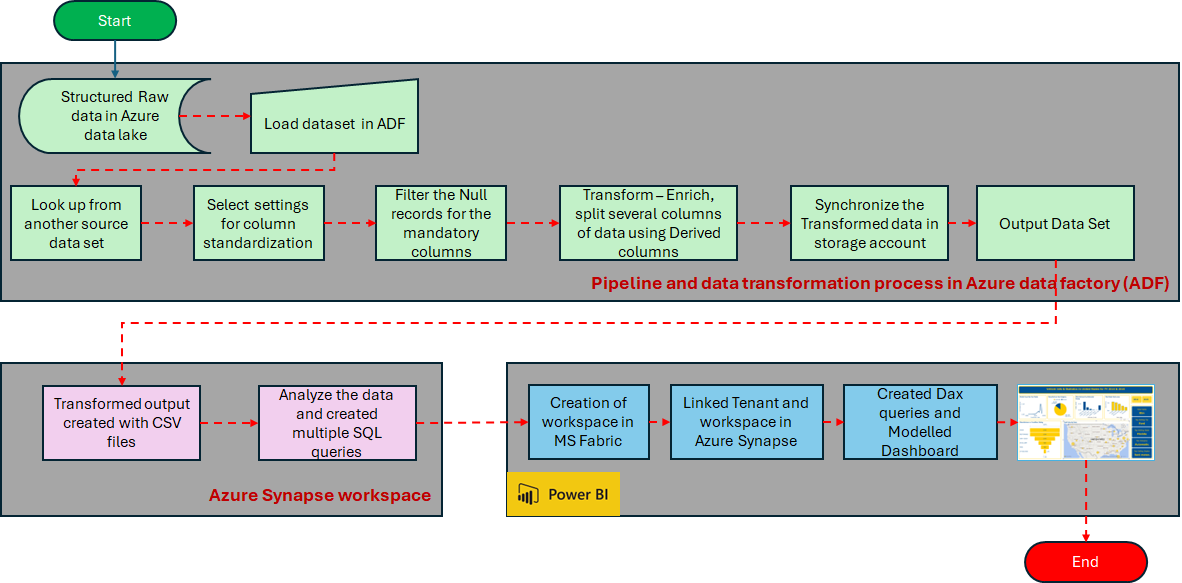
# Assignment: Big Data Solution in Azure

## Objective:

The goal of this project is to develop a comprehensive Big Data solution for sales data using Azure, focusing on integrating various Azure services to manage and analyse large-scale data. By working with data ingestion, ETL processes, and visualization, this project aims to showcase how data can be efficiently processed, stored, and visualized for insights.

## Scope of the Project and Components:

* Data Storage: Azure Blob Storage
  + Used to store large volumes of structured and unstructured data.
* Data Ingestion: Azure Data Factory
* ETL Process: Extraction, Transformation & Loading in ADF
  + Facilitates data ingestion and ETL operations, allowing the seamless movement of data between various sources and the Azure environment.
* Data Analysis: Azure Synapse Analytics
  + Supports advanced analytics by providing a powerful SQL-based engine for data analysis.
* Visualization: Power BI
  + A visualization tool for representing analysed data in dashboards and reports, providing actionable insights.
* Scalability & Security
* Data: Vehicle Sales Data from Kaggle website. <https://www.kaggle.com/datasets/syedanwarafridi/vehicle-sales-data>



## Implementation:

### Step 1: Create and Configure Azure Storage Account and Data Lake Gen2

**Create a Storage Account**

Sign into the [Azure portal](https://portal.azure.com/) at <https://portal.azure.com> and navigate to create a resource and select Azure Storage account and click on create and fill in the necessary details as

* Resource group
* Storage account name
* Region
* Performance
* Replication
* Storage account type
* Hierarchical namespace

to create a Data Lake Gen2 storage account. Click **Review + Create** and then **Create.**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Create Containers:**

After the Storage Account is created, navigate to the Storage Account dashboard. Click on Containers in the left-hand menu. Click **+ Container** to create a new container with a name structdata (sales data) and unstructdata (log data). In structdata container, click on **Upload** and upload your sales data csv file from the disk/folder. And repeat the process for unstruct data-log file. Data sources from

* structdata – sales data from Kaggle
* unstructdata – log data (fake) created for sales using python scripts

A screenshot of a computer

Description automatically generated

### Step 2: Create an Azure Data Factory

Sign into the [Azure portal](https://portal.azure.com/) at <https://portal.azure.com> and navigate to create a resource and select Azure Storage account and click on create and fill in the necessary details as Subscription, Resource group, Name, Region, Version and click **Review + Create** and then **Create.**

A screenshot of a computer

Description automatically generated

After deployment is complete, click **Lauch studio** and it opens the new tab in ADF home page.

A screenshot of a computer

Description automatically generated

**Build ETL Pipeline**

**Creating a Pipeline:**

* + In your Azure Data Factory instance, navigate to the **Data** tab.
  + Select **Pipelines** and click **+ New pipeline** to create a new pipeline named **LoadSalesData**.
  + In the pipeline menu, under **Move and transform**, select **Data flow** and drag it onto the workspace.

**1. Source (Data Extraction):**

* + Drag the **Data flow** activity onto the pipeline canvas. In the **Properties** pane, rename this data flow from **Data Flow** to **LoadRawData** and configure the source settings.
  + For **Source type**, select **Dataset**, and add your dataset by clicking **+ New**.
  + Choose **Azure Data Lake Storage Gen2** as your storage account, and select **Delimited Text** for CSV file format, depending on your storage account settings.
  + In this Vehicle Sales data, the 'State' column contains codes such as 'CA' and 'NM.' To add state abbreviations, create an additional table and add to the storage account in structdata.

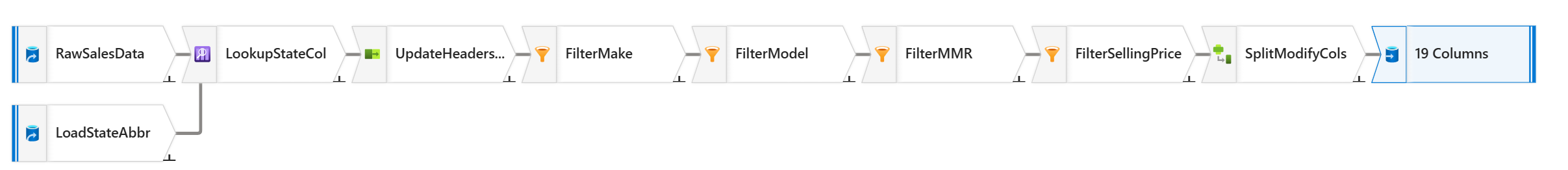
1. **Transformation**:

The Vehicle Sales dataset includes over 500K car sales records for the financial years 2014 and 2015. Various modifications have been applied to ensure data quality. The transformations include lookup, select, filter, and derived columns.

* **Lookup:** Add a condition in the lookup transformation to join data by matching the state name with the state code.
* **Select:** Use a select condition in ADF to update all headers and exclude the 'AlphaCode' column.
* **Filter:** Filter out records where 'Make,' 'Model,' 'MMR,' or 'SellingPrice' fields are empty, as records lacking this basic information are not useful. This step ensures data cleanliness.
* **Derived Column:** Apply transformations in the derived column step to modify dataset columns according to specified conditions.
  + Condition: iif(isNull(Condition), 49, Condition)
  + Odometer: iif(isNull(Odometer), 0, Odometer)
  + Day\_of\_sale: iif(isNull(SaleDate) || SaleDate == '', 'Mon', substring(SaleDate, 0, 4))
  + Sale\_Date: toString(toTimestamp(substring(trim (SaleDate), 5, 11), 'MMM dd yyyy'), 'yyyy-MM-dd')
  + Condition\_Rating: iif(Condition>=1 && Condition <= 10, 'EXCELLENT', iif(Condition >= 11 && Condition <= 20, 'FINE', iif(Condition >= 21 && Condition <= 30, 'VERY GOOD', iif(Condition >= 31 && Condition <= 40, 'GOOD', iif(Condition >= 41 && Condition <= 50, 'RESTORABLE', 'Unrestorable')))))
  + Transmission: iif(isNull(Transmission) || Transmission == 'manual', 'Manual', iif(Transmission == 'automatic' , 'Automatic', 'Hybrid'))
  + Exterior\_Color: iif(isNull(Exterior\_Color) || trim(Exterior\_Color) == '' || !isNull(Exterior\_Color) && !regexMatch(lower(Exterior\_Color), '^[a-z]+$'), 'Neutral', initCap(Exterior\_Color) )
  + Interior\_Color: iif(isNull(Interior\_Color) || trim(Interior\_Color) == ' ' || !isNull(Interior\_Color) && !regexMatch(lower(Interior\_Color), '^[a-z]+$'), 'Neutral', initCap(Interior\_Color))
  + Model: iif(isNull(Model) || isDate(Model), 'Unknown', Model)
  + Car\_Trim: iif(isNull(Trim), 'Unknown', Trim)
  + Body: iif(isNull(Body), 'Unknown', Body)
  + State\_Abbr: iif(isNull(State\_Abbr), 'Canada', State\_Abbr)

1. **Sink (Data Loading):**

* In the pipeline, add a **Sink** ’DimSales’ transformation to define the destination for transformed data to a new container ‘pipelinesink’ and point to the target Data Lake Gen2, not chose SQL database based on cost management. The final pipeline, is structured as follows



**Debug the Data Flow:**

At the top of the data flow designer, enabled **Data flow debug**. Review the default configuration and select **OK**, then wait for the debug cluster to start. In the data flow designer, select the **DimSales** sink and view its **Data preview** tab and review it.

**Schedule the Pipeline:**

* After completing the **LoadRawData** pipeline, click **Publish** to save your changes. Then, select **Add trigger** to run the pipeline. Go to the **Monitor** page to track the pipeline’s progress. The process should complete in under 5 minutes, display a **Succeeded** status, and the table should contain the data loaded by the pipeline.
* Select **New/Edit** to create a new trigger for scheduling. Configure the trigger to run periodically (e.g., daily or every hour) to simulate real-time data ingestion. Define the start time and frequency for the trigger, then click **OK**.

A screenshot of a computer

Description automatically generated

### Step 3: Create and Configure a Database in Azure Synapse Analytics

1. **Create an Azure Synapse Workspace:**

* Set up a new Synapse Workspace in the same Resource group and configure the serverless pool option for cost management.
* Import the data transformed and loaded from Data Factory into Azure Synapse. Using Data tab, go to Linked Service and configure Data Factory to export data directly from Data Lake or Blob storage into Synapse workspace.
* Since this vehicle sales dataset exceeds 80 MB in size, it has been split into four separate files to meet the 20 MB storage limit per file. Each file contains a portion of the data, enabling efficient storage and management within the 20 MB constraint.

1. **Perform Large-Scale Data Analysis:**

* For serverless pool, use SQL queries to analyse the data such as
* For Sales analysis
  + Average Sales per Year
  + Car\_Condition\_Rating\_Count
  + Monthwise\_Growth\_Percentage
  + Top\_5\_Sellers
  + Transmission\_categorywise
  + Yearly\_Sales
* When analysing the data, since the dataset is split into four files, use a wildcard to load all four files simultaneously. For example,

'https://storage1982sales2015.dfs.core.windows.net/pipelinesink/transformoutput/**part-\***'

A screenshot of a computer

Description automatically generated

* Once data analysis begins, it becomes easier to debug, clean, and identify gaps in the data more accurately. Additionally, applying filters individually, rather than combining them with AND or OR operations, proves to be more efficient.

### Step 4: Visualise data in Power BI

Integrate with Power BI

* Connect Power BI through Microsoft Fabric or Synapse when using a dedicated SQL pool. For a serverless pool, open Power BI, add sink data from the storage account, and create the report.
* Open Power BI web, and upload the report to the designated workspace.
* In Synapse, go to **Manage**, select **Linked Services**, click **+New**, and add Power BI with the same workspace. Now, the report will be accessible on the **Develop** page for modifications.

A screenshot of a map with graphs and charts

Description automatically generated

### Step 5: Scalability and Security

**1. Scalability**

* **Adjust Azure Synapse SQL Pool Settings**:
  + In Azure Synapse Analytics, go to your **SQL pool settings** and adjust **Data Warehouse Units (DWUs)**. Increasing DWUs will allocate more resources, enhancing processing power and allowing the system to handle larger data volumes. Similarly, reduce DWUs during low demand to optimize costs.
  + Using a **serverless SQL pool** in Azure Synapse Analytics, you won't have the option to adjust **Data Warehouse Units (DWUs)**, as serverless pools operate on a different model. Instead, here are the steps you can take to manage scalability
    - For Serverless SQL Pool, optimize queries, use partitioned data, leverage materialized views, monitor and analyse query performance, control data scanning and cost management.
* **Enable Auto-scaling in Data Lake**:
  + Enable **auto-scaling** for Azure Data Lake to dynamically manage storage needs. This feature automatically scales storage capacity as data volumes grow, ensuring efficient storage management without manual intervention.
  + Under the **Data management** section, click on **Lifecycle management**. Create rules for automatic data tiering based on the age of data or access patterns (e.g., move data to cooler storage tiers).
  + Use **Azure Monitor** or **Azure Storage Analytics** to track metrics related to your data lake.
  + Overall, while Azure Data Lake Storage itself doesn't require manual auto-scaling settings, following these best practices can help ensure that your data management is efficient, costs are controlled, and your system performs well as data volumes grow.

**2. Security**

* **Implement Access Controls in Azure Storage and Synapse**:
  + Use **Azure Active Directory (AAD)** to manage access to data sources and tools. Set up role-based access control (RBAC) policies to specify who can view, edit, or manage data within Azure Storage and Synapse, ensuring that only authorized users have access to sensitive information.
* **Enable Encryption for Data at Rest and in Transit**:
  + Configure **data encryption** for all stored data (data at rest) and enable **SSL/TLS** for data in transit to protect it during transfer. Azure Storage and Synapse provide options to encrypt data by default, adding a layer of security to your data both while stored and while moving through networks.

A screenshot of a computer

Description automatically generated

## Conclusion:

This project provides hands-on experience in building a scalable, secure Big Data solution in Azure. Through the integration of various services, it demonstrates the power of Azure in handling and processing large datasets while enabling insightful visualization. This end-to-end approach to data management and analysis establishes a strong foundation in Big Data and Azure services.