Building an Azure Business Intelligence Solution End to End

Hands On Workshop

Paul Andrew | Senior Consultant

Terry McCann | Principal Consultant

Simon Whiteley | Cloud Architect

adatis

Microsoft Partner | Gold Data Analytics, Gold Data Platform, Gold Cloud Platform https://github.com/Adatis

ModernDataWarehouseWorkshop

adatis

Agenda for the Day

Module 1 Microsoft Azure	Module 2 Storage Uploading Data Data Lake	Module 3 Real-time Data Streaming Power BI	Module 4 U-SQL - Data Transformation Basics
Module 5 USQL - Advanced Analytics Cognitive Services	Module 6 Data Factory Orchestration Dynamic Pipelines	Module 7 Data Presentation & Consumption Power BI Models	Module 8 Other Services Q&A

Module Agenda

U-SQL as a Framework

Scale Out .Net, R & Python

Data Lake in Production

Limitations & Considerations

Data Lake in Production

Storage Handling Code Generation

Module Agenda

U-SQL as a Framework

Scale Out .Net, R & Python

Data Lake in Production

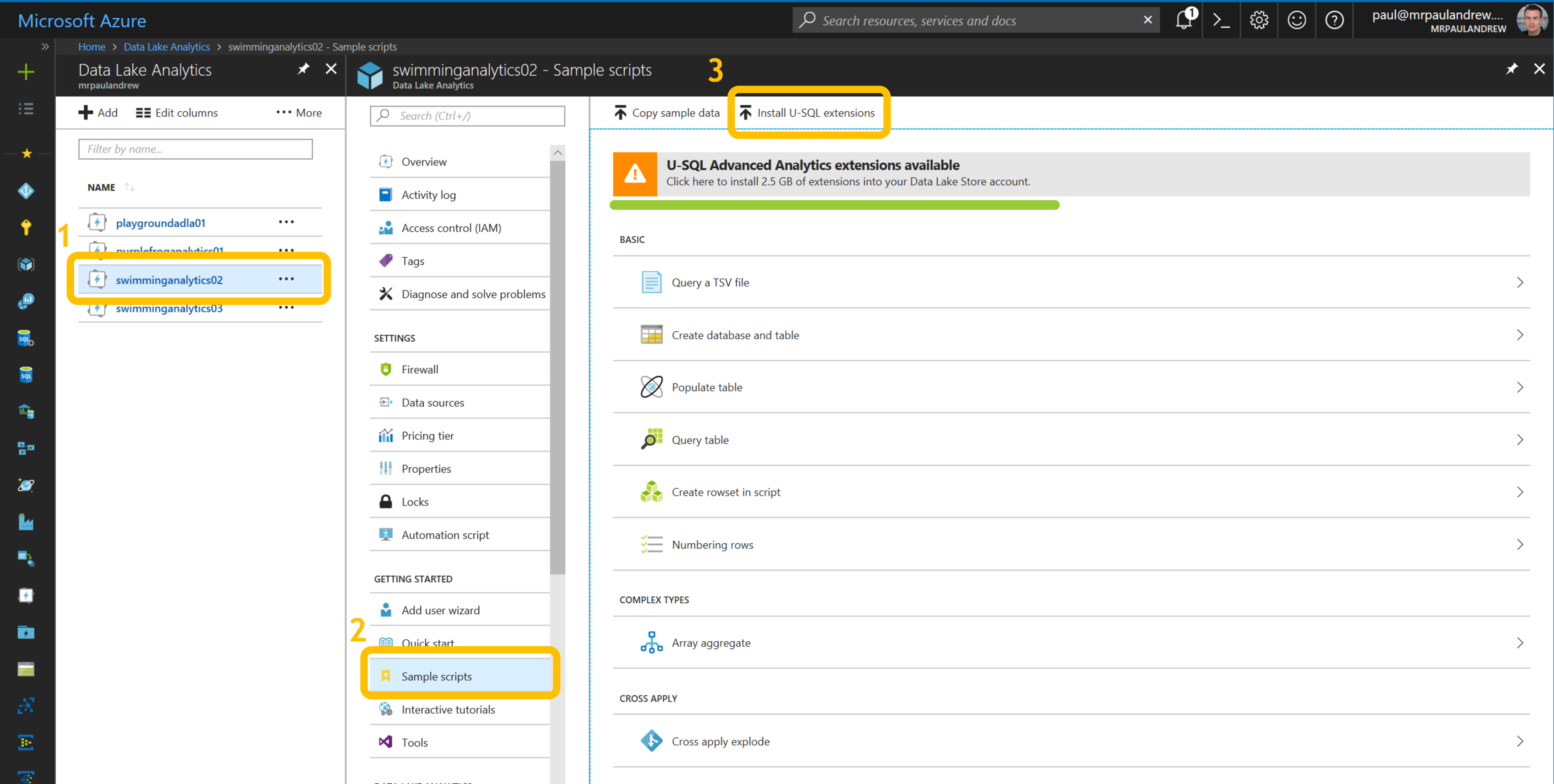
Limitations & Considerations

Data Lake in Production

Storage Handling Code Generation

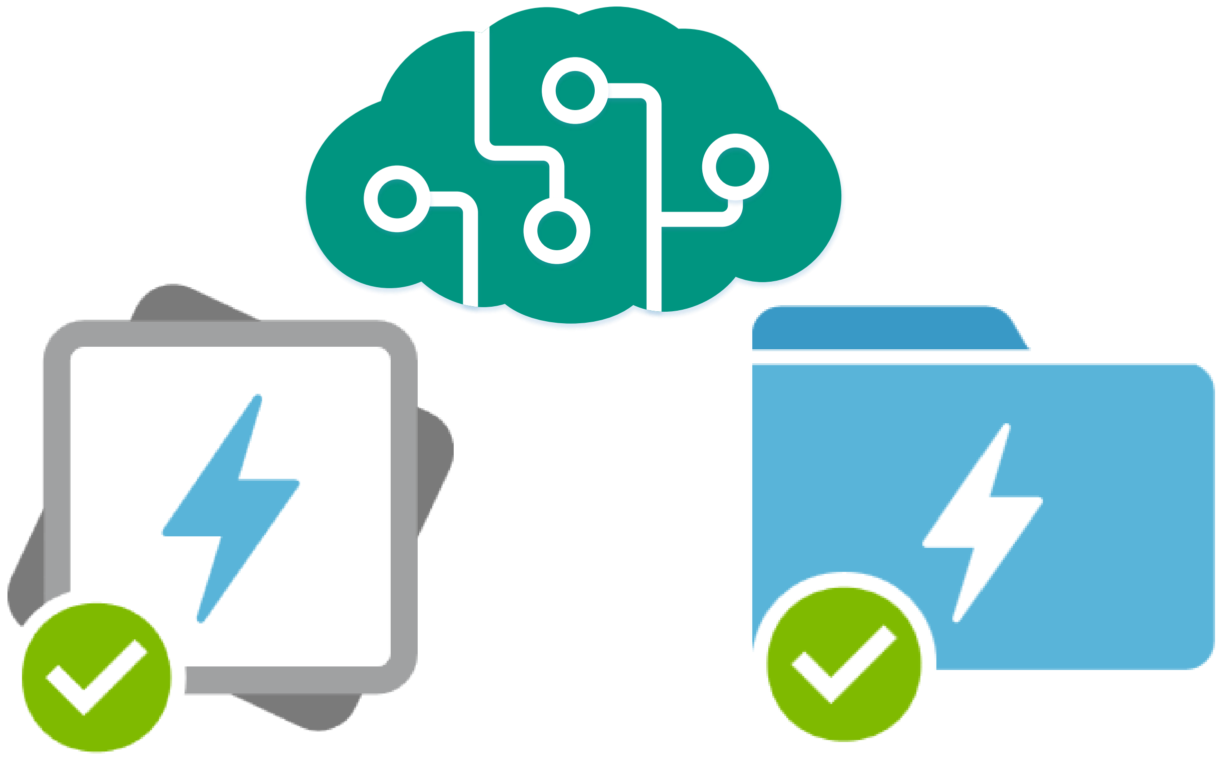
U-SQL As A Framework

Getting the U-SQL Extensions



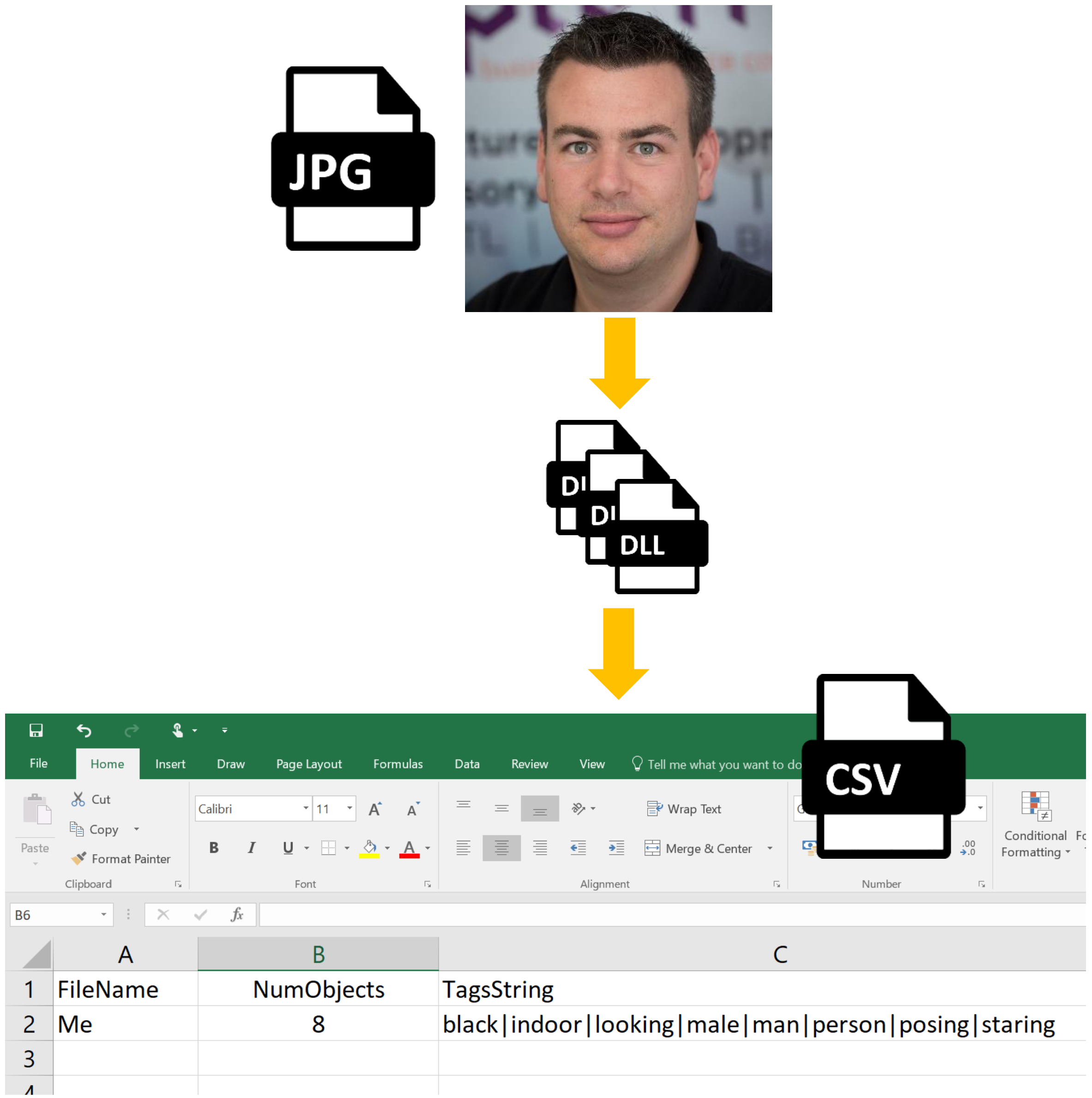
U-SQL Advanced Analytics extensions available
Click here to install 2.5 GB of extensions into your Data Lake Store account.

Azure Data Lake with Cognitive Services



U-SQL Image Tagging with Cognitive Services

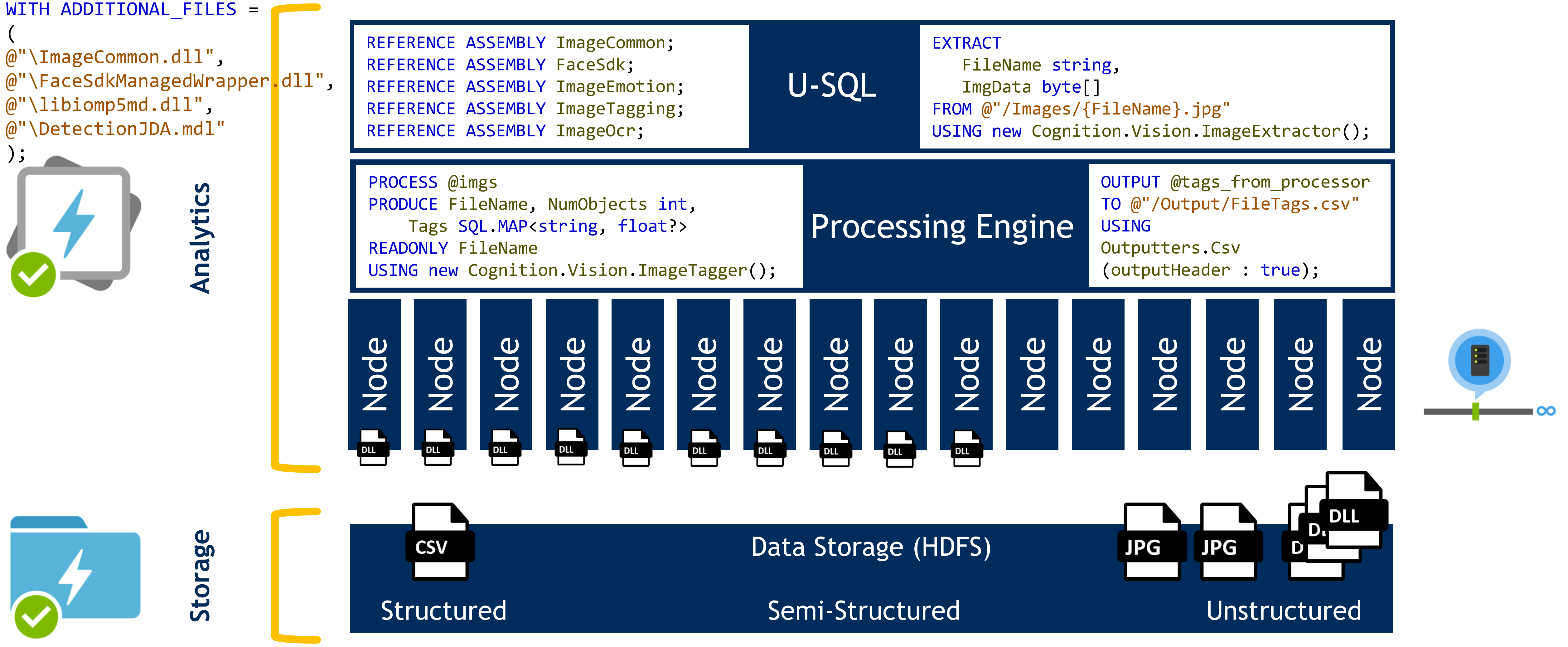
```
// Load Assemblies
REFERENCE ASSEMBLY ImageCommon;
REFERENCE ASSEMBLY FaceSdk;
REFERENCE ASSEMBLY ImageEmotion;
REFERENCE ASSEMBLY ImageTagging;
REFERENCE ASSEMBLY ImageOcr;

// Load in images
@imgs =
    EXTRACT FileName string, ImgData byte[]
    FROM @"/Images/{FileName}.jpg"
    USING new Cognition.Vision.ImageExtractor();

//Tagging processor
@tags_from_processor =
    PROCESS @imgs
    PRODUCE FileName, NumObjects int, Tags SQL.MAP<string, float?>
    READONLY FileName USING new Cognition.Vision.ImageTagger();

@tags_from_processor_serialized =
    SELECT
        FileName,
        NumObjects,
        String.Join
        ("|", Tags.Select(x => String.Format("{0}", x.Key))) AS TagsString
    FROM
        @tags_from_processor;

//Output
OUTPUT @tags_from_processor_serialized
TO @"/Output/FileTags.csv"
USING Outputters.Csv(outputHeader : true);
```



	A	B	C
1	FileName	NumObjects	TagsString
2	Me	8	black|indoor|looking|male|man|person|posing|staring
3			

Scale Out Cognitive Service with Azure Data Lake

```
CREATE ASSEMBLY IF NOT EXISTS [ImageCommon]
FROM @"\usqlext\assembly\cognition\vision\common\ImageIO.dll"
WITH ADDITIONAL_FILES =
(
@"\ImageCommon.dll",
@"\FaceSdkManagedWrapper.dll",
@"\libiomp5md.dll",
@"\DetectionJDA.mdl"
);
```



```
REFERENCE ASSEMBLY ImageCommon;
REFERENCE ASSEMBLY FaceSdk;
REFERENCE ASSEMBLY ImageEmotion;
REFERENCE ASSEMBLY ImageTagging;
REFERENCE ASSEMBLY ImageOcr;
```

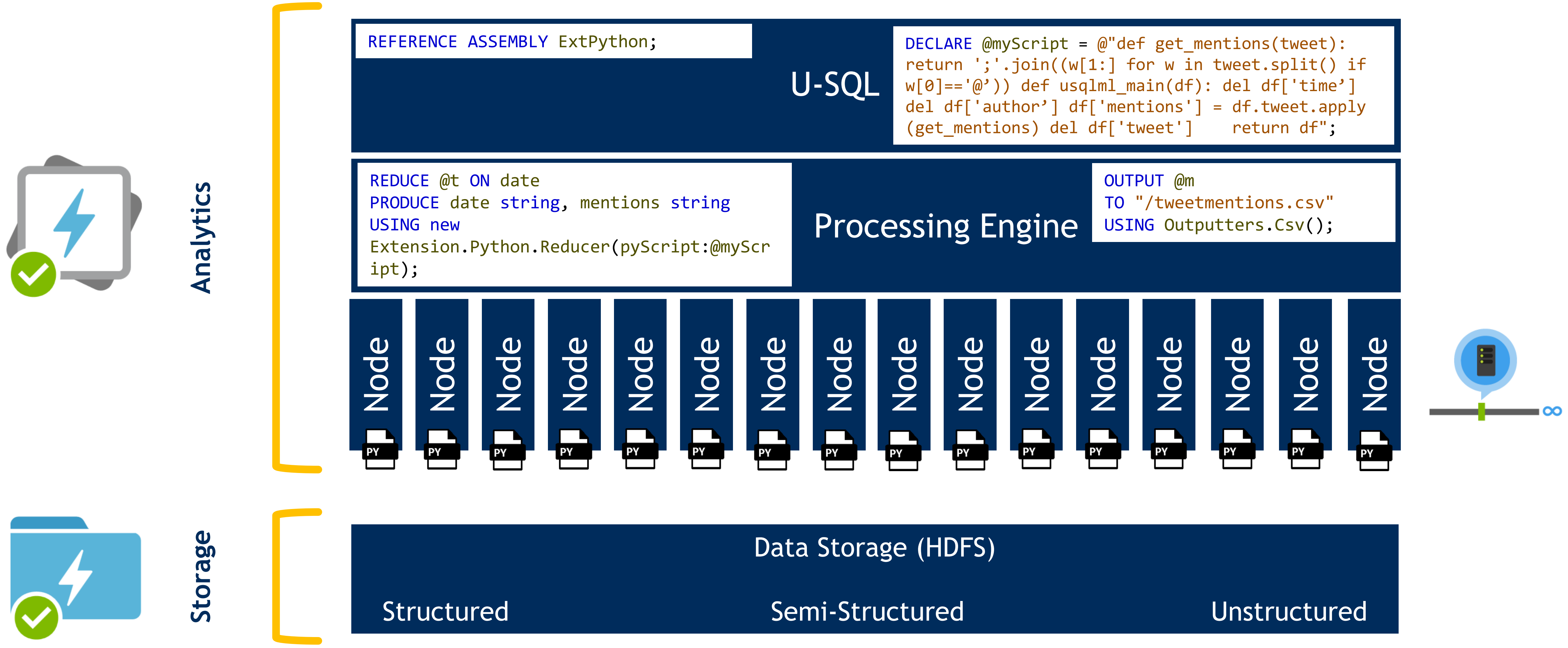
```
EXTRACT
   FileName string,
   ImgData byte[]
FROM @"/Images/{FileName}.jpg"
USING new Cognition.Vision.ImageExtractor();
```

```
PROCESS @imgs
PRODUCE FileName, NumObjects int,
    Tags SQL.MAP<string, float?>
READONLY FileName
USING new Cognition.Vision.ImageTagger();
```

```
OUTPUT @tags_from_processor
TO @"/Output/FileTags.csv"
USING
Outputters.Csv
(outputHeader : true);
```

https://docs.microsoft.com/en-us/azure/data-lake-analytics/data-lake-analytics-u-sql-cognitive

Scale Out Python with Azure Data Lake



```
REFERENCE ASSEMBLY ExtPython;
```

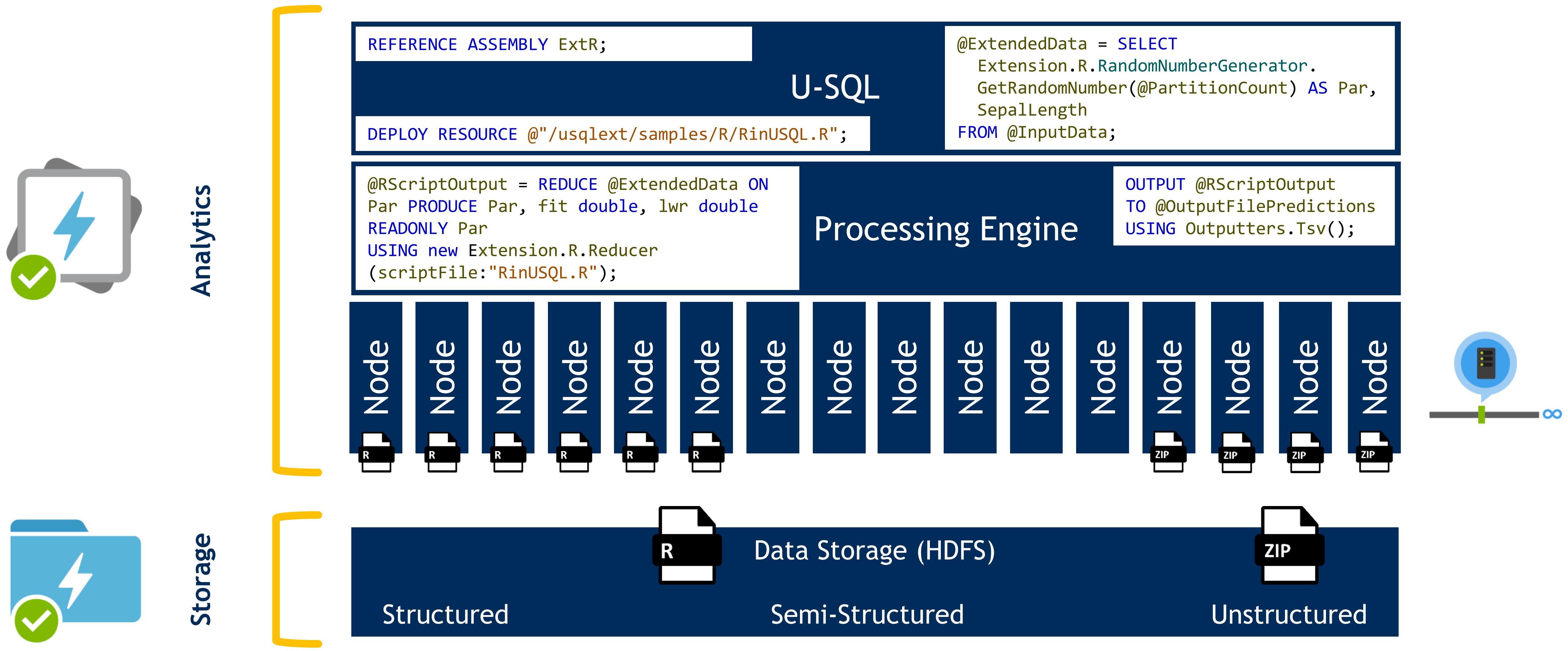
```
DECLARE @myScript = @"def get_mentions(tweet):
return ';'.join((w[1:] for w in tweet.split() if
w[0]=='@')) def usqlml_main(df): del df['time']
del df['author'] df['mentions'] = df.tweet.apply
(get_mentions) del df['tweet']    return df";
```

```
REDUCE @t ON date
PRODUCE date string, mentions string
USING new
Extension.Python.Reducer(pyScript:@myScr
ipt);
```

```
OUTPUT @m
TO "/tweetmentions.csv"
USING Outputters.Csv();
```

https://docs.microsoft.com/en-us/azure/data-lake-analytics/data-lake-analytics-u-sql-python-extensions

Scale Out R with Azure Data Lake

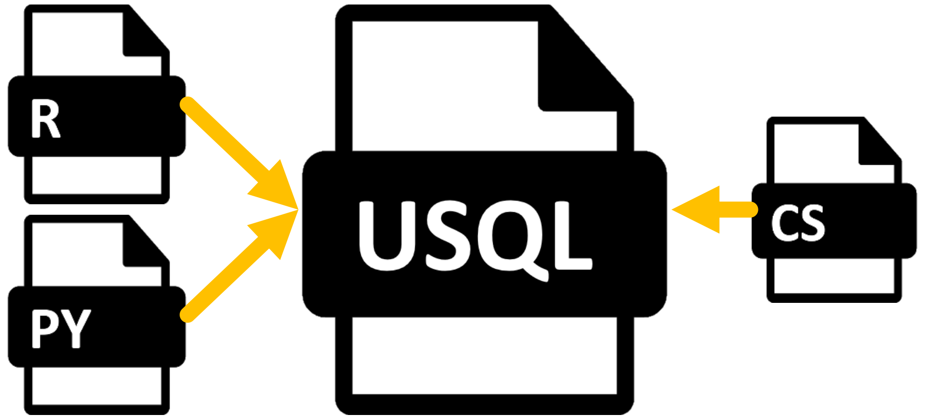


```
REFERENCE ASSEMBLY ExtR;

DEPLOY RESOURCE @"/usqlext/samples/R/RinUSQL.R";

@ExtendedData = SELECT
  Extension.R.RandomNumberGenerator.
  GetRandomNumber(@PartitionCount) AS Par,
  SepalLength
FROM @InputData;

@RScriptOutput = REDUCE @ExtendedData ON
Par PRODUCE Par, fit double, lwr double
READONLY Par
USING new Extension.R.Reducer
(scriptFile:"RinUSQL.R");

OUTPUT @RScriptOutput
TO @OutputFilePredictions
USING Outputters.Tsv();
```

https://docs.microsoft.com/en-us/azure/data-lake-analytics/data-lake-analytics-u-sql-r-extensions

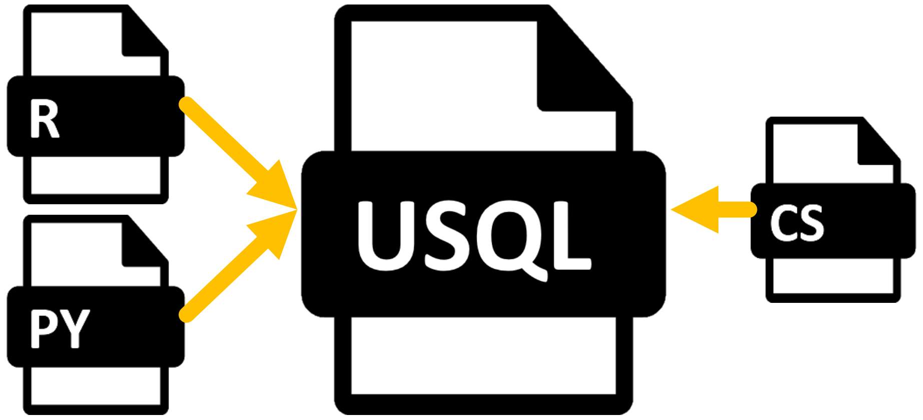
What is U-SQL? **Again**

```
@SizeAndCount =
    SELECT
        [ModifiedDate].ToString("yyyy") AS Year,
        [FileName].Substring([FileName].IndexOf(".") + 1, 3) AS FileExtension,
        COUNT(0) AS RecordCount,
        Math.Ceiling(Convert.ToDecimal(SUM([Size]))) AS FileSizeTotalsMB,
        Math.Ceiling(Convert.ToDecimal(SUM([Size])/1024)) AS FileSizeTotalsGB
    FROM
        @Raw
    WHERE
        [ActualFileName] == "FileDetailsTest.csv"
    GROUP BY
        [ModifiedDate].ToString("yyyy"),
        [FileName].Substring([FileName].IndexOf(".") + 1, 3);
```



What is U-SQL?

Answer: A highly scalable hybrid query framework.



Module Agenda

U-SQL as a Framework

Scale Out .Net, R & Python

Data Lake in Production

Limitations & Considerations

Data Lake in Production

Storage Handling
Code Generation

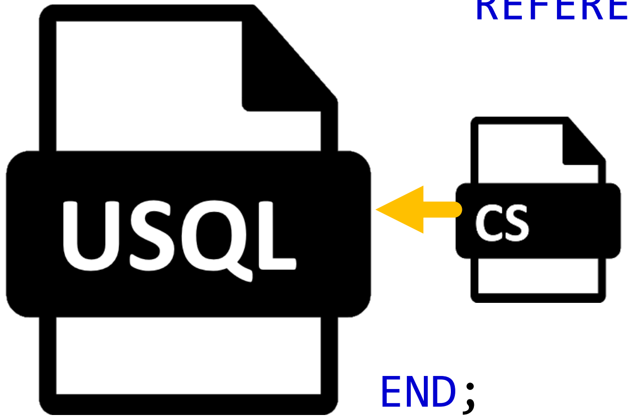
U-SQL and C# Code Behind in Production **Consideration 1**

Development - Adhoc Job Submit

```
// Auto-generated header code
// Generated Code Behind Header

CREATE ASSEMBLY [__codeBehind_1xkprrnp.trv]
FROM 0x4D5A900003000000040000000;

REFERENCE ASSEMBLY [__codeBehind_1xkprrnp.trv];
USING Stuff = [__codeBehind_1xkprrnp.trv];

// Generated Code Behind Header
// Auto-generated header code ended
// User script:

SELECT
   Stuff.Method([Value]) AS Result
FROM
   @Stuff;

DROP ASSEMBLY
[__codeBehind_1xkprrnp.trv];
```

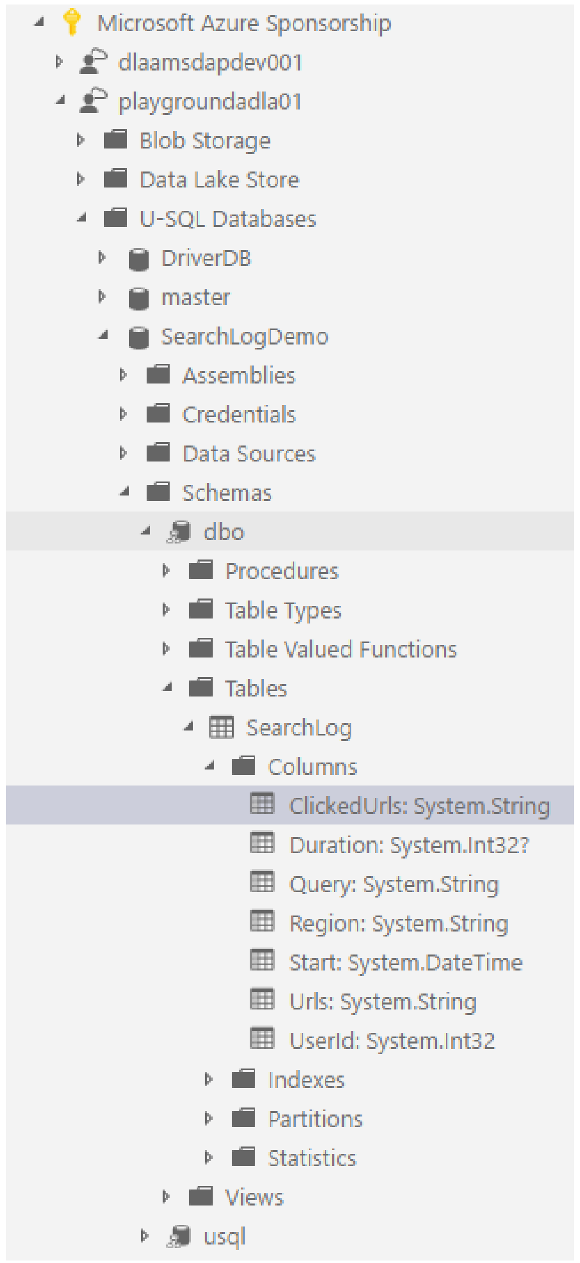
Production - Create A Stored Procedure

```
// Assemblies from class library

CREATE ASSEMBLY IF NOT EXISTS [Stuff]
FROM @"\CustomStringMethods.dll";


// User code wrapped in proc

CREATE PROCEDURE StoredProc01()
AS
BEGIN
        REFERENCE ASSEMBLY [Stuff];

              SELECT
              Stuff.Method([Value]) AS Result
              FROM
              @Stuff;

   END;
```



Data Lake Analytics As A Logical Instance

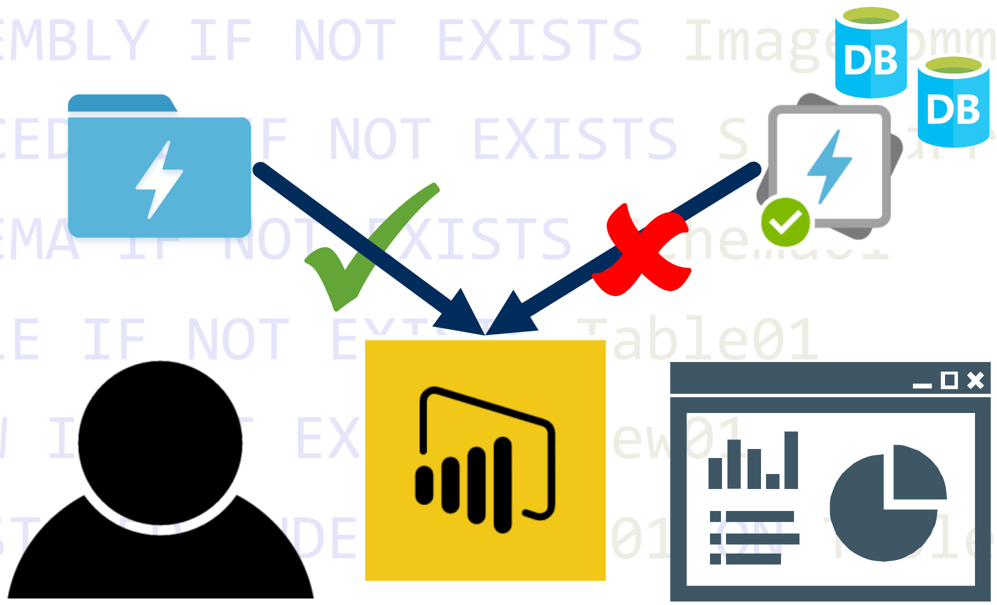
Consideration 2



```
◢ Microsoft Azure Sponsorship
  ▷ dlaamsdapdev001
  ◢ playgroundadla01
    ▷ Blob Storage
    ▷ Data Lake Store
    ◢ U-SQL Databases
      ▷ DriverDB
      ▷ master
      ◢ SearchLogDemo
        ▷ Assemblies
        ▷ Credentials
        ▷ Data Sources
        ◢ Schemas
          ◢ dbo
            ▷ Procedures
            ▷ Table Types
            ▷ Table Valued Functions
            ◢ Tables
              ◢ SearchLog
                ◢ Columns
                    ClickedUrls: System.String
                    Duration: System.Int32?
                    Query: System.String
                    Region: System.String
                    Start: System.DateTime
                    Urls: System.String
                    UserId: System.Int32
                ▷ Indexes
                ▷ Partitions
                ▷ Statistics
            ▷ Views
          ▷ usql
```

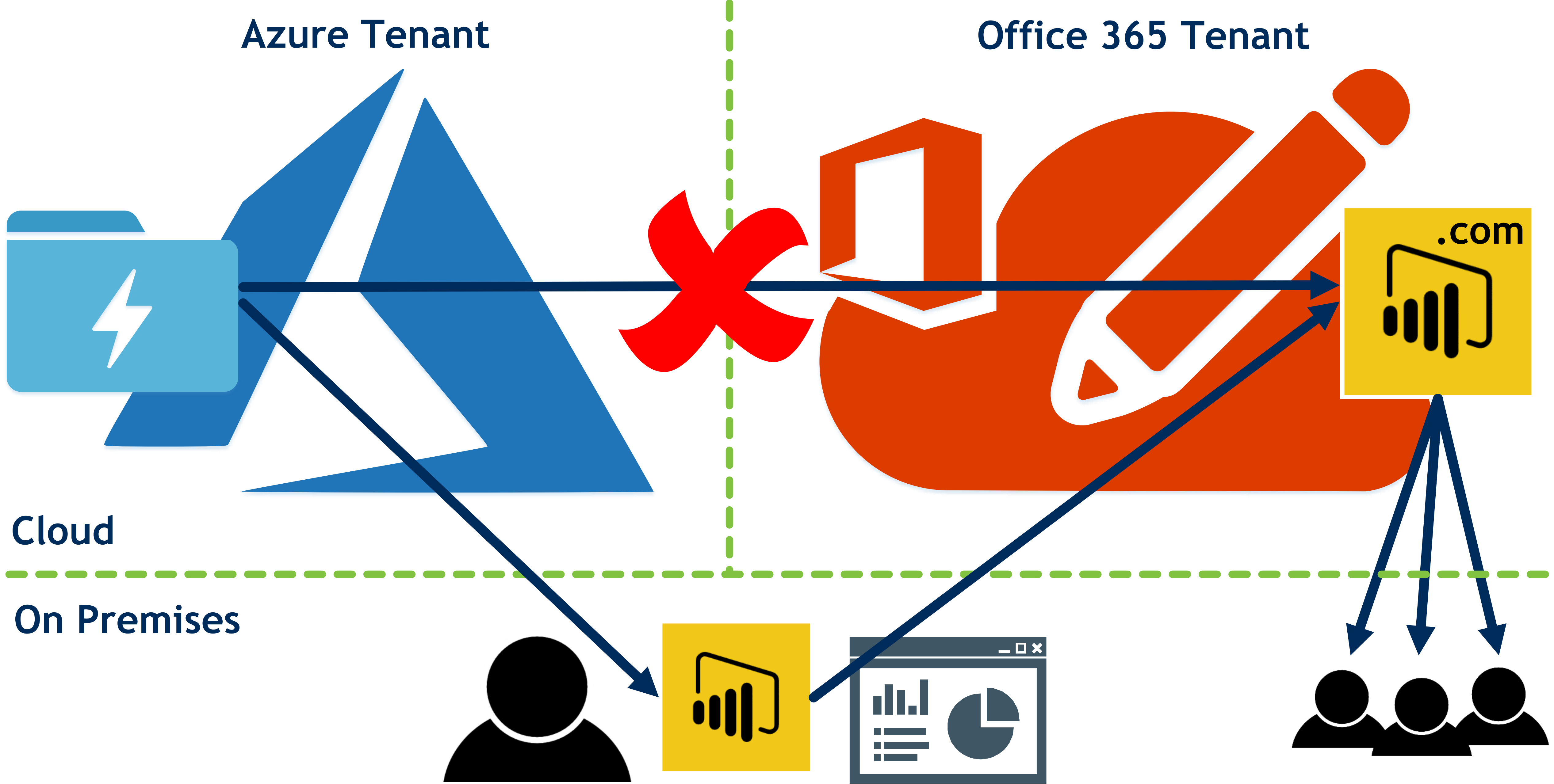
catalog

```
CREATE DATABASE IF NOT EXISTS BaseOfData01
CREATE ASSEMBLY IF NOT EXISTS Imag    mmon
CREATE PROCED    F NOT EXISTS S    roc01
CREATE SCHEMA IF NOT   XISTS    hema01
CREATE TABLE IF NOT EX    Table01
CREATE VIEW I     EX    ew01
CREATE CLUST     DE    01 ON Table01
CREATE STATISTICS Stats01 ON Table01
```

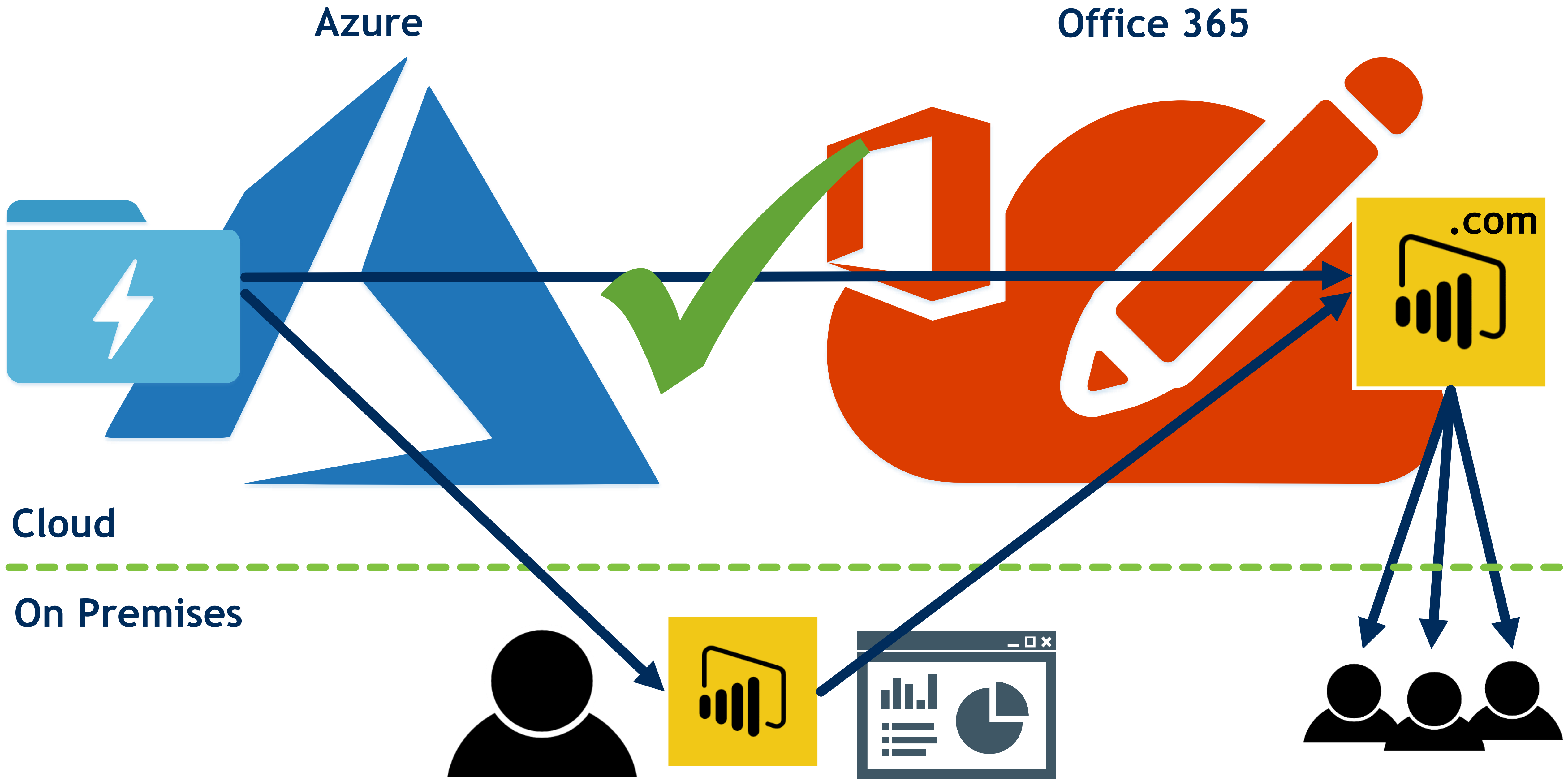


Consuming Azure Data Lake

Consideration 3



Consuming Azure Data Lake **Consideration 3**



Module Agenda

U-SQL as a Framework

Scale Out .Net, R & Python

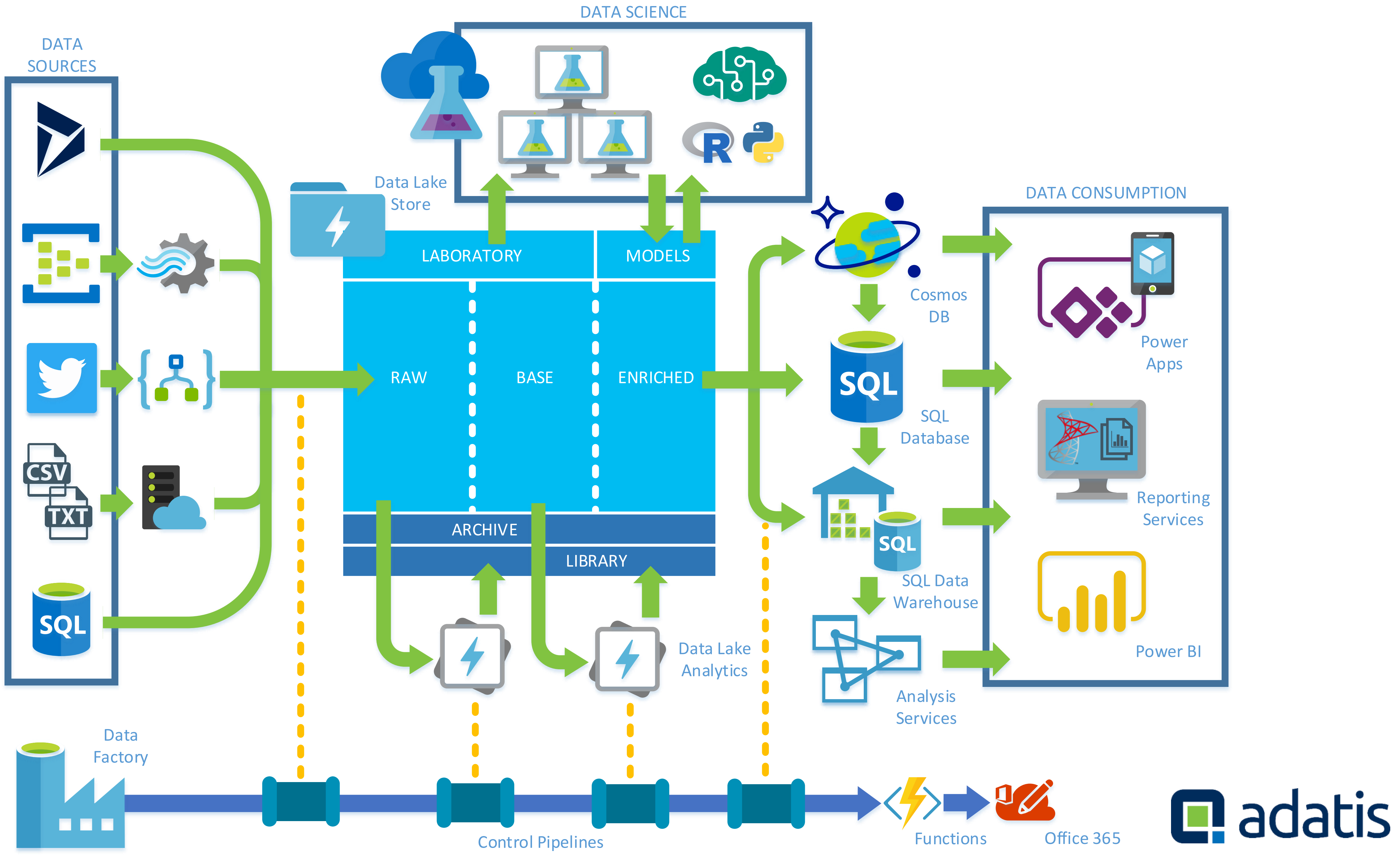
Data Lake in Production

Limitations & Considerations

Data Lake in Production

Storage Handling
Code Generation

The Modern Data Analytics Platform



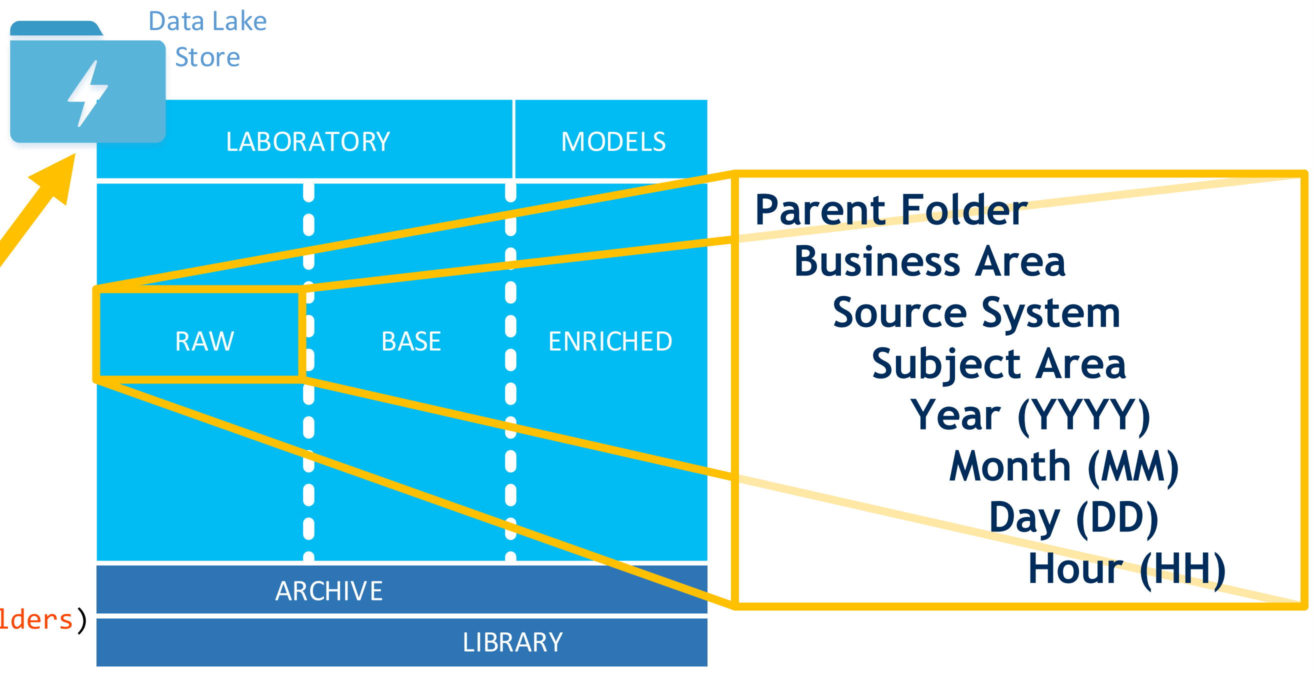
Setting Up Azure Data Lake Storage in Production

http://blogs.adatis.co.uk/ustoldfield/post/Shaping-The-Lake-Data-Lake-Framework

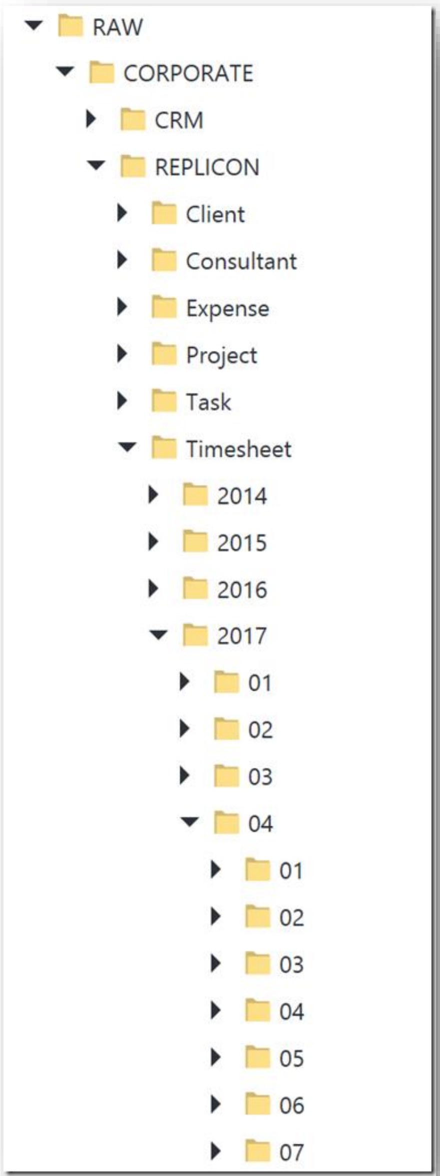
JSON

```json
[
{
"path": "//RAW",
"security": []
},
{
"path": "//BASE",
"security": []
},
{
"path": "//ENRICHED",
"security": []
},
{
"path": "//MODEL",
"security": []
},
{
"path": "//LIBRARY",
"security": []
},
{
"path": "//LABORATORY",
"security": []
},
{
"path": "//UTILITIES",
"security": []
}
]
```

PS1

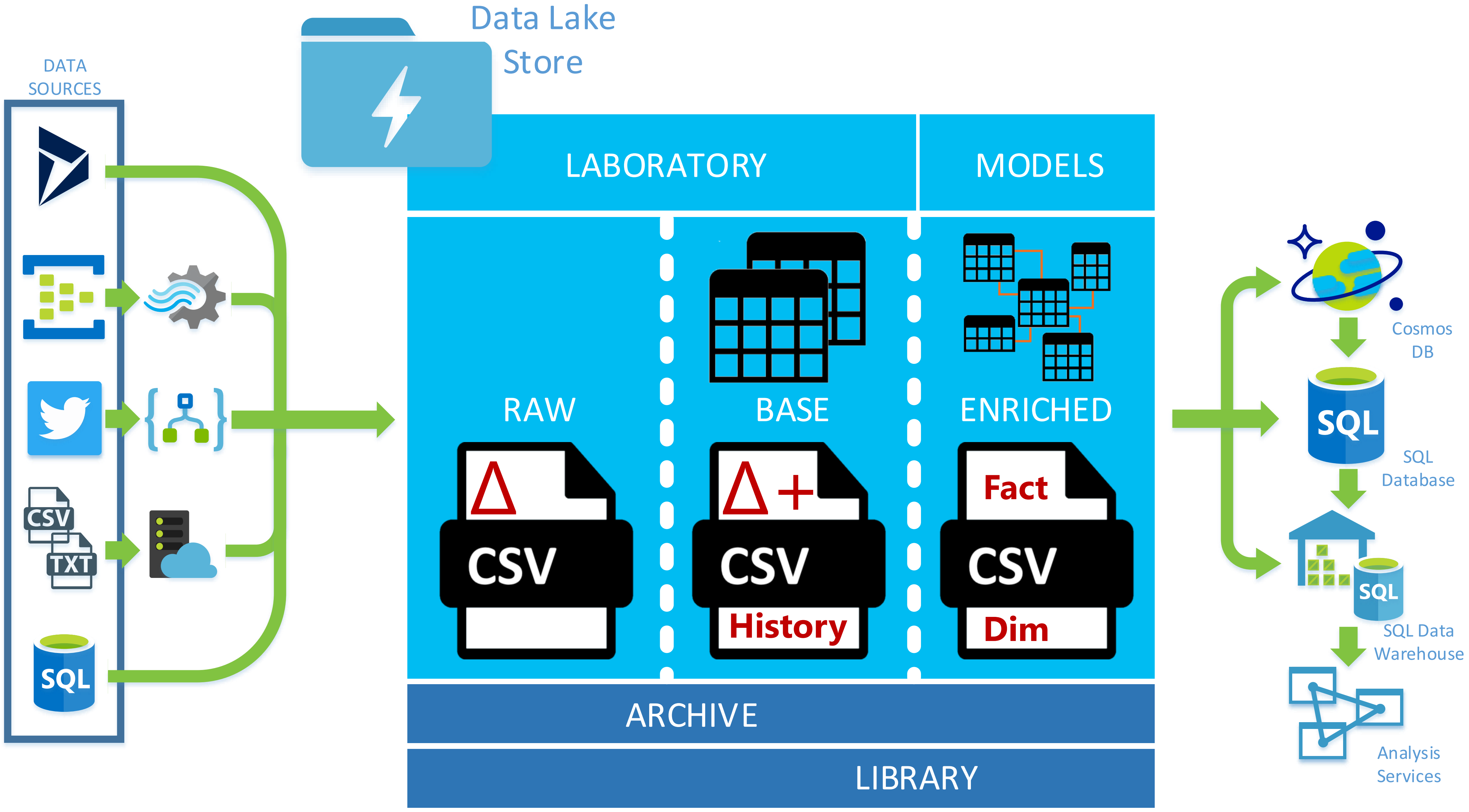


```powershell
foreach ($folder in $folders)
{
    $path = $myrootdir + $folder.path

    $dl = Get-AzureRmDataLakeStoreItem -AccountName $dataLakeStoreName -Path $path

    if (!$dl)
    {
        New-AzureRmDataLakeStoreItem -Folder -AccountName $dataLakeStoreName -Path $path
        Write-Host "New Path Created" -ForegroundColor Cyan
    }
    else
    {
        Write-Host $path " already exists in account "  $dataLakeStoreName -ForegroundColor Cyan
    }   }
```



```
RAW
  CORPORATE
    CRM
    REPLICON
      Client
      Consultant
      Expense
      Project
      Task
      Timesheet
        2014
        2015
        2016
        2017
          01
          02
          03
          04
            01
            02
            03
            04
            05
            06
            07
```

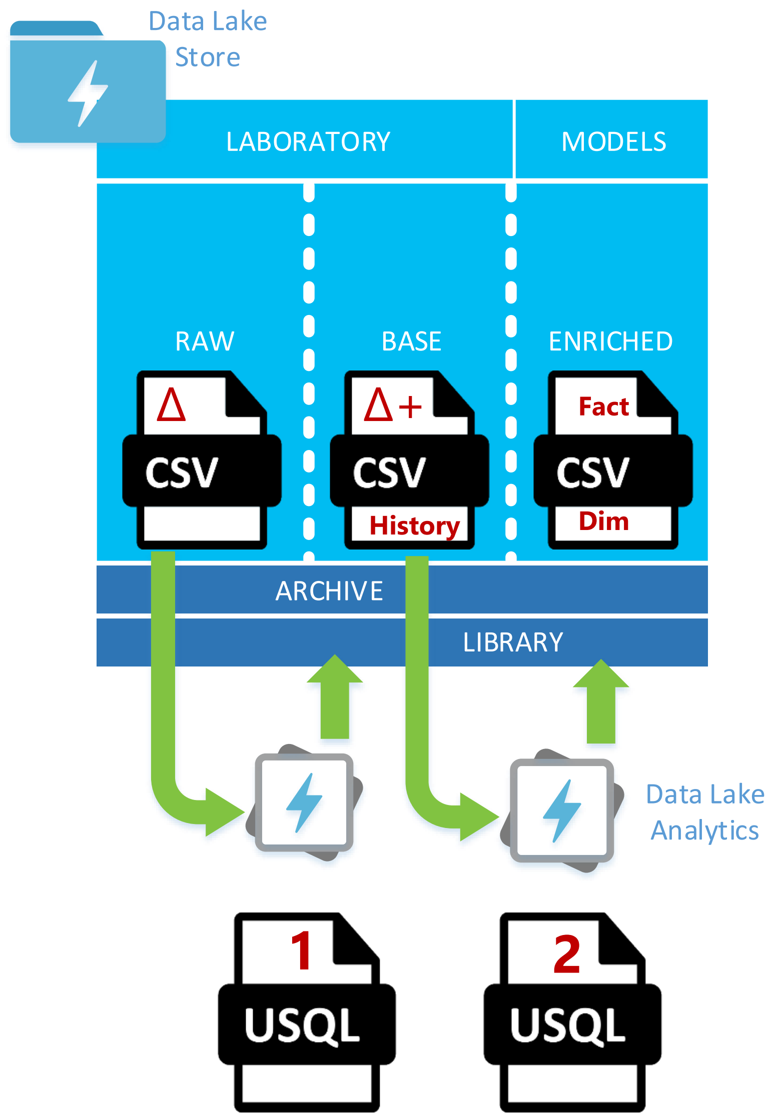
Using Azure Data Lake Storage in Production



Using Azure Data Lake Analytics in Production

1. RAW to BASE

```
/*
INSERT
CODE
HERE
*/
```



2. BASE to ENRICHED

```
/*
INSERT
CODE
HERE
*/
```

Using Azure Data Lake Analytics in Production

1. RAW to BASE

https://semanticinsight.wordpress.com/2018/04/19/finer-points-usql-merging-datasets-part-2/

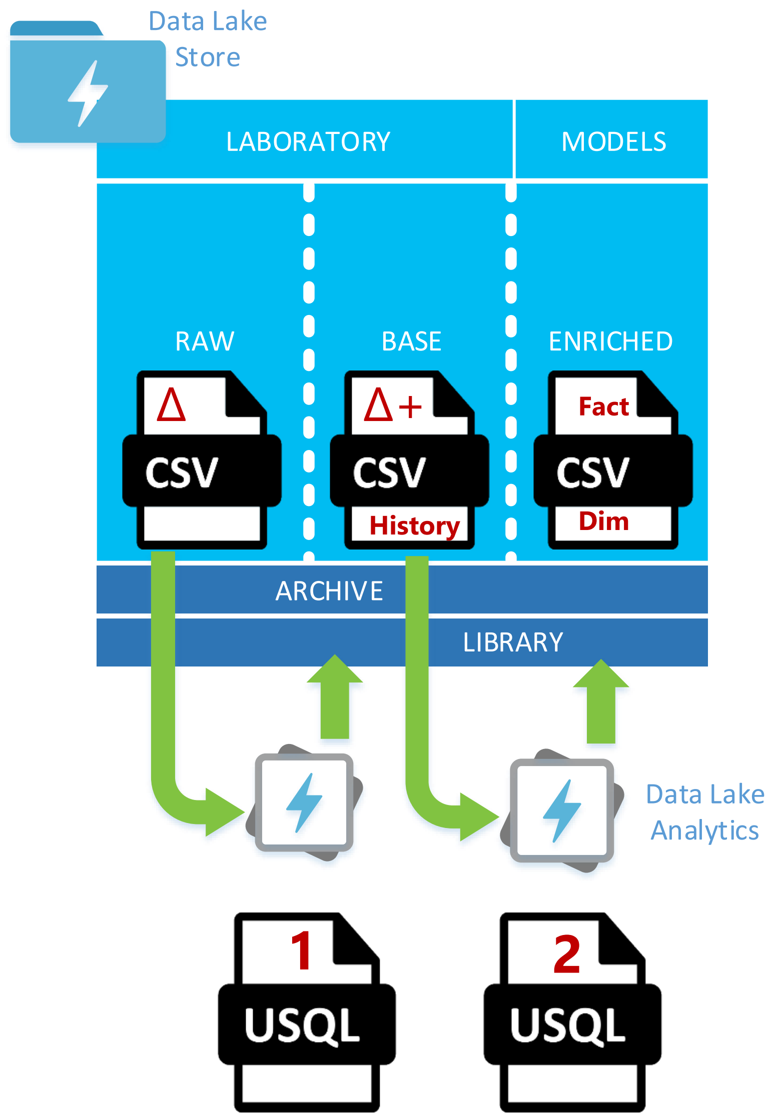
```
DECLARE @LocalRunDate string = "20180604";

DECLARE @InputFilePath string =
"/RAW/SourceSystem/{yyyy}/{mm}/{dd}/FileName.csv";

@Extracted =
    EXTRACT
        [Columns] string,
        //virtual columns
        [yyyy] string,
        [mm] string,
        [dd] string
    FROM
        @InputFilePath
    USING
        Extractors.Csv(skipFirstNRows:1);


@Delta =
    SELECT
        *
    FROM
        @Extracted
    WHERE
        [yyyy] + [mm] + [dd] == @LocalRunDate;
```

```
@Merged =
    SELECT
        (string)([checkFlag] ? [src_Field1] : [tgt_Field1]) AS Field1,
        (string)([checkFlag] ? [src_Field2] : [tgt_Field2]) AS Field2
    FROM
        (
        SELECT
            (
                ([source].[PK] == [target].[PK] & [source].[PK] != null)
                || ([PK].[PK] == null) ? true : false
            ) AS checkFlag,
            //source data
            source.[Field1] AS src_Field1,
             source.[Field2] AS src_Field2,
            //target data
            target.[Field1] AS tgt_Field1,
             target.[Field2] AS tgt_Field2
        FROM
            @Delta AS source
            FULL OUTER JOIN [base].[Table1] AS target
                ON [source].[PK] == [target].[PK]
        ) AS dataMerge;

TRUNCATE TABLE [base].[Table1];
INSERT INTO [base].[Table1] ([Field1],[Field2])
SELECT [Field1],[Field2] FROM @Merged;
```

Using Azure Data Lake Analytics in Production

1. RAW to BASE

```
CREATE PROCEDURE IF NOT EXISTS
[base].[DeltaLoad_Table1]
        (
        @LocalRunDate string
        )
AS
BEGIN

DECLARE @InputFilePath string =
"/RAW/SourceSystem/{yyyy}/{mm}/{dd}/FileName.csv";

@Extracted =
    EXTRACT
        [Columns] string,
        //virtual columns
        [yyyy] string,
        [mm] string,
        [dd] string
    FROM
        @InputFilePath
    USING
        Extractors.Tsv(skipFirstNRows:1, quoting : true);

@Delta =
    SELECT
        *
    FROM
        @Extracted
    WHERE
        [yyyy] + [mm] + [dd] == @LocalRunDate;

@Merged =
    SELECT
        (string)([checkFlag] ? [src_Field1] : [tgt_Field1]) AS Field1,
(string)([checkFlag] ? [src_Field2] : [tgt_Field2]) AS Field2
    FROM
        (
        SELECT
            (
                ([source].[PK] == [target].[PK] & [source].[PK] != null)
                || ([PK].[PK] == null) ? true : false
            ) AS checkFlag,
            //source data
            source.[Field1] AS src_Field1,
source.[Field2] AS src_Field2,
            //target data
            target.[Field1] AS tgt_Field1,
target.[Field2] AS tgt_Field2
        FROM
            @Delta AS source
            FULL OUTER JOIN [base].[Table1] AS target
                ON [source].[PK] == [target].[PK]
        ) AS dataMerge;

TRUNCATE TABLE [base].[Table1];
INSERT INTO [base].[Table1] ([Field1],[Field2]) SELECT [Field1],[Field2] FROM
@Merged;

END;
```



2. BASE to ENRICHED

```
/*
INSERT
CODE
HERE
*/
```

Using Azure Data Lake Analytics in Production

2. BASE to ENRICHED

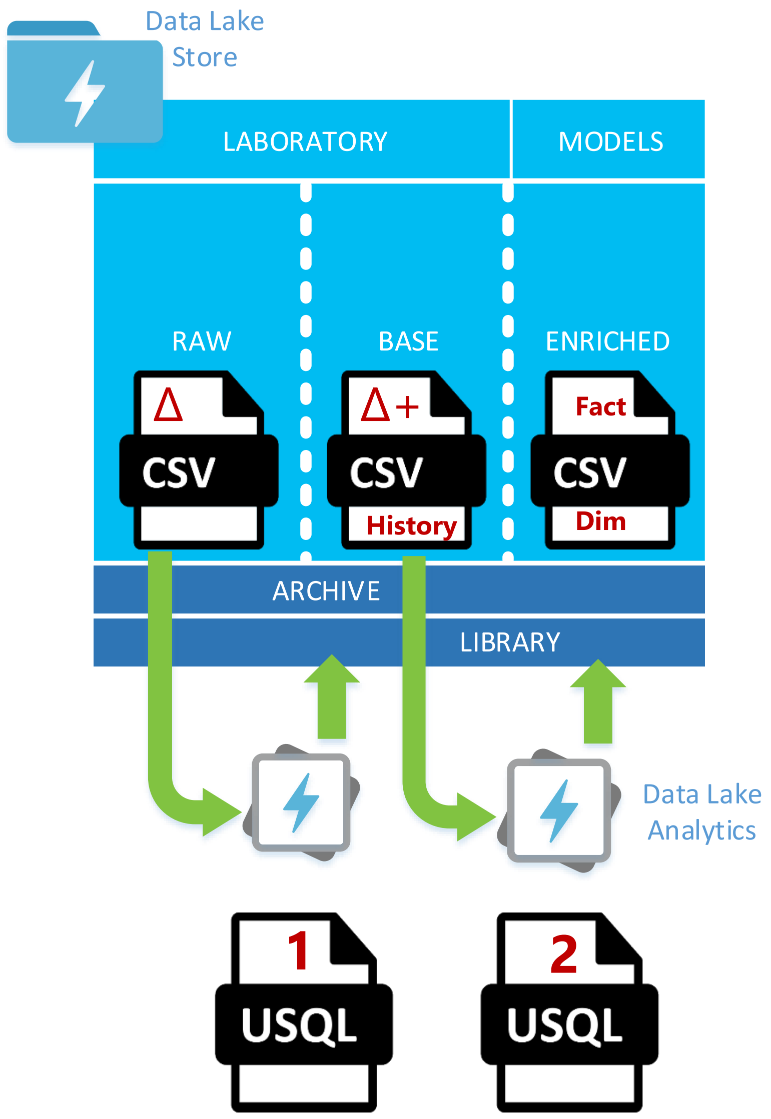
```
@AllSourceData =
    SELECT
        *
    FROM
        [SourceSystem1].[base].[Sales]

    UNION ALL

    SELECT
        *
    FROM
        [SourceSystem2].[base].[Sales];


@Transformation =
    /* INSERT CODE HERE */

@Lookups =
    /* INSERT CODE HERE */

@Aggregates =
    /* INSERT CODE HERE */
```

```
@OutputDataset =
    SELECT
        *,
        [SourceSystemKey],
        [LastUpdatedDate]
    FROM
        @Aggregates,@Lookups;

DECLARE @OutputPath = "/ENRICHED/Warehouse/Fact/Sales.csv";

OUTPUT @OutputDataset
TO @outputLocation
USING Outputters.Csv();
```

Using Azure Data Lake Analytics in Production

1. RAW to BASE

```
CREATE PROCEDURE IF NOT EXISTS
[base].[DeltaLoad_Table1]
        (
        @LocalRunDate string
        )
AS
BEGIN

DECLARE @InputFilePath string =
"/RAW/SourceSystem/{yyyy}/{mm}/{dd}/FileName.csv";

@Extracted =
    EXTRACT
        [Columns] string,
        //virtual columns
        [yyyy] string,
        [mm] string,
        [dd] string
    FROM
        @InputFilePath
    USING
        Extractors.Tsv(skipFirstNRows:1, quoting : true);

@Delta =
    SELECT
        *
    FROM
        @Extracted
    WHERE
        [yyyy] + [mm] + [dd] == @LocalRunDate;

@Merged =
    SELECT
        (string)([checkFlag] ? [src_Field1] : [tgt_Field1]) AS Field1,
(string)([checkFlag] ? [src_Field2] : [tgt_Field2]) AS Field2
    FROM
        (
        SELECT
            (
                ([source].[PK] == [target].[PK] & [source].[PK] != null)
                || ([PK].[PK] == null) ? true : false
            ) AS checkFlag,
            //source data
            source.[Field1] AS src_Field1,
source.[Field2] AS src_Field2,
            //target data
            target.[Field1] AS tgt_Field1,
target.[Field2] AS tgt_Field2
        FROM
            @Delta AS source
            FULL OUTER JOIN [base].[Table1] AS target
                ON [source].[PK] == [target].[PK]
        ) AS dataMerge;

TRUNCATE TABLE [base].[Table1];
INSERT INTO [base].[Table1] ([Field1],[Field2]) SELECT [Field1],[Field2] FROM
@Merged;

END;
```



2. BASE to ENRICHED

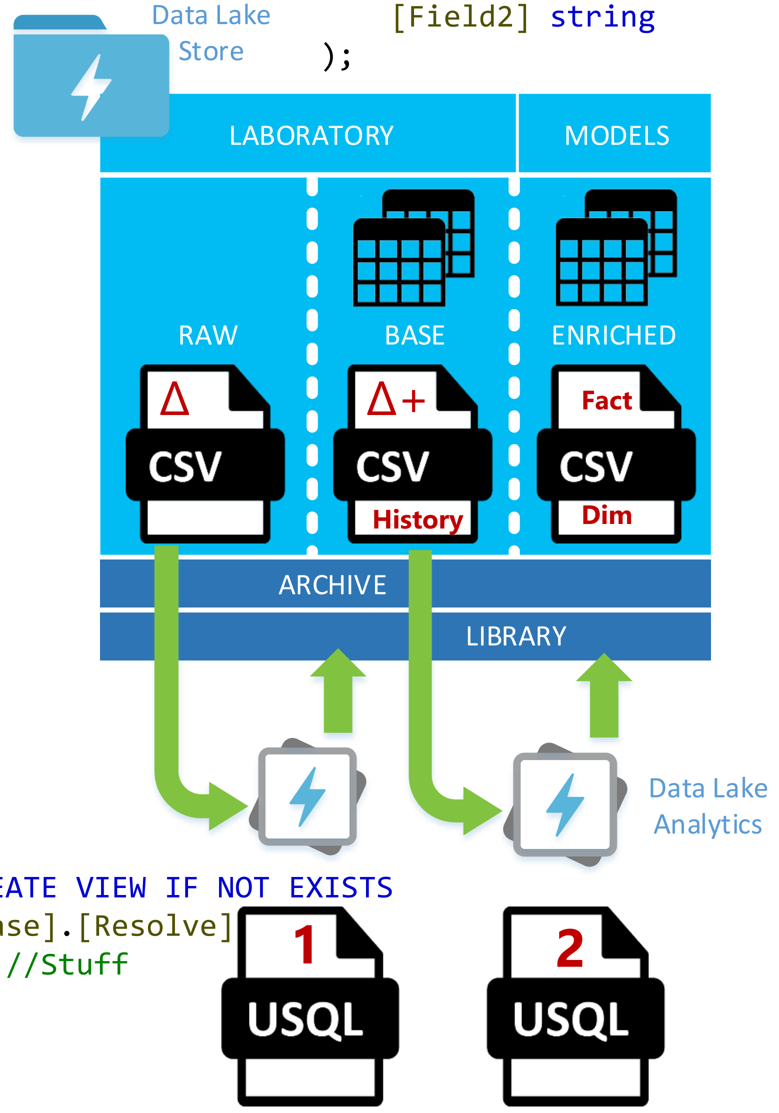
```
CREATE PROCEDURE IF NOT EXISTS [fact].[Sales]
AS
BEGIN

    @AllSourceData =
        SELECT
            *
        FROM
            [SourceSystem1].[base].[Sales]

        UNION ALL

        SELECT
            *
        FROM
            [SourceSystem2].[base].[Sales];

    @Transformation =
        /* INSERT CODE HERE */

    @Lookups =
        /* INSERT CODE HERE */


    @OutputDataset =
        SELECT
            *,
            [SourceSystemKey],
            [LastUpdatedDate]
        FROM
            @Aggregates,@Lookups;

    DECLARE @OutputPath = "/ENRICHED/Warehouse/Fact/Sales.csv";

    OUTPUT @OutputDataset
    TO @outputLocation
    USING Outputters.Csv();


END;
```

Using Azure Data Lake Analytics in Production

1. RAW to BASE

```
CREATE PROCEDURE IF NOT EXISTS
[base].[DeltaLoad_Table1]
        (
        @LocalRunDate string
        )
AS
BEGIN

DECLARE @InputFilePath string =
"/RAW/SourceSystem/{yyyy}/{mm}/{dd}/FileName.csv";

@Extracted =
    EXTRACT
        [Columns] string,
        //virtual columns
        [yyyy] string,
        [mm] string,
        [dd] string
    FROM
        @InputFilePath
    USING
        Extractors.Tsv(skipFirstNRows:1, quoting : true);

@Delta =
    SELECT
        *
    FROM
        @Extracted
    WHERE
        [yyyy] + [mm] + [dd] == @LocalRunDate;

@Merged =
    SELECT
        (string)([checkFlag] ? [src_Field1] : [tgt_Field1]) AS Field1,
(string)([checkFlag] ? [src_Field2] : [tgt_Field2]) AS Field2
    FROM
        (
        SELECT
            (
                ([source].[PK] == [target].[PK] & [source].[PK] != null)
                || ([PK].[PK] == null) ? true : false
            ) AS checkFlag,
            //source data
            source.[Field1] AS src_Field1,
source.[Field2] AS src_Field2,
            //target data
            target.[Field1] AS tgt_Field1,
target.[Field2] AS tgt_Field2
        FROM
            @Delta AS source
            FULL OUTER JOIN [base].[Table1] AS target
                ON [source].[PK] == [target].[PK]
        ) AS dataMerge;

TRUNCATE TABLE [base].[Table1];
INSERT INTO [base].[Table1] ([Field1],[Field2]) SELECT [Field1],[Field2] FROM
@Merged;

END;
```

```
CREATE TABLE IF NOT EXISTS [base].[Table1]
        (
            [Field1] string,
            [Field2] string
        );
```



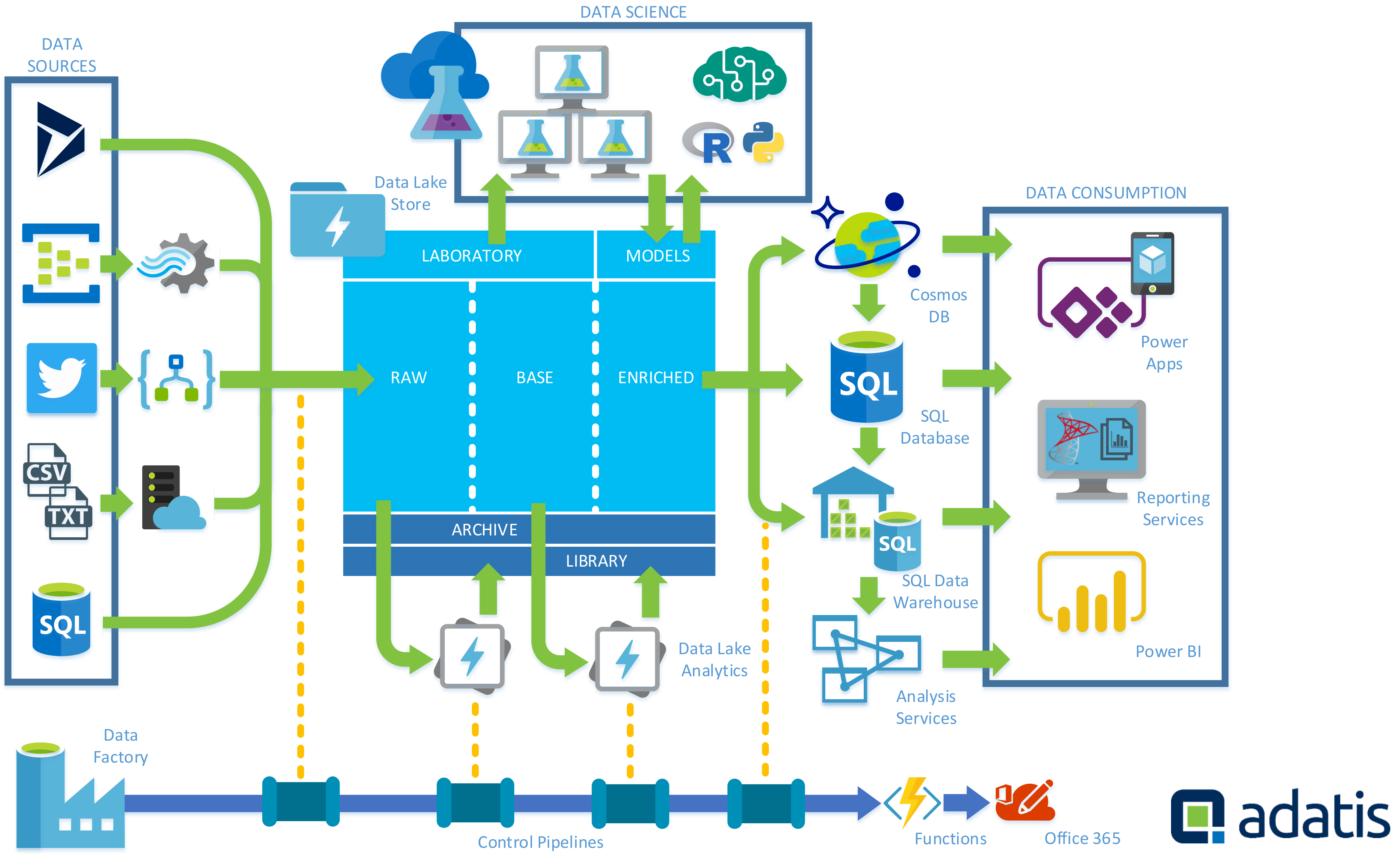
```
CREATE VIEW IF NOT EXISTS
[base].[Resolve]
AS //Stuff
```

2. BASE to ENRICHED

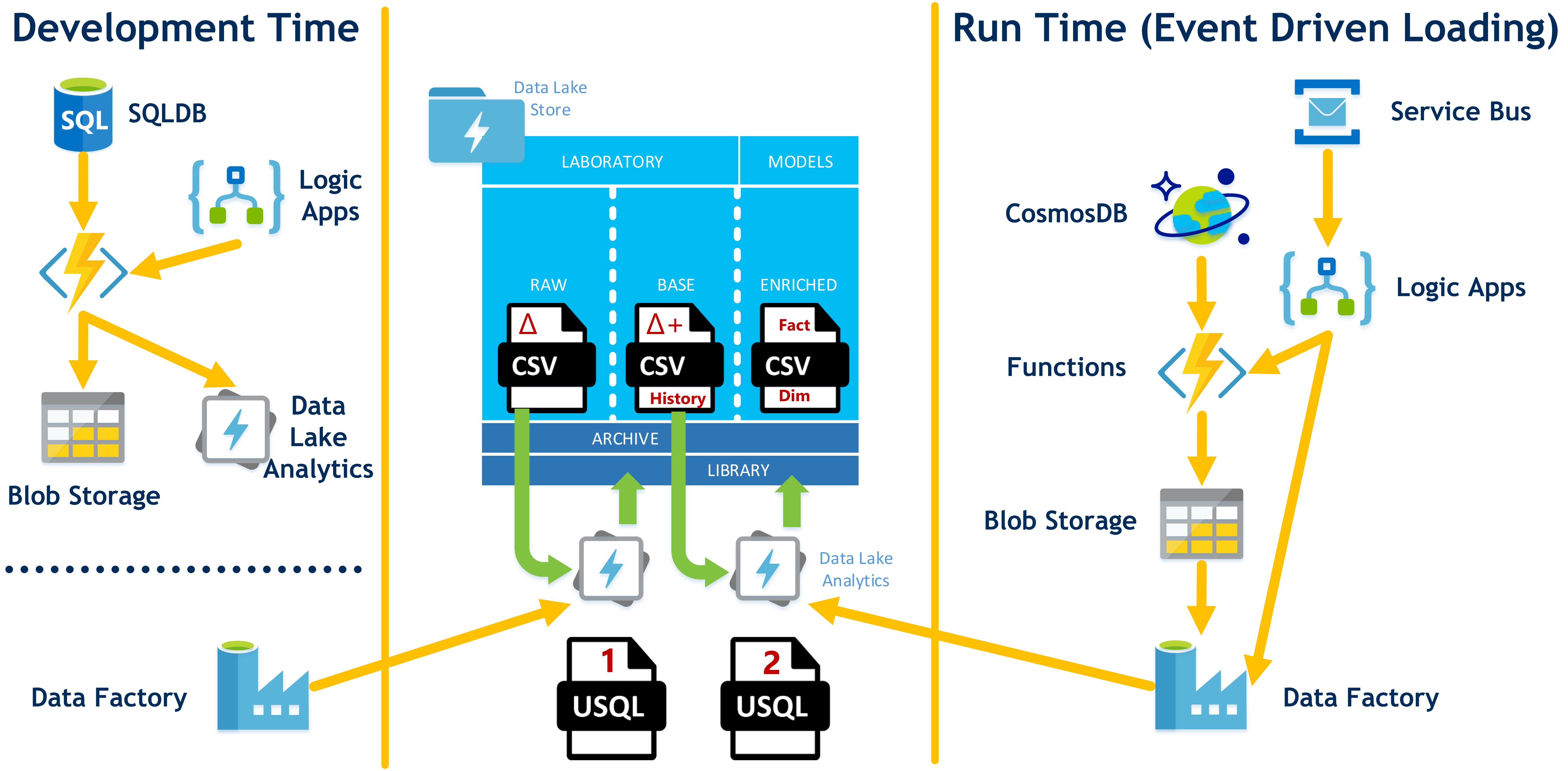
```
CREATE PROCEDURE IF NOT EXISTS [fact].[Sales]
AS
BEGIN

    @AllSourceData =
        SELECT
            *
        FROM
            [SourceSystem1].[base].[Sales]

        UNION ALL

        SELECT
            *
        FROM
            [SourceSystem2].[base].[Sales];

    @Transformation =
        /* INSERT CODE HERE */

    @Lookups =
        /* INSERT CODE HERE */


    @OutputDataset =
        SELECT
            *,
            [SourceSystemKey],
            [LastUpdatedDate]
        FROM
            @Aggregates,@Lookups;

    DECLARE @OutputPath = "/ENRICHED/Warehouse/Fact/Sales.csv";

    OUTPUT @OutputDataset
    TO @outputLocation
    USING Outputters.Csv();


END;
```

adatis

The Modern Data Analytics Platform



How do we generate our U-SQL?



Agenda for the Day

Module 1 Microsoft Azure	Module 2 Storage Uploading Data Data Lake	Module 3 Real-time Data Streaming Power BI	Module 4 U-SQL - Data Transformation Basics
Module 5 USQL - Advanced Analytics Cognitive Services	Module 6 Data Factory Orchestration Dynamic Pipelines	Module 7 Data Presentation & Consumption Power BI Models	Module 8 Other Services Q&A