

Working with Power BI Dataflows



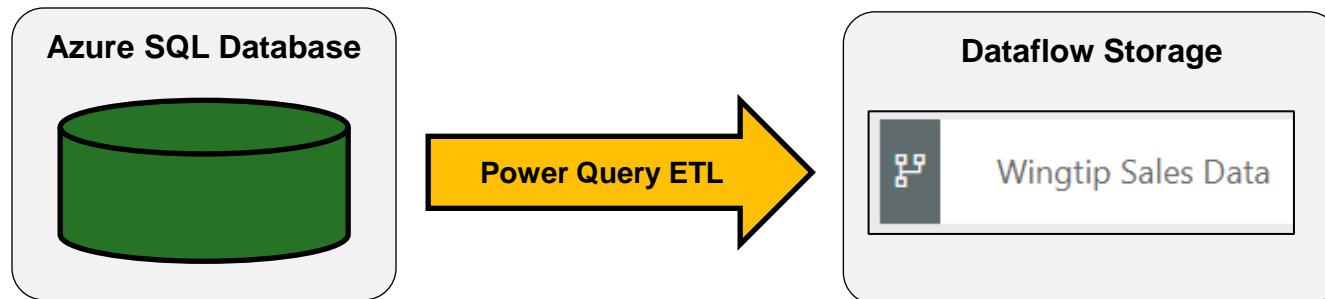
Agenda

- Understanding Dataflow Architecture
- Creating and Consuming Dataflows
- Importing and Exporting Dataflows
- Using Premium Dataflow Features
- Creating Dataflows using Code



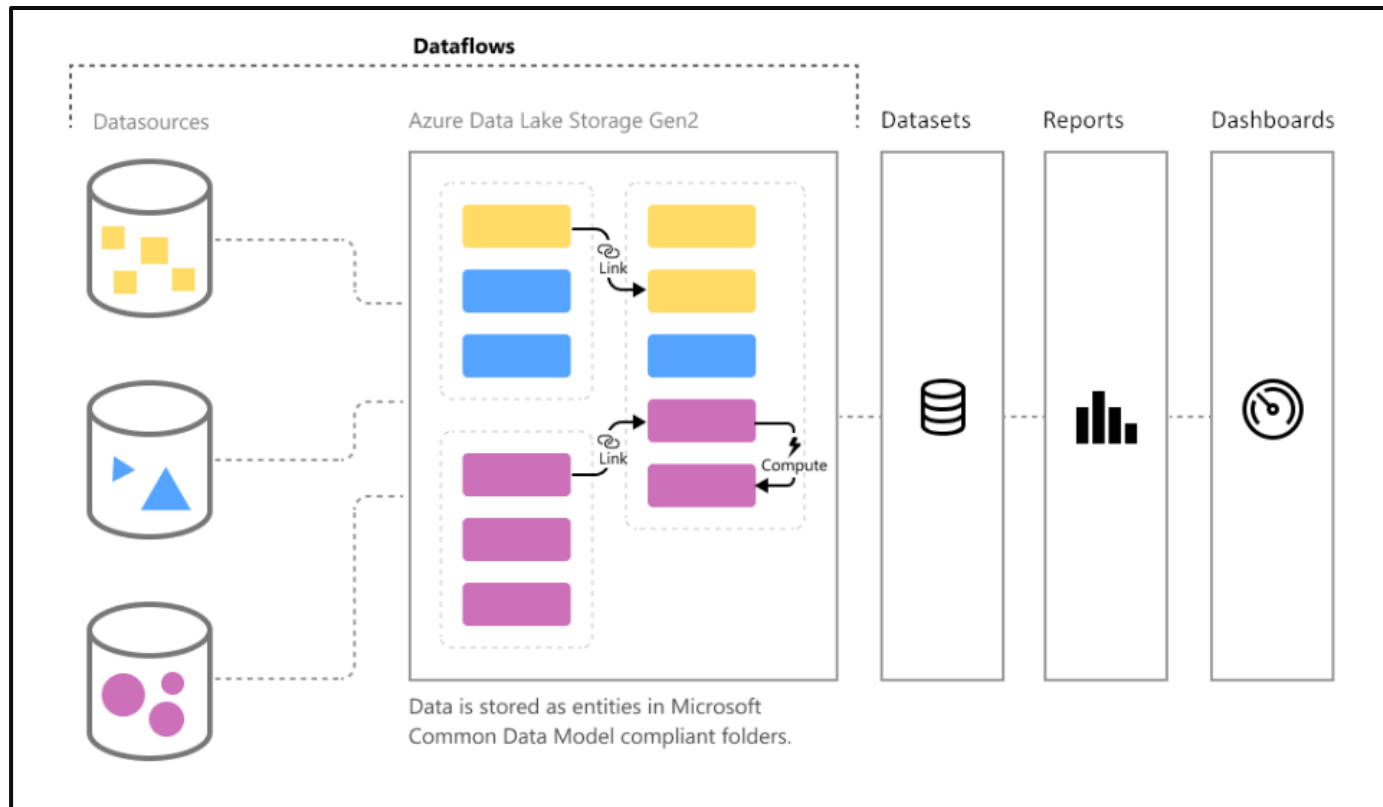
Motivation for Dataflows

- Dataflows provide self-service ETL that scales
 - Intuitive and familiar authoring using Power Query
 - Seamless integration with Power BI Desktop
 - Dataflow storage designed for interoperability & big data
 - Data storage format based on Common Data Model



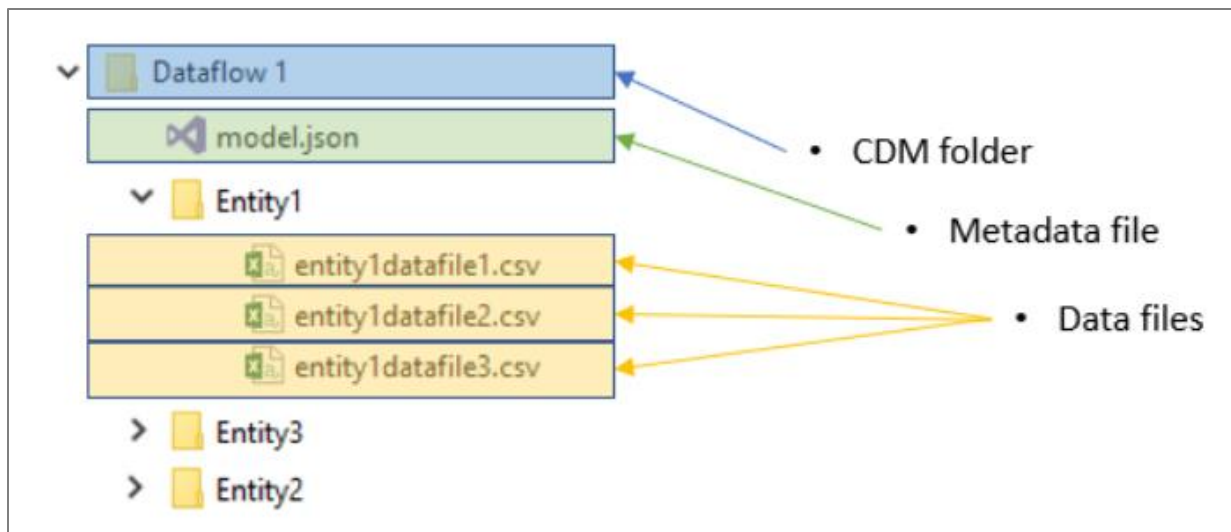
Dataflow Architecture

- Dataflows use Azure Data Lake Gen2 Storage
 - Storage designed to meet requirements of big data
 - Dataflows serialized in format defined by Common Data Model (CDM)



Dataflow Storage Details

- Serialization format is defined by the Common Data Model specification
 - Dataflow metadata stored in **model.json** file
 - Dataflow data rows stored in CSV files
 - By default, Power BI manages dataflow storage behind the scenes
- Dataflows can be configured to write data to Azure storage account
 - Allows access to dataflows by service other than Power BI
 - Allows with an organization to store larger data volumes



Common Data Model Metadata

- Dataflow output is stored in CDM format
 - model.json file contains metadata about entities
 - model.json file contains M code for queries

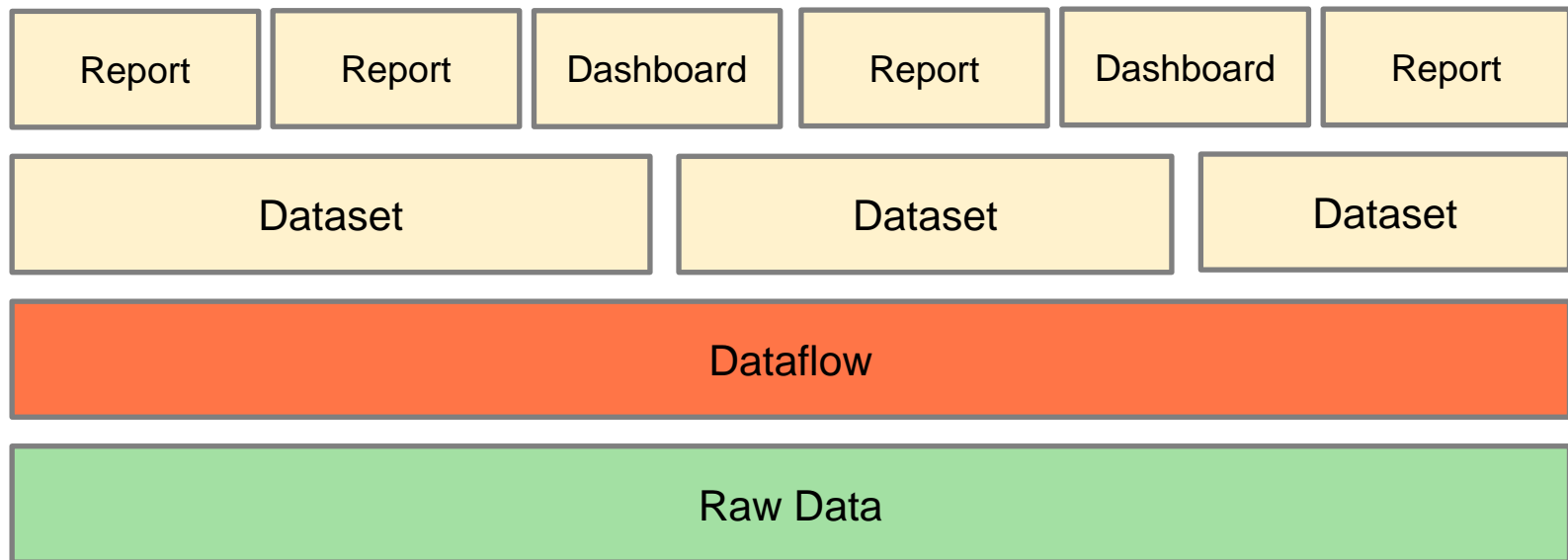
```
{
  "name": "Wingtip Sales Dataflow",
  "description": "A sample dataflow",
  "version": "1.0",
  "culture": "en-US",
  "modifiedTime": "2019-10-21T17:54:50.1618626+00:00",
  "pbi:mashup": {
    "fastCombine": false,
    "allowNativeQueries": false,
    "queriesMetadata": {
      "Customers": { "queryId": "58d2a7e0-0298-4d94-8285-7af1f3d54b15", "query": "SELECT * FROM Customers" },
      "Products": { "queryId": "10577951-df4b-407c-b6fb-c923880ba1ed", "query": "SELECT * FROM Products" },
      "Orders": { "queryId": "ad08816d-be0d-4f6f-b19e-755f23c8fb0f", "query": "SELECT * FROM Orders" },
      "Sales": { "queryId": "4613190e-da33-4a3a-af5d-a567cbde4dd2", "query": "SELECT * FROM Sales" }
    }
  },
  "document": "section Section1;\r\nshared Customers = let\r\n Source = s",
  "entities": [
    {
      "type": "LocalEntity",
      "name": "Customers",
```

```
{
  "type": "LocalEntity", "name": "Products", "description": "",
  "pbi:refreshPolicy": { "type": "FullRefreshPolicy", "location": "Products.csv" },
  "attributes": [
    { "name": "ProductId", "dataType": "int64" },
    { "name": "Product", "dataType": "string" },
    { "name": "Description", "dataType": "string" },
    { "name": "Category", "dataType": "string" },
    { "name": "Subcategory", "dataType": "string" },
    { "name": "UnitCost", "dataType": "decimal" },
    { "name": "ListPrice", "dataType": "decimal" },
    { "name": "Product Image", "dataType": "string" }
  ],
  "partitions": [
    {
      "name": "Part001",
      "refreshTime": "2019-10-21T17:59:55.5031318+00:00",
      "location": "https://wabieus2cdsap1.blob.core.windows.net:443/913b7aae-5"
    }
  ]
},
```



Designing Power BI Solutions with Dataflows

- Dataflows used to collect all data
 - Power BI Desktop projects import data from dataflows
 - Complex ETL work no longer required in Power BI Desktop projects



Dataflow Benefits

- Replaces other ETL tools (*Azure Data Factory, Power Automate*)
- Decouples ETL work from datasets in PBIX projects
- Enable sharing of source tables of data between datasets
- Reduces number of queries on live data sources
- Centralizes efforts to clean and prepare data
- Share tables that have no source (calendar tables)



Dataflow Disadvantages

- Adds extra complexity
- Data must be refreshed in 2 phases
- Does not support data modelling features of DAX
- Some features require dedicated capacities



Licensing with Dataflows

- Dataflow creation requires Power BI pro
 - Dataflows can only be created in app workspaces
 - Dataflows cannot be created in personal workspaces
- Dedicated capacity add extra dataflow features
 - Ability to handle larger data volumes
 - Better refresh performance
 - Linked and computed entities
 - AI features to transform data



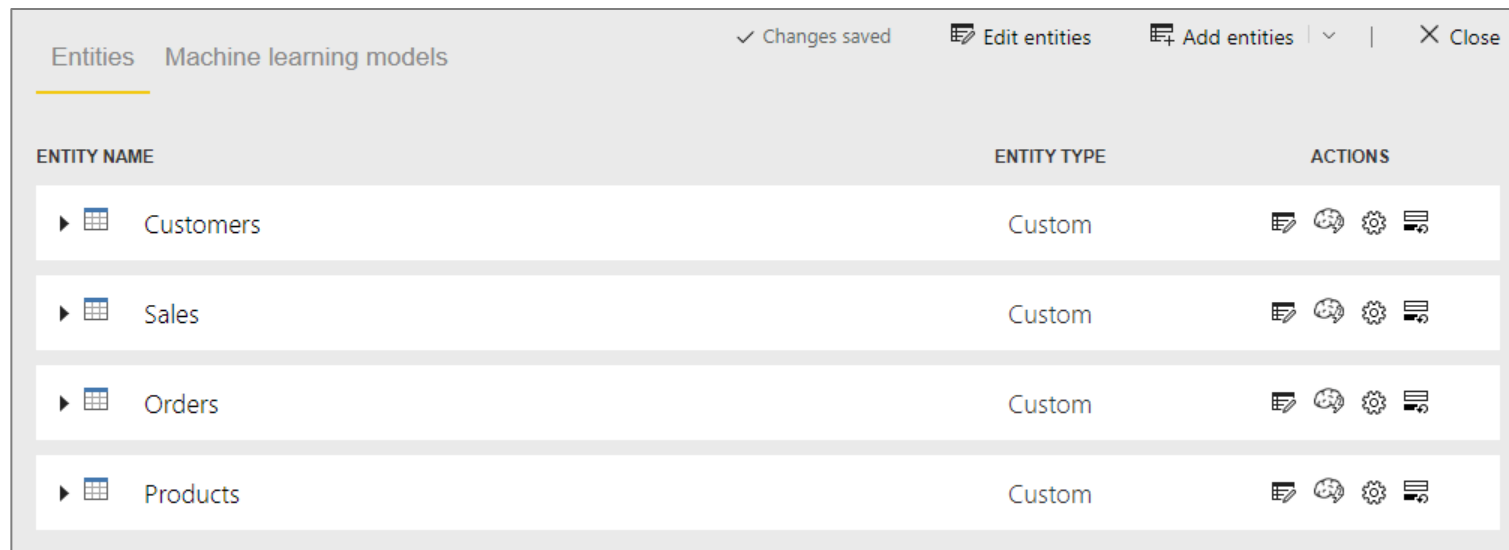
Agenda





















- ✓ Understanding Dataflow Architecture
- Creating and Consuming Dataflows
 - Importing and Exporting Dataflows
 - Using Premium Dataflow Features
 - Creating Dataflows using Code



Dataflow Entities

- A **dataflow** exists within a workspace
 - Dataflow contains one or more **entities**
 - Entity is a table with well-defined schema
 - Entity data populated by running a query (M code)



ENTITY NAME	ENTITY TYPE	ACTIONS
▶  Customers	Custom	   
▶  Sales	Custom	   
▶  Orders	Custom	   
▶  Products	Custom	   



Power Query in the Browser

- Editing experience familiar to Power Query users

The screenshot shows the 'Edit queries' window in Power Query. The window has a title bar 'Power Query' and a close button. Below the title bar is a ribbon with tabs: 'Get data', 'Manage parameters', 'Refresh', 'Options', 'Manage columns', 'Transform table', 'Reduce rows', 'Add column', and 'AI insights'. The 'Transform table' tab is active. The main area displays a table with columns: CustomerId, Customer, City, State, Zipcode, Gender, BirthDate, and Customer Type. The table contains 17 rows of customer data. On the left, there is a 'Queries' pane with a list of queries: Customers, Sales, Orders, and Products. On the right, there is a 'Query settings' pane with fields for 'Name' (Customers) and 'Entity type' (Custom). Below these is a list of 'Applied steps' including Source, Navigation, Choose columns, Merged columns, Replaced Female Values, Replaced Male Values, Changed column type, Inserted conditional columns, and Removed columns. The 'Changed column type 1' step is selected. At the bottom right, there are 'Cancel' and 'Save & close' buttons.

Power Query

Edit queries

Get data Manage parameters Refresh Options Manage columns Transform table Reduce rows Add column AI insights

Queries

- Customers
- Sales
- Orders
- Products

Table.TransformColumnTypes(#"Removed columns", {"Customer Type", type text})

	CustomerId	Customer	City	State	Zipcode	Gender	BirthDate	Customer Type
1	1	Nina Diaz	Eureka	CA	95501	Female	4/11/1966	One-time Customer
2	2	Melinda Carter	Napa	CA	94558	Female	6/6/1976	One-time Customer
3	3	Pam Miller	Napa	CA	94558	Female	9/8/1952	One-time Customer
4	4	Merle Blackwell	Sacramento	CA	95823	Female	9/12/1939	One-time Customer
5	5	Ariel Hale	Sacramento	CA	95818	Male	9/15/1965	One-time Customer
6	6	Randy Carter	Sacramento	CA	95818	Male	7/14/1953	One-time Customer
7	7	Lillie Hinton	Eureka	CA	95501	Female	2/3/1992	One-time Customer
8	8	Ladonna Moody	Napa	CA	94559	Female	4/5/1949	One-time Customer
9	9	Buddy McKay	Bend	OR	97701	Male	5/10/1989	One-time Customer
10	10	Warren Sykes	Sacramento	CA	95818	Male	6/17/1960	One-time Customer
11	11	Jan Rutledge	Portland	OR	97216	Female	11/26/1981	One-time Customer
12	12	Dallas Lester	Eugene	OR	97402	Male	3/26/1973	One-time Customer
13	13	Matthew Zimmerm...	Portland	OR	97220	Male	4/5/1988	One-time Customer
14	14	Sheryl Hernandez	Sacramento	CA	95823	Female	11/8/1974	One-time Customer
15	15	Bradley Cannon	Redmond	WA	98052	Male	11/10/1957	One-time Customer
16	16	Markus Soto	Beaverton	OR	97005	Male	5/12/1975	One-time Customer
17	17	Patricia Davis	Medford	OR	97504	Female	6/13/1973	One-time Customer

Query settings

Name: Customers

Entity type: Custom

Applied steps


- Source
- Navigation
- Choose columns
- Merged columns
- Replaced Female Values
- Replaced Male Values
- Changed column type
- Inserted conditional columns
- Removed columns
- Changed column type 1

Cancel Save & close



Refreshing a Dataflow

- Dataflow can be refreshed on demand or scheduled

NAME ↑	ACTIONS	LAST REFRESH	NEXT REFRESH
 Wingtip Sales Data	  ...	N/A	N/A

- Dataflow displays last refresh time

Dashboards

Reports

Workbooks

Datasets

Dataflows

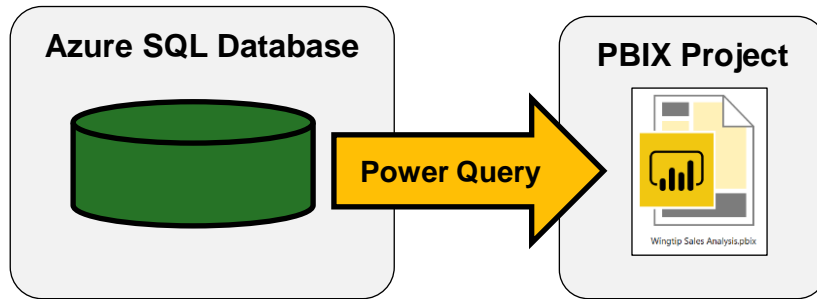
Showing 1 item

NAME ↑	ACTIONS	LAST REFRESH	NEXT REFRESH
<div><div><div></div></div><div>Wingtip Sales Data</div></div>	<div><div><div></div></div><div><div></div><div></div></div><div>...</div><div><div></div></div></div>	2/17/2020, 3:39:12	N/A

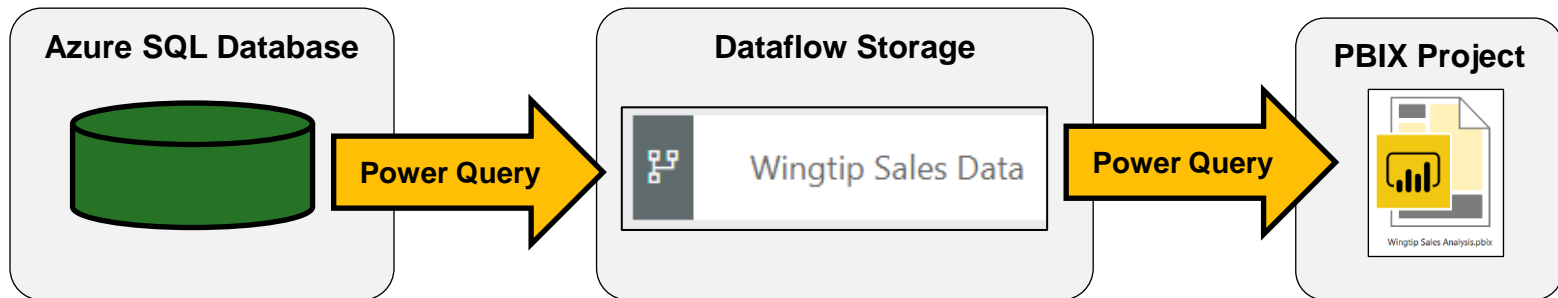


Dataflows and Power BI Desktop

- Power BI Desktop project without dataflow

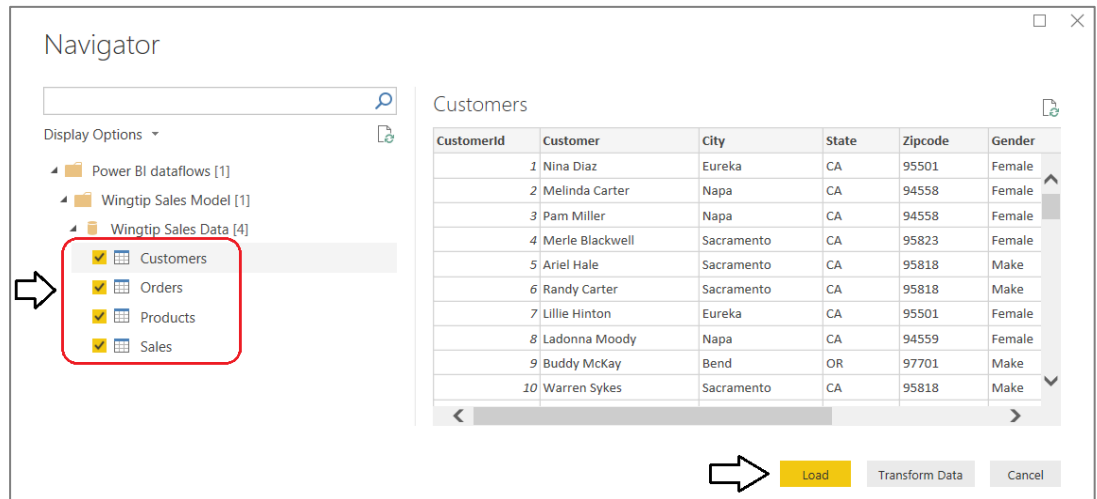
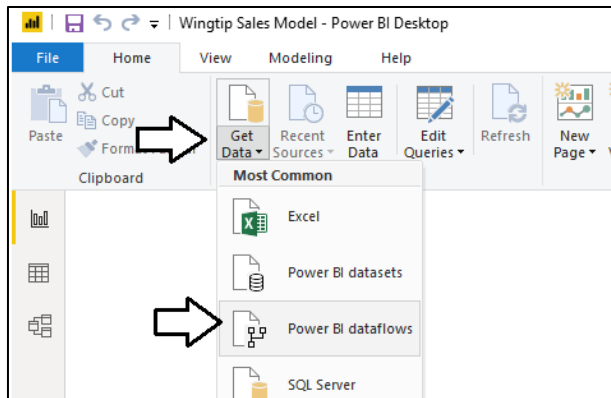


- Power BI Desktop project with dataflow

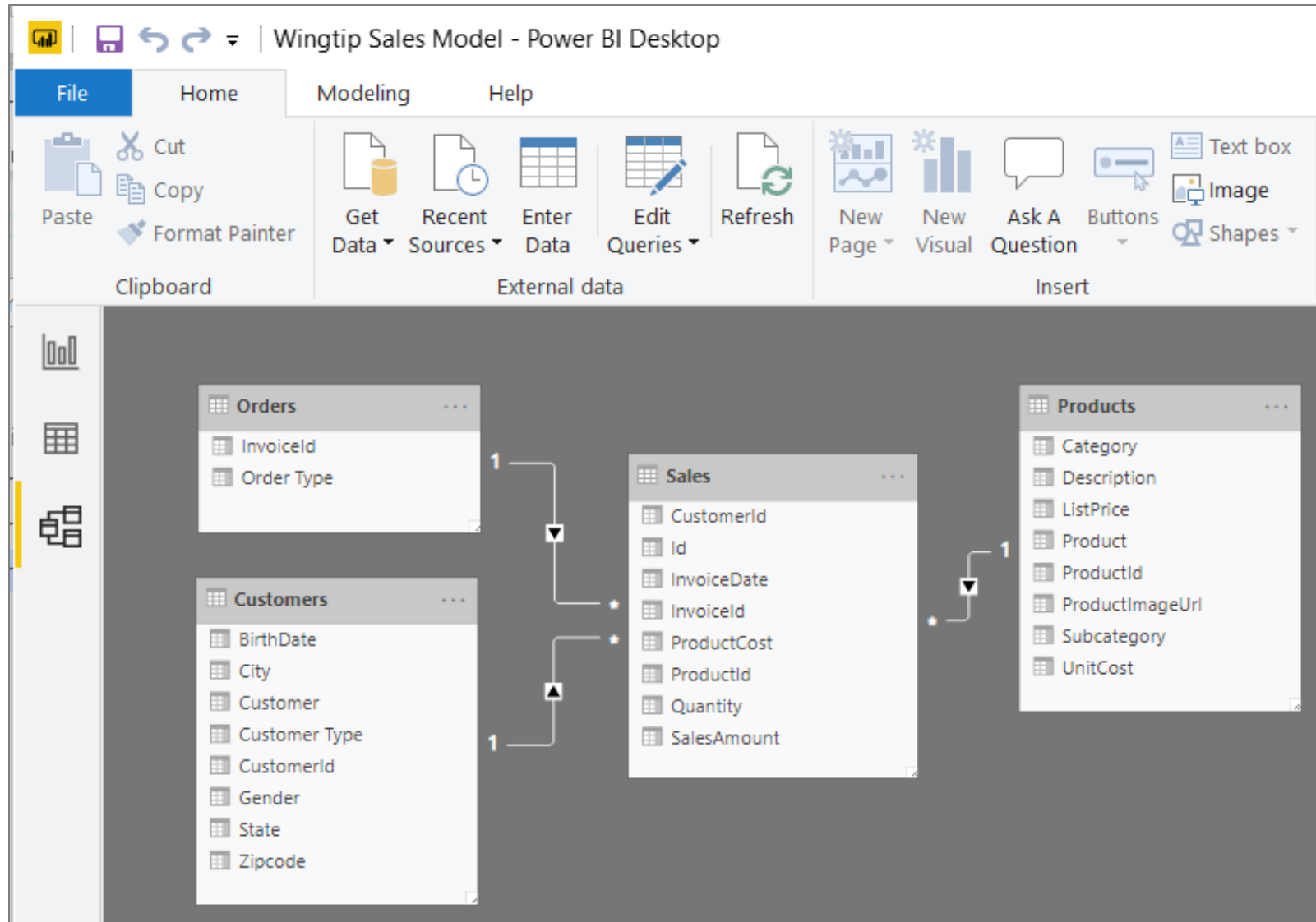


Consuming Dataflows

- Dataflow entities consumed by Power BI Desktop
 - Use **Power BI dataflows** data source
 - Dataset with dataflow can be published to any app workspace



Dataflow Provides Starting Point for Data Model



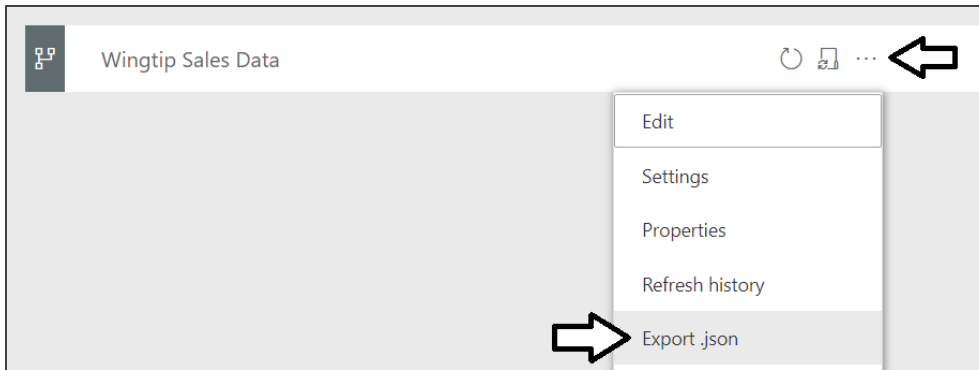
Agenda

- ✓ Understanding Dataflow Architecture
- ✓ Creating and Consuming Dataflows
- Importing and Exporting Dataflows
 - Using Premium Dataflow Features
 - Creating Dataflows using Code

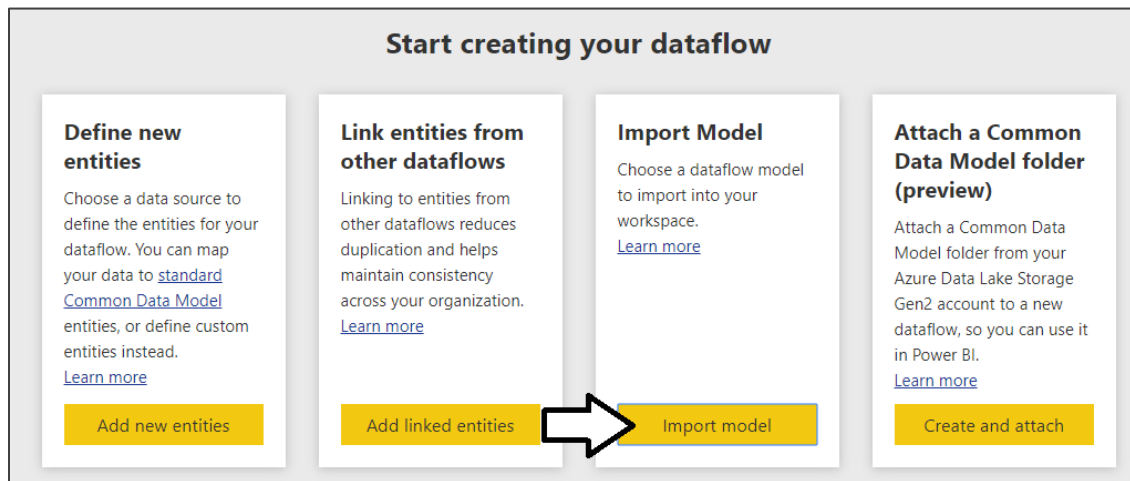


Importing and Exporting Dataflows

- Dataflow can be exported as model.json



- model.json can be imported to create new dataflow



Agenda

- ✓ Understanding Dataflow Architecture
- ✓ Creating and Consuming Dataflows
- ✓ Importing and Exporting Dataflows
- Using Premium Dataflow Features
 - Creating Dataflows using Code



Dataflow Premium Features

- Linked entities
- Computed entities
- Artificial intelligence (AI) features
- Incremental refresh
- Parallel execution of transforms



Linked Entities

- Linked entities let you share data between
 - Different dataflows in the same Workspace
 - Different dataflows in different Workspaces
- Creating linked entity does not duplicate source data
 - You can use existing entity in another workspace as a source
 - Uses the same M code as dataset uses to get data from an entity
 - Linked entities are read-only
 - If you want further transformations you create computed entity
 - Diagram view makes it easy to see usage of linked entities



Computed Entities

- Computed entities built on top of other entities
 - Allows entities use other entities as data sources
 - Computed entity uses the persisted output of the source entity
- Useful scenarios
 - You are creating multiple entities within a dataflow from the same raw data, and don't want to get data from the original data source more than once
 - You are hitting problems as a result of the Power Query engine's habit of requesting data multiple times during a single query execution



AI Features

- Dataflows in Power BI Premium includes AI features
 - Call Cognitive Services functions
 - Detect sentiment, language, tag images
 - Extract key phrases

AI insights


Cognitive Services [4]

- CognitiveServices.TagImages
- CognitiveServices.ExtractKeyPhrases
- CognitiveServices.DetectLanguage
- CognitiveServices.ScoreSentiment**

CognitiveServices.ScoreSentiment


Measure the positive or negative sentiment of words and phrases. Sentiment is rated on a scale of zero to one, with one being the most positive.

Text *

 Description

▼

LanguageISOCode

 en-US

▼

Show ▼

Apply

Cancel



Incremental Refresh

- Incremental refresh can be configured for entities
 - Very helpful to speed up refresh time for large tables

Incremental refresh settings

Sales

Incremental refresh updates only data that's changed, to speed refresh, reduce capacity usage, and store historic data. [Learn more](#)

☒ On

Choose a DateTime field to filter by

Timestamp ▼

Store rows from the past

4 Years ▼

Refresh rows from the past

7 Days ▼



Agenda

- ✓ Understanding Dataflow Architecture
- ✓ Creating and Consuming Dataflows
- ✓ Importing and Exporting Dataflows
- ✓ Using Premium Dataflow Features
- Creating Dataflows using Code



Creating Dataflows using Code

- Power BI Service API provides dataflow management
 - Dataflows created using Imports and model.json file

```
static void CreateDataflow(Guid appworkspaceId, string modelJsonPath) {  
    string AccessToken = GetAccessTokenWithUserPassword(scopesManageWorkspaceAssets);  
    var pbiClient = new PowerBIClient(new Uri(urlPowerBiRestApiRoot),  
                                     new TokenCredentials(AccessToken, "Bearer"));  
    MemoryStream stream = new MemoryStream(Properties.Resources.model_json);  
    var import = pbiClient.Imports.PostImportWithFileInGroup(appworkspaceId, stream, "model.json");  
    Console.WriteLine("Dataflow creation completed");  
}
```

- Dataflows can be refreshed on demand with code

```
RefreshRequest refreshRequest = new RefreshRequest(NotifyOption.MailOnCompletion);  
pbiClient.Dataflows.RefreshDataflowAsync(appworkspaceId, dataflow.ObjectId, refreshRequest);
```



Summary

- Understanding Dataflow Architecture
- Creating and Consuming Dataflows
- Importing and Exporting Dataflows
- Using Premium Dataflow Features
- Creating Dataflows using Code

