자습서 - 스크립팅의 다음 단계

Qlik Sense®
May 2024

Contents

1 자습서 시작! 5
1.1 학습 내용 5
1.2 학습 대상 5
1.3 패키지 콘텐츠 5
1.4 이 자습서의 단원 6
1.5 추가 자료 및 리소스 6
2 LOAD 및 SELECT 문 7
3 데이터 변환 8
3.1 Crosstable 접두사 사용 8
Crosstable 접두사 8
메모리 캐시 정리 12
3.2 Join 및 Keep을 사용한 테이블 결합 12
Join 13
Join 사용 13
Keep 16
Inner 16
Left 18
Right 19
3.3 인터 레코드 함수 사용: Peek, Previous 및 Exists 20
Peek() 20
Previous() 21
Exists() 21
Peek() 및 Previous() 사용 21
Exists() 사용 24
3.4 간격 일치 및 반복 로드 27
IntervalMatch() 접두사 사용 27
While 루프 및 반복 로드 IterNo() 사용 29
열린 간격 및 닫힌 간격 31
4 데이터 정리 32
4.1 매핑 테이블 32
규칙: 32
4.2 Mapping 함수 및 문 32
4.3 Mapping 접두사 32
4.4 ApplyMap() 함수 33
4.5 MapSubstring() 함수 35
4.6 Map ... Using 37
5 계층 구조 데이터 처리 39
5.1 Hierarchy 접두사 39
5.2 HierarchyBelongsTo 접두사 40
인증 41
6 QVD 파일 44
6.1 QVD 파일 만들기 45
Store 45
6.2 QVD 파일에서 데이터 읽기 46
Buffer 47

6.3 감사합니다. 50

1 자습서 시작!

자습서에 오신 것을 환영합니다. 이 자습서에서는 Qlik Sense에서 고급 스크립팅을 수행하는 방법에 대해 소개합니다.

스크립팅의 기본 사항에 익숙해졌으면 데이터를 Qlik Sense에 로드할 때 보다 정교한 작업을 수행할 수 있습니다. 예를 들어 교차 표를 사용한 데이터 변환, 데이터 정리, QVD 파일로 알려진 Qlik 데이터 파일에서 데이터를 만들고 로드하는 등의 작업을 수행합니다.

1.1 학습 내용

After completing this tutorial, you should be comfortable with loading data using some of the more advanced scripting functions in Qlik Sense.

1.2 학습 대상

Qlik Sense의 스크립팅 기본 사항에 익숙해야 합니다. 다시 말해 스크립트를 사용하여 데이터를 로드하고 조작할 수 있어야 합니다.

이 작업을 아직 수행하지 않은 경우 초보자를 위한 스크립팅 자습서를 완료하는 것이 좋습니다.

데이터 로드 편집기에 액세스하고 Qlik Sense Enterprise on Windows에서 데이터를 로드할 수 있어야 합니다.

해당 지침은 Qlik Sense Cloud Business에도 일반적으로 적용됩니다.

1.3 패키지 콘텐츠

다운로드한 zip 패키지에는 자습서를 완료하는 데 필요한 다음 데이터 파일이 포함되어 있습니다.

- *Cutlery.xlsx*
- *Data.xlsx*
- *Events.txt*
- *Employees.xlsx*
- *Intervals.txt*
- *Product.xlsx*
- *Salesman.xlsx*
- *Transactions.csv*
- *Winedistricts.txt*

또한 패키지에는 *Advanced Scripting Tutorial* 앱의 복사본도 포함되어 있습니다. 앱의 추가 스크립트 섹션에는 이 자습서에서 만드는 다른 앱의 스크립트가 포함되어 있습니다. 허브로 앱을 업로드할 수 있습니다.

학습 효과를 최대화하려면 자습서에 설명된 대로 직접 앱을 작성하는 것이 좋습니다. 또한 작업할 데이터를 로드하기 위해 자습서에 설명된 대로 데이터 파일을 업로드하고 연결해야 합니다.

문제가 있는 경우 앱을 통해 문제를 해결할 수 있습니다. 각 단원과 관련된 스크립트 세그먼트를 표시했습니다.

1.4 이 자습서의 단원

이 자습서를 완료하려면 Qlik Sense를 사용하는 환경에 따라 3-4시간이 걸립니다. 항목은 순서대로 완료되도록 설계되었습니다. 그러나 중단했다가 언제든지 다시 돌아올 수 있습니다. 다행히도 테스트는 없습니다.

데이터 변환
Crosstable 접두사 사용
Join 및 Keep을 사용한 테이블 결합
인터 레코드 함수 사용: Peek, Previous 및 Exists
간격 일치 및 반복 로드
데이터 정리
계층 구조 데이터 처리
QVD 파일

1.5 추가 자료 및 리소스

- Qlik에서는 보다 자세한 정보를 알아볼 수 있도록 다양한 리소스를 제공합니다.
- Qlik 온라인 도움말을 사용할 수 있습니다.
- Qlik Continuous Classroom에서 교육(무료 온라인 과정 포함)이 제공됩니다.
- Qlik Community에서 토론 포럼, 블로그 등을 찾을 수 있습니다.

2 LOAD 및 SELECT 문

LOAD 및 SELECT 문을 사용하여 데이터를 Qlik Sense에 로드할 수 있습니다. 이러한 각 문은 내부 테이블을 생성합니다. LOAD는 파일에서 데이터를 로드하는 데 사용되고 SELECT는 데이터베이스에서 데이터를 로드하는 데 사용됩니다.

이 자습서에서는 파일의 데이터를 사용하므로 LOAD 문을 사용합니다.

또한 선행 LOAD를 사용하여 로드된 데이터의 내용을 편집할 수도 있습니다. 예를 들어, 필드 이름을 바꾸려면 LOAD 문을 사용해야 합니다. SELECT 문은 필드 이름을 변경할 수 없습니다.

Qlik Sense에 데이터를 로드할 때 다음과 같은 규칙이 적용됩니다.

- Qlik Sense는 LOAD 또는 SELECT 문으로 생성된 테이블을 구분하지 않습니다. 이는 여러 테이블이 로드될 때 테이블이 LOAD 또는 SELECT 문 또는 이 둘의 조합으로 로드되어도 상관이 없음을 의미합니다.
- 문이나 데이터베이스에 있는 원래 테이블 내의 필드 순서는 Qlik Sense 논리에서 중요하지 않습니다.
- 필드 이름은 대/소문자를 구분하며 데이터 테이블 간에 연결을 설정하는 데 사용됩니다. 따라서 원하는 데이터 모델을 연결하기 위해 로드 스크립트에서 필드 이름을 변경해야 합니다.

3 데이터 변환

앱에서 데이터를 사용하기 전에 데이터 로드 편집기에서 데이터를 변환하고 조작할 수 있습니다.

데이터 조작의 장점 중 하나는 테이블에서 몇 개의 선택한 열만 로드하는 등, 파일에서 데이터의 하위 집합만 로드하도록 선택하여 더 효과적으로 데이터를 처리할 수 있다는 점입니다. 또한 데이터를 여러 번 로드하여 원시 데이터를 여러 개의 새 논리 테이블로 분할할 수도 있습니다. 둘 이상의 소스에서 데이터를 로드하고 Qlik Sense에서 테이블 하나로 병합하는 것도 가능합니다.

다음 연습에서는 Crosstable 접두사를 사용하여 데이터를 로드하는 방법을 보여 줍니다. 테이블 조인 방법, Peek 및 Previous와 같은 인터 레코드 함수 사용 방법 및 While Load를 사용하여 동일한 행을 여러 번 로드하는 방법도 배우게 됩니다.

3.1 Crosstable 접두사 사용

교차 표는 두 개의 머리글 데이터 직교 목록 사이의 값을 표로 나타내는 일반적인 유형의 표입니다. 데이터의 교차 표가 있는 경우에는 언제든지 Crosstable 접두사를 사용하여 데이터를 변환하고 원하는 필드를 만들 수 있습니다.

Crosstable 접두사

다음 *Product* 테이블에는 월별로 열 하나와 제품별로 행이 하나 있습니다.

Product 테이블

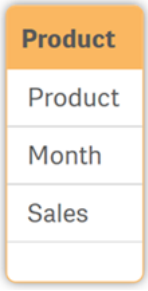
Product	Jan 2014	Feb 2014	Mar 2014	Apr 2014	May 2014	Jun 2014
A	100	98	100	83	103	82
B	284	279	297	305	294	292
C	50	53	50	54	49	51

테이블을 로드하면 *Product*에 해당하는 필드 하나와 각 월에 해당하는 필드 하나가 있는 테이블이 생성됩니다.

Product 필드와 각 월에 해당하는 필드 하나가 있는 Product 테이블

이 데이터를 분석하려고 한다면 하나의 필드에 모든 숫자가 있고 다른 필드에는 모든 월이 있는 편이 훨씬 사용하기 좋습니다. 이 경우, 각 범주(*Product, Month, Sales*)에 대해 하나의 열이 있는 3열 테이블이 좋습니다.

Product, Month 및 Sales 필드가 있는 Product 테이블



Crosstable 접두사는 *Month*에 대해 열이 하나 있고 *Sales*에 대해서는 다른 열이 있는 테이블로 데이터를 변환합니다. 이를 표현하는 다른 방법은 필드 이름을 지정하고 이를 필드 값으로 변환하는 것입니다.

다음과 같이 하십시오.

1. 새 앱을 만들고 *Advanced Scripting Tutorial*로 지정합니다.
2. **데이터 로드 편집기**에서 새 스크립트 섹션을 추가합니다.
3. 섹션 이름을 *Product*로 지정합니다.
4. 오른쪽 메뉴의 **AttachedFiles**에서 **데이터 선택**을 클릭합니다.
5. 업로드한 다음 *Product.xlsx*를 선택합니다.
6. **데이터 선택** 창에서 *Product* 테이블을 선택합니다.

   > ***필드 이름** 아래에서, 데이터를 로드할 때 테이블 필드의 이름을 포함하도록 **포함된 필드 이름**이 선택되어 있는지 확인합니다.*

7. **스크립트 삽입**을 클릭합니다.
   스크립트는 다음과 같이 표시되어야 합니다.

```
LOAD
    Product,
    "Jan 2014",
    "Feb 2014",
    "Mar 2014",
    "Apr 2014",
    "May 2014",
    "Jun 2014"
FROM [lib://AttachedFiles/Product.xlsx]
(ooxml, embedded labels, table is Product);
```

8. **데이터 로드**를 클릭합니다.
9. **데이터 모델 뷰어**를 엽니다. 데이터 모델은 다음과 같이 표시됩니다.

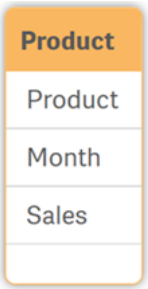
   Product 필드와 각 월에 해당하는 필드 하나가 있는 Product 테이블

10. **데이터 로드 편집기**에서 *Product* 탭을 클릭합니다.
11. LOAD 문 위에 다음을 입력합니다.

    ```
    CrossTable(Month, Sales)
    ```

12. **데이터 로드**를 클릭합니다.
13. **데이터 모델 뷰어**를 엽니다. 데이터 모델은 다음과 같이 표시됩니다.

    Product, Month 및 Sales 필드가 있는 Product 테이블

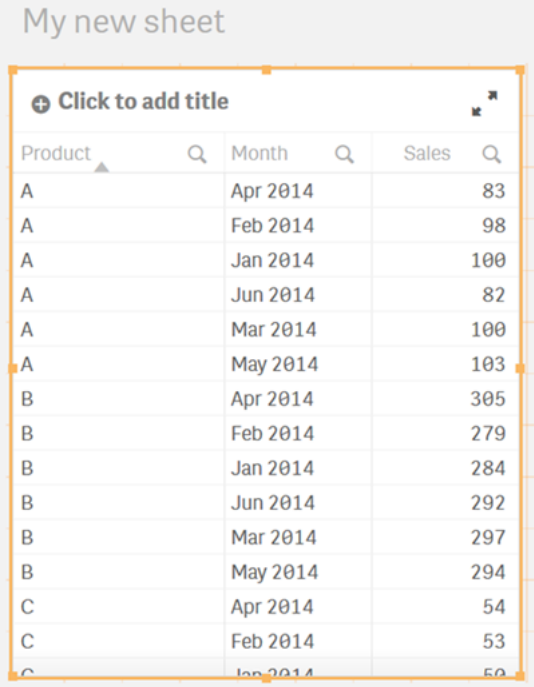


일반적으로 입력 데이터에는 한정자 필드 및 내부 키(위의 예에서는 *Product*)로 하나의 열만 있습니다. 그러나 여러 개를 가질 수도 있습니다. 그런 경우, 모든 한정자 필드가 LOAD 문의 특성 필드 앞에

나열되어야 하며 Crosstable 접두사에 대한 세 번째 매개 변수를 사용하여 한정하는 필드의 수를 정의해야 합니다. Crosstable 키워드 앞에 선행 LOAD 또는 접두사는 올 수 없습니다. 그러나 자동 연결은 사용할 수 있습니다.

Qlik Sense의 테이블에서 데이터는 다음과 같이 표시됩니다.

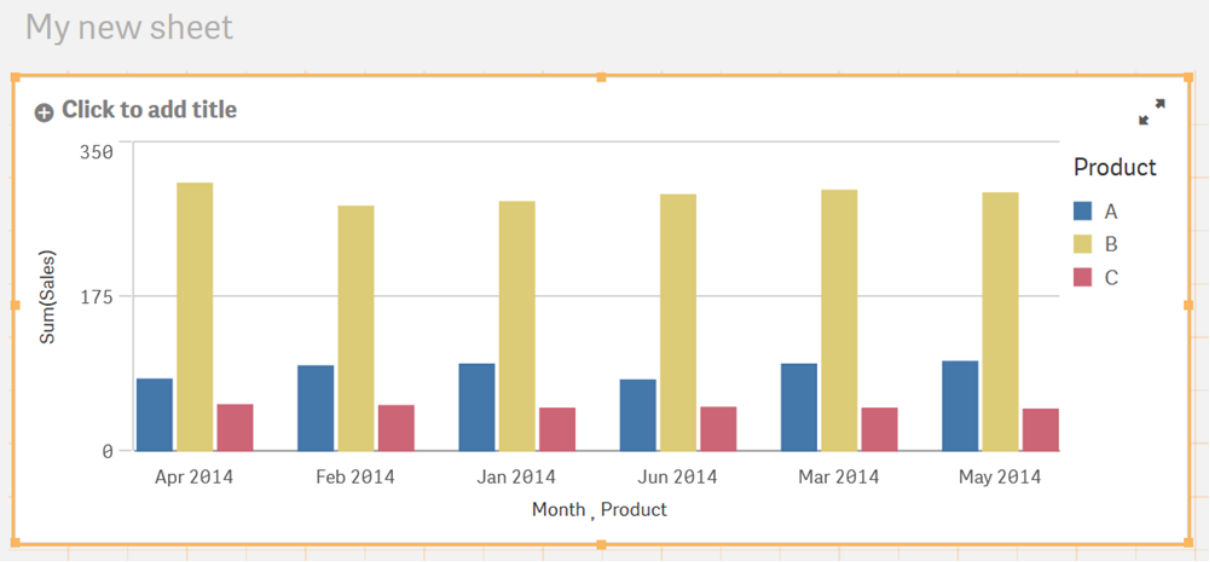
Crosstable 접두사를 사용하여 로드된 데이터를 보여 주는 테이블



Product	Month	Sales
A	Apr 2014	83
A	Feb 2014	98
A	Jan 2014	100
A	Jun 2014	82
A	Mar 2014	100
A	May 2014	103
B	Apr 2014	305
B	Feb 2014	279
B	Jan 2014	284
B	Jun 2014	292
B	Mar 2014	297
B	May 2014	294
C	Apr 2014	54
C	Feb 2014	53

이제 데이터를 사용하여 막대형 차트 등을 만들 수 있습니다.

Crosstable 접두사를 사용하여 로드된 데이터를 보여 주는 막대형 차트



Crosstable에 대한 자세한 내용은 Qlik Community에서 다음 블로그 게시물을 참조하십시오. The Crosstable Load(Crosstable Load) 동작은 QlikView 컨텍스트로 설명됩니다. 그러나 논리는 Qlik Sense에 동일하게 적용됩니다.

특성 필드에 대해서는 숫자 해석이 제대로 작동하지 않습니다. 즉, 열 헤더로 월이 있는 경우에는 자동으로 해석되지 않습니다. 이를 해결하려면 Crosstable 접두사를 사용하여 임시 테이블을 만들고 두 번째 통과를 실행하여 다음 예와 같이 해석하는 것입니다.

다음은 예일뿐입니다. Qlik Sense에서 완료할 연습은 수반되지 않습니다.

```
tmpData:
Crosstable (MonthText, Sales)
LOAD Product, [Jan 2014], [Feb 2014], [Mar 2014], [Apr 2014], [May 2014], [Jun 2014]
FROM ...

Final:
LOAD Product,
Date(Date#(MonthText,'MMM YYYY'),'MMM YYYY') as Month,
Sales
Resident tmpData;
Drop Table tmpData;
```

메모리 캐시 정리

사용자가 만든 테이블을 삭제하여 메모리 캐시를 정리할 수 있습니다. 이전 섹션에서처럼 임시 테이블에 로드한 경우 더 이상 필요하지 않으면 제거해야 합니다. 예:

```
DROP TABLE Table1, Table2, Table3, Table4;
DROP TABLES Table1, Table2, Table3, Table4;
```

필드도 삭제할 수 있습니다. 예:

```
DROP FIELD Field1, Field2, Field3, Field4;
DROP FIELDS Field1, Field2, Field3, Field4;
DROP FIELD Field1 from Table1;
DROP FIELDS Field1 from Table1;
```

표시된 대로 키워드 TABLE과 FIELD는 단수 또는 복수일 수 있습니다.

3.2 Join 및 Keep을 사용한 테이블 결합

조인은 두 테이블을 하나로 결합하는 연산입니다. 결과 테이블의 레코드는 원래 테이블에 있는 레코드의 조합이며, 결과 테이블 내의 일정한 조합에 기여하는 2개의 레코드가 하나 또는 여러 공통 필드에 대해 공통의 값을 가지는 이른바 자연 조인이라 불리는 조인이 일반적으로 수행됩니다. Qlik Sense의 경우 스크립트에서 조인을 수행하여 논리 테이블을 생성할 수 있습니다.

이미 스크립트에 있는 테이블을 조인할 수 있습니다. 그러면 Qlik Sense 논리에서는 별도의 테이블이 아니라 조인의 결과인 단일 내부 테이블을 보게 됩니다. 경우에 따라서는 이러한 작업 방식이 필요할 수 있지만 단점이 되는 경우도 있습니다.

- 로드한 테이블이 커져서 Qlik Sense가 느리게 작동할 수 있습니다.
- 일부 정보가 손실될 수 있습니다. 원래 테이블의 빈도(레코드의 수)를 더 이상 사용할 수 없게 될 수 있습니다.

테이블을 Qlik Sense에 저장하기 전에 두 테이블 중 하나 또는 둘 다 테이블 데이터의 교집합으로 축소하는 효과를 제공하는 Keep 기능은 명시적 조인을 사용해야 하는 경우의 수를 줄이기 위해 설계되었습니다.

이 설명서에서는 일반적으로 내부 테이블이 생성되기 전에 수행되는 조인에 대해 조인이라는 용어를 사용합니다. 내부 테이블이 생성된 후 수행되는 연결도 기본적으로 조인입니다.

Join

Join을 수행하는 가장 간단한 방법은 스크립트에서 join 접두사를 사용하여 내부 테이블을 다른 명명된 테이블이나 최근에 생성한 테이블과 조인하는 것입니다. 조인은 외부 조인이 되어 두 테이블에서 가능한 모든 값의 조합을 만듭니다.

```
LOAD  a, b, c from table1.csv;
join LOAD a, d from table2.csv;
```

결과 내부 테이블에는 a, b, c, d 필드가 생깁니다. 레코드 수는 두 테이블의 필드 값에 따라 달라집니다.

조인하려는 필드의 이름이 완전히 동일해야 합니다. 조인하려는 필드의 수는 임의입니다. 일반적으로 테이블은 하나 이상의 공통 필드를 가집니다. 공통 필드가 없을 경우 테이블의 카티션 곱이 생성됩니다. 모든 필드가 공통인 경우도 가능하지만 일반적으로 의미가 없습니다. 이전에 로드한 테이블의 이름을 Join 문에 지정하지 않은 경우 Join 접두사는 최근에 만든 테이블을 사용합니다. 따라서 두 문의 순서는 임의가 아닙니다.

Join 사용

Qlik Sense 스크립트 언어에서 명시적인 Join 접두사는 두 테이블의 완전 조인을 수행합니다. 결과는 한 테이블입니다. 이러한 조인으로 인해 큰 테이블이 될 수 있습니다.

다음과 같이 하십시오.

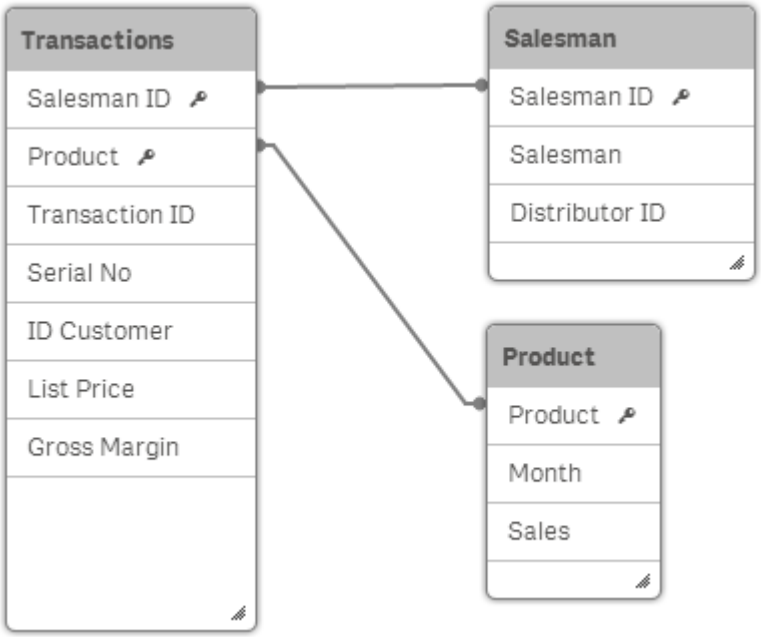
1. *Advanced Scripting Tutorial* 앱을 엽니다.
2. **데이터 로드 편집기**에서 새 스크립트 섹션을 추가합니다.
3. *Transactions* 섹션을 호출합니다.
4. 오른쪽 메뉴의 **AttachedFiles**에서 **데이터 선택**을 클릭합니다.
5. 업로드한 다음 *Transactions.csv*를 선택합니다.

***필드 이름** 아래에서, 데이터를 로드할 때 테이블 필드의 이름을 포함하도록 **포함된 필드 이름**이 선택되어 있는지 확인합니다.*

6. **데이터 선택** 창에서 **스크립트 삽입**을 클릭합니다.
7. 업로드한 다음 *Salesman.xlsx*를 선택합니다.
8. **데이터 선택** 창에서 **스크립트 삽입**을 클릭합니다.
   스크립트는 다음과 같이 표시되어야 합니다.

```
LOAD
    "Transaction ID",
    "Salesman ID",
    Product,
    "Serial No",
    "ID Customer",
    "List Price",
    "Gross Margin"
FROM [lib://AttachedFiles/Transactions.csv]
(txt, codepage is 28591, embedded labels, delimiter is ',', msq);

LOAD
    "Salesman ID",
    Salesman,
    "Distributor ID"
FROM [lib://AttachedFiles/Salesman.xlsx]
(ooxml, embedded labels, table is Salesman);
```

9. **데이터 로드**를 클릭합니다.
10. **데이터 모델 뷰어**를 엽니다. 데이터 모델은 다음과 같이 표시됩니다.
    데이터 모델: Transactions, Salesman 및 Product 테이블

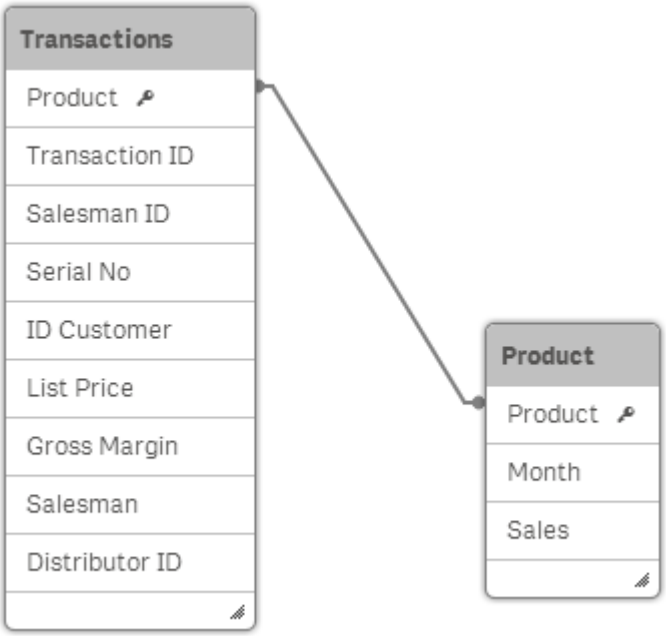


그러나 반드시 *Transactions* 및 *Salesman* 테이블을 별도로 구성할 필요는 없으며, 두 테이블을 조인하는 것이 더 나을 수 있습니다.

다음과 같이 하십시오.

1. 조인된 테이블의 이름을 설정하려면 첫 번째 LOAD 문 위에 다음 줄을 추가합니다.
   ```
   Transactions:
   ```
2. *Transactions* 및 *Salesman* 테이블을 조인하려면 두 번째 LOAD 문 위에 다음 줄을 추가합니다.
   ```
   Join(Transactions)
   ```
   스크립트는 다음과 같이 표시되어야 합니다.
   ```
   Transactions:
   LOAD
       "Transaction ID",
       "Salesman ID",
       Product,
       "Serial No",
       "ID Customer",
       "List Price",
       "Gross Margin"
   FROM [lib://AttachedFiles/Transactions.csv]
   (txt, codepage is 28591, embedded labels, delimiter is ',', msq);

   Join(Transactions)
   LOAD
       "Salesman ID",
       Salesman,
       "Distributor ID"
   FROM [lib://AttachedFiles/Salesman.xlsx]
   (ooxml, embedded labels, table is Salesman);
   ```
3. **데이터 로드**를 클릭합니다.
4. **데이터 모델 뷰어**를 엽니다. 데이터 모델은 다음과 같이 표시됩니다.

데이터 모델: Transactions 및 Product 테이블



이제 *Transactions* 및 *Salesman* 테이블의 모든 필드가 단일 *Transactions* 테이블로 결합됩니다.

Join을 사용하는 경우에 대해 자세히 알아보려면 Qlik Community에서 다음 블로그 게시물을 참조하십시오. To Join or not to Join *(조인하려면 또는 조인하지 않으려면),* Mapping as an Alternative to Joining *(조인에 대한 대안으로 매핑) 동작은 QlikView 컨텍스트로 설명됩니다. 그러나 논리는 Qlik Sense에 동일하게 적용됩니다.*

Keep

Qlik Sense의 주요 기능 중에는 테이블을 조인하는 대신 테이블을 연결하여 메모리 사용량 축소, 속도 향상, 그리고 엄청난 유연성 향상을 제공하는 기능이 있습니다. Keep 기능은 명시적 조인을 사용해야 하는 경우의 수를 줄이기 위해 설계되었습니다.

두 개의 LOAD 또는 SELECT 문 사이에 Keep 접두사를 사용하면 Qlik Sense에 저장하기 전에 하나 또는 두 테이블을 테이블 데이터의 교집합으로 축소합니다. Keep 접두사 앞에는 항상 Inner, Left 또는 Right 키워드 중 하나가 와야 합니다. 테이블에서 레코드의 선택은 해당하는 조인과 동일한 방법으로 수행됩니다. 그러나 두 테이블은 조인되지 않으며 Qlik Sense에 별도의 명명된 두 테이블로 저장됩니다.

Inner

데이터 로드 스크립트에서 Join 및 Keep 접두사 앞에 Inner 접두사가 올 수 있습니다.

이 접두사가 Join 앞에 사용될 경우 두 테이블 간에 내부 조인을 수행하도록 지정됩니다. 결과 테이블은 양쪽의 전체 데이터 셋이 포함된 두 테이블 간의 조합만 포함합니다.

Keep 앞에 사용할 경우 Qlik Sense에 저장되기 전에 두 테이블을 공통 교집합으로 축소하도록 지정됩니다.

이 예에서는 소스 테이블 *Table1* 및 *Table2*를 사용합니다.

이 경우는 예일뿐입니다. Qlik Sense에서 완료할 연습은 수반되지 않습니다.

Table 1

A	B
1	aa
2	cc
3	ee

Table2

A	C
1	xx
4	yy

Inner Join

먼저 테이블에 대한 Inner Join을 수행하여 두 테이블에 존재하는 유일한 레코드인 하나의 행만 포함된 *VTable*을 만들고 두 테이블에서 데이터가 결합되도록 합니다.

```
VTable:
SELECT * from Table1;
inner join SELECT * from Table2;
```

VTable

A	B	C
1	aa	xx

Inner Keep

대신 Inner Keep을 수행해도 두 테이블이 생깁니다. 두 테이블은 공통 필드 *A*를 통해 연결됩니다.

```
VTab1:
SELECT * from Table1;
VTab2:
inner keep SELECT * from Table2;
```

VTab1

A	B
1	aa

VTab2

A	C
1	xx

Left

데이터 로드 스크립트에서 Join 및 Keep 접두사 앞에 left 접두사가 올 수 있습니다.

이 접두사가 Join 앞에 사용될 경우 두 테이블 간에 왼쪽 조인을 수행하도록 지정됩니다. 결과 테이블은 첫 번째 테이블의 전체 데이터 셋이 포함된 두 테이블 간의 조합만 포함합니다.

Keep 앞에 사용할 경우 Qlik Sense에 저장되기 전에 두 번째 테이블을 첫 번째 테이블과의 공통 교집합으로 축소하도록 지정됩니다.

이 예에서는 소스 테이블 *Table1* 및 *Table2*를 사용합니다.

Table1

A	B
1	aa
2	cc
3	ee

Table2

A	C
1	xx
4	yy

먼저 테이블에 대한 Left Join을 수행하여 *Table1*의 모든 행이 포함되고 *Table2*의 해당 행에 있는 필드와 결합된 *VTable*을 만듭니다.

```
VTable:
SELECT * from Table1;
left join SELECT * from Table2;
```

VTable

A	B	C
1	aa	xx
2	cc	-
3	ee	-

대신 Left Keep을 수행해도 두 테이블이 생깁니다. 두 테이블은 공통 필드 *A*를 통해 연결됩니다.

```
VTab1:
SELECT * from Table1;
VTab2:
left keep SELECT * from Table2;
```

VTab1

A	B
1	aa
2	cc
3	ee

VTab2

A	C
1	xx

Right

Qlik Sense 스크립트 언어에서 Join 및 Keep 접두사 앞에 right 접두사가 올 수 있습니다.

이 접두사가 Join 앞에 사용될 경우 두 테이블 간에 오른쪽 조인을 수행하도록 지정됩니다. 결과 테이블은 두 번째 테이블의 전체 데이터 셋이 포함된 두 테이블 간의 조합만 포함합니다.

Keep 앞에 사용할 경우 Qlik Sense에 저장되기 전에 첫 번째 테이블을 두 번째 테이블과의 공통 교집합으로 축소하도록 지정됩니다.

이 예에서는 소스 테이블 *Table1* 및 *Table2*를 사용합니다.

Table1

A	B
1	aa
2	cc
3	ee

Table2

A	C
1	xx
4	yy

먼저 테이블에 대한 Right Join을 수행하여 *Table2*의 모든 행이 포함되고 *Table1*의 해당 행에 있는 필드와 결합된 *VTable*을 만듭니다.

```
VTable:
SELECT * from Table1;
right join SELECT * from Table2;
```

VTable

A	B	C
1	aa	xx
4	-	yy

대신 Right Keep을 수행해도 두 테이블이 생깁니다. 두 테이블은 공통 필드 *A*를 통해 연결됩니다.

```
VTab1:
SELECT * from Table1;
VTab2:
right keep SELECT * from Table2;
```

VTab1

A	B
1	aa

VTab2

A	C
1	xx
4	yy

3.3 인터 레코드 함수 사용: Peek, Previous 및 Exists

이 함수는 현재 레코드를 평가하기 위해 이전에 로드된 데이터 레코드의 값이 필요할 때 사용됩니다.

자습서의 이 부분에서는 Peek(), Previous() 및 Exists() 함수를 사용해 봅니다.

Peek()

Peek()는 이미 로드된 행에 대한 테이블의 필드 값을 반환합니다. 테이블처럼 행 번호를 지정할 수 있습니다. 행 번호를 지정하지 않으면 이전에 마지막으로 로드된 레코드가 사용됩니다.

구문:

```
Peek(fieldname [ , row [ , tablename ] ] )
```

행은 정수여야 합니다. 0은 첫 번째 레코드, 1은 두 번째 레코드와 같은 식으로 이어집니다. 음수는 테이블 끝을 기준으로 한 순서를 나타냅니다. 1은 마지막으로 읽은 레코드를 나타냅니다.

행을 지정하지 않으면 -1로 가정됩니다.

*Tablename*은 끝 부분에 콜론이 없는 테이블 레이블입니다. *tablename*을 지정하지 않으면 현재 테이블이 사용됩니다. **LOAD** 문 외부에서 사용했거나 다른 테이블을 참조할 경우 *tablename*을 반드시 포함해야 합니다.

Previous()

Previous()는 **where** 절로 인해 무시되지 않은 이전 입력 레코드의 데이터를 사용하여 **expr** 표현식의 값을 찾습니다. 이 함수는 내부 테이블의 첫 번째 레코드에 대해 NULL을 반환합니다.

구문:

```
Previous(expression)
```

Previous() 함수는 레코드에 더 심층적으로 액세스하기 위해 중첩될 수 있습니다. 입력 소스에서 데이터를 직접 가져오므로 관련 데이터베이스에 저장되지 않았더라도 Qlik Sense에 로드되지 않은 필드를 참조할 수 있습니다.

Exists()

Exists()는 특정 필드 값이 데이터 로드 스크립트의 필드에 이미 로드되었는지 여부를 판단합니다. 이 함수는 TRUE 또는 FALSE를 반환하므로 **LOAD** 문의 **where** 절 또는 **IF** 문에서 사용할 수 있습니다.

구문:

```
Exists(field [, expression ] )
```

이 필드는 스크립트에 의해 현재까지 로드된 데이터에 존재해야 합니다. *Expression*은 지정된 필드에서 검색할 필드 값을 평가하는 표현식입니다. 생략하면 지정된 필드의 현재 레코드 값이 사용됩니다.

Peek() 및 Previous() 사용

Peek() 및 Previous()는 가장 간단한 형식으로, 테이블 내의 특정 값을 식별하는 데 사용됩니다. 다음은 이 연습에서 로드할 *Employees* 테이블의 샘플 데이터입니다.

Employees 테이블의 샘플 데이터

Date	Hired	Terminated
1/1/2011	6	0
2/1/2011	4	2
3/1/2011	6	1
4/1/2011	5	2

현재는 월, 고용 및 종료에 대한 데이터만 수집하므로 Peek() 및 Previous() 함수를 사용하여 *Employee Count* 및 *Employee Var*에 대한 필드를 추가함으로써 총 직원의 월별 차이를 표시할 것입니다.

다음과 같이 하십시오.

1. *Advanced Scripting Tutorial* 앱을 엽니다.
2. **데이터 로드 편집기**에서 새 스크립트 섹션을 추가합니다.
3. *Employees* 섹션을 호출합니다.

4. 오른쪽 메뉴의 **AttachedFiles**에서 **데이터 선택**을 클릭합니다.
5. 업로드한 다음 *Employees.xlsx*를 선택합니다.

   Field names 아래에서, 데이터를 로드할 때 테이블 필드의 이름을 포함하도록 Embedded field names가 선택되어 있는지 확인합니다.

6. **데이터 선택** 창에서 **스크립트 삽입**을 클릭합니다.
   스크립트는 다음과 같이 표시되어야 합니다.

```
LOAD
    "Date",
    Hired,
    Terminated
FROM [lib://AttachedFiles/Employees.xlsx]
(ooxml, embedded labels, table is Sheet1);
```

7. 이제 다음과 같도록 스크립트를 수정합니다.

```
[Employees Init]:
LOAD
    rowno() as Row,
    Date(Date) as Date,
    Hired,
    Terminated,
    If(rowno()=1, Hired-Terminated,  peek([Employee Count], -1)+(Hired-Terminated)) as
[Employee Count]
FROM [lib://AttachedFiles/Employees.xlsx]
(ooxml, embedded labels, table is Sheet1);
```

   Excel 시트에서 *Date* 필드의 날짜 형식은 MM/DD/YYYY입니다. 시스템 변수의 서식을 사용하여 날짜가 올바르게 해석되도록 *Date* 필드에 Date 함수가 적용됩니다.

   Peek() 함수를 사용하면 정의된 필드에 대해 로드된 모든 값을 식별할 수 있습니다. 이 표현식에서 먼저 rowno()가 1인지 확인합니다. 1인 경우에는 *Employee Count*가 존재하지 않으므로 *Hired*에서 *Terminated*를 뺀 차로 필드를 채웁니다.

   rowno()가 1보다 큰 경우에는 마지막 월의 *Employee Count*를 조회하고 해당 숫자를 해당 월의 *Hired*에서 *Terminated* 직원 수를 뺀 차이 값에 더합니다.

   Peek() 함수에서는 (-1)을 사용하고 있음도 알 수 있습니다. 이를 통해 Qlik Sense에 현재 레코드 위의 레코드를 조회하도록 알려줍니다. (-1)이 지정되지 않으면 Qlik Sense에서는 이전 레코드를 조회하는 것으로 간주합니다.

8. 스크립트의 끝 부분에 다음을 추가합니다.

```
[Employee Count]:
LOAD
       Row,
       Date,
       Hired,
       Terminated,
       [Employee Count],
       If(rowno()=1,0,[Employee Count]-Previous([Employee Count])) as [Employee Var]
```

```
Resident [Employees Init] Order By Row asc;
Drop Table [Employees Init];
```

Previous() 함수를 사용하면 정의된 필드에 대해 로드된 마지막 값을 식별할 수 있습니다. 이 표현식에서 먼저 rowno()가 1인지 확인합니다. 1인 경우에는 이전 월의 *Employee Count*에 대한 레코드가 없기 때문에 Employee Var이 없습니다. 따라서 값에 그냥 0을 입력합니다.

rowno()가 1보다 큰 경우에는 *Employee Var*이 존재하는 것을 알고 있으므로 마지막 월의 *Employee Count*를 조회하고 해당 숫자를 현재 월의 *Employee Count*에서 빼서 *Employee Var* 필드의 값을 생성합니다.

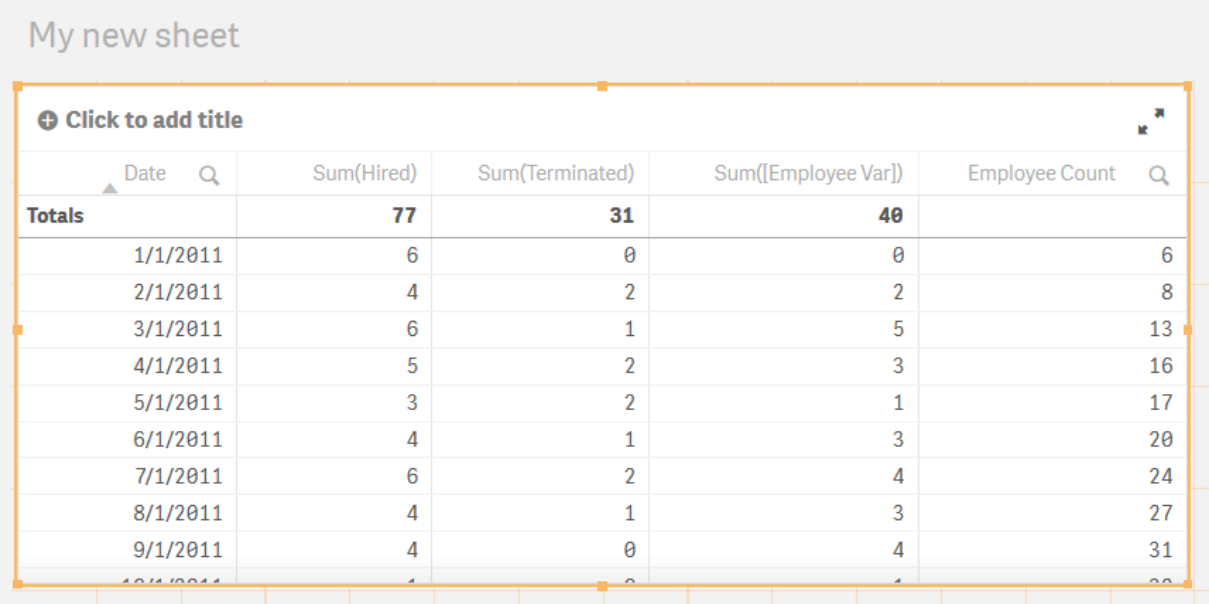
스크립트는 다음과 같이 표시되어야 합니다.

```
[Employees Init]:
LOAD
    rowno() as Row,
    Date(Date) as Date,
    Hired,
    Terminated,
    If(rowno()=1, Hired-Terminated,  peek([Employee Count], -1)+(Hired-Terminated)) as
[Employee Count]
FROM [lib://AttachedFiles/Employees.xlsx]
(ooxml, embedded labels, table is Sheet1);

[Employee Count]:
LOAD
    Row,
    Date,
    Hired,
    Terminated,
    [Employee Count],
    If(rowno()=1,0,[Employee Count]-Previous([Employee Count])) as [Employee Var]
Resident [Employees Init] Order By Row asc;
Drop Table [Employees Init];
```

9. **데이터 로드**를 클릭합니다.

   앱 개요에서 새 시트에 *Date, Hired, Terminated, Employee Count* 및 *Employee Var*을 테이블 열로 사용하여 테이블을 만듭니다. 결과 테이블은 다음과 같습니다.

스크립트에 Peek 및 Previous를 사용한 테이블



Date	Sum(Hired)	Sum(Terminated)	Sum([Employee Var])	Employee Count
Totals	**77**	**31**	**40**	
1/1/2011	6	0	0	6
2/1/2011	4	2	2	8
3/1/2011	6	1	5	13
4/1/2011	5	2	3	16
5/1/2011	3	2	1	17
6/1/2011	4	1	3	20
7/1/2011	6	2	4	24
8/1/2011	4	1	3	27
9/1/2011	4	0	4	31

Peek() 및 Previous()를 사용하면 사용자가 테이블 내의 정의된 행을 대상으로 지정할 수 있습니다. 두 함수의 가장 큰 차이는 Peek() 함수를 사용하면 사용자가 이전에 스크립트에 로드되지 않은 필드를 볼 수 있는 반면 Previous() 함수는 이전에 로드된 필드만 조회할 수 있다는 점입니다. Previous()는 LOAD 문에 대한 입력에서 작동하지만 Peek()는 LOAD 문의 출력에서 작동합니다. (RecNo()와 RowNo()의 차이와 동일) 즉, Where 절이 있으면 두 함수가 다르게 동작합니다.

따라서 현재 값과 이전 값을 비교하여 표시해야 하는 경우에는 Previous() 함수가 더 적합할 수 있습니다. 예에서는 월별 직원 변동을 계산했습니다.

이전에 테이블에 로드되지 않은 필드를 대상으로 지정하거나 특정 행을 대상으로 지정해야 하는 경우에는 Peek() 함수가 더 적합할 수 있습니다. 이는 이전 월의 *Employee Count*를 조회하여 *Employee Count*를 계산한 후 현재 월에 대한 고용 및 종료 직원 간의 차를 더했던 예에서 알 수 있습니다. *Employee Count*는 원본 파일의 필드가 아님을 기억하십시오.

Peek() 및 Previous()를 사용하는 경우에 대한 자세한 내용은 Qlik Community의 블로그 게시물 Peek() vs Previous() – When to Use Each를 참조하십시오. 동작은 QlikView 컨텍스트로 설명됩니다. 그러나 논리는 Qlik Sense에 동일하게 적용됩니다.

Exists() 사용

Exists() 함수는 종종 관련 데이터가 데이터 모델에 이미 로드된 경우 데이터를 로드하기 위해 스크립트에서 Where 절과 함께 사용됩니다.

또한 다음 예에서는 Dual() 함수를 사용하여 문자열에 숫자 값을 할당합니다.

다음과 같이 하십시오.

1. 새 앱을 만들고 이름을 지정합니다.
2. **데이터 로드 편집기**에서 새 스크립트 섹션을 추가합니다.
3. *People* 섹션을 호출합니다.
4. 다음 스크립트를 입력합니다.

```
//Add dummy people data
PeopleTemp:
LOAD * INLINE [
PersonID, Person
1, Jane
2, Joe
3, Shawn
4, Sue
5, Frank
6, Mike
7, Gloria
8, Mary
9, Steven,
10, Bill
];

//Add dummy age data
AgeTemp:
LOAD * INLINE [
PersonID, Age
1, 23
2, 45
3, 43
4, 30
5, 40
6, 32
7, 45
8, 54
9,
10, 61
11, 21
12, 39
];

//LOAD new table with people
People:
NoConcatenate LOAD
    PersonID,
    Person
Resident PeopleTemp;

Drop Table PeopleTemp;

//Add age and age bucket fields to the People table
Left Join (People)
```

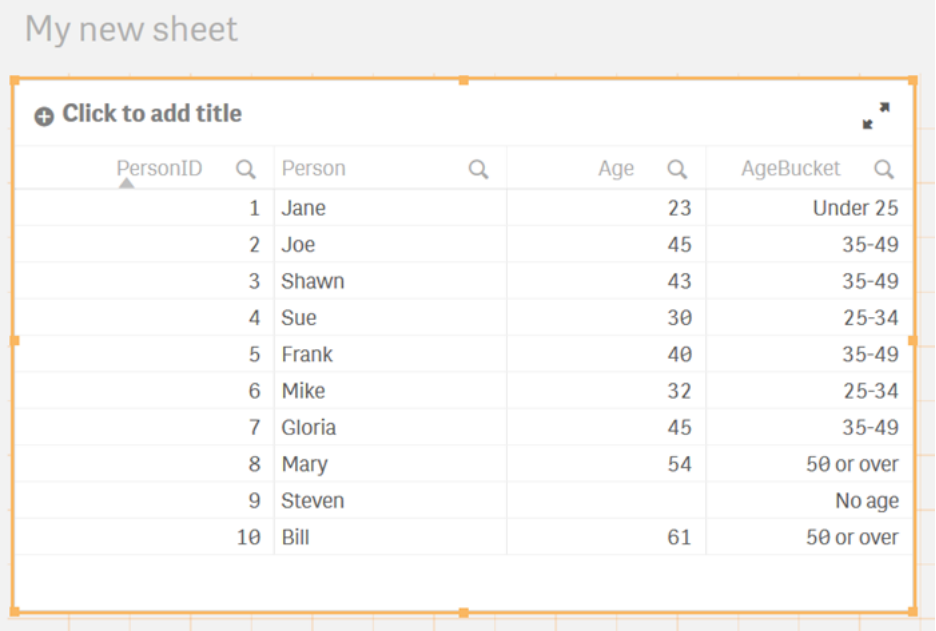
```
LOAD
    PersonID,
    Age,
       If(IsNull(Age) or Age='', Dual('No age', 5),
        If(Age<25, Dual('Under 25', 1),
         If(Age>=25 and Age <35, Dual('25-34', 2),
          If(Age>=35 and Age<50, Dual('35-49' , 3),
           If(Age>=50, Dual('50 or over', 4)
            ))))) as AgeBucket
Resident AgeTemp
Where Exists(PersonID);

DROP Table AgeTemp;
```

5. **데이터 로드**를 클릭합니다.
   스크립트에서는 *PersonID*가 데이터 모델에 이미 로드된 경우에만 *Age* 및 *AgeBucket* 필드가 로드됩니다.

   AgeTemp 테이블에는 *PersonID* 11 및 12에 대해 나열된 연령대가 있지만 이 ID들은 데이터 모델(*People* 테이블)에서 로드되지 않았으므로 Where Exists(PersonID) 절에 의해 제외됩니다. 또한 이 절은 다음과 같이 작성할 수도 있습니다. Where Exists(PersonID, PersonID).

   스크립트의 결과는 다음과 같습니다.

   스크립트에 Exists를 사용한 테이블

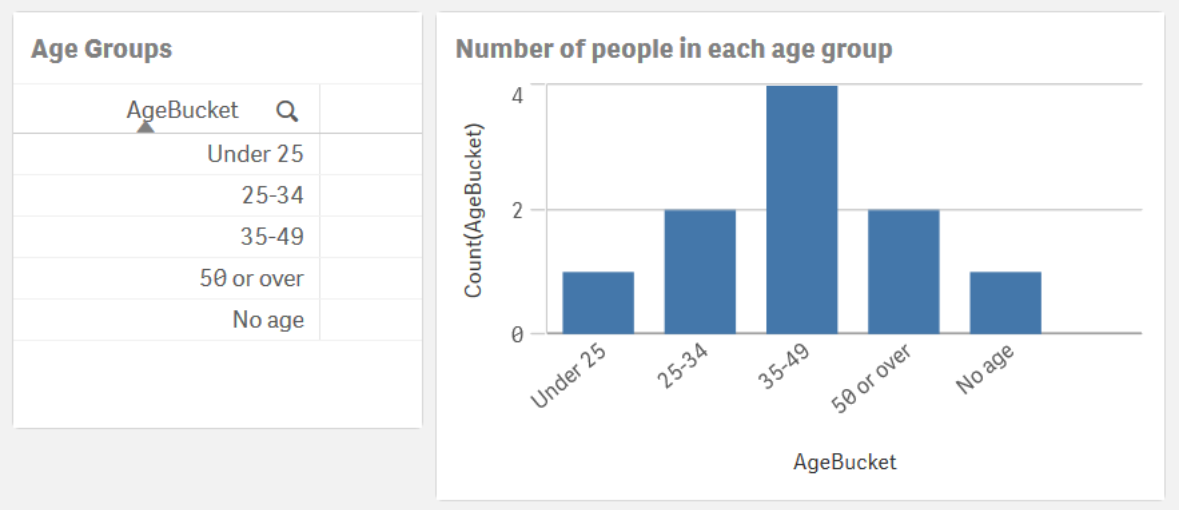


PersonID	Person	Age	AgeBucket
1	Jane	23	Under 25
2	Joe	45	35-49
3	Shawn	43	35-49
4	Sue	30	25-34
5	Frank	40	35-49
6	Mike	32	25-34
7	Gloria	45	35-49
8	Mary	54	50 or over
9	Steven		No age
10	Bill	61	50 or over

   AgeTemp 테이블의 *PersonID* 중 데이터 모델로 로드된 것이 없다면 *Age* 및 *AgeBucket* 필드가 *People* 테이블에 조인되지 않았을 수 있습니다. Exists() 함수를 사용하면 데이터 모델에서 레코드/데이터가 분리되는 것, 즉 연결된 사람이 없는 *Age* 및 *AgeBucket* 필드를 방지할 수 있습니다.

6. 새 시트를 만들고 이름을 지정합니다.
7. 새 시트를 열고 **시트 편집**을 클릭합니다.

8. 차원 *AgeBucket*이 있는 시트에 일반 테이블을 추가하고 시각화 이름을 *Age Groups*로 지정합니다.
9. 차원 *AgeBucket* 및 측정값 *Count([AgeBucket])*이 있는 시트에 막대형 차트를 추가합니다. 시각화 이름을 *Number of people in each age group*으로 지정합니다.
10. 기본 설정에 따라 테이블 및 막대형 차트의 속성을 조정하고 **완료**를 클릭합니다.
    시트가 다음과 같이 표시됩니다.
    연령별로 그룹화된 시트



Dual() 함수는 문자열에 숫자 값을 할당할 필요가 있는 경우 스크립트 또는 차트 표현식에서 유용합니다.

위 스크립트에 연령대를 로드하는 응용 프로그램이 있고 이 연령대를 버킷에 넣기로 했으므로 연령대 버킷과 실제 연령대 간의 비교에 기반하여 시각화를 만들 수 있습니다. 25세 미만, 25 ~ 35세 사이 등의 버킷이 있습니다. Dual() 함수를 사용하면 연령대 버킷에 숫자 값에 할당하여 나중에 목록 상자 또는 차트에서 연령대 버킷을 정렬하는 데 사용할 수 있습니다. 따라서 앱 시트에서와 같이 정렬에서도 목록 끝부분에 "No age"가 추가됩니다.

> *Exists() 및 Dual()에 대한 자세한 내용은 Qlik Community의 다음 블로그 게시물을 참조하십시오.* Dual & Exists – Useful Functions *(Dual & Exists – 유용한 함수)*

3.4 간격 일치 및 반복 로드

LOAD 또는 SELECT 문에 Intervalmatch 접두사를 사용하면 불연속 숫자 값을 하나 이상의 숫자 간격과 연결할 수 있습니다. 예를 들어, 프로덕션 환경에서 사용할 수 있는 매우 강력한 기능입니다.

IntervalMatch() 접두사 사용

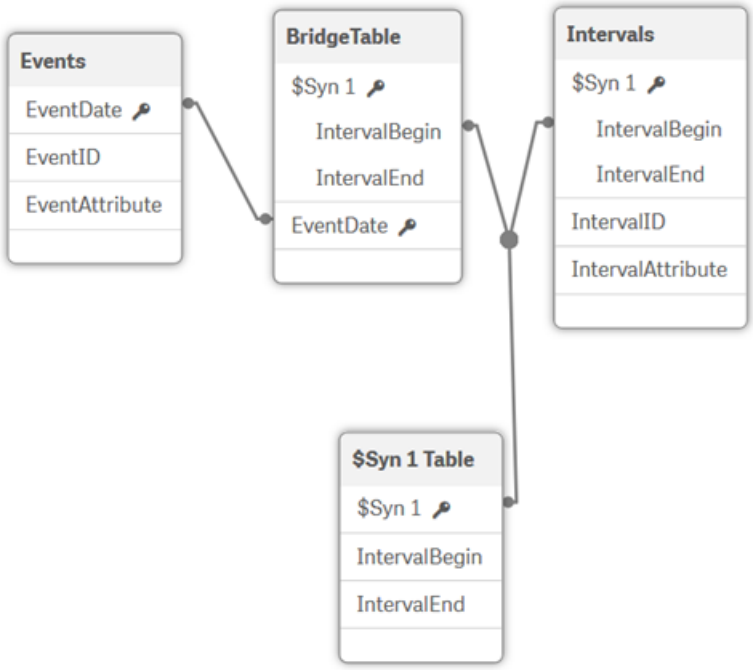
가장 기본적인 간격 일치는 한 테이블에 숫자 또는 날짜(이벤트)의 목록이 있고 두 번째 테이블에 간격의 목록이 있는 경우입니다. 두 테이블을 연결하는 것이 목적입니다. 일반적으로 이것은 다대다 관계이므로, 하나의 간격에 많은 날짜가 속해 있을 수 있으며 하나의 날짜가 여러 간격에 속할 수도 있습니다. 이 문제를 해결하기 위해서는 두 원본 테이블 사이에 브리지 테이블을 만들어야 합니다. 이 작업은 여러 방법으로 수행할 수 있습니다.

Qlik Sense에서 이 문제를 해결하는 가장 간단한 방법은 LOAD 또는 SELECT 문 앞에 IntervalMatch() 접두사를 사용하는 것입니다. LOAD/SELECT 문에는 간격을 정의하는 두 개의 필드인 From 및 To만 포함되어야 합니다. 그러면 IntervalMatch() 접두사가 로드된 간격과 접두사에 대한 매개 변수로 지정된 이전에 로드된 숫자 필드 사이의 모든 조합을 생성합니다.

다음과 같이 하십시오.

1. 새 앱을 만들고 이름을 지정합니다.
2. **데이터 로드 편집기**에서 새 스크립트 섹션을 추가합니다.
3. *Events* 섹션을 호출합니다.
4. 오른쪽 메뉴의 **AttachedFiles**에서 **데이터 선택**을 클릭합니다.
5. 업로드한 다음 *Events.txt*를 선택합니다.
6. **데이터 선택** 창에서 **스크립트 삽입**을 클릭합니다.
7. 업로드한 다음 *Intervals.txt*를 선택합니다.
8. **데이터 선택** 창에서 **스크립트 삽입**을 클릭합니다.
9. 스크립트에서 첫 번째 테이블 이름을 *Events*로 지정하고 두 번째 테이블은 *Intervals*로 지정합니다.
10. 스크립트의 끝에 IntervalMatch를 추가하여 처음 두 테이블을 연결하는 세 번째 테이블을 만듭니다.
    ```
    BridgeTable:
    IntervalMatch (EventDate)
    LOAD distinct IntervalBegin, IntervalEnd
    Resident Intervals;
    ```
11. 스크립트는 다음과 같이 표시되어야 합니다.
    ```
    Events:
    LOAD
        EventID,
        EventDate,
        EventAttribute
    FROM [lib://AttachedFiles/Events.txt]
    (txt, utf8, embedded labels, delimiter is '\t', msq);

    Intervals:
    LOAD
        IntervalID,
        IntervalAttribute,
        IntervalBegin,
        IntervalEnd
    FROM [lib://AttachedFiles/Intervals.txt]
    (txt, utf8, embedded labels, delimiter is '\t', msq);

    BridgeTable:
    IntervalMatch (EventDate)
    LOAD distinct IntervalBegin, IntervalEnd
    Resident Intervals;
    ```
12. **데이터 로드**를 클릭합니다.
13. **데이터 모델 뷰어**를 엽니다. 데이터 모델은 다음과 같이 표시됩니다.

데이터 모델: Events, BridgeTable, Intervals 및 $Syn1 테이블



데이터 모델에는 자체적으로 *IntervalBegin* 가상 키로 증명될 복합 키(*IntervalEnd* 및 Qlik Sense 필드)가 포함됩니다.

기본 테이블은 다음과 같습니다.

- 이벤트당 정확히 하나의 레코드를 포함하는 *Events* 테이블.
- 간격당 정확히 하나의 레코드를 포함하는 *Intervals* 테이블.
- 이벤트 및 간격의 조합당 정확히 하나의 레코드가 포함되고 두 이전 테이블을 연결하는 브리지 테이블.

간격이 중첩되는 경우는 하나의 이벤트가 여러 간격에 속할 수 있습니다. 또한 간격은 당연히 이에 속하는 여러 이벤트를 가질 수 있습니다.

이 데이터 모델은 정규화되고 소형이라는 점에서 최적입니다. *Events* 테이블 및 *Intervals* 테이블은 모두 변경되지 않으며 원래 레코드 수를 포함합니다. 이들 테이블에서 수행되는 모든 Qlik Sense 계산(예: Count(EventID))은 올바르게 작동 및 평가됩니다.

IntervalMatch()에 대한 자세한 내용은 Qlik Community에서 다음 블로그 게시물을 참조하십시오. Using IntervalMatch() *(IntervalMatch() 사용)*

While 루프 및 반복 로드 IterNo() 사용

간격의 하위 및 상위 경계 사이의 열거 가능한 값을 생성하는 While 루프 및 IterNo()를 사용하여 거의 동일한 브리지 테이블을 만들 수 있습니다.

LOAD 문 내의 루프는 While 절을 사용하여 만들 수 있습니다. 예:

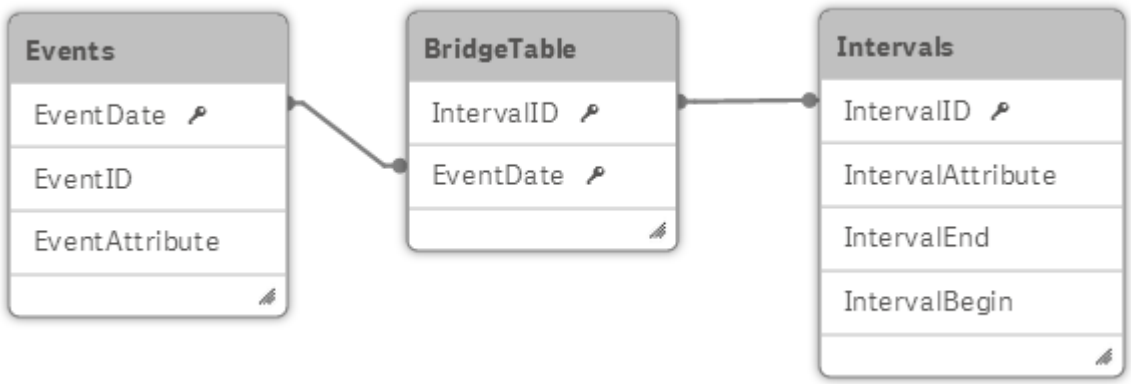
```
LOAD Date, IterNo() as Iteration From … While IterNo() <= 4;
```

이러한 LOAD 문은 각 입력 레코드에 따라 반복되며 While 절의 표현식이 참인 동안 반복해서 로드합니다. IterNo() 함수는 첫 번째 반복 시 "1"을 반환하고 두 번째 반복 시 "2"를 반환하는 식으로 계속 이어집니다.

간격에 대한 기본 키인 IntervalID가 있으므로 스크립트에서의 유일한 차이는 브리지 테이블을 만드는 방법뿐입니다.

다음과 같이 하십시오.

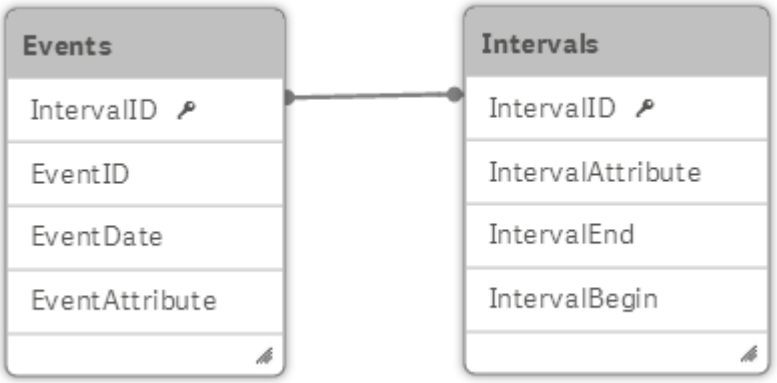
1. 기존 Bridgetable 문을 다음 스크립트로 바꿉니다.
   ```
   BridgeTable:
   LOAD distinct * Where Exists(EventDate);
   LOAD IntervalBegin + IterNo() - 1 as EventDate, IntervalID
       Resident Intervals
       While IntervalBegin + IterNo() - 1 <= IntervalEnd;
   ```
2. **데이터 로드**를 클릭합니다.
3. **데이터 모델 뷰어**를 엽니다. 데이터 모델은 다음과 같이 표시됩니다.
   데이터 모델: Events, BridgeTable 및 Intervals 테이블



   일반적으로 세 개의 테이블이 있는 솔루션이 가장 좋은데, 간격과 이벤트 사이의 다대다 관계를 허용하기 때문입니다. 그러나 대게 하나의 이벤트가 하나의 단일 간격에만 속할 수 있다고 알고 있습니다. 이 경우 브리지 테이블은 실제로 필요하지 않습니다. *IntervalID*는 이벤트 테이블에 직접 저장할 수 있습니다. 이렇게 하려면 여러 가지 방법이 있지만 Bridgetable을 *Events* 테이블과 조인하는 방법이 가장 유용합니다.

4. 스크립트의 끝 부분에 다음 스크립트를 추가합니다.
   ```
   Join (Events)
   LOAD EventDate, IntervalID
   Resident BridgeTable;

   Drop Table BridgeTable;
   ```
5. **데이터 로드**를 클릭합니다.
6. **데이터 모델 뷰어**를 엽니다. 데이터 모델은 다음과 같이 표시됩니다.

데이터 모델: Events 및 Intervals 테이블



열린 간격 및 닫힌 간격

간격이 열려 있는지 또는 닫혀 있는지는 종료 지점이 간격에 포함되어 있는지 여부에 의해 결정됩니다.

- 복수의 종료 지점이 포함되어 있으면 닫힌 간격입니다.
  [a,b] = {x ∈ ℝ | a ≤ x ≤ b}
- 복수의 종료 지점이 포함되어 있지 않으면 열린 간격입니다.
  ]a,b[= {x ∈ ℝ | a < x < b}
- 종료 지점 하나가 포함되어 있으면 반만 열린 간격입니다.
  [a,b[= {x ∈ ℝ | a ≤ x < b}

간격이 중첩되고 숫자가 둘 이상의 간격에 속할 수 있는 경우에는 보통 닫힌 간격을 사용해야 합니다.

그러나 간격을 중첩시키지 않으려는 경우에는 숫자가 하나의 간격에만 속하도록 하려고 할 수 있습니다. 따라서 한 지점이 한 간격의 끝이고 동시에 다음 간격의 시작인 경우에 문제가 생깁니다. 이 값에 있는 숫자는 양쪽 간격에 모두 속하게 됩니다. 따라서 반만 열린 간격이 필요합니다.

이 문제에 대한 실질적인 솔루션은 모든 간격의 종료 값에서 약간을 빼서 닫힌 간격을 만들어 간격이 중첩되지 않도록 하는 것입니다. 숫자가 날짜인 경우는 가장 간단한 방법이 해당 날짜의 마지막 밀리초를 반환하는 DayEnd() 함수를 사용하는 것입니다.

```
Intervals:
LOAD…, DayEnd(IntervalEnd – 1) as IntervalEnd From Intervals;
```

수동으로 작은 양을 뺄 수도 있습니다. 그렇게 하는 경우, 연산에서 52개의 유효 이진 숫자(14개의 십진 숫자)로 올림 처리되므로 빼는 양이 너무 작지 않도록 하십시오. 너무 작은 양을 사용하면 차이를 식별할 수 없어서 원래 숫자를 다시 사용하게 됩니다.

4 데이터 정리

Qlik Sense에 로드한 소스 데이터가 Qlik Sense 앱에서 반드시 필요한 방식으로 유지되지 않는 경우가 있습니다. Qlik Sense에서는 데이터를 적합한 형식으로 변환할 수 있도록 해주는 다수의 함수 및 문을 제공합니다.

매핑은 Qlik Sense 스크립트에서 해당 스크립트를 실행할 때 필드 값 또는 이름을 바꾸거나 수정하는 데 사용할 수 있습니다. 따라서 매핑을 사용하면 데이터를 정리하여 더 일관성 있게 만들거나 필드 값의 일부 또는 전체를 바꿀 수 있습니다.

서로 다른 테이블에서 데이터를 로드할 때 필드 값의 의미가 같더라도 이름이 항상 일관성 있게 지정되는 것은 아닙니다. 일관성 결여는 연결에 방해가 되므로 해결할 필요가 있습니다. 필드 값을 비교하는 매핑 테이블을 만들면 이 문제를 매끄럽게 해결할 수 있습니다.

4.1 매핑 테이블

Mapping load 또는 Mapping select를 통해 로드된 테이블은 일반 테이블과는 다르게 처리됩니다. 이러한 테이블은 별도의 메모리 영역에 저장되며 스크립트 실행 시 매핑 테이블로만 사용됩니다. 스크립트 실행 후에는 이들 테이블이 자동으로 삭제됩니다.

규칙:

- 매핑 테이블에는 두 개의 열, 즉 비교 값을 포함하는 첫째 열과 원하는 매핑 값을 포함하는 둘째 열이 있어야 합니다.
- 두 열에는 이름을 지정해야 하지만 이름은 중요하지 않습니다. 열 이름은 일반 내부 테이블의 필드 이름과 연결되지 않습니다.

4.2 Mapping 함수 및 문

이 자습서에서는 다음 매핑 함수/문을 설명합니다.

• Mapping 접두사

• ApplyMap()

• MapSubstring()

• Map ... Using 문

• Unmap 문

4.3 Mapping 접두사

Mapping 접두사는 스크립트에서 매핑 테이블을 만드는 데 사용됩니다. 그런 다음, 매핑 테이블을 ApplyMap() 함수, MapSubstring() 함수 또는 Map ... Using 문에서 사용할 수 있습니다.

다음과 같이 하십시오.

1. 새 앱을 만들고 이름을 지정합니다.
2. **데이터 로드 편집기**에서 새 스크립트 섹션을 추가합니다.
3. *Countries* 섹션을 호출합니다.
4. 다음 스크립트를 입력합니다.

```
CountryMap:
MAPPING LOAD * INLINE [
Country, NewCountry
U.S.A., US
U.S., US
United States, US
United States of America, US
];
```

CountryMap 테이블에는 다음 두 열이 저장됩니다. *Country* 및 *NewCountry*를 참조하십시오. *Country* 열에는 *Country* 필드에 국가를 입력한 다양한 방법을 저장합니다. *NewCountry* 열에는 해당 값의 매핑 방법을 저장합니다. 이 매핑 테이블은 *Country* 필드에 일관된 *US* 국가 값을 저장하는 데 사용됩니다. 예를 들어 *Country* 필드에 *U.S.A.*가 저장되면 이를 *US*로 매핑합니다.

4.4 ApplyMap() 함수

ApplyMap()을 사용하면 이전에 만든 매핑 테이블에 기반하여 필드의 데이터를 바꿀 수 있습니다. ApplyMap() 함수를 사용하려면 먼저 매핑 테이블을 로드해야 합니다. 로드할 *Data.xlsx* 테이블의 데이터는 다음과 같습니다.

데이터 테이블

ID	이름	Country	코드
1	John Black	U.S.A.	SDFGBS1DI
2	Steve Johnson	U.S.	2ABC
3	Mary White	United States	DJY3DFE34
4	Susan McDaniels	u	DEF5556
5	Dean Smith	US	KSD111DKFJ1

국가가 다양한 방법으로 입력되었음을 알 수 있습니다. 국가 필드를 일관성 있게 만들기 위해 매핑 테이블이 로드된 후 **ApplyMap()** 함수가 사용됩니다.

다음과 같이 하십시오.

1. 위에서 입력한 스크립트 아래에서 *Data.xlsx*를 선택하고 로드한 다음 스크립트를 삽입합니다.
2. 새로 만든 LOAD 문 위에 다음을 입력합니다.

```
Data:
```

스크립트는 다음과 같이 표시되어야 합니다.

```
CountryMap:
MAPPING LOAD * INLINE [
    Country, NewCountry
    U.S.A., US
    U.S., US
    United States, US
    United States of America, US
];

Data:
LOAD
    ID,
    Name,
    Country,
    Code
FROM [lib://AttachedFiles/Data.xlsx]
(ooxml, embedded labels, table is Sheet1);
```

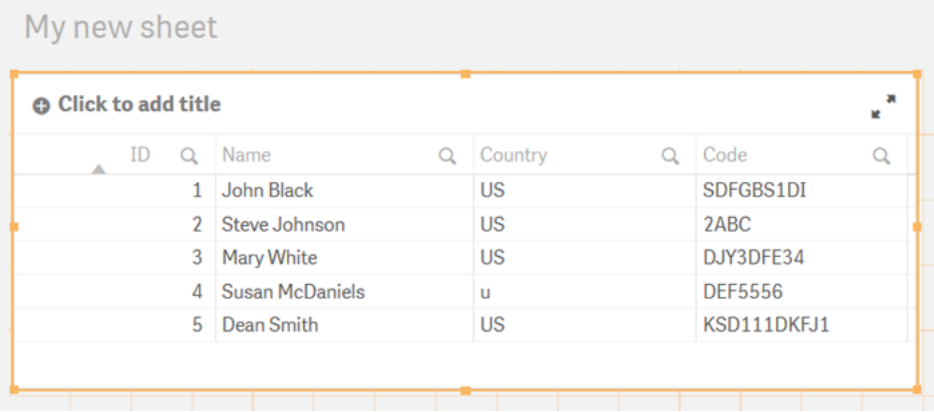
3. 아래와 같이 `Country,`가 포함된 줄을 수정합니다.

   ```
   ApplyMap('CountryMap', Country) as Country,
   ```

   ApplyMap() 함수의 첫 번째 매개 변수에는 작은따옴표로 묶인 맵 이름이 있습니다. 두 번째 매개 변수는 바꿀 데이터가 포함된 필드입니다.

4. **데이터 로드**를 클릭합니다.
   결과 테이블은 다음과 같습니다.

   ApplyMap() 함수를 사용하여 로드된 데이터를 보여 주는 테이블

   *United States*의 다양한 철자는 모두 *US*로 변경되었습니다. 철자가 올바르지 않은 레코드가 하나 있으므로 ApplyMap() 함수는 해당 필드 값을 변경하지 않았습니다. ApplyMap() 함수를 사용하면 매핑 테이블에 일치 값이 없는 경우 세 번째 매개 변수를 사용하여 기본 표현식을 추가할 수 있습니다.

5. 국가가 잘못 입력되었을 수 있는 경우를 처리하기 위해 ApplyMap() 함수의 세 번째 매개 변수로 `'US'`를 추가합니다.

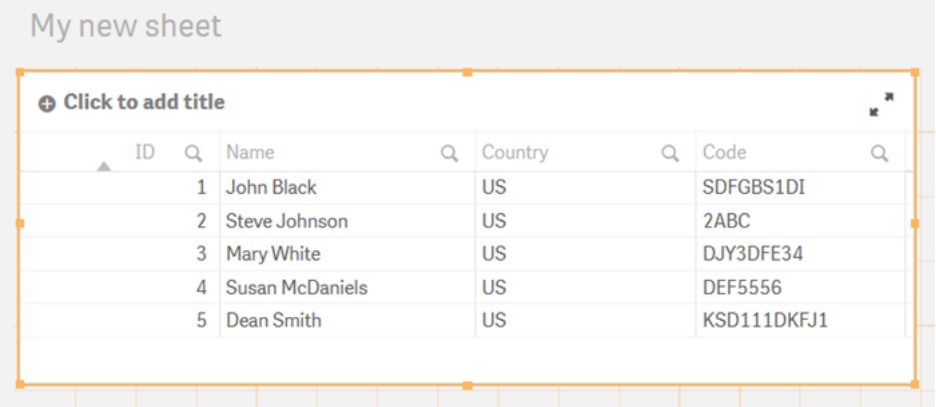
   ```
   ApplyMap('CountryMap', Country, 'US') as Country,
   ```

   스크립트는 다음과 같이 표시되어야 합니다.

```
CountryMap:
MAPPING LOAD * INLINE [
    Country, NewCountry
    U.S.A., US
    U.S., US
    United States, US
    United States of America, US
];

Data:
LOAD
    ID,
    Name,
    ApplyMap('CountryMap', Country, 'US') as Country,
    Code
FROM [lib://AttachedFiles/Data.xlsx]
(ooxml, embedded labels, table is Sheet1);
```

6. **데이터 로드**를 클릭합니다.
   결과 테이블은 다음과 같습니다.

   ApplyMap 함수를 사용하여 로드된 데이터를 보여 주는 테이블



ApplyMap()에 대한 자세한 내용은 Qlik Community에서 다음 블로그 게시물을 참조하십시오.
Don't join - use Applymap instead *(조인하지 말고 대신 Applymap 사용)*

4.5 MapSubstring() 함수

MapSubstring() 함수를 사용하면 필드의 일부분을 매핑할 수 있습니다.

이제 ApplyMap()으로 생성된 테이블에서 숫자를 텍스트로 쓰려고 하며, 따라서 숫자 데이터를 텍스트로 바꾸는 데 MapSubstring() 함수가 사용됩니다.

이 작업을 수행하기 위해서는 먼저 매핑 테이블을 만들어야 합니다.

다음과 같이 하십시오.

1. *CountryMap* 섹션과 *Data* 섹션 사이에 다음 스크립트 줄을 추가합니다.

```
CodeMap:
MAPPING LOAD * INLINE [
F1, F2
1, one
2, two
3, three
4, four
5, five
11, eleven
];
```

   CodeMap 테이블에서는 숫자 1 ~ 5 및 11이 매핑됩니다.

2. 스크립트의 *Data* 섹션에서 Code 문을 다음과 같이 수정합니다.

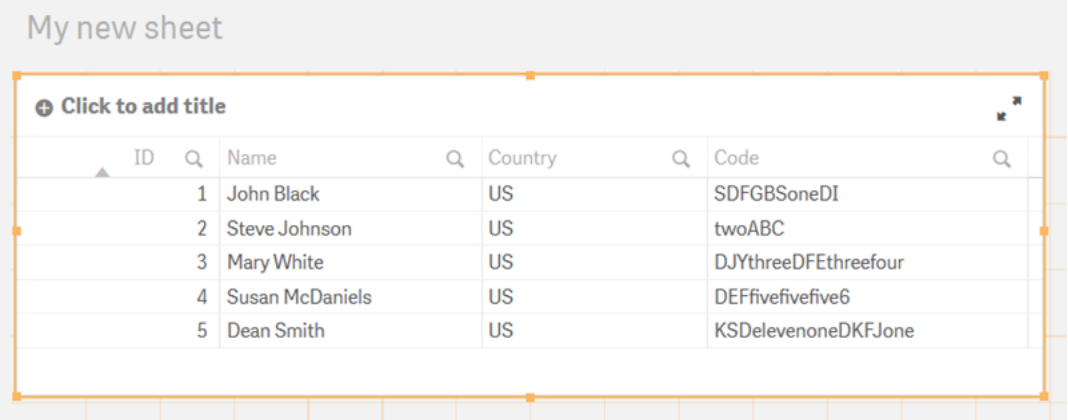
```
MapSubString('CodeMap', Code) as Code
```

   스크립트는 다음과 같이 표시되어야 합니다.

```
CountryMap:
MAPPING LOAD * INLINE [
    Country, NewCountry
    U.S.A., US
    U.S., US
    United States, US
    United States of America, US
];

CodeMap:
MAPPING LOAD * INLINE [
F1, F2
1, one
2, two
3, three
4, four
5, five
11, eleven
];

Data:
LOAD
    ID,
    Name,
    ApplyMap('CountryMap', Country, 'US') as Country,
    MapSubString('CodeMap', Code) as Code
FROM [lib://AttachedFiles/Data.xlsx]
(ooxml, embedded labels, table is Sheet1);
```

3. **데이터 로드**를 클릭합니다.
   결과 테이블은 다음과 같습니다.

MapSubString 함수를 사용하여 로드된 데이터를 보여 주는 테이블



숫자는 *Code* 필드의 텍스트로 바뀌었습니다. ID=3 및 ID=4의 경우처럼 숫자가 두 번 이상 나타나면 텍스트도 반복됩니다. ID=4. *Susan McDaniels*는 코드에 6이 있습니다. 6은 *CodeMap* 테이블에서 매핑되어 있지 않으므로 변경되지 않은 상태를 유지합니다. ID=5, *Dean Smith*는 코드에 111이 있습니다. 이는 'elevenone'으로 매핑되었습니다.

MapSubstring()에 대한 자세한 내용은 Qlik Community에서 다음 블로그 게시물을 참조하십시오. Mapping ... and not the geographical kind *(Mapping ... 지리적 종류가 아님)*

4.6 Map ... Using

Map ... Using 문도 필드에 매핑을 적용하는 데에 사용할 수 있습니다. 하지만 ApplyMap()과는 약간 다르게 작동됩니다. ApplyMap()은 필드 이름이 발견될 때마다 매핑을 처리하지만 Map ... Using은 해당 값이 내부 테이블의 필드 이름 아래에 저장된 경우에 매핑을 처리합니다.

예를 살펴보도록 하겠습니다. 스크립트에서 *Country* 필드를 여러 번 로드했고 필드가 로드될 때마다 매핑을 적용한다고 가정해봅니다. 이 자습서의 앞부분에서 설명한 대로 ApplyMap() 함수를 사용하거나 Map ... Using을 사용할 수 있습니다.

Map ... Using을 사용하면 필드가 내부 테이블에 저장될 때 필드에 매핑이 적용됩니다. 그러므로 아래 예에서는 Data1 테이블의 *Country* 필드에 매핑이 적용되고 *Data2* 테이블의 *Country2* 필드에는 적용되지 않습니다. Map ... Using 문은 *Country*로 이름이 지정된 필드에만 적용될 수 있기 때문입니다. *Country2* 필드가 내부 테이블에 저장되면 더 이상 이름이 *Country*로 지정되지 않습니다. *Country2* 테이블에 매핑이 적용되도록 하려면 ApplyMap() 함수를 사용해야 합니다.

Unmap 문은 Map ... Using 문을 끝내므로 *Country*가 Unmap 문 뒤에 로드되면 *CountryMap*은 적용되지 않습니다.

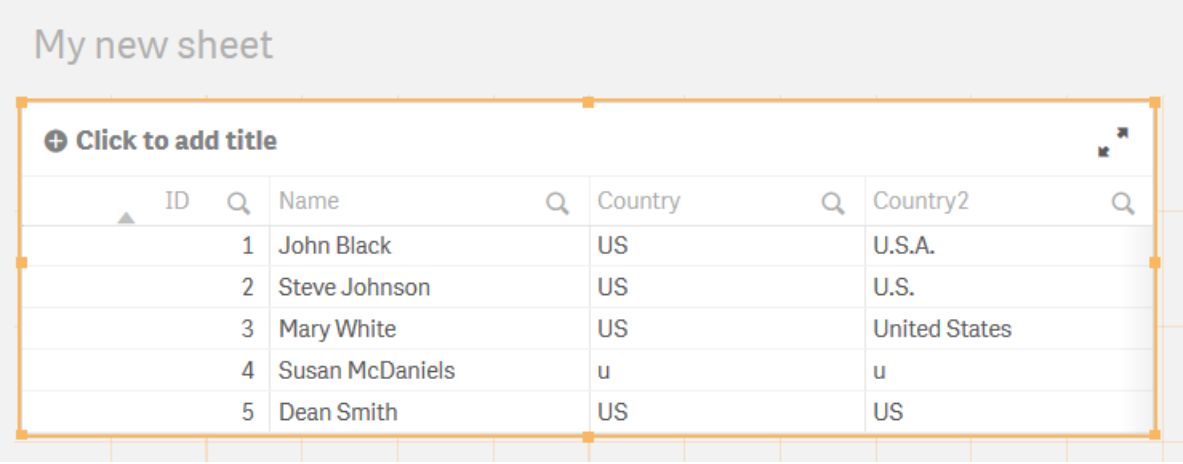
다음과 같이 하십시오.

1. *Data* 테이블에 대한 스크립트를 다음으로 대체합니다.

```
Map Country Using CountryMap;
Data1:
    LOAD
    ID,
    Name,
    Country
FROM [lib://AttachedFiles/Data.xlsx]
(ooxml, embedded labels, table is Sheet1);

Data2:
LOAD
    ID,
    Country as Country2
FROM [lib://AttachedFiles/Data.xlsx]
(ooxml, embedded labels, table is Sheet1);
UNMAP;
```

2. **데이터 로드**를 클릭합니다.
   결과 테이블은 다음과 같습니다.

   Map ... Using 함수를 사용하여 로드된 데이터를 보여 주는 테이블



5 계층 구조 데이터 처리

계층 구조는 모든 Business Intelligence 솔루션에서 중요한 부분으로, 본질적으로 다양한 수준의 세분성이 포함된 차원을 설명하는 데 사용됩니다. 어떤 차원은 단순하고 직관적이지만, 어떤 차원은 복잡하고 까다롭게 때문에 올바르게 모델링하려면 많은 생각을 해야 합니다.

계층 구조의 위쪽에서 아래쪽으로 향하면서 멤버는 점차 더 세분화됩니다. 예를 들어, Market, Country, State 및 City 수준이 있는 차원에서 멤버 Americas는 계층 구조의 최상위 수준에 표시되고 멤버 U.S.A.는 두 번째 수준, 멤버 California 는 세 번째 수준, San Francisco는 최하위 수준에 표시됩니다. California는 U.S.A.보다 더 구체적이며 San Francisco는 California보다 더 구체적입니다.

관계형 모델에 계층 구조를 저장하는 것은 여러 솔루션에서 공통적인 과제입니다. 여기에는 여러 가지 접근 방법이 있습니다.

- 수평 계층 구조
- 인접 목록 모델
- 경로 열거 방법
- 중첩된 집합 모델
- 상위 항목 목록

이 자습서의 목적에 맞게 상위 항목 목록을 만들 예정인데, 쿼리에서 직접 사용할 수 있는 형식으로 계층 구조가 표현되기 때문입니다. 다른 접근 방법에 대한 자세한 내용은 Qlik Community에서 확인할 수 있습니다.

5.1 Hierarchy 접두사

Hierarchy 접두사는 인접 노드 테이블을 로드하는 LOAD 또는 SELECT 문의 앞에 넣는 스크립트 명령입니다. LOAD 문에는 3개 이상의 필드 (노드에 대한 고유 키에 해당하는 ID, 부모에 대한 참조 및 이름)가 있어야 합니다.

이 접두사는 로드된 테이블을, 계층 구조의 각 수준에 대해 하나씩 여러 개의 추가 열을 가진 확장된 노드 테이블로 변환합니다.

다음과 같이 하십시오.

1. 새 앱을 만들고 이름을 지정합니다.
2. **데이터 로드 편집기**에서 새 스크립트 섹션을 추가합니다.
3. *Wine* 섹션을 호출합니다.
4. 오른쪽 메뉴의 **AttachedFiles**에서 **데이터 선택**을 클릭합니다.
5. 업로드한 다음 *Winedistricts.txt*를 선택합니다.
6. **데이터 선택** 창에서 *Lbound* 및 *RBound* 필드를 선택 취소하여 로드되지 않도록 합니다.
7. **스크립트 삽입**을 클릭합니다.
8. LOAD 문 위에 다음을 입력합니다.

```
Hierarchy (NodeID, ParentID, NodeName)
```

스크립트는 다음과 같이 표시되어야 합니다.

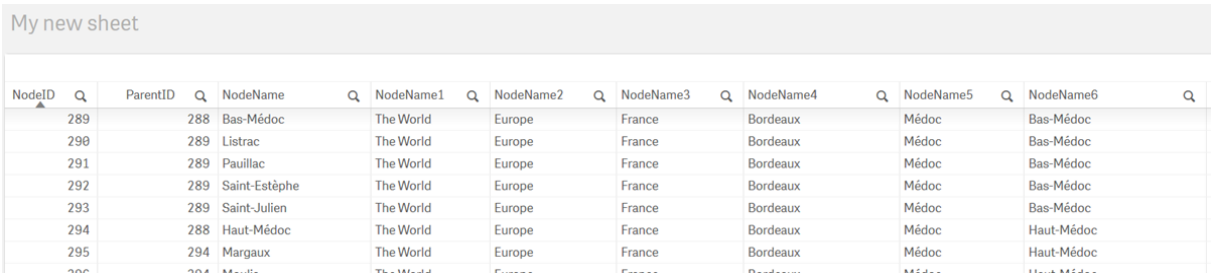
```
Hierarchy (NodeID, ParentID, NodeName)
LOAD
    NodeID,
    ParentID,
    NodeName
FROM [lib://AttachedFiles/Winedistricts.txt]
(txt, utf8, embedded labels, delimiter is '\t', msq);
```

9. **데이터 로드**를 클릭합니다.
10. **데이터 모델 뷰어**의 **미리 보기** 섹션을 사용하여 결과 테이블을 봅니다.
    결과로 생성되는 확장된 노드 테이블은 정확하게 소스 테이블과 같은 수의 레코드를 갖습니다. 노드 당 하나. 확장된 노드 테이블은 관계형 모델에서 계층 구조를 분석하는 데 필요한 여러 요구 사항을 충족시키므로 매우 유용합니다.

    - 모든 노드 이름이 하나의 동일한 열에 나타나므로 검색에 사용할 수 있습니다.
    - 또한 여러 노드 수준이 각각 하나의 필드로 확장되며 이 필드는 드릴다운 그룹에서 또는 피벗 테이블의 차원으로 사용할 수 있습니다.
    - 또한, 여러 노드 수준이 각각 하나의 필드로 확장되며 이 필드는 드릴 다운 그룹에서 사용할 수 있습니다.
    - 노드에 대해 고유한 경로를 포함하여 올바른 순서의 모든 상위 항목을 나열할 수 있습니다.
    - 노드의 깊이(즉, 루트로부터의 거리)를 포함할 수 있습니다.

    결과 테이블은 다음과 같습니다.

    Hierarchy 접두사를 사용하여 로드된 데이터 샘플을 보여 주는 테이블


My new sheet

NodeID	ParentID	NodeName	NodeName1	NodeName2	NodeName3	NodeName4	NodeName5	NodeName6
289	288	Bas-Médoc	The World	Europe	France	Bordeaux	Médoc	Bas-Médoc
290	289	Listrac	The World	Europe	France	Bordeaux	Médoc	Bas-Médoc
291	289	Pauillac	The World	Europe	France	Bordeaux	Médoc	Bas-Médoc
292	289	Saint-Estèphe	The World	Europe	France	Bordeaux	Médoc	Bas-Médoc
293	289	Saint-Julien	The World	Europe	France	Bordeaux	Médoc	Bas-Médoc
294	288	Haut-Médoc	The World	Europe	France	Bordeaux	Médoc	Haut-Médoc
295	294	Margaux	The World	Europe	France	Bordeaux	Médoc	Haut-Médoc

5.2 HierarchyBelongsTo 접두사

Hierarchy 접두사와 마찬가지로 HierarchyBelongsTo 접두사는 인접 노드 테이블을 로드하는 LOAD 또는 SELECT 문의 앞에 넣는 스크립트 명령입니다.

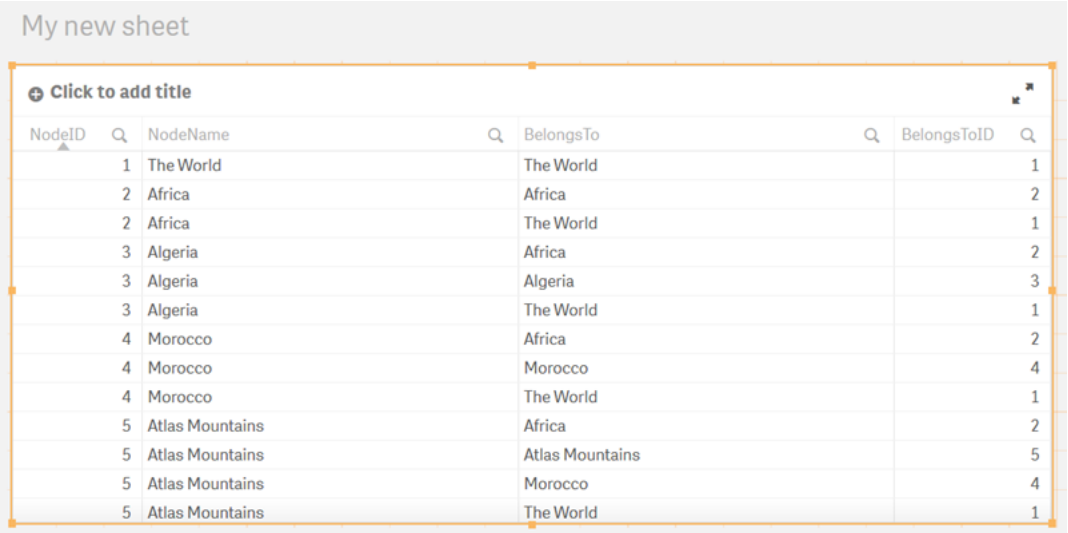
또한 LOAD 문에는 3개 이상의 필드 (노드에 대한 고유 키에 해당하는 ID, 부모에 대한 참조 및 이름)가 있어야 합니다. 이 접두사는 로드된 테이블을 상위 항목 테이블로 변환하는 데, 이 상위 항목 테이블에는 개별 레코드로 나열된 상위 항목 및 하위 항목의 모든 조합이 포함되어 있습니다. 따라서 특정 노드의 모든 상위 항목 또는 하위 항목을 매우 쉽게 찾을 수 있습니다.

다음과 같이 하십시오.

1. **데이터 로드 편집기**에서 다음과 같이 Hierarchy 문을 수정합니다.
   ```
   HierarchyBelongsTo (NodeID, ParentID, NodeName, BelongsToID, BelongsTo)
   ```
2. **데이터 로드**를 클릭합니다.
3. **데이터 모델 뷰어**의 **미리 보기** 섹션을 사용하여 결과 테이블을 봅니다.
   상위 항목 테이블은 관계형 모델에서 계층 구조를 분석하는 데 필요한 여러 요구 사항을 충족시킵니다.
   - 노드 ID가 단일 노드를 나타내는 경우 상위 ID는 계층 구조의 전체 트리 및 하위 트리를 나타냅니다.
   - 모든 노드 이름은 노드 역할과 트리 역할에서 모두 존재하며, 둘 모두 검색에 사용할 수 있습니다.
   - 노드 깊이 및 상위 항목 깊이 간의 깊이 차(즉, 하위 트리의 루트로부터의 거리)를 포함시킬 수 있습니다.

   결과 테이블은 다음과 같습니다.

   HierarchyBelongsTo 접두사를 사용하여 로드된 데이터를 보여 주는 테이블



NodeID	NodeName	BelongsTo	BelongsToID
1	The World	The World	1
2	Africa	Africa	2
2	Africa	The World	1
3	Algeria	Africa	2
3	Algeria	Algeria	3
3	Algeria	The World	1
4	Morocco	Africa	2
4	Morocco	Morocco	4
4	Morocco	The World	1
5	Atlas Mountains	Africa	2
5	Atlas Mountains	Atlas Mountains	5
5	Atlas Mountains	Morocco	4
5	Atlas Mountains	The World	1

인증

계층 구조가 인증에 사용되는 것이 드문 일은 아닙니다. 일례로 조직 계층 구조가 있습니다. 각 관리자는 자신의 부서(모든 하위 부서 포함)에 속하는 모든 것을 볼 수 있는 권한을 가져야 합니다. 그러나 다른 부서를 볼 권한을 가질 필요는 없습니다.

조직 계층 구조의 예

따라서 여러 사람이 해당 조직의 여러 하위 트리를 볼 수 있게 됩니다. 인증 테이블은 다음과 같을 수 있습니다.

인증 테이블

ACCESS	NTNAME	PERSON	POSITION	PERMISSIONS
사용자	ACME\JRL	John	CPO	HR
사용자	ACME\CAH	Carol	CEO	CEO
사용자	ACME\JER	James	엔지니어링 임원	엔지니어링
사용자	ACME\DBK	Diana	CFO	재무
사용자	ACME\RNL	Bob	COO	판매
사용자	ACME\LFD	Larry	CTO	제품

이 사례에서 *Carol*은 *CEO* 및 그 이하에 속하는 모든 내용을 볼 수 있으며, *Larry*는 *Product* 조직, *James*는 *Engineering* 조직만 볼 수 있습니다.

종종 계층 구조는 인접 노드 테이블에 저장됩니다. 이 예에서 이 문제를 해결하려면 HierarchyBelongsTo를 사용하여 인접 노드 테이블을 로드하고 상위 필드 *Tree*의 이름을 지정하기만 하면 됩니다.

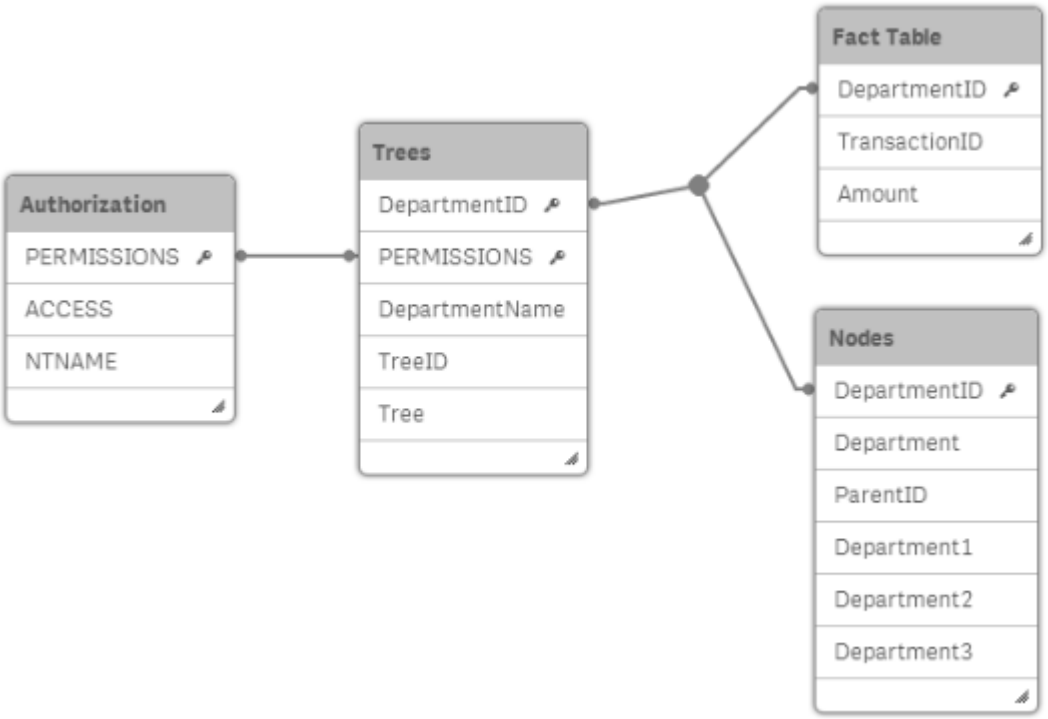
Section Access를 사용하려는 경우 *Tree*의 대문자 복사본을 로드하고 이 새 필드 *PERMISSIONS*를 호출해야 합니다. 마지막으로 인증 테이블을 로드해야 합니다. 다음 마지막 두 단계는 다음 스크립트 줄을 사용하여 수행할 수 있습니다. TempTrees 테이블은 HierarchyBelongsTo 문에 의해 생성된 테이블입니다.

다음은 예일뿐입니다. Qlik Sense에서 완료할 연습은 수반되지 않습니다.

```
Trees:
LOAD *,
    Upper(Tree) as PERMISSIONS
    Resident TempTrees;
Drop Table TempTrees;

Section Access;
Authorization:
LOAD   ACCESS,
       NTNAME,
       UPPER(Permissions) as PERMISSIONS
From Organization;
Section Application;
```

이 예제는 다음 모델을 생성합니다.

데이터 모델: Authorization, Trees, Fact 및 Nodes 테이블



6 QVD 파일

QVD (QlikView Data) 파일은 Qlik Sense 또는 QlikView에서 내보낸 데이터의 테이블을 포함한 파일입니다. QVD는 네이티브 Qlik 형식이며 Qlik Sense 또는 QlikView에서만 읽고 쓸 수 있습니다. 파일 형식은 Qlik Sense 스크립트에서 데이터를 읽는 속도에 최적화되어 있지만 크기는 매우 작습니다. 파일 형식은 스크립트에서 데이터를 읽는 속도에 최적화되어 있지만 크기는 매우 작습니다. QVD 파일에서 데이터를 읽는 속도는 일반적으로 다른 데이터 소스에서 데이터를 읽는 것보다 10-100배 정도 빠릅니다.

QVD 파일은 표준(빠름) 모드와 최적화(매우 빠름) 모드에서 읽을 수 있습니다. 모드 선택은 Qlik Sense 스크립트 엔진에서 자동으로 결정합니다. 최적화 모드는 모든 로드 파일을 아무런 변형(필드에 적용되는 수식) 없이 읽을 때에만 사용할 수 있지만 필드 이름 변경은 허용됩니다. Qlik Sense에서 레코드를 압축 해제하게 만드는 Where 절은 최적화된 로드 또한 비활성화합니다.

QVD 파일은 정확하게 하나의 데이터 테이블을 포함하며 다음 세 가지 부분으로 구성됩니다.

- 테이블의 필드, 후속 정보 레이아웃 및 기타 메타데이터를 설명하는 XML 헤더(UTF-8 문자 집합)
- 바이트 형식의 기호 테이블
- 비트 형식의 실제 테이블 데이터

QVD 파일은 다양한 용도로 사용될 수 있습니다. 네 가지 중요 용도는 쉽게 확인할 수 있습니다. 어느 경우에도 다음 중 둘 이상의 용도를 적용할 수 있습니다.

- 데이터 로드 속도 향상

  QVD 파일에서 입력 데이터의 불변하는 또는 느리게 변하는 블록을 버퍼링하면 큰 데이터 셋에 대한 스크립트 실행 속도가 상당히 빨라집니다.

- 데이터베이스 서버의 부하 감소

  외부 데이터 소스에서 가져오는 데이터의 양이 상당히 감소할 수 있습니다. 이로 인해 외부 데이터베이스의 워크로드와 네트워크 트래픽이 감소하게 됩니다. 게다가 여러 Qlik Sense 스크립트에서 동일한 데이터를 공유할 때 해당 데이터를 소스 데이터베이스에서 QVD 파일로 한 번만 로드하면 됩니다. 다른 응용 프로그램에서도 이 QVD 파일을 통해 동일한 데이터를 사용할 수 있습니다.

- 여러 Qlik Sense 응용 프로그램의 데이터 통합.

  Binary 스크립트 문을 사용하면 한 Qlik Sense 응용 프로그램의 데이터만 다른 프로그램으로 로드할 수 있습니다. 하지만 QVD 파일을 사용하면 Qlik Sense 스크립트가 여러 Qlik Sense 응용 프로그램의 데이터를 조합할 수 있습니다. 이 기능으로 인해 응용 프로그램에서 여러 비즈니스 단위 등의 유사한 데이터를 통합할 수 있게 되었습니다.

- 증분 로드

  많은 경우, QVD 기능을 사용하면 크기가 커지고 있는 데이터베이스에서 새로운 레코드를 배타적으로 로드하여 증분 로드를 수월하게 수행할 수 있습니다.

Qlik Community가 Qlik 응용 프로그램 자동화를 사용하여 QVD 로드 시간을 개선하는 방법을 알아보려면 다시 로드를 개선하기 위해 자동화를 사용하여 QVD를 분할하는 방법을 참조하십시오.

6.1 QVD 파일 만들기

QVD 파일은 두 가지 방법으로 만들 수 있습니다.

- Qlik Sense 스크립트에서 Store 명령을 사용하여 명시적 생성 및 명명.
  이전에 읽은 테이블 또는 그 일부를 선택한 위치의 명시적으로 명명한 파일로 내보내도록 스크립트에 명시합니다.
- 스크립트에서 자동 생성 및 유지 관리.
  load 또는 select 문 앞에 Buffer 접두사를 추가하면 Qlik Sense가 특정 상황에서 데이터를 다시 로드할 때 원본 데이터 소스 대신 사용할 수 있는 QVD 파일이 자동으로 생성됩니다.

위의 방법으로 생성된 QVD 파일들은 읽기 속도에 차이가 없습니다.

Store

이 스크립트 문은 명시적으로 명명된 QVD, CSV 또는 txt 파일을 만듭니다.

구문:

```
Store[ *fieldlist from] table into filename [ format-spec ];
```

또한 이 문은 하나의 데이터 테이블에서만 필드를 내보낼 수 있습니다. 여러 테이블의 필드를 내보내는 경우 내보낼 데이터 테이블을 만들려면 스크립트에서 미리 명시적 조인을 수행해야 합니다.

텍스트 값은 UTF-8 형식의 CSV 파일로 내보냅니다. 구분 기호를 지정할 수 있습니다. 자세한 내용은 **LOAD**를 참조하십시오. CSV 파일에 대한 store 문은 BIFF 내보내기를 지원하지 않습니다.

예:

```
Store mytable into [lib://AttachedFiles/xyz.qvd];
Store * from mytable into [lib://FolderConnection/xyz.qvd];
Store myfield from mytable into 'lib://FolderConnection/xyz.qvd';
Store myfield as renamedfield, myfield2 as renamedfield2 from mytable into
[lib://AttachedFiles/xyz.qvd];
Store mytable into 'lib://FolderConnection/myfile.txt';
Store * from mytable into 'lib://FolderConnection/myfile.csv';
```

다음과 같이 하십시오.

1. *Advanced Scripting Tutorial* 앱을 엽니다.
2. *Product* 스크립트 섹션을 클릭합니다.
3. 스크립트의 끝 부분에 다음을 추가합니다.
   ```
   Store * from Product into [lib://AttachedFiles/ProductData.qvd](qvd);
   ```
   스크립트는 다음과 같이 표시되어야 합니다.

```
CrossTable(Month, Sales)
LOAD
    Product,
    "Jan 2014",
    "Feb 2014",
    "Mar 2014",
    "Apr 2014",
    "May 2014"
FROM [lib://AttachedFiles/Product.xlsx]
(ooxml, embedded labels, table is Product);

Store * from Product into [lib://AttachedFiles/ProductData.qvd](qvd);
```

4. **데이터 로드**를 클릭합니다.
   이제 *Product.qvd* 파일이 파일 목록에 표시됩니다.

   이 데이터 파일은 **Crosstable** 스크립트의 결과이며 각 범주(Product, Month, Sales)에 대해 하나의 열을 갖는 3열 테이블입니다. 이제 이 데이터 파일을 전체 *Product* 스크립트 섹션을 바꾸는 데 사용할 수 있습니다.

6.2 QVD 파일에서 데이터 읽기

다음과 같은 방법을 통해 Qlik Sense에서 QVD 파일을 읽거나 액세스할 수 있습니다.

- 명시적 데이터 소스로 QVD 파일 로드. QVD 파일은 다른 형식의 텍스트 파일(csv, fix, dif, biff 등)과 마찬가지로 Qlik Sense 스크립트에서 load 문을 통해 참조할 수 있습니다.

  예:

  ```
  LOAD * from 'lib://FolderConnection/xyz.qvd' (qvd);
  LOAD fieldname1, fieldname2 from [lib://FolderConnection/xyz.qvd] (qvd);
  LOAD fieldname1 as newfieldname1, fieldname2 as newfieldname2 from
  [lib://AttachedFiles/xyz.qvd](qvd);
  ```

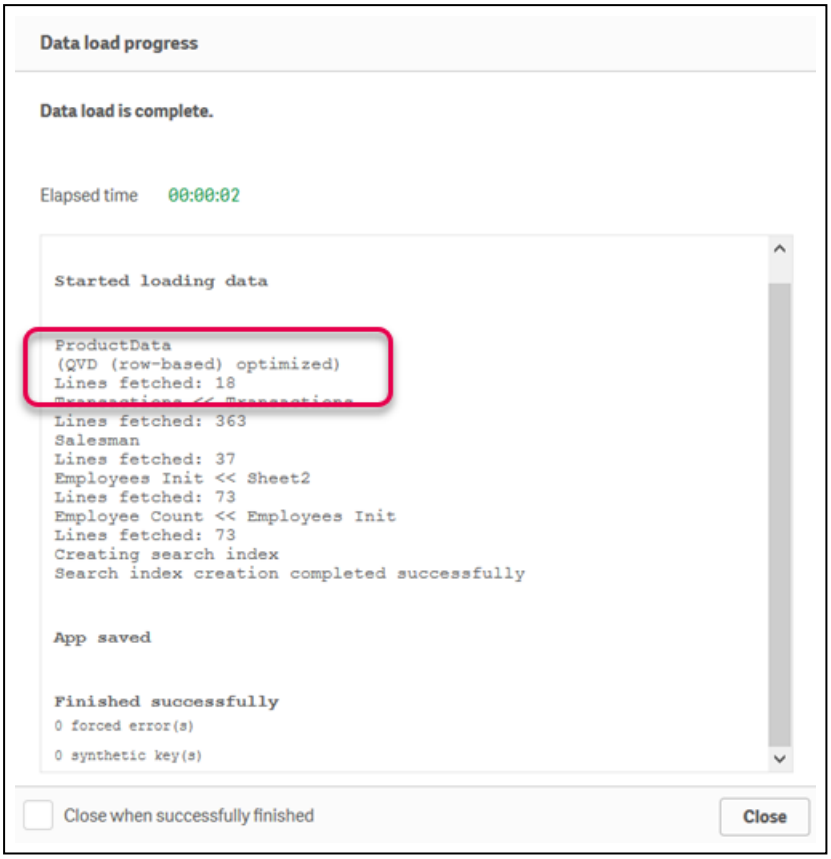
- 버퍼링된 QVD 파일 자동 로드. load 또는 select 문에 buffer 접두사를 사용하면 읽기를 위한 명시적 문을 사용할 필요가 없습니다. 원래 LOAD 또는 SELECT 문을 사용하여 데이터를 얻는 것과는 대조적으로 Qlik Sense에서는 QVD 파일의 데이터 사용 범위를 결정합니다.
- 스크립트에서 QVD 파일 액세스. 여러 스크립트 함수(모두 QVD로 시작)를 QVD 파일의 XML 헤더에 있는 데이터에서 다양한 정보를 검색하는 데 사용할 수 있습니다.

다음과 같이 하십시오.

1. *Product* 스크립트 섹션의 전체 스크립트를 주석 처리합니다.
2. 다음 스크립트를 입력합니다.
   ```
   Load * from [lib://AttachedFiles/ProductData.qvd](qvd);
   ```

3. **데이터 로드**를 클릭합니다.
   QVD 파일에서 데이터가 로드됩니다.

데이터 로드 진행률 창



증분 로드에 QVD 파일 사용에 대한 자세한 내용은 Qlik Community의 다음 블로그 게시물을 참조하십시오. Overview of Qlik Incremental Loading *(Qlik 증분 로드 개요)*

Buffer

QVD 파일은 Buffer 접두사를 통해 자동으로 생성 및 유지 관리할 수 있습니다. 이 접두사는 스크립트에서 대부분의 LOAD 및 SELECT 문에 사용할 수 있습니다. 이 접두사는 해당 문의 결과를 캐시/버퍼링하는 데 QVD 파일을 사용하도록 지정합니다.

구문:

```
Buffer [ (option [ , option])] ( loadstatement | selectstatement )
       option::= incremental | stale [after] amount [(days | hours)]
```

옵션이 사용되지 않으면 스크립트가 처음 실행될 때 생성된 QVD 버퍼가 무제한으로 사용됩니다.

```
Buffer load * from MyTable;
```

stale [after] amount [(days | hours)]

Amount는 기간을 지정하는 수입니다. Decimals를 사용할 수 있습니다. 단위를 생략한 경우 days가 사용됩니다.

stale after 옵션은 원본 데이터에 단순 타임스탬프가 없는 데이터베이스 소스에서 일반적으로 사용됩니다. stale after 절은 QVD 버퍼가 생성된 이후 기간이 얼마나 지나면 더 이상 유효하지 않은 것으로 간주할지 지정합니다. 이 기간이 되기 전까지는 QVD 버퍼가 데이터 소스로 사용되며 그 이후로는 원본 데이터 소스가 사용됩니다. 그러면 QVD 버퍼 파일이 자동으로 업데이트되고 새로운 기간이 시작됩니다.

```
Buffer (stale after 7 days) load * from MyTable;
```

Incremental

incremental 옵션은 원본 파일의 일부분만 읽는 기능을 사용합니다. 파일의 이전 크기는 QVD 파일의 XML 헤더에 저장됩니다. 이 기능은 특히 로그 파일에 유용합니다. 이전 항목에서 로드된 모든 레코드는 QVD 파일에서 읽어들이고 새로운 후속 레코드는 원본 소스에서 읽어들여서 최종적으로 업데이트된 QVD 파일을 만듭니다.

incremental 옵션은 LOAD 문 및 텍스트 파일에만 사용할 수 있으며 증분 로드는 이전 데이터가 변경 또는 삭제된 경우 사용할 수 없습니다.

```
Buffer (incremental) load * from MyLog.log;
```

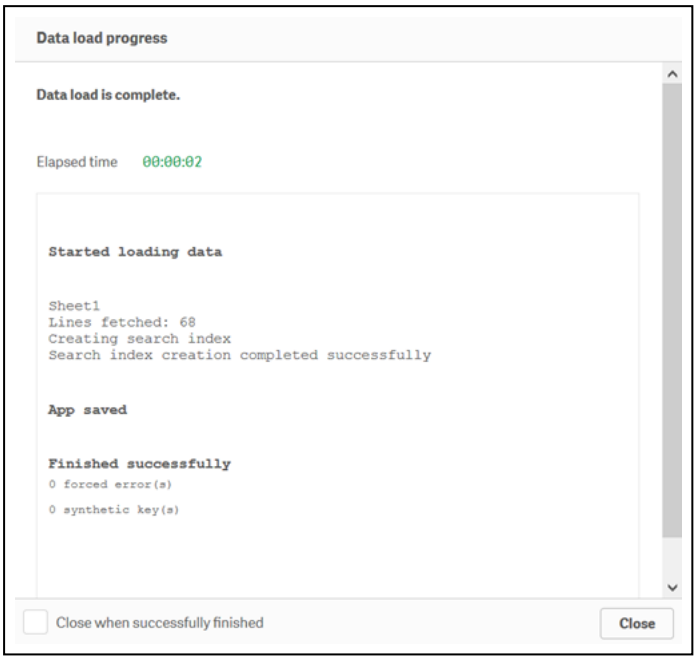
일반적으로 QVD 버퍼는 앱 내에서 버퍼를 만든 전체 스크립트가 실행되는 동안 더 이상 참조되지 않거나 버퍼를 만든 앱이 더 이상 존재하지 않으면 제거됩니다. Store 문은 버퍼의 내용을 QVD 또는 CSV 파일로 유지하려는 경우에 사용해야 합니다.

다음과 같이 하십시오.

1. 새 앱을 만들고 이름을 지정합니다.
2. **데이터 로드 편집기**에서 새 스크립트 섹션을 추가합니다.
3. 오른쪽 메뉴의 **AttachedFiles**에서 **데이터 선택**을 클릭합니다.
4. 업로드한 다음 *Cutlery.xlsx*를 선택합니다.
5. **데이터 선택** 창에서 **스크립트 삽입**을 클릭합니다.
6. LOAD 문의 필드를 주석 처리하고 LOAD 문을 다음으로 변경합니다.
   ```
   Buffer LOAD *
   ```
   스크립트는 다음과 같이 표시되어야 합니다.
   ```
   Buffer LOAD *
          //     "date",
          //     item,
          //     quantity
          FROM [lib://AttachedFiles/Cutlery.xlsx]
          (ooxml, embedded labels, table is Sheet1);
   ```
7. **데이터 로드**를 클릭합니다.

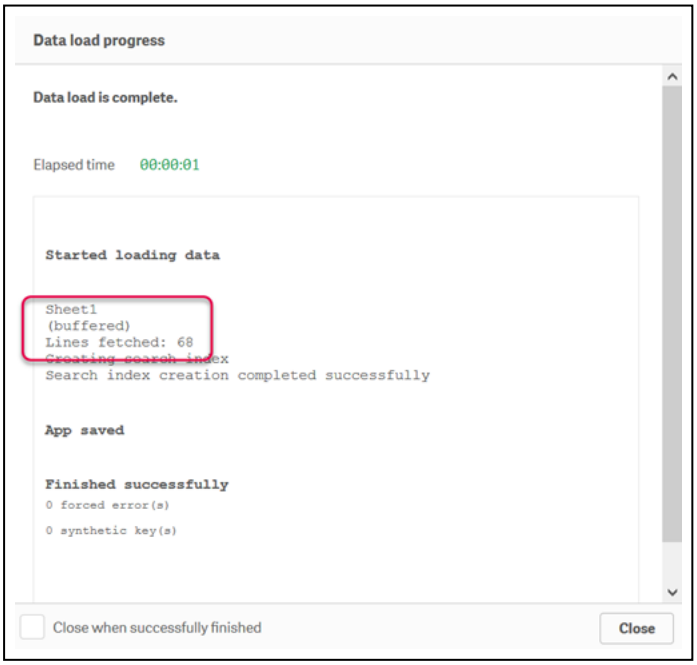
데이터를 처음 로드하는 경우 *Cutlery.xlsx*에서 로드됩니다.

데이터 로드 진행률 창



Buffer 문도 QVD 파일을 만들고 Qlik Sense에 저장합니다. Qlik Sense Enterprise on Windows 배포에서는 Qlik Sense 서버의 디렉터리에 저장됩니다.

8. **데이터 로드**를 다시 클릭합니다.
9. 데이터를 처음 로드한 경우 이번에는 Buffer 문에서 만든 QVD 파일에서 데이터가 로드됩니다.
   데이터 로드 진행률 창



6.3 감사합니다.

이제 이 자습서가 끝났습니다. Qlik Sense에서 스크립트를 작성하는 데 필요한 많은 지식을 얻으셨기 바랍니다. 추가적으로 제공되는 다양한 교육에 대한 자세한 정보를 얻으려면 Qlik 웹 사이트를 방문하십시오.