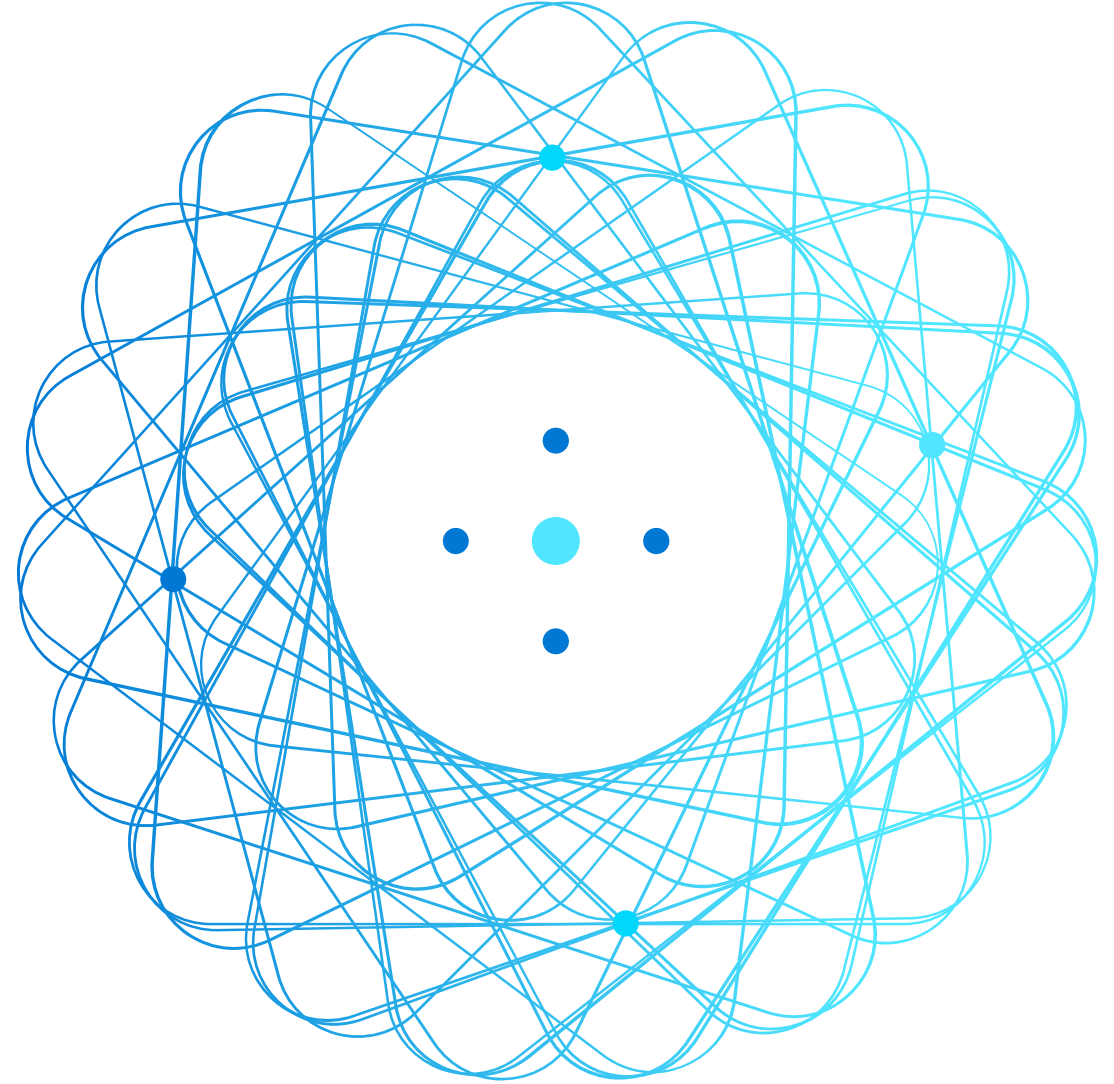


Work with data warehouses using Azure Synapse Analytics



Agenda



Analyze data in a relational data warehouse



Load data into a relational data warehouse

Analyze data in a relational data warehouse



Design a data warehouse schema

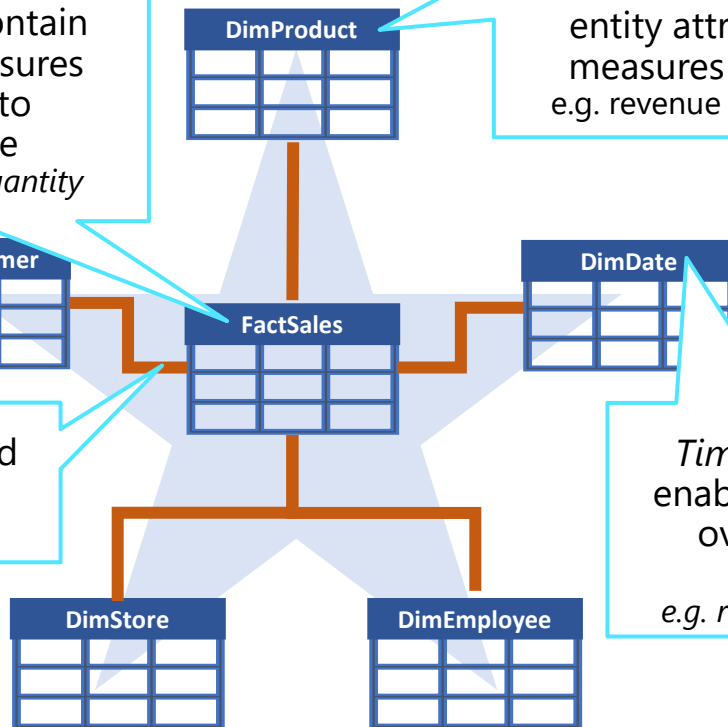
Star schema

Fact tables contain numeric measures you want to aggregate
e.g. revenue, quantity

Dimension tables contain entity attributes by which measures are aggregated
e.g. revenue by *product category*

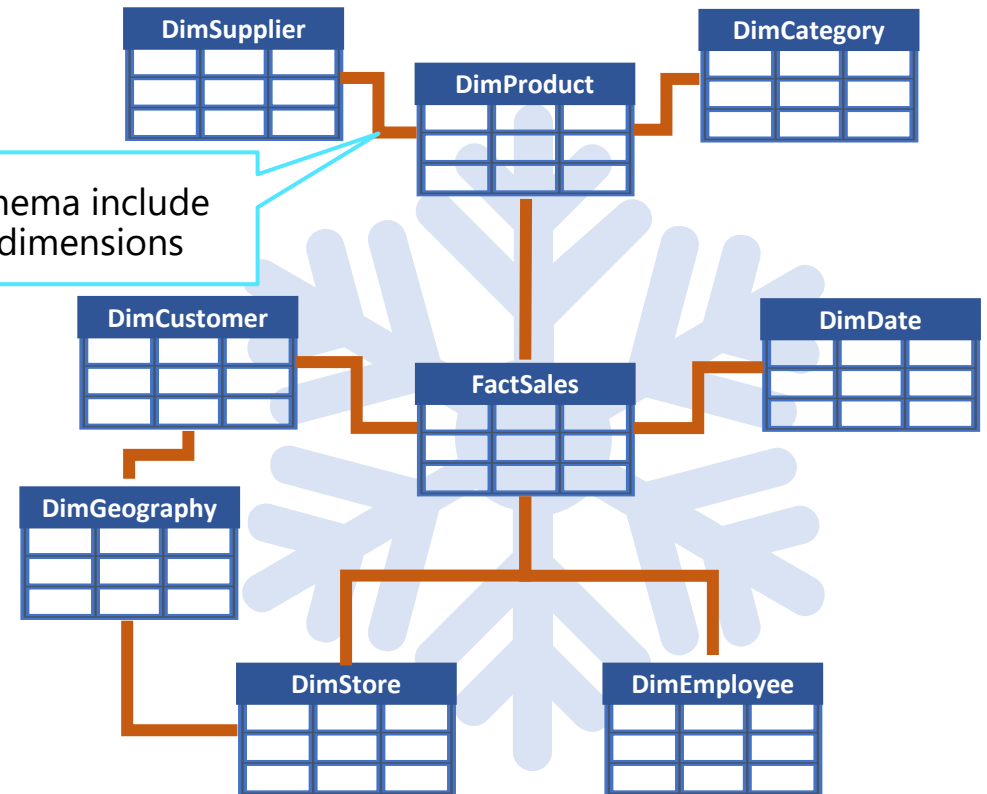
Facts are related to dimensions using *keys*

Time dimensions enable aggregation over temporal periods
e.g. revenue by *month*



Snowflake schema

Snowflake schema include *normalized* dimensions



Dimension keys

Surrogate key

- Uniquely identifies an instance of a dimension entity (i.e. a row)
- Usually a simple integer value
- Must be unique in the dimension table

Alternate key

- Identifies an entity in the operational source system
- Often a *business* key (e.g. a product code or customer ID) or a *natural* key (e.g. a datetime value in a time dimension)
- Can be duplicated in the dimension table to represent the same entity at different points in time

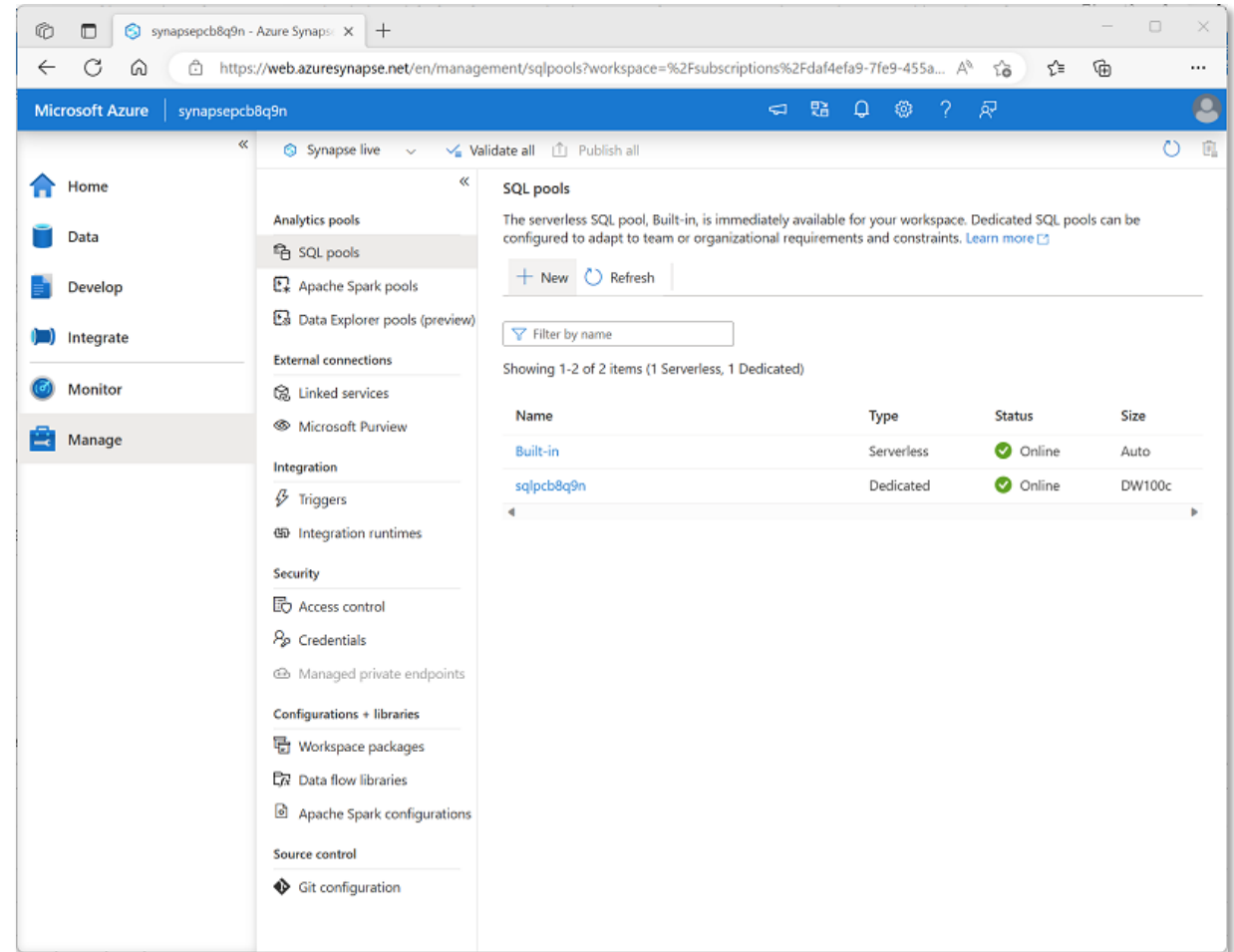
CustomerKey	CustomerAltKey	Name	Email	Street	City	PostalCode	CountryRegion
123	I-543*	Navin Jones	navin1@contoso.com	1 Main St.	Seattle	90000	United States
124	R-589	Mary Smith	mary2@contoso.com	234 190th Ave	Buffalo	50001	United States
125	I-321	Antoine Dubois	antoine1@contoso.com	2 Rue Jolie	Paris	20098	France
126	I-543*	Navin Jones	navin1@contoso.com	24 125th Ave.	New York	50000	United States
...

* This customer moved from Seattle to New York, so a new record with the same alternate key but a new surrogate key was added.

Create a relational data warehouse in Azure Synapse Analytics

Create a *dedicated* SQL pool

- Specify name and size
- Pause and resume pool as needed
- The pool provides a relational database instance in which you can create, load, and query tables



Considerations for creating data warehouse tables

Data integrity constraints

- Foreign key and unique constraints are not supported
- You must implement logic to ensure referential integrity between facts and dimensions

Indexes

- The default index type is CLUSTERED COLUMNSTORE – use this in most cases
- For field types not supported in COLUMNSTORE indexes, use a CLUSTERED index on appropriate columns

Data distribution

- Use **hash** distribution to distribute fact tables across compute nodes
- Use **replicated** distribution for small dimension tables to avoid data shuffling; but for dimension tables too large to store on each compute node, use **hash** distribution
- Use **round-robin** distribution for staging tables to evenly distribute data across compute nodes

```
CREATE TABLE schema.table_name
(
    column_name DATA_TYPE, NULLABILITY,
    ...
)
WITH
(
    DISTRIBUTION = HASH(column_name)
                | REPLICATE
                | ROUND_ROBIN,

    INDEX_TYPE
)
```

External tables

Use external tables to define table metadata for files in a data lake

- Data is managed independently from the table
- Useful for reading data into staging tables directly from the data lake

```
CREATE EXTERNAL DATA SOURCE StagedFiles
WITH (
    LOCATION = 'https://.../file/location'
);
GO

CREATE EXTERNAL FILE FORMAT ParquetFormat
WITH (
    FORMAT_TYPE = PARQUET,
    DATA_COMPRESSION =
        'org.apache.hadoop.io.compress.SnappyCodec'
);
GO

CREATE EXTERNAL TABLE dbo.ExternalStageProduct
(
    ProductID NVARCHAR(10) NOT NULL,
    ProductName NVARCHAR(200) NOT NULL,
    ...
)
WITH
(
    DATA_SOURCE = StagedFiles,
    LOCATION = folder_name/*.parquet',
    FILE_FORMAT = ParquetFormat
);
GO
```


Demo: Explore a data warehouse

You can try this for yourself later by following the instructions at the link below:

<https://aka.ms/mslearn-synapse-dw>



Knowledge check



In which of the following table types should an insurance company store details of customer attributes by which claims will be aggregated?

- ☐ Staging table
- ☒ Dimension table
- ☐ Fact table



You create a dimension table for product data, assigning a unique numeric key for each row in a column named ProductKey. The ProductKey is only defined in the data warehouse. What kind of key is ProductKey?

- ☒ A surrogate key
- ☐ An alternate key
- ☐ A business key



What distribution option would be best for a sales fact table that will contain billions of records?

- ☒ HASH
- ☐ ROUND_ROBIN
- ☐ REPLICATE

Load data into a relational data warehouse



Load staging tables

Use an external table to query files in the data lake



```
CREATE EXTERNAL TABLE dbo.ExternalStageProduct
(
    ProductID NVARCHAR(10) NOT NULL,
    ProductName NVARCHAR(200) NOT NULL,
    ...
)
WITH
(
    DATA_SOURCE = StagedFiles,
    LOCATION = folder_name/*.parquet',
    FILE_FORMAT = ParquetFormat
);
GO
```

or

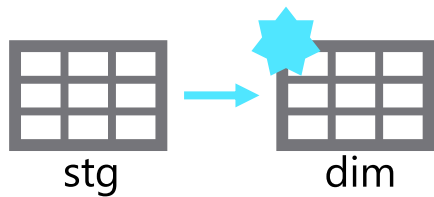
Use COPY to load data from files into the database



```
COPY INTO dbo.StageProduct
(ProductID, ProductName, ...)
FROM 'https://mydatalake.../data/products/*.parquet'
WITH
(
    FILE_TYPE = 'PARQUET',
    MAXERRORS = 0,
    IDENTITY_INSERT = 'OFF'
);
```

Load dimension tables from staging tables

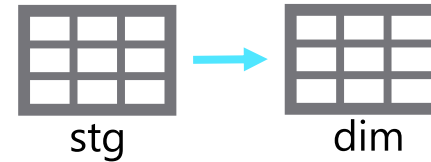
Use CREATE TABLE AS to create a new table



No support for IDENTITY surrogate key generation
Use a UNION to combine staged and existing data, then rename tables to replace old table with new one

or

Use INSERT.. SELECT to load an existing table



Supports IDENTITY surrogate key in dimension table
Easier to implement for repeated loads than CTAS

```
CREATE TABLE dbo.DimProduct
WITH
(
    DISTRIBUTION = REPLICATE,
    CLUSTERED COLUMNSTORE INDEX
)
AS
SELECT ROW_NUMBER() OVER(ORDER BY ProdID) AS ProdKey,
       ProdID AS ProdAltKey,
       ProductName,
       ProductCategory,
       Color,
       Size,
       ListPrice,
       Discontinued
FROM dbo.StageProduct;
```

```
INSERT INTO dbo.DimCustomer
SELECT CustomerNo AS CustomerAltKey,
       CustomerName,
       EmailAddress,
       Phone,
       StreetAddress,
       City,
       PostalCode,
       CountryRegion
FROM dbo.StageCustomers
```

Load time dimensions

- Initialize the table with the required timespan
- Extend by adding new rows periodically as required

Scripting this in SQL may be time-consuming in a dedicated SQL pool – it may be more efficient to prepare the data in Microsoft Excel or an external script and import it using the COPY statement

```
-- Create a temporary table for the dates we need
CREATE TABLE #TmpStageDate (DateVal DATE NOT NULL)

-- Populate the temp table with a range of dates
DECLARE @StartDate DATE
DECLARE @EndDate DATE
SET @StartDate = '2019-01-01'
SET @EndDate = '2022-12-31'
DECLARE @LoopDate DATE
SET @LoopDate = @StartDate
WHILE @LoopDate <= @EndDate
BEGIN
    INSERT INTO #TmpStageDate VALUES
    (
        @LoopDate
    )
    SET @LoopDate = DATEADD(dd, 1, @LoopDate)
END

-- Insert the dates and calculated attributes into the dimension table
INSERT INTO dbo.DimDate
SELECT  CAST(CONVERT(VARCHAR(8), DateVal, 112) AS int) , -- date key
        DateVal, -- date alt key
        Day(DateVal) -- day number of month
        -- ,other derived temporal fields as required
FROM #TmpStageDate
GO
```

Load slowly changing dimension tables

Types of slowly changing dimensions

Type 0: No changes allowed

DateKey	DateAltKey	Day	Month	Year
20230101	01-01-2023	Sunday	January	2023

Type 1: Changes made inline in dimension row

StoreKey	StoreAltKey	StoreName
123	EH199J	High Street Store Town Central Store

Type 2: Changes result in a new version of the dimension entity (a new row)

CustomerKey	CustomerAltKey	Name	Address	City	DateFrom	DateTo	IsCurrent
1211	jo@contoso.com	Jo Smith	9999 Main St	Seattle	20190101	20230105	False
2996	jo@contoso.com	Jo Smith	1234 9 th Ave	Boston	20230106		True

Load slowly changing dimension tables

Loading techniques

Combine INSERT and UPDATE statements

```
-- New customers
INSERT INTO dbo.DimCustomer
SELECT stg.*
FROM dbo.StageCustomers AS stg
WHERE NOT EXISTS
    (SELECT * FROM dbo.DimCustomer AS dim
     WHERE dim.CustomerAltKey = stg.CustNo);

-- Type 1 updates (name)
UPDATE dbo.DimCustomer
SET CustomerName = stg.CustomerName
FROM dbo.StageCustomers AS stg
WHERE dbo.DimCustomer.CustomerAltKey = stg.CustNo;

-- Type 2 updates (StreetAddress)
INSERT INTO dbo.DimCustomer
SELECT stg.*
FROM dbo.StageCustomers AS stg
JOIN dbo.DimCustomer AS dim
ON stg.CustNo = dim.CustomerAltKey
AND stg.StreetAddress <> dim.StreetAddress;
```

or

Use the MERGE statement

```
MERGE dbo.DimProduct AS tgt
      USING (SELECT * FROM dbo.StageProducts) AS src
      ON src.ProductID = tgt.ProductBusinessKey
      WHEN MATCHED THEN
          -- Type 1 updates
          UPDATE SET
              tgt.ProductName = src.ProductName,
              tgt.ProductCategory = src.ProductCategory,
              tgt.Color = src.Color,
              tgt.Size = src.Size,
              tgt.ListPrice = src.ListPrice,
              tgt.Discontinued = src.Discontinued
      WHEN NOT MATCHED THEN
          -- New products
          INSERT VALUES
              (src.ProductID,
               src.ProductName,
               src.ProductCategory,
               src.Color,
               src.Size,
               src.ListPrice,
               src.Discontinued);
```


Load fact tables

- Use an INSERT statement
- Look up surrogate keys in dimension tables based on alternate key:
 - Simple case, get the most recently loaded dimension instance (maximum incrementing surrogate key)
 - Or use an *IsCurrent* flag field
 - Or use *start* and *end* dates to find the right instance for the fact time

```
INSERT INTO dbo.FactSales
SELECT  (SELECT MAX(DateKey)
        FROM dbo.DimDate
        WHERE FullDateAlternateKey = stg.OrderDate) AS OrderDateKey,
        (SELECT MAX(CustomerKey)
        FROM dbo.DimCustomer
        WHERE CustomerAlternateKey = stg.CustNo) AS CustomerKey,
        (SELECT MAX(ProductKey)
        FROM dbo.DimProduct
        WHERE ProductAlternateKey = stg.ProductID) AS ProductKey,
        (SELECT MAX(StoreKey)
        FROM dbo.DimStore
        WHERE StoreAlternateKey = stg.StoreID) AS StoreKey,
        OrderNumber,
        OrderLineItem,
        OrderQuantity,
        UnitPrice,
        Discount,
        Tax,
        SalesAmount
FROM dbo.StageSales AS stg
```

Perform post-load optimization

Rebuild indexes

```
ALTER INDEX ALL ON dbo.DimProduct REBUILD
```

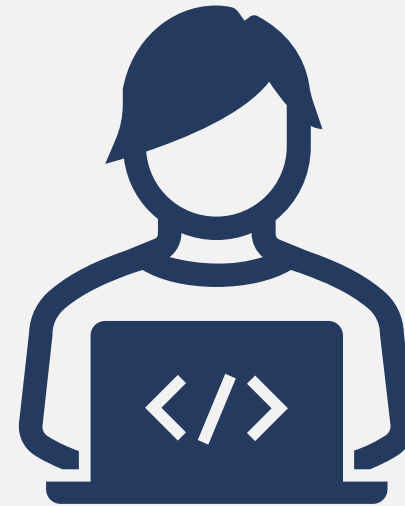
Update statistics

```
CREATE STATISTICS productcategory_stats  
ON dbo.DimProduct (ProductCategory);
```

Exercise: Load data into a data warehouse

Use the hosted lab environment provided,
or view the lab instructions at the link
below:

<https://aka.ms/mslearn-load-data-into-warehouse>



Knowledge check



In which order should you load tables in the data warehouse?

- ☒ Staging tables, then dimension tables, then fact tables
 - ☐ Staging tables, then fact tables, then dimension tables
 - ☐ Dimension tables, then staging tables, then fact tables
-



Which command should you use to load a staging table with data from files in the data lake?

- ☒ COPY
 - ☐ LOAD
 - ☐ INSERT
-



When a customer changes their phone number, the change should be made in the existing row for that customer in the dimension table. What type of slowly changing dimension is this?

- ☐ Type 0
- ☒ Type 1
- ☐ Type 2

Further reading



Work with Data Warehouses using Azure Synapse Analytics
<https://aka.ms/mslearn-synapse-data-warehouse>