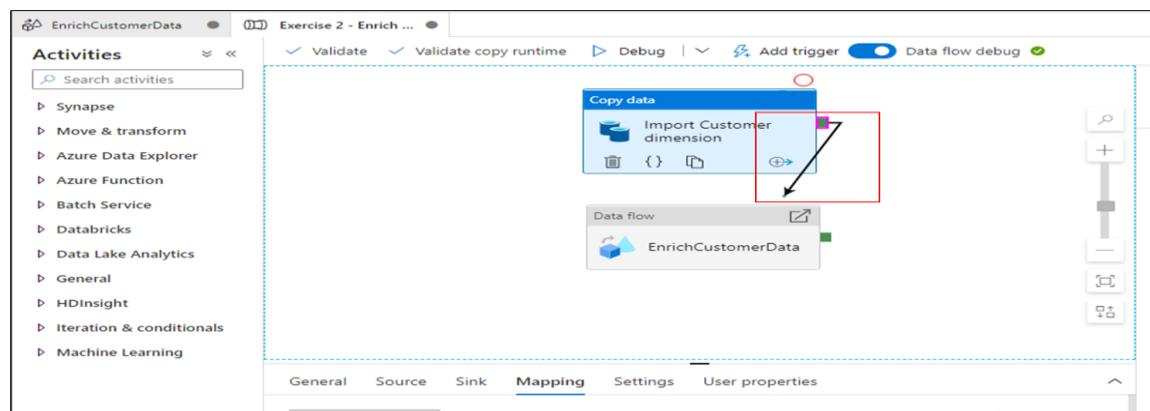


38. Add dynamic content 블레이드에서 아래 내용을 입력하고 Finish 버튼을 클릭합니다.

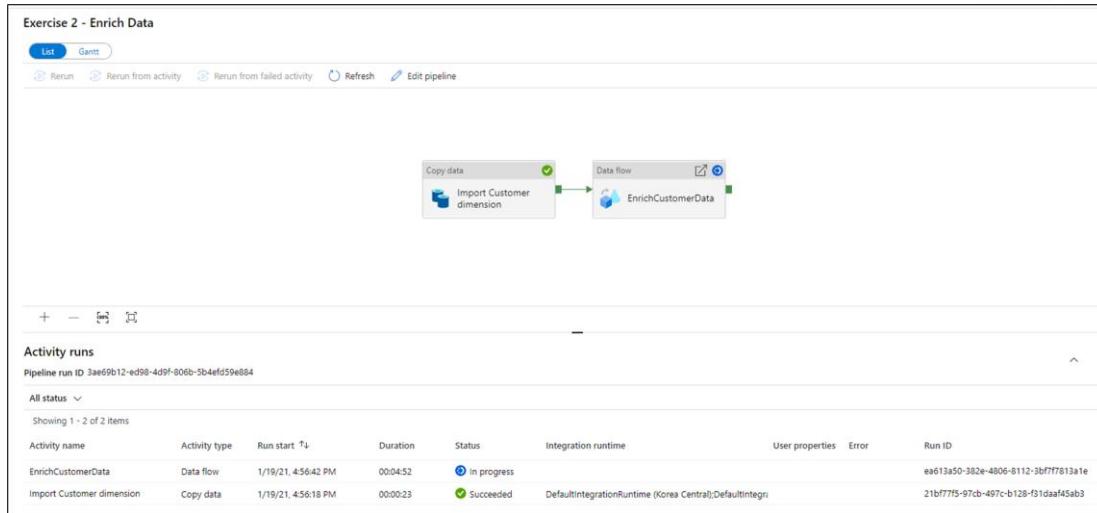
```
@substring(pipeline().RunId,0,8)
```



39. Copy Data Activity의 Output Point를 Drag하여 Data Flow Activity와 연결 합니다.



40. 상단의 **Publish all** 버튼을 클릭하여 작성 내용을 저장하고 **Add Trigger**의 **Trigger Now**를 클릭하여 완성된 Pipeline을 실행 합니다.



41. Pipeline이 실행 완료 되면 Data Hub로 이동 하여 SQLPool01에 테이블이 생성되었고, asadatalake01의 Staging/Customer/ 아래 파일이 생성 되었는지 확인합니다.

The screenshot shows the Azure Data Lake Storage Gen2 interface. It displays two windows: one for the 'Data' workspace and one for the 'staging' folder under the 'wwi' workspace. In the 'staging' folder, there is a single file named 'enrichedcustomer\_3ae69b12.parquet', which is highlighted with a red box. This file was created by the pipeline execution.

## 5. Exercise 3: Power BI Integration

이 연습에서는 Azure Synapse Analytics에서 Power BI 보고서를 작성합니다.

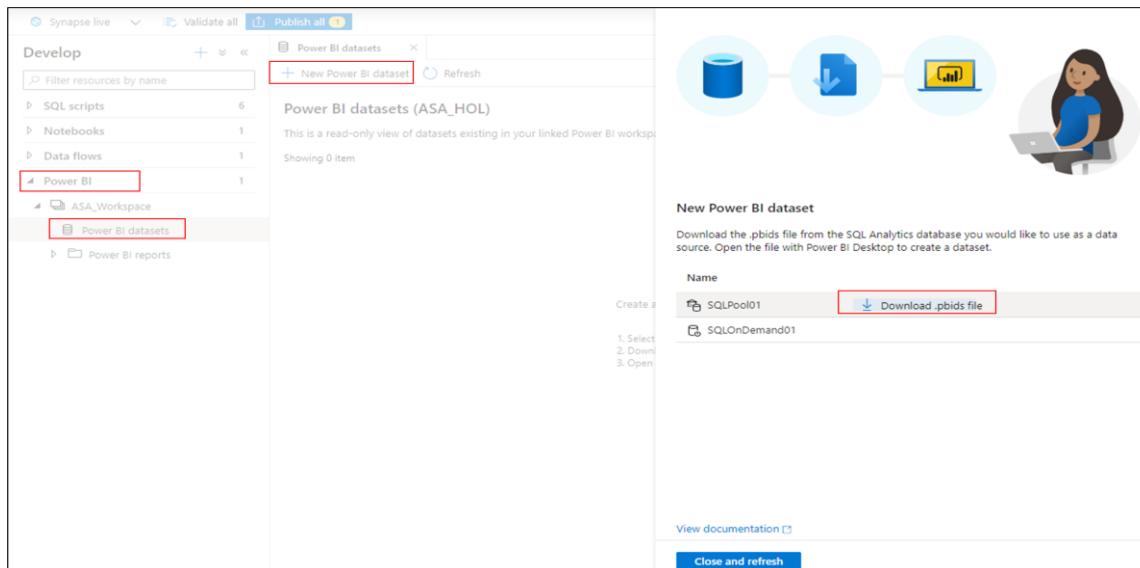
데이터 탐색, 분석 및 해석의 시각적 접근 방식은 기술 사용자 (데이터 엔지니어, 데이터 과학자)와 비즈니스 사용자 모두에게 필수적인 도구 중 하나입니다. 매우 유연하고 성능이 뛰어난 데이터 표현 계층을 갖는 것은 모든 최신 데이터 플랫폼의 필수 요소입니다.

Azure Synapse Analytics는 검증되고 매우 성공적인 데이터 프레젠테이션 및 탐색 플랫폼인 Power BI와 기본적으로 통합됩니다. Power BI 환경은 Synapse Studio 내에서 사용할 수 있습니다.

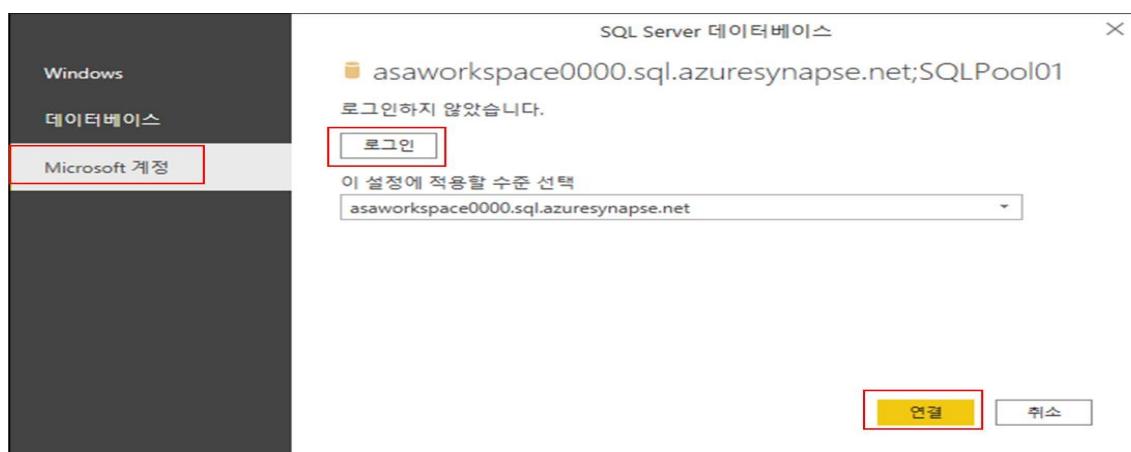
### 5.1. Task 1: Create a Power BI dataset in Synapse

- 원쪽 메뉴에서 **Develop** Hub로 이동하여 **Power BI → Power BI datasets** 를 클릭합니다.

상단의 **+New Power BI dataset**을 클릭하고 **SQLPool01** 오른쪽의 **Download .pbids file**을 다운로드 합니다.



- 다운로드 한 pbids 파일을 실행 시키면 Power BI Desktop이 실행 되고 로그인 Popup이 뜨는데 왼쪽 Microsoft 계정을 클릭하고 Azure 계정으로 로그인 후 연결 버튼을 클릭합니다.



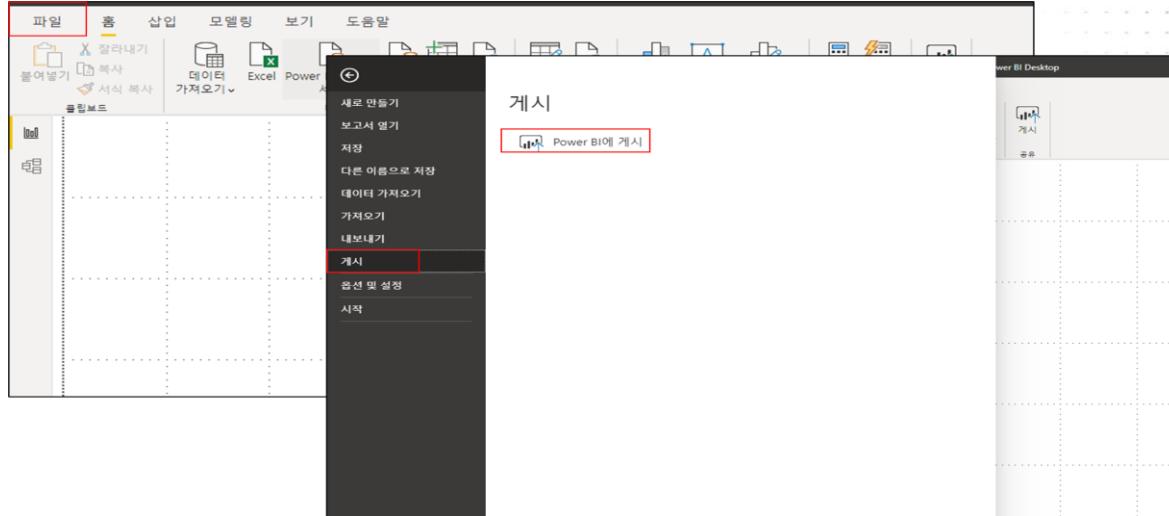
3. 탐색창에 SQLPool01의 Table 목록이 조회 됩니다. 여기서 **wwi.FactSale** 테이블을 선택하고 **로드** 버튼을 클릭합니다.

The screenshot shows the Power BI Data Explorer interface. On the left, there is a tree view of tables under the connection 'asaworkspace0000.sql.azuresynapse.net: S...'. The 'wwi.FactSale' table is selected and highlighted with a red box. On the right, a preview of the 'wwi.FactSale' table is displayed in a grid format with columns: SaleKey, CityKey, CustomerKey, BillToCustomerKey, StockItemKey, and Inv. The first few rows of data are visible. At the bottom right of the preview area, there are three buttons: '로드' (Load) with a red box around it, '데이터 변환' (Data Transformation), and '취소' (Cancel).

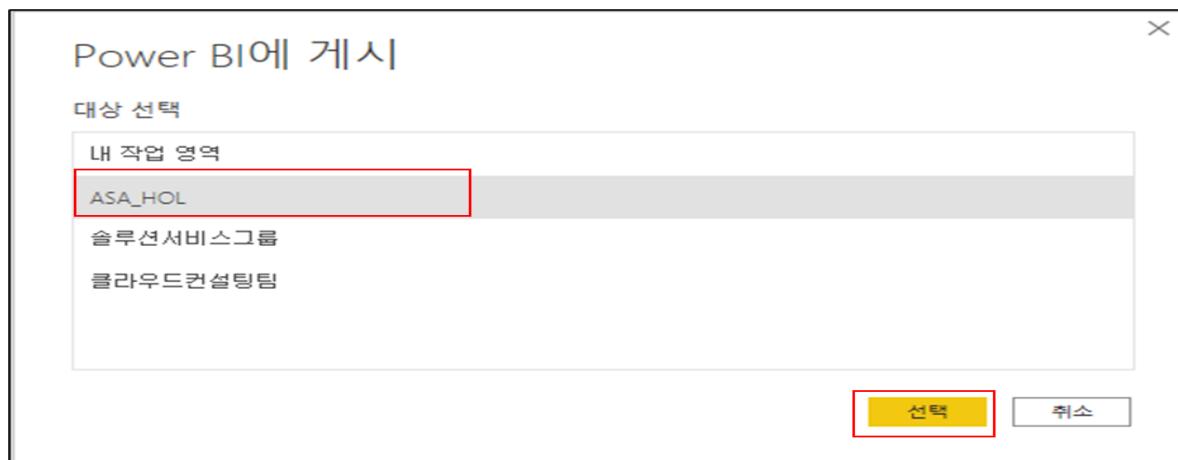
4. 연결 설정 Popup에서 **DirectQuery**를 선택하고 **확인** 버튼을 클릭합니다.



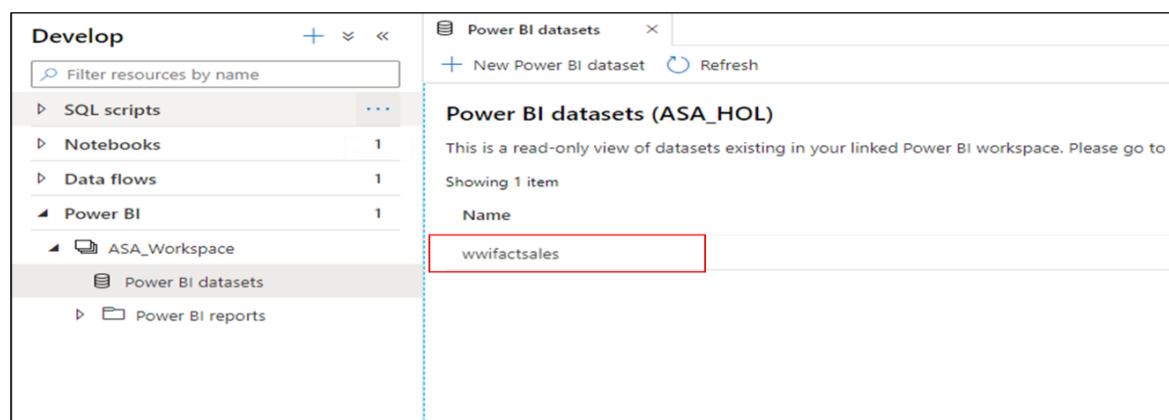
5. 데이터 로드가 완료 되면 왼쪽 상단의 **File** 메뉴를 클릭, 게시를 선택하고 **Power BI에 게시**를 클릭합니다.



6. 먼저 변경내용을 로컬에 파일로(wwifactsales.pbix) 저장하고 Power BI에 게시할 Workspace를 선택 합니다. Workspace이름은 사용자 마다 다릅니다.

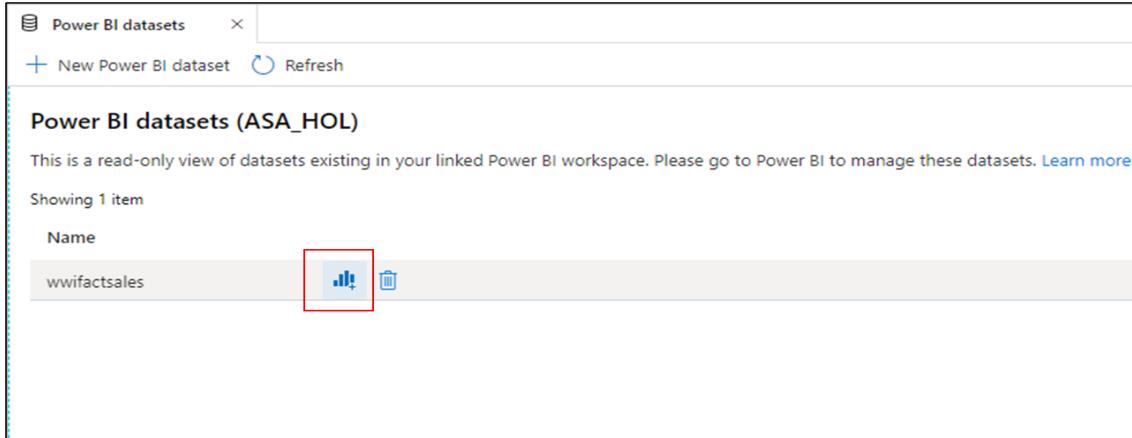


7. 다시 Azure Synapse Studio로 돌아와서 **Close and refresh** 버튼을 클릭하고, 방금 게시한 데이터 셋이 보이는지 확인합니다. 만약 작업전에 **Close and refresh** 버튼을 클릭하였다면 상단의 **Refresh**버튼을 클릭하여 재 조회합니다.



## 5.2. Task 2: Create a Power BI report in Synapse

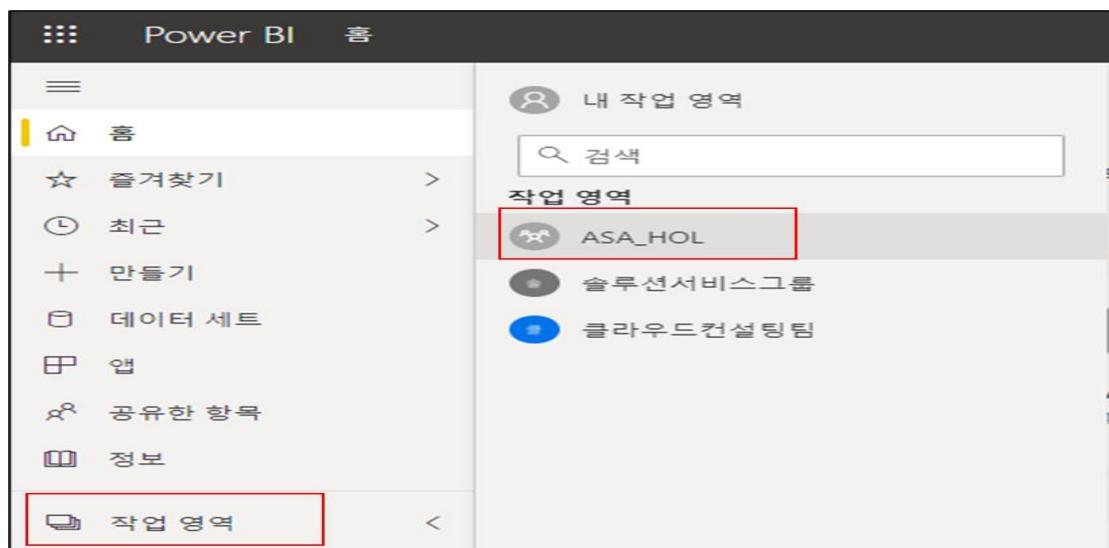
- Task 1에서 추가한 **wwifactsales** Power BI dataset 위로 마우스를 올리면 **New Power BI Report** 아이콘이 나타납니다. 아이콘을 클릭합니다.



- Power BI Report designer 화면이 새로운 Tab에서 실행 됩니다.



- Web Browser에서 [www.powerbi.com](http://www.powerbi.com) 으로 접속해서 로그인합니다. 로그인 후 Power BI 홈 메뉴에서 아래쪽에 작업영역 을 클릭 하고 데이터 세트을 저장 했던 작업영역(Workspace)를 선택합니다.

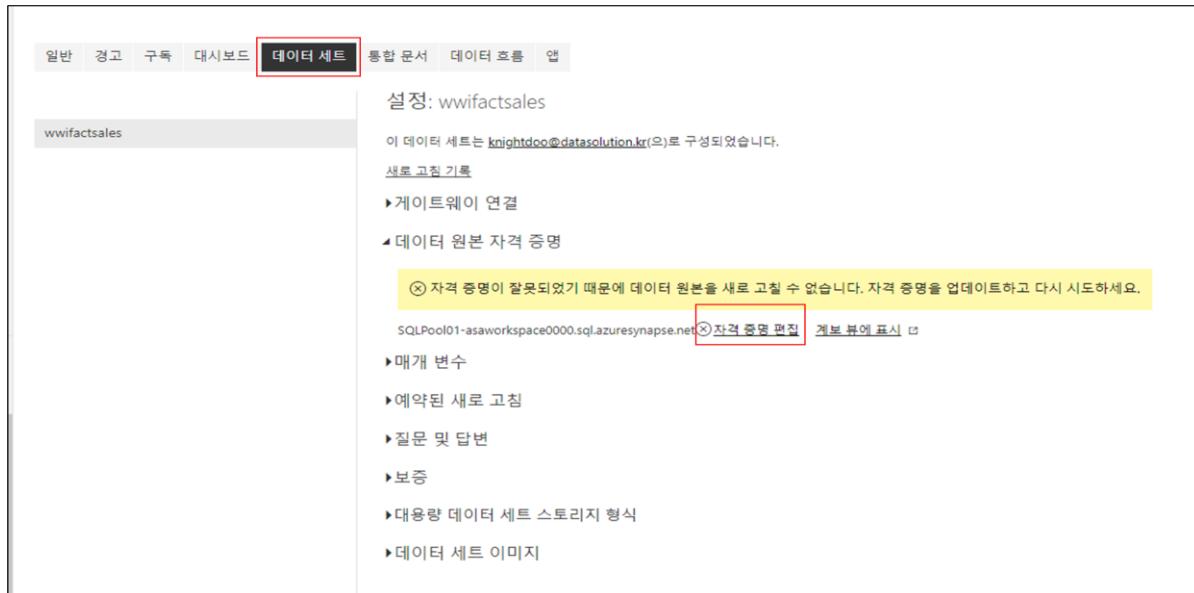


4. 오른쪽 상단의 설정 (톱니바퀴모양 아이콘) 버튼을 클릭하고 설정을 선택합니다.



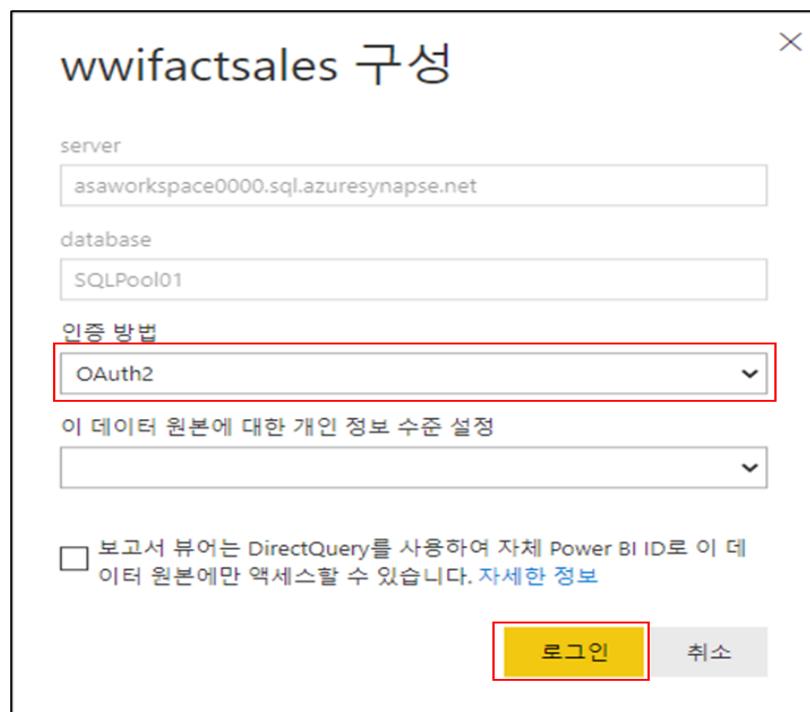
The screenshot shows the Power BI service interface with the 'ASA\_HOL' workspace selected. The top navigation bar includes 'Power BI', 'ASA\_HOL', and a search bar. On the left, there's a sidebar with options like '종', '즐겨찾기', '최근', '만들기', '데이터 세트', '업', and '공유한 항목'. The main content area displays a table with two rows: 'wwifactsales' (reporter, ASA\_HOL, 21.1.27.오전 10:21:07) and 'wwifactsales' (데이터 세트, ASA\_HOL, 21.1.27.오전 10:21:07). A red box highlights the gear icon in the top right corner of the ribbon.

5. 데이터 세트를 선택하고 자격증명 편집을 클릭합니다.



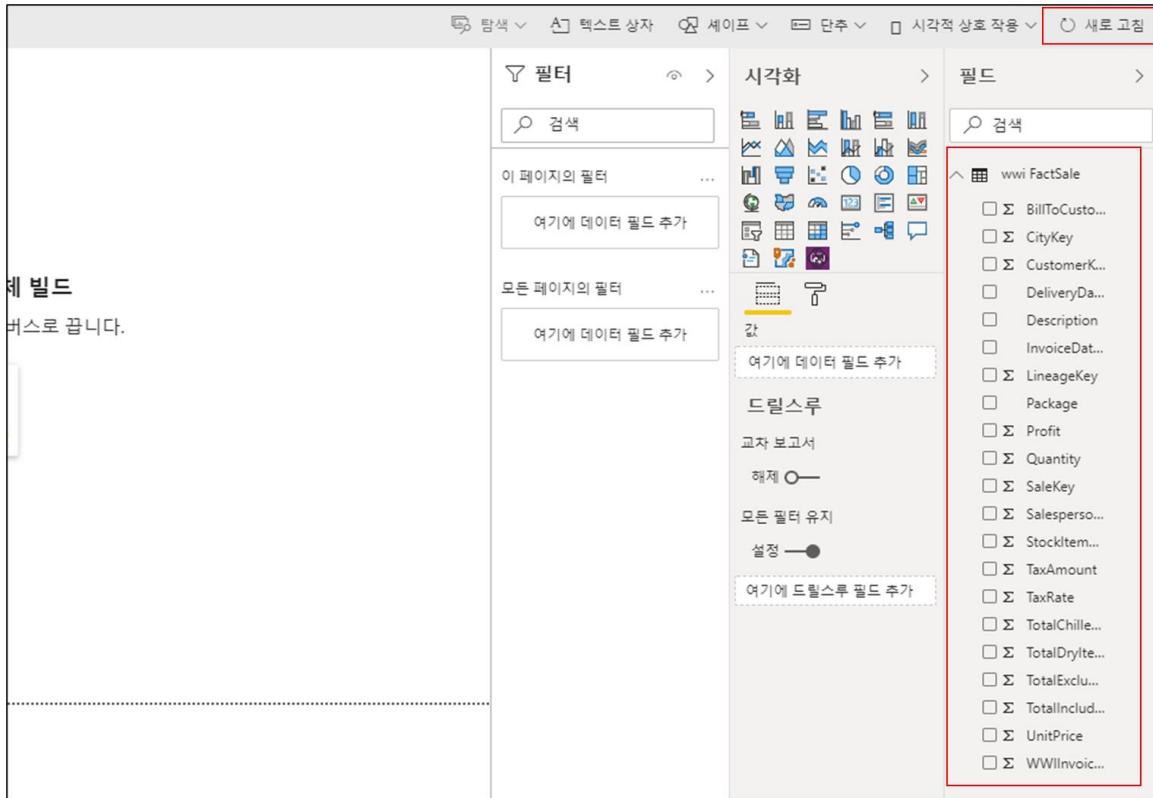
The screenshot shows the configuration page for the 'wwifactsales' data set. The top navigation bar has tabs: '일반', '경고', '구축', '대시보드', '데이터 세트' (which is highlighted with a red box), '통합 문서', '데이터 흐름', and '앱'. The main content area shows the connection settings for 'wwifactsales': '설정: wwifactsales' and '이 데이터 세트는 knightdoo@datasolution.kr(으)로 구성되었습니다.' Below this, there are several sections with icons: '▶ 게이트웨이 연결' (red box), '◀ 데이터 원본 자격 증명' (yellow box), '▶ 매개 변수', '▶ 예약된 새로 고침', '▶ 질문 및 답변', '▶ 보증', '▶ 대용량 데이터 세트 스토리지 형식', and '▶ 데이터 세트 이미지'. A red box highlights the 'Edit' button in the top right corner of the configuration page.

6. wwifactsales 구성 Popup에서 인증방법을 OAuth2로 변경하고, 로그인 합니다.

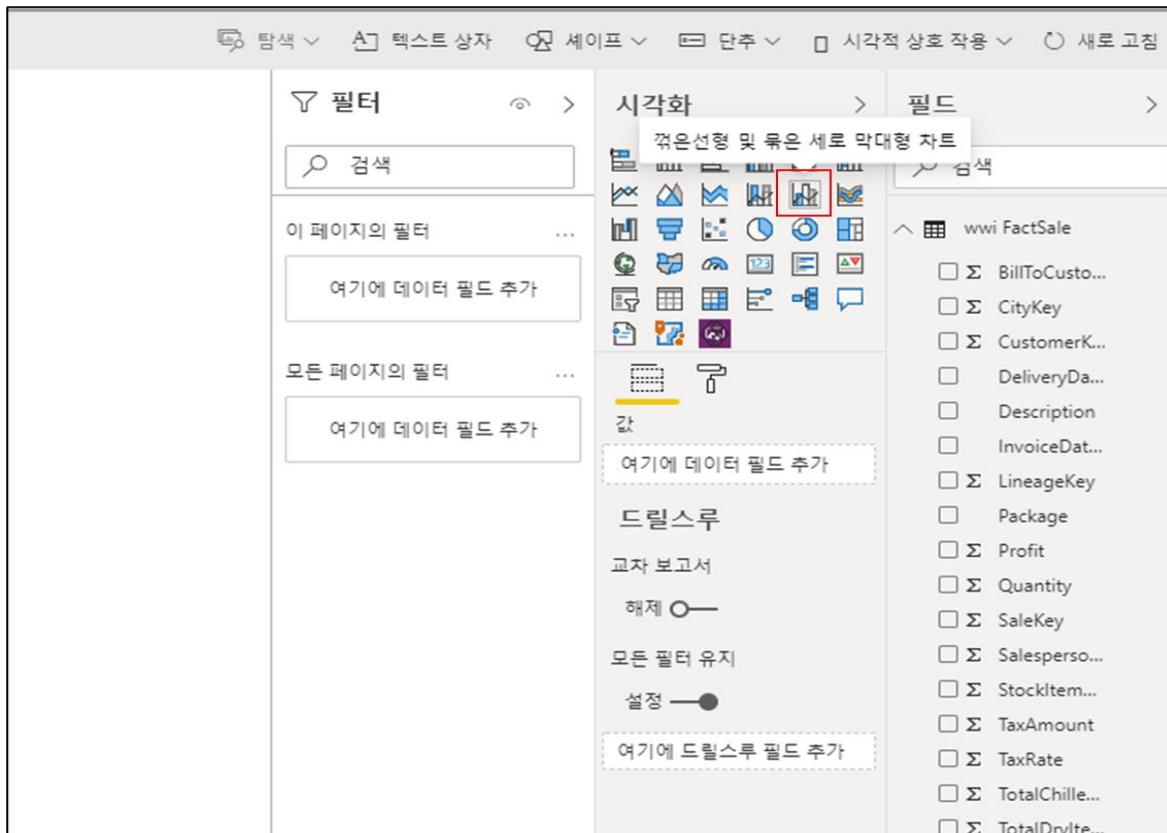


The screenshot shows the 'wwifactsales 구성' (Configure) dialog box. It contains fields for 'server' (asaworkspace0000.sql.azuresynapse.net) and 'database' (SQLPool01). A red box highlights the '인증 방법' (Authentication Method) dropdown, which is currently set to 'OAuth2'. Below it is a section for '이 데이터 원본에 대한 개인 정보 수준 설정' (Personal information level settings for this data source). At the bottom, there's a checkbox for '보고서 뷰어는 DirectQuery를 사용하여 자체 Power BI ID로 이 데이터 원본에만 액세스할 수 있습니다.' (Report viewer uses DirectQuery to access this data source only via its own Power BI ID) and a red box highlights the '로그인' (Login) button.

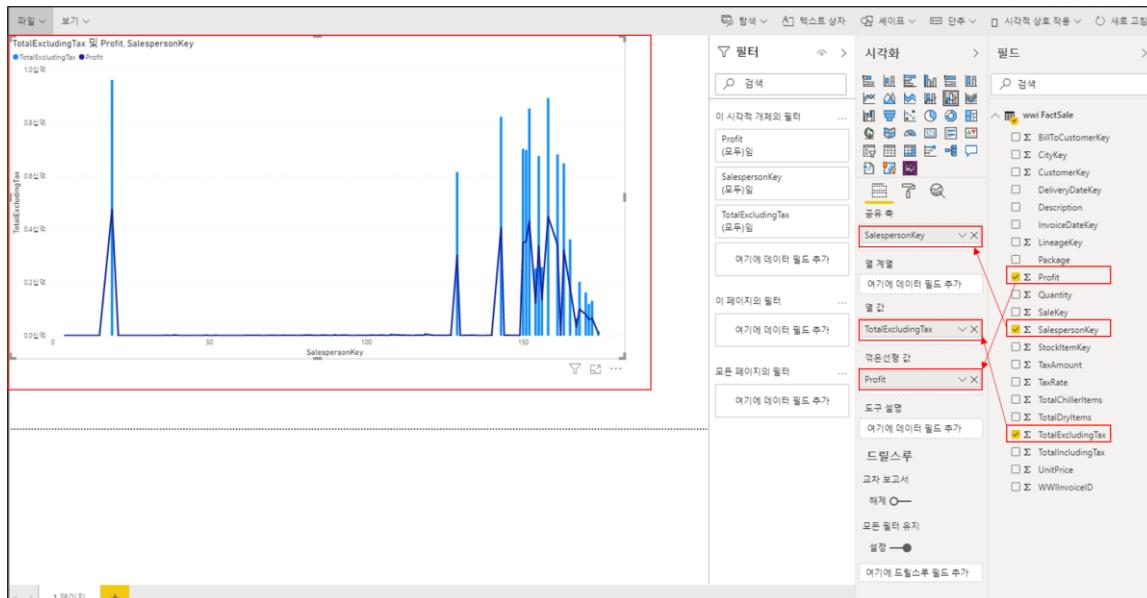
7. 로그인이 완료 되면 다시 Azure Synapse Analytics Studio로 돌아와서 Power BI 보고서 오른쪽 상단의 **새로고침**을 클릭합니다. 필드영역에서 wwi.FactSale 테이블의 컬럼들이 조회 됩니다.



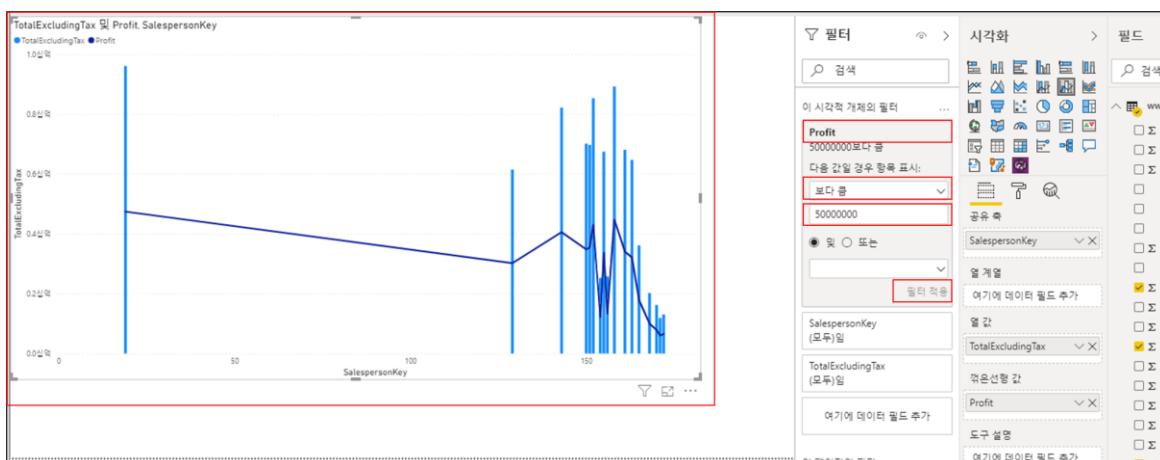
8. 시작화 영역에서 꺾은선형 및 뮤은 세로 막대형 차트 아이콘을 클릭합니다.



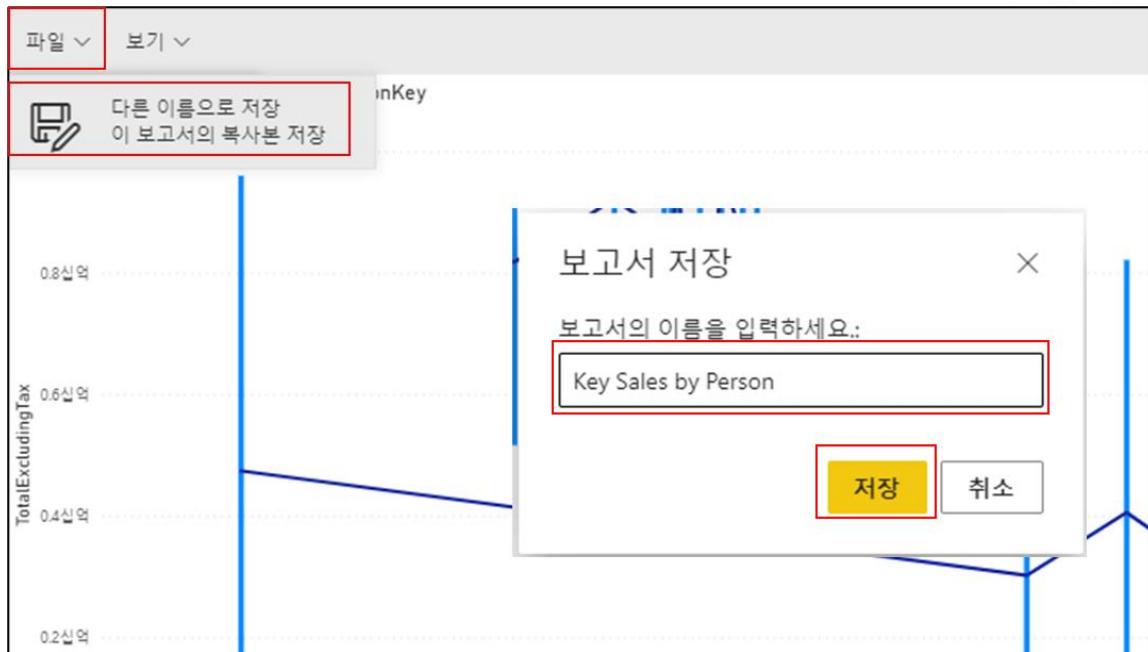
9. 차트의 공유 축에 SalespersonKey 필드, 열 값에 TotalExcludingTax 필드, 꺽은선형 값에 Profit 필드를 각각 Drag하여 매핑 합니다. 매핑이 완료되면 잠시 후 왼쪽 차트에 그래프가 표시 됩니다.



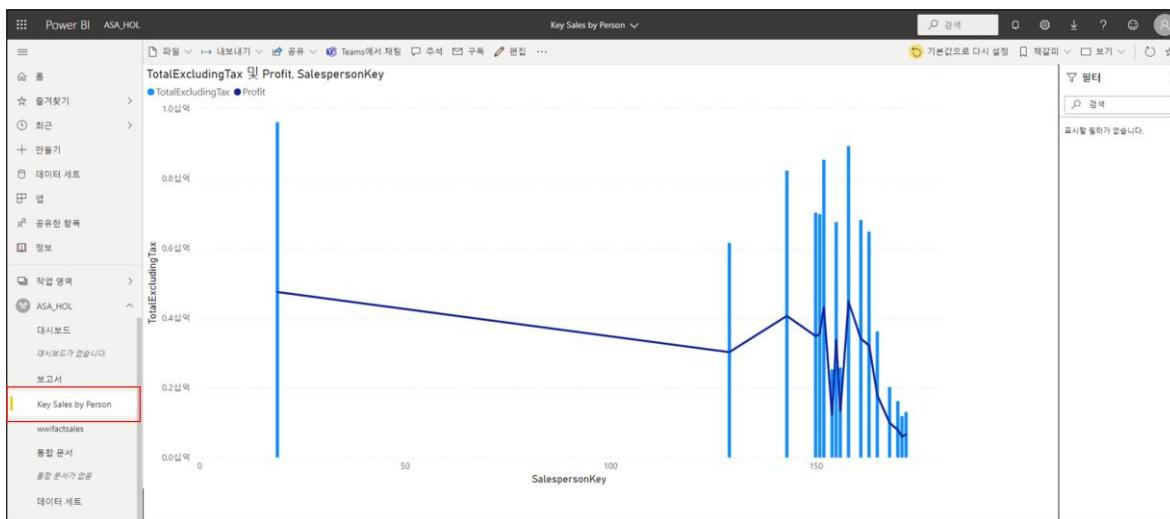
10. 필터 영역에서 **Profit** 필드 오른쪽에 **▼**를 클릭하고 조건 부분에서 **보다 큼**을 선택, 값을 **50000000**을 입력 합니다. 필터적용을 클릭하면 왼쪽의 그래프에 필터가 적용되어 변경 됩니다.



11. 보고서 왼쪽 상단의 파일 메뉴에서 다른 이름으로 저장을 클릭하고 보고서 이름을 Key Sales by Person으로 입력하고 저장 버튼을 클릭하여 저장합니다.



12. 저장 된 Power BI 보고서는 Power BI 홈의 보고서에 게시되고 다른 사용자와도 공유 할 수 있습니다.



### 5.3. Task 3: View the SQL query

1. Synapse Studio 왼쪽메뉴에서 Monitor를 선택하여 Monitor Hub로 이동합니다. Monitor Hub의 Integration의 SQL requests를 클릭하고 오른쪽 상단의 Pool을 SQLPool01로 선택합니다. SQLPool01에서 수행 된 Query 목록이 조회 됩니다. Power BI에서 수행한 Request ID 위에 마우스를 올리면 문서모양의 Request content 아이콘이 나타나는데 이를 클릭합니다.

| Request ID | Request content                 | Submit time         | Duration | Status    | Queued duration | Workload group |
|------------|---------------------------------|---------------------|----------|-----------|-----------------|----------------|
| QID11750   | EXEC sys.sp_set_session_context | 1/27/21, 2:08:19 PM | 0s       | Completed | 0s              |                |
| QID11748   | SELECT TOP (1000001) *          | 1/27/21, 2:08:14 PM | 5s       | Completed | 0s              |                |
| QID11747   | EXEC sys.sp_set_session_context | 1/27/21, 2:08:14 PM | 0s       | Completed | 0s              |                |
| QID11746   | EXEC sys.sp_set_session_context | 1/27/21, 2:08:14 PM | 5s       | Completed | 0s              |                |
| QID11745   | exec [sp_executesql] @_0x02e1   | 1/27/21, 2:08:14 PM | 5s       | Completed | 0s              |                |
| QID11744   | exec [sp_executesql] @_0x02e1   | 1/27/21, 2:08:14 PM | 5s       | Completed | 0s              |                |
| QID11742   | EXEC sys.sp_set_session_context | 1/27/21, 2:06:38 PM | 0s       | Completed | 0s              |                |
| QID11741   | SELECT TOP (1000001) *          | 1/27/21, 2:06:34 PM | 4s       | Completed | 0s              | smallrc        |
| QID11740   | EXEC sys.sp_set_session_context | 1/27/21, 2:06:34 PM | 0s       | Completed | 0s              |                |
| QID11739   | EXEC sys.sp_set_session_context | 1/27/21, 2:06:33 PM | 5s       | Completed | 0s              |                |
| QID11738   | EXEC sys.sp_set_session_context | 1/27/21, 2:06:25 PM | 0s       | Completed | 0s              |                |
| QID11737   | SELECT TOP (1000001) *          | 1/27/21, 2:06:21 PM | 4s       | Completed | 0s              | smallrc        |
| QID11736   | EXEC sys.sp_set_session_context | 1/27/21, 2:06:21 PM | 0s       | Completed | 0s              |                |
| QID11735   | EXEC sys.sp_set_session_context | 1/27/21, 2:06:21 PM | 4s       | Completed | 0s              |                |

2. 실제 수행 된 Query를 확인 할 수 있습니다.

```

Request content

QID11741

SELECT
TOP <1000001> *
FROM
(
(
(
select [$Table].[SaleKey] as [SaleKey],
[$Table].[CityKey] as [CityKey],
[$Table].[CustomerKey] as [CustomerKey],
[$Table].[BillToCustomerKey] as [BillToCustomerKey],
[$Table].[StockItemKey] as [StockItemKey],
[$Table].[InvoiceDateKey] as [InvoiceDateKey],
[$Table].[DeliveryDateKey] as [DeliveryDateKey],
[$Table].[SalespersonKey] as [SalespersonKey],
[$Table].[WWIInvoiceID] as [WWIInvoiceID],
[$Table].[Description] as [Description],
[$Table].[Package] as [Package],
[$Table].[Quantity] as [Quantity],
[$Table].[UnitPrice] as [UnitPrice],
[$Table].[TaxRate] as [TaxRate],
[$Table].[TotalExcludingTax] as [TotalExcludingTax],
[$Table].[TaxAmount] as [TaxAmount],
[$Table].[Profit] as [Profit],
[$Table].[TotalIncludingTax] as [TotalIncludingTax],
[$Table].[TotalDryItems] as [TotalDryItems],
[$Table].[TotalChillerItems] as [TotalChillerItems],
[$Table].[LineageKey] as [LineageKey]
from [wwi].[FactSale] as [$Table]
)
)
AS [t0]
GROUP BY [t0].[SalespersonKey]
)
AS [MainTable]

```

Close

## 6. Exercise 4: High-Performance Analysis with Azure Synapse Dedicated SQL Pools

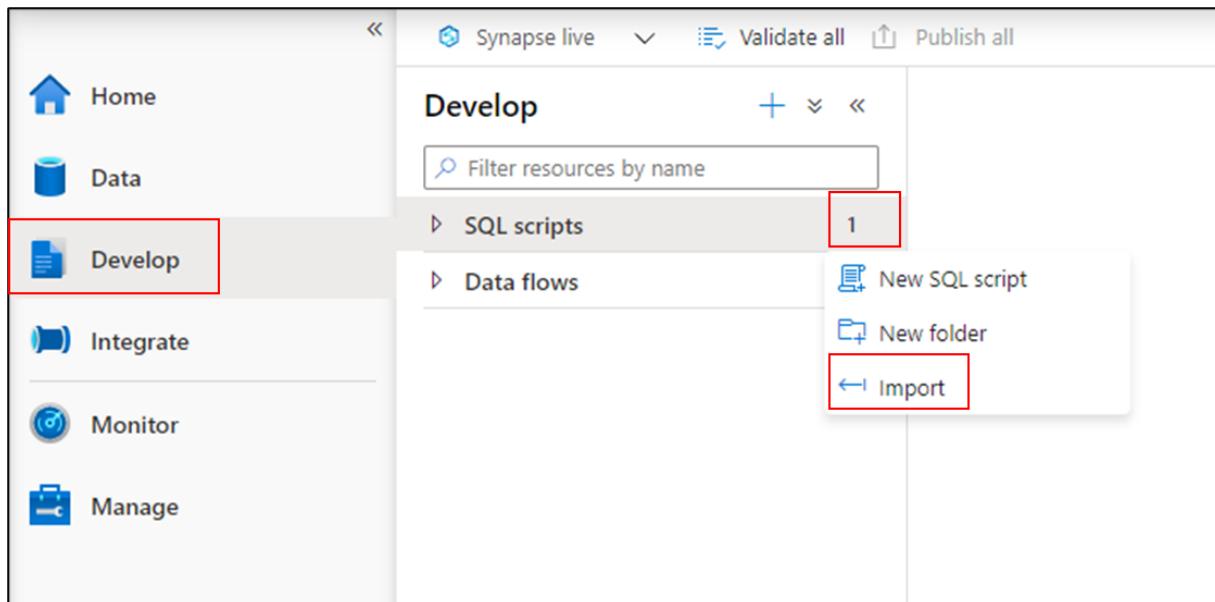
이 연습에서는 쿼리 및 차트 시각화를 사용하여 고객 세부 정보를 이해하려고합니다. 또한 다양한 쿼리의 성능을 탐색합니다.

SQL 데이터웨어 하우스는 오랫동안 데이터 플랫폼의 중심이었습니다. 최신 데이터웨어 하우스는 현재 데이터 볼륨에 관계없이 고성능, 분산 및 관리 워크로드를 제공 할 수 있습니다.

Azure Synapse Analytics의 Azure Synapse SQL 풀은 이전 Azure SQL 데이터웨어 하우스를 새롭게 구현 한 것입니다. 모든 최신 SQL 데이터웨어 하우징 기능을 제공하는 동시에 다른 모든 Synapse 서비스와의 고급 통합을 활용합니다..

### 6.1. Task 1: Use a dedicated SQL pool query to understand a dataset

1. 왼쪽 메뉴에서 **Develop** Hub로 이동하여 **SQL scripts** 오른쪽 ... 버튼을 클릭하고 **Import**를 선택 하여 [Exercise 4 - Analyze Transactions.sql](#) 파일을 Import 합니다.



2. **Connect to**를 **SQLPool01**을 선택하고 **Run** 버튼을 클릭하여 쿼리를 실행합니다.



```
1 SELECT
2     CustomerKey,
3     SUM(Quantity) as Quantity
4 FROM
5     wwi.FactSale
6 GROUP BY
7     CustomerKey
```

3. 쿼리 결과의 View 항목이 Chart를 클릭하고 아래 내용으로 설정합니다.

- ✓ Chart type: **Column** 선택
- ✓ Category column: **(none)** 선택
- ✓ Legend(Series) columns: **Customerkey** 선택



## 6.2. Task 2: Investigate query performance and table design

1. **Exercise 4 - Investigate query performance.sql** 파일을 Import하고 **Connect to** 를 **SQLPool01**로 변경 합니다.

```
1  SELECT count(*) from wwi_perf.FactSale_Slow
2
3  SELECT count(*) from wwi_perf.FactSale_Fast
4
5  SELECT
6      FS.CustomerKey
7      ,MIN(FS.Quantity) as MinQuantity
8      ,MAX(FS.Quantity) as MaxQuantity
9      ,AVG(FS.TaxRate) as AvgTaxRate
10     ,AVG(FS.TaxAmount) as AvgTaxAmount
11     ,AVG(FS.TotalExcludingTax) as AverageSaleWithoutTax
12     ,AVG(FS.TotalIncludingTax) as AverageSaleWithTax
13     ,COUNT(DISTINCT FS.StockItemKey) as DistinctStockItems
14     ,COUNT(DISTINCT DC.Country) as DistinctCountries
15  FROM
16      wwi_perf.FactSale_Slow FS
17      join wwi.DimCity DC ON
```

2. 첫번째 라인을 블록으로 지정 후 Run 버튼을 클릭하여 쿼리를 수행합니다.

83,449,830건이 Count 됨을 확인합니다.

동일하게 두번째 라인을 블록으로 지정 후 쿼리를 수행합니다.

결과는 동일하게 83,449,830건이 Count 됩니다.

```
1 SELECT count(*) from wwi_perf.FactSale_Slow
2
3 SELECT count(*) from wwi_perf.FactSale_Fast
4
5 SELECT
6     FS.CustomerKey
7     ,MIN(FS.Quantity) as MinQuantity
8     ,MAX(FS.Quantity) as MaxQuantity
9     ,AVG(FS.TaxRate) as AvgTaxRate
10    ,AVG(FS.TaxAmount) as AvgTaxAmount
11    ,AVG(FS.TotalExcludingTax) as AverageSaleWithoutTax
```

Results Messages

View Table Chart Export results

Search

(No Column Name)

83449830

3. 3번째 쿼리 (Line 5~20)를 블록 지정 후 쿼리를 실행 합니다.

3~4회 쿼리 수행을 반복하고 쿼리 수행 시간을 확인합니다.

```
4
5 SELECT
6     FS.CustomerKey
7     ,MIN(FS.Quantity) as MinQuantity
8     ,MAX(FS.Quantity) as MaxQuantity
9     ,AVG(FS.TaxRate) as AvgTaxRate
10    ,AVG(FS.TaxAmount) as AvgTaxAmount
11    ,AVG(FS.TotalExcludingTax) as AverageSaleWithoutTax
12    ,AVG(FS.TotalIncludingTax) as AverageSaleWithTax
13    ,COUNT(DISTINCT FS.StockItemKey) as DistinctStockItems
14    ,COUNT(DISTINCT DC.Country) as DistinctCountries
15
16 FROM
17     wwi_perf.FactSale_Slow FS
18     join wwi.DimCity DC ON
19         DC.CityKey = FS.CityKey
20     GROUP BY
21         FS.CustomerKey
22
23 SELECT
```

Results Messages

View Table Chart Export results

Search

| CustomerKey | MinQuantity | MaxQuantity | AvgTaxRate | AvgTaxAmount | AverageSaleWi... | Average |
|-------------|-------------|-------------|------------|--------------|------------------|---------|
| 221         | 1           | 324         | 14.955089  | 120.229161   | 805.417964       | 925.647 |
| 1           | 1           | 252         | 14.987684  | 112.816945   | 752.449261       | 865.266 |
| 381         | 1           | 324         | 14.942129  | 126.328888   | 845.895833       | 972.224 |

00:00:04 Query executed successfully.

4. 4번째 쿼리 (Line 22~37)를 블록 지정 후 쿼리를 수행 합니다.

3~4회 쿼리 수행을 반복하고 쿼리 수행 시간을 확인합니다.

```
20    FS.CustomerKey
21
22    SELECT
23        FS.CustomerKey
24        ,MIN(FS.Quantity) as MinQuantity
25        ,MAX(FS.Quantity) as MaxQuantity
26        ,AVG(FS.TaxRate) as AvgTaxRate
27        ,AVG(FS.TaxAmount) as AvgTaxAmount
28        ,AVG(FS.TotalExcludingTax) as AverageSaleWithoutTax
29        ,AVG(FS.TotalIncludingTax) as AverageSaleWithTax
30        ,COUNT(DISTINCT FS.StockItemKey) as DistinctStockItems
31        ,COUNT(DISTINCT DC.Country) as DistinctCountries
32    FROM
33        wwi_perf.FactSale_Fast FS
34        join wwi.DimCity DC ON
35            DC.CityKey = FS.CityKey
36    GROUP BY
37        FS.CustomerKey
```

Results Messages

View Table Chart Export results

Search

| CustomerKey | MinQuantity | MaxQuantity | AvgTaxRate | AvgTaxAmount | AverageSaleWi... | Average... |
|-------------|-------------|-------------|------------|--------------|------------------|------------|
| 221         | 1           | 324         | 14.955089  | 120.229161   | 805.417964       | 925.64     |
| 391         | 1           | 240         | 14.952229  | 106.924044   | 716.311624       | 823.23     |
| 86          | 1           | 360         | 14.956395  | 139.184476   | 929.486191       | 1068.6     |

00:00:01 Query executed successfully.

5. 동일한 데이터가 저장 되어있는 두테이블의 쿼리속도가 차이나는 이유는 두 테이블의 분산 디자인이 다르기 때문입니다.

FactSale\_Slow 는 Round\_Robin 분산이고, FactSale\_Fast는 Hash 분산으로 생성 되었습니다.

```
CREATE TABLE [wwi_perf].[FactSale_Slow]
(
    [SaleKey] [bigint] NOT NULL,
    [CityKey] [int] NOT NULL,
    [CustomerKey] [int] NOT NULL,
    [BillToCustomerKey] [int] NOT NULL,
    [StockItemKey] [int] NOT NULL,
    [InvoiceDateKey] [date] NOT NULL,
    [DeliveryDateKey] [date] NULL,
    [SalespersonKey] [int] NOT NULL,
    [WWIInvoiceID] [int] NOT NULL,
    [Description] [nvarchar](100) NOT NULL,
    [Package] [nvarchar](50) NOT NULL,
    [Quantity] [int] NOT NULL,
    [UnitPrice] [decimal](18,2) NOT NULL,
    [TaxRate] [decimal](18,3) NOT NULL,
    [TotalExcludingTax] [decimal](18,2) NOT NULL,
    [TaxAmount] [decimal](18,2) NOT NULL,
    [Profit] [decimal](18,2) NOT NULL,
    [TotalIncludingTax] [decimal](18,2) NOT NULL,
    [TotalDryItems] [int] NOT NULL,
    [TotalChillerItems] [int] NOT NULL,
    [LineageKey] [int] NOT NULL
)
WITH
(
    DISTRIBUTION = ROUND_ROBIN,
    CLUSTERED COLUMNSTORE INDEX
)
GO
```

```
CREATE TABLE [wwi_perf].[FactSale_Fast]
(
    [SaleKey] [bigint] NOT NULL,
    [CityKey] [int] NOT NULL,
    [CustomerKey] [int] NOT NULL,
    [BillToCustomerKey] [int] NOT NULL,
    [StockItemKey] [int] NOT NULL,
    [InvoiceDateKey] [date] NOT NULL,
    [DeliveryDateKey] [date] NULL,
    [SalespersonKey] [int] NOT NULL,
    [WWIInvoiceID] [int] NOT NULL,
    [Description] [nvarchar](100) NOT NULL,
    [Package] [nvarchar](50) NOT NULL,
    [Quantity] [int] NOT NULL,
    [UnitPrice] [decimal](18,2) NOT NULL,
    [TaxRate] [decimal](18,3) NOT NULL,
    [TotalExcludingTax] [decimal](18,2) NOT NULL,
    [TaxAmount] [decimal](18,2) NOT NULL,
    [Profit] [decimal](18,2) NOT NULL,
    [TotalIncludingTax] [decimal](18,2) NOT NULL,
    [TotalDryItems] [int] NOT NULL,
    [TotalChillerItems] [int] NOT NULL,
    [LineageKey] [int] NOT NULL
)
WITH
(
    DISTRIBUTION = HASH ( [CustomerKey] ),
    CLUSTERED COLUMNSTORE INDEX
)
GO
```

## 7. Exercise 5 - Data Science with Azure Synapse Spark

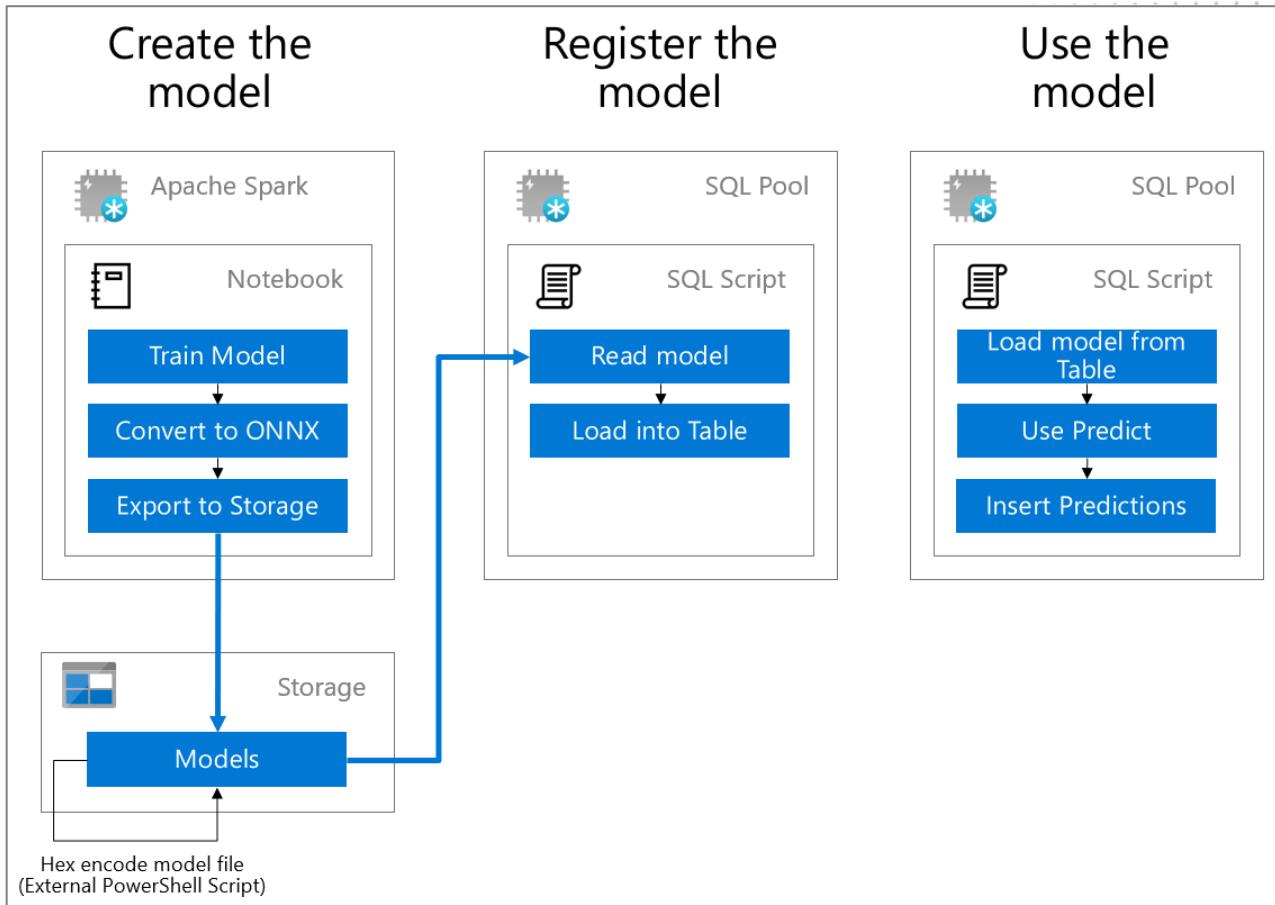


그림 출처: <https://github.com/solliancenet/azure-synapse-analytics-day/blob/master/media/ex05-model-registration-process.png>

이 연습에서는 Python을 이용하여 생성된 모델을 활용하여 Azure Synapse Analytics 전용 SQL 풀에서 T-SQL PREDICT 문을 사용하여 예측을 수행합니다.

Azure Synapse Analytics는 전용 SQL 풀에서 직접 학습 된 모델 (ONNX 형식) 사용을 지원합니다. 이것이 실제로 의미하는 바는 데이터 엔지니어가 이러한 모델을 사용하여 SQL 풀 데이터베이스 테이블에 저장된 테이블 형식 데이터에 대한 예측을 수행하는 T-SQL 쿼리를 작성할 수 있다는 것입니다.

모델은 Scikit-Learn 데이터분석 라이브러리를 사용하여 Synapse Notebook 또는 Azure Databricks에서 Python 기반으로 학습 할 수 있습니다. Spark ML에 포함 되어있는 ML 라이브러리를 사용하여 모델을 생성, 학습 시킬 수도 있습니다.

모델은 Azure Machine Learning 자동화 ML을 사용하는 등 다른 접근 방식을 사용하여 학습 할 수도 있습니다. 주요 요구 사항은 모델 형식이 ONNX에서 지원 되어야 한다는 것입니다.

## 7.1. Task 1: Making predictions with a trained model

1. Data Hub로 이동하여 **asastore01** 아래 **models** 컨테이너 아래 **onnx, hex** 두개의 폴더를 추가 합니다.

The screenshot shows the Azure Data Lake Storage Gen2 Data Hub interface. On the left, there's a navigation pane with 'Workspace' selected. The main area shows a file browser with the path 'Exercise 5 - Predict with...'. The 'models' folder is expanded, revealing 'hex' and 'onnx' subfolders, which are both highlighted with red boxes. Other visible items include 'Azure Data Lake Storage Gen2', 'asaworkspace0000', 'asadatalake01', and 'staging'.

2. hex 폴더로 이동하여 Upload 버튼을 클릭하여 **model.onnx.hex** 파일을 업로드 합니다.

The screenshot shows the 'hex' folder within the 'models' folder. The 'Upload' button at the top of the list view is highlighted with a red box. A single file, 'model.onnx.hex', is listed below it, showing its last modified date as '2021. 1. 26. 오전 10:33:35'. The path 'models > hex' is shown at the top of the list view.

3. hex로 변환 된 모델을 Synapse external table을 이용하여 등록 합니다. 먼저 Develop Hub로 이동하여 SQL scripts 의 Import를 클릭하여 3개의 파일을 하나씩 Import 합니다.

Import 파일 : [Exercise 5 - Register model.sql](#)

[Exercise 5 - Create Sample Data for Predict.sql](#)

[Exercise 5 - Predict with model.sql](#)

The screenshot shows the Develop Hub interface. In the 'SQL scripts' section, three specific files are selected and highlighted with a red box: 'Exercise 5 - Create Sample Data for ...', 'Exercise 5 - Predict with model', and 'Exercise 5 - Register model'. To the right of these, the 'Import' button is also highlighted with a red box. The 'models' folder is visible in the background.

4. Exercise 5 – Register model을 열고 9 Line의 <Storage Access Key>와 15 Line의 <Storage Name>을 사용자의 스토리지 계정 (asablobstorageXXXX)의 Access Key와 스토리지 계정 이름 asablobstorageXXXX로 수정합니다.

스토리지 계정의 Access Key는 Azure Portal을 통해 storage계정 리소스에서 왼쪽 엑세스 키 메뉴를 통해 확인 할 수 있습니다.

The screenshot shows the Azure Storage Account settings for 'asablobstorage0000'. In the 'Access Keys' section, two keys are listed: 'key1' and 'key2'. The 'key1' value is redacted with a large red box. Below each key is a 'Secret' field where the key values are also redacted.

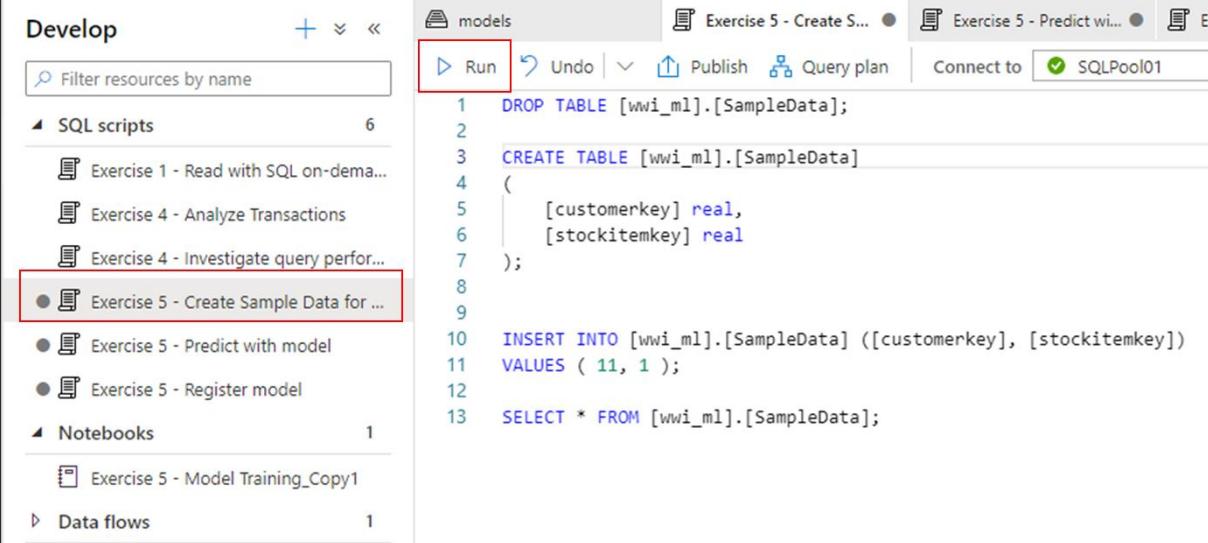
5. 수정 된 쿼리를 차례로 수행하여 모델을 등록 합니다. 쿼리는 Storage 계정의 Hex 모델 파일을 External Table로 선언 하여 MLModel 테이블에 저장합니다.

```

1 -- Use poly to load model into the model table
2 CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'fQv2fKq#FN7r'
3
4 -- Create a database scoped credential with Azure storage account key (not a Shared Access S
5
6 CREATE DATABASE SCOPED CREDENTIAL StorageCredential
7 WITH
8 IDENTITY = 'SHARED ACCESS SIGNATURE'
9 , SECRET = 'MciQyJPYb9x/z9L+tMDAcBhQkT2XHSJtcy5V/mIZg9ieHN4EHRccp6/2CTvxECsXKNdtNsE00bWcMj9
10 ;
11
12 -- Create an external data source with CREDENTIAL option.
13 CREATE EXTERNAL DATA SOURCE ModelStorage
14 WITH
15 ( LOCATION = 'abfss://models@asablobstorage0000.dfs.core.windows.net'
16 , CREDENTIAL = StorageCredential
17 , TYPE = HADOOP
18 )
19 ;
20 CREATE EXTERNAL FILE FORMAT csv
21 WITH (
22 FORMAT_TYPE = DELIMITEDTEXT,
23 FORMAT_OPTIONS (
24 FIELD_TERMINATOR = ',',
25 STRING_DELIMITER = '',
26 DATE_FORMAT = '',
27 USE_TYPE_DEFAULT = False

```

6. Exercise 5 – create Sample Data for Predict 쿼리를 수행하여 예측에 사용할 입력 데이터를 생성합니다.



```

Develop + <> models Exercise 5 - Create S... Exercise 5 - Predict wi...
Filter resources by name
SQL scripts 6 Undo Publish Query plan Connect to SQLPool01
Exercise 1 - Read with SQL on-dema...
Exercise 4 - Analyze Transactions
Exercise 4 - Investigate query perfor...
Exercise 5 - Create Sample Data for ...
Exercise 5 - Predict with model
Exercise 5 - Register model
Notebooks 1
Exercise 5 - Model Training_Copy1
Data flows 1

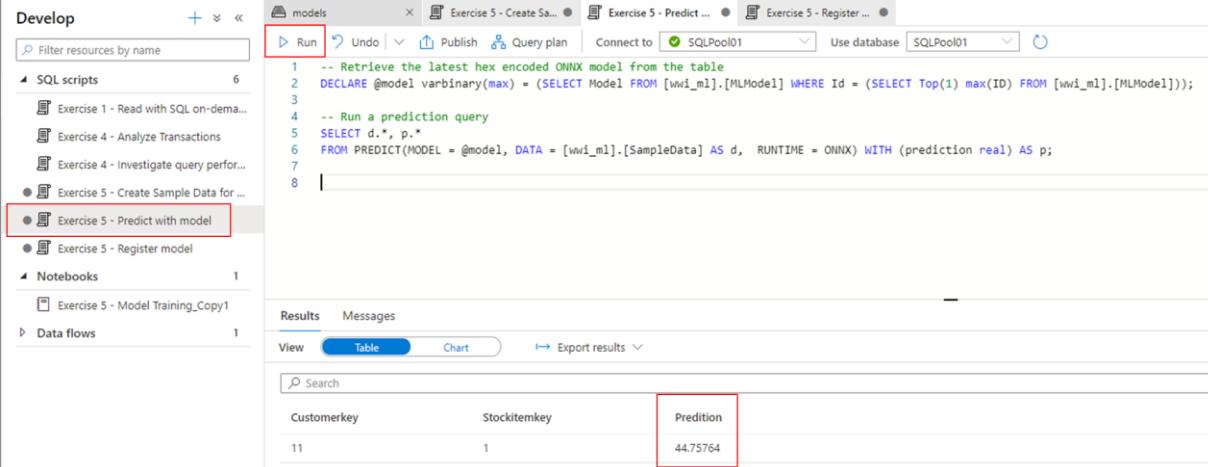
```

```

1 DROP TABLE [wwi_ml].[SampleData];
2
3 CREATE TABLE [wwi_ml].[SampleData]
4 (
5     [customerkey] real,
6     [stockitemkey] real
7 );
8
9
10 INSERT INTO [wwi_ml].[SampleData] ([customerkey], [stockitemkey])
11 VALUES ( 11, 1 );
12
13 SELECT * FROM [wwi_ml].[SampleData];

```

7. Exrcice 5 – Predict with model 쿼리 수행 합니다. 입력 한 데이터에 대한 예측 결과를 확인합니다.



```

Develop + <> models Exercise 5 - Create S... Exercise 5 - Predict ...
Filter resources by name
SQL scripts 6 Undo Publish Query plan Connect to SQLPool01 Use database SQLPool01
Exercise 1 - Read with SQL on-dema...
Exercise 4 - Analyze Transactions
Exercise 4 - Investigate query perfor...
Exercise 5 - Create Sample Data for ...
Exercise 5 - Predict with model
Exercise 5 - Register model
Notebooks 1
Exercise 5 - Model Training_Copy1
Data flows 1

```

```

1 -- Retrieve the latest hex encoded ONNX model from the table
2 DECLARE @model varbinary(max) = (SELECT Model FROM [wwi_ml].[MLModel] WHERE Id = (SELECT Top(1) max(ID) FROM [wwi_ml].[MLModel]));
3
4 -- Run a prediction query
5 SELECT d.* , p.*
6 FROM PREDICT(MODEL = @model, DATA = [wwi_ml].[SampleData] AS d, RUNTIME = ONNX) WITH (prediction real) AS p;
7
8

```

| Customerkey | Stockitemkey | Prediction |
|-------------|--------------|------------|
| 11          | 1            | 44.75764   |

## 7.2. Task 2: Examining the model training and registration process

1. Notebooks의 Import를 클릭하여 **Exercise 5 - Model Training.ipynb** 파일을 Import 합니다.

```

df = spark.read.load('abfss://dev@asaprimarystorage0000.dfs.core.windows.net/bronze/ww1-factsale.csv', format="csv")
## If header exists uncomment line below
3 , header=True, sep="|"
4 )

```

The following cells load the source CSV file into a Spark DataFrame and create a temporary view that can be used to query the data with Spark SQL.

```

Cell 4
[2] 1 df.createOrReplaceTempView("facts")

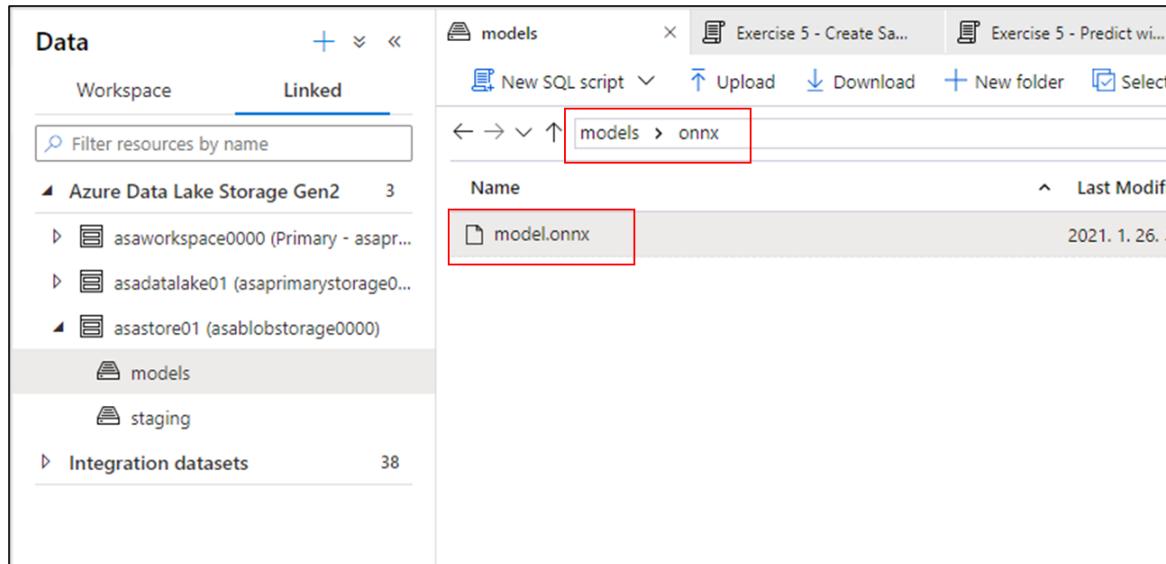
Cell 5
[3] 1 display(spark.sql("SELECT * FROM facts WHERE `Customer Key` == '11' ORDER BY `Stock Item Key`"))

```

| Sale Key | City Key | Customer Key | Bill To Customer... | Stock Item Key | Invoice Date Key | Delivery Date Key | Salesperson Key |
|----------|----------|--------------|---------------------|----------------|------------------|-------------------|-----------------|
| 11869077 | 100413   | 11           | 1                   | 1              | 2012-12-20       | 2012-12-21        | 156             |

2. 위의 Notebook은 Data Lake (Primary Storage)의 CSV 파일 (ww1-factsale.csv) 데이터를 이용하여 모델을 학습하고, ONNX 모델로 변환하여 Data Lake (Blob Storage)에 Upload 합니다.

ONNX 모델 원본 파일은 models 컨테이너 아래 onnx 폴더에 ONNX모델을 Hex로 인코딩한 파일은 hex 폴더에 Upload 합니다.



3. Task 1의 모델 등록 작업을 수행하여 모델을 테이블에 등록하고 T-SQL Predict를 이용하여 예측을 수행 합니다.

금일 실습에 사용된 실습 내용 및 자료는 <https://github.com/solliancenet/azure-synapse-analytics-day>를 기반으로 수정 및 보완이 되어 작성 되었습니다.