MAX - Assortment Planning System

Application Description

Environment : Azure Cloud environment

Data Type : Batch and Real Time

Business Teams and Stake Holders:

1. Application used by Store Manager
2. Corporate Teams

Application Data Sources

- Inventory Data [Terradata / Netezza] DWH consists of the following
    1. Daily/Weekly inventory
    2. item hierarchy
    3. Item store
    4. Clearance

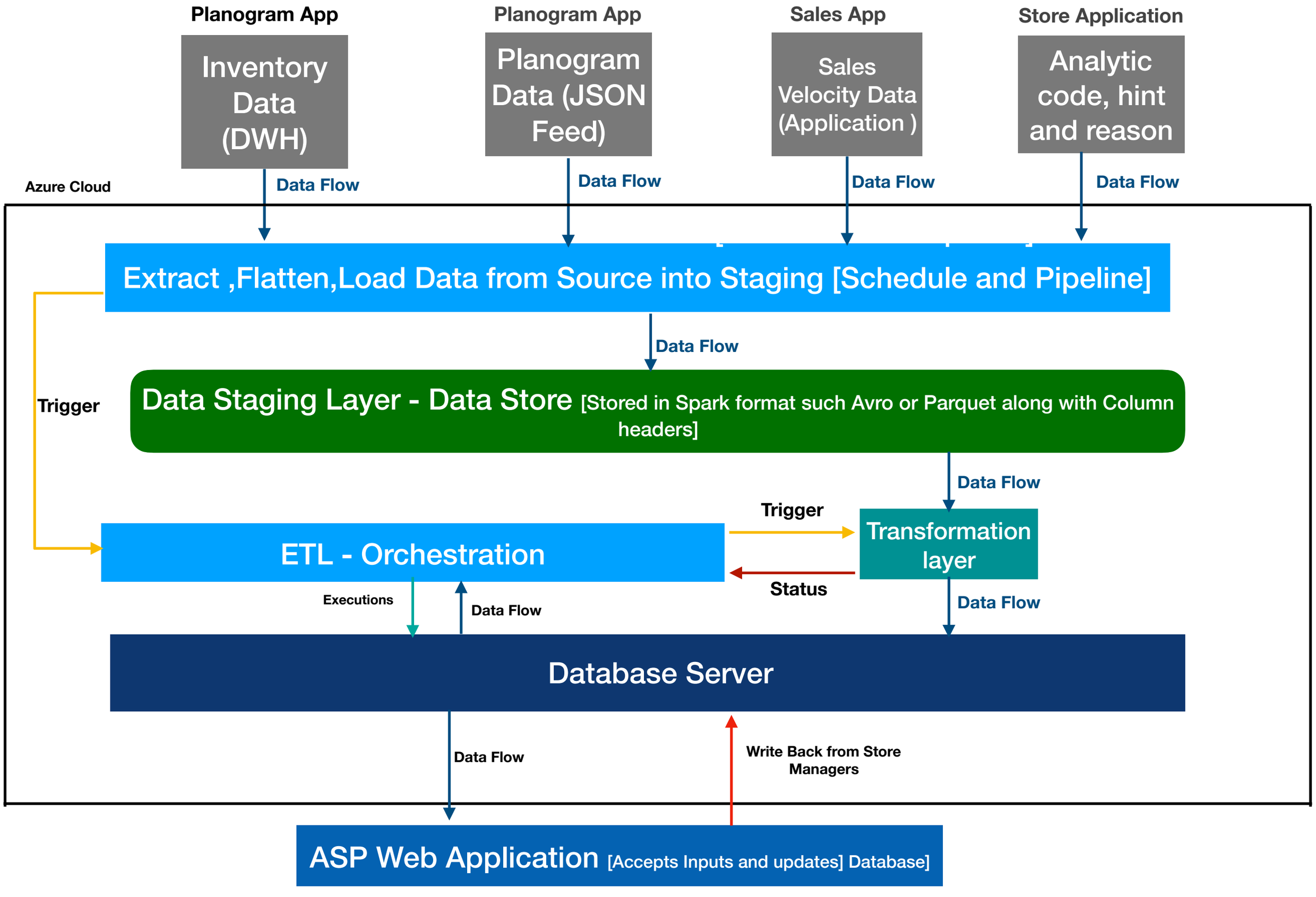
- Planogram Data from Oracle data are available as JSON Feeds

- Sales Velocity is available from different Application (Handshake needs to be created to extract the data)

- Integration of productivity Inventory analytic code, hint Text and reason Code.

- Stored Manager’s reconciliation will happen at the Target level and will be direct write to Target through Assortment planning application.

Data Layer Diagram



Cloud Components and Reasoning

Azure Cloud Components

Azure Data Factory is used as it supports

- ETL orchestration,Event Based Triggers, Integration with majority of the Azure Service like Databricks , Data lake , RDMS.
- Configurable to any level of Monitoring , Notification and easy data transmission.

Azure Data Bricks -

- Optimised for Big Data , PySpark & Spark SQL, Machine Learning.
- Flexible to install all Python Packages and integrated with ODBC along with easy read and write to all the major Data file formats.

Azure Data Lake -

- Optimised to store Spark native Formats and
- Structure folder format and along with indexing.

Azure RDBMS - ACID transactions after read write and logging

Azure Key-vault - Storing Credentials for Data Lake, RDMS.

Azure Integration Runtime -
used to integrate Azure Data factory with any on - premise data store to extract data.

Data Application Architecture

Sources

Store Application

WEB APPLICATION

Sales App

WEB APPLICATION

Planogram App

ORACLE

JSON

Inventory DWH

Metadata

Summary data

Raw data

Data warehouse

Web Activity

1

Web Activity

Event Triggered/ Scheduled Triggered ADF Instance 1

Integration Runtimes

Self-hosted Integration Runtime

Data Pre-Check Notebook

2

Data Lake - Staging

Azure Data Lake Storage Gen2

3

Stage Data flow

Access ADLS through Key vault

Azure Databricks

Azure Databricks

Power BI

Power BI

Spark ODBC

5

ADF 1 Triggers ADF 2 Through Web Hook Activity

4

ADF Instance 2

AzureKeyVault

Azure Key Vault

AZURE RDBMS

6

SQL

Retrieval

Write Back

7

ASP Web Application

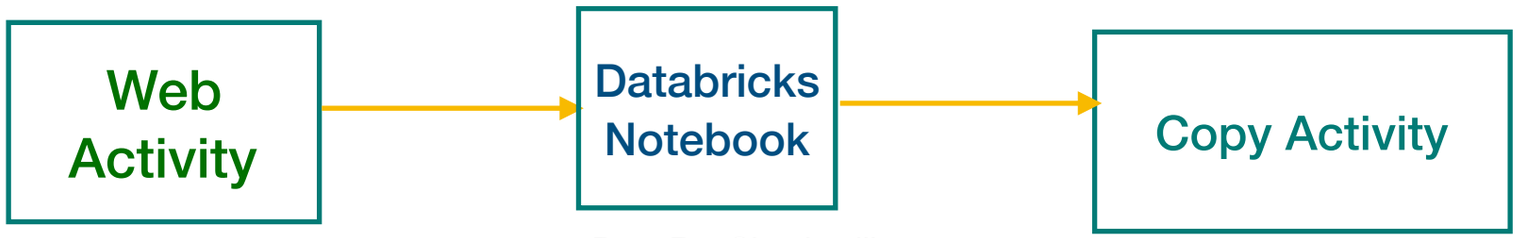
Numbers encircled in grey, represents order of flow of Data from source to target(Database)

Extract and Load Pipeline Design

ASP Pipeline

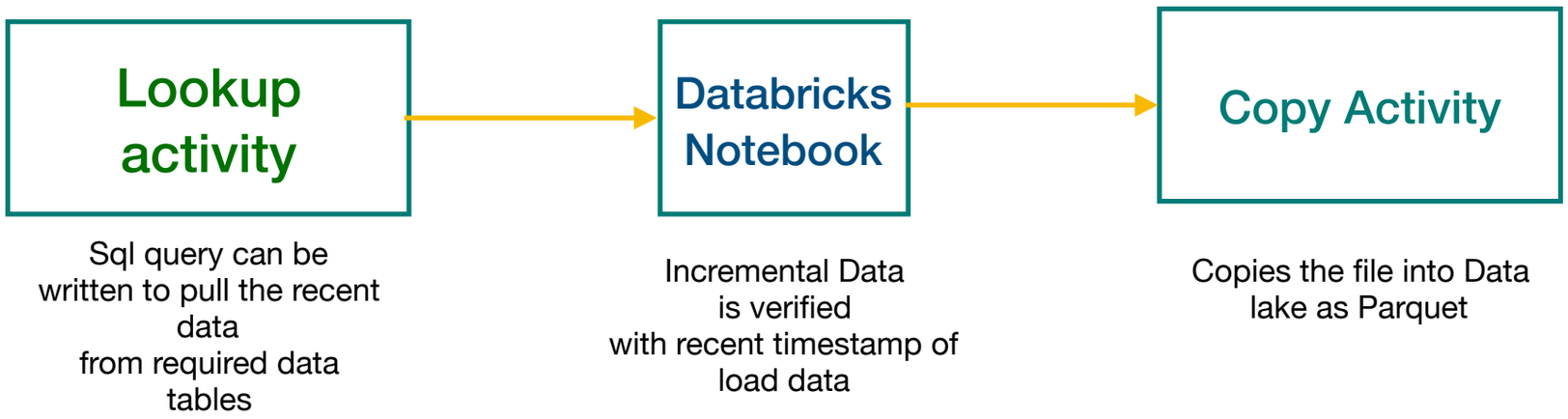
WEB Application Pipeline

Generic Pipe Line Design - to pull and push data from Web Application (SalesVelocity and Store Application) into Data Lake



DWH Pipeline

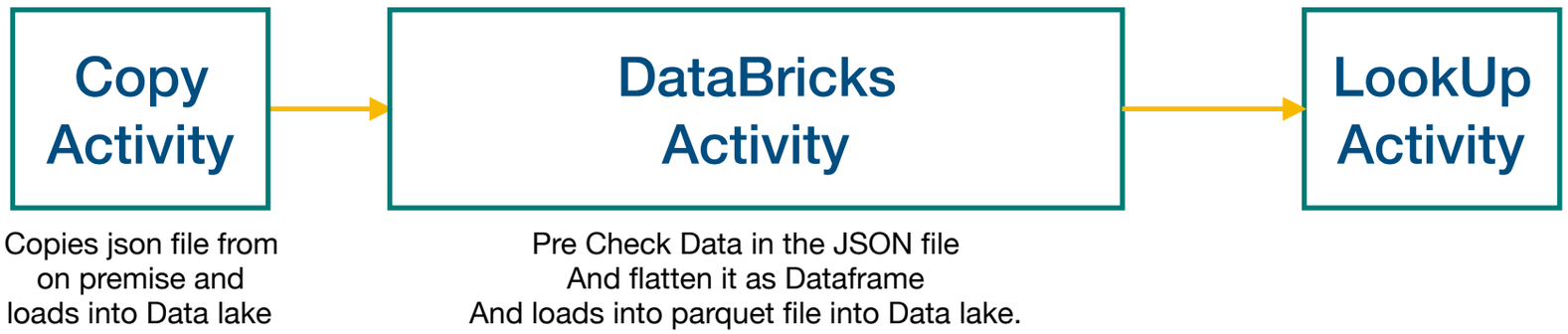
Generic Pipe Line Design - will be configured with linked Service and Integration runtime environment to pull data from DWH



Pipeline Design for Data Extraction

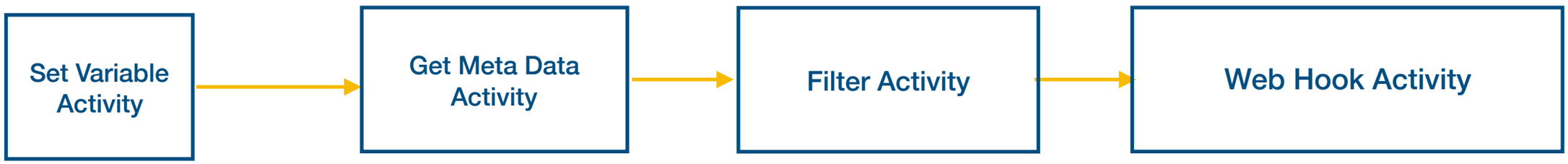
JSON File Extract Pipeline

Pipeline to copy Json file from on premise location and store it as Parquet file in Data lake



Pipeline Design for Web App Data Extraction

Master ASP Pipeline - ADF Instance 1



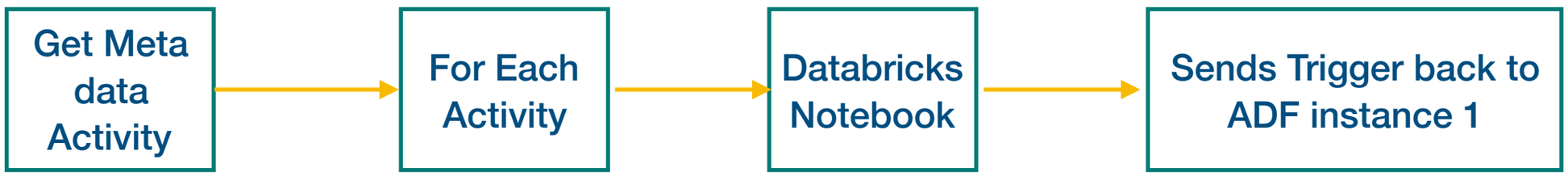
Set today's system date and timestamp

Get last happened Activity from Azure RDBMS

Check for files in Staging for today's data

Triggers the Data transformation pipeline Instance 2 ADF

ADF Pipeline - Instance 2



To check and get list of the recently uploaded Files to Staging Data Lake

For Each file based on the Particular Date Time Stamp will Be sent to Notebook as pipeline variable

Read the Parquet file in the Data Lake Staging Location and Pushes Data through ODBC Into RDBMS

Note :

- The Databricks integrated with ADF instance 2 will transform the data and
- Rank the products based on various parameters and also print the Hint Text

Order of Pipeline Trigger

Assortment Planning System Pipeline list and Execution Order

- WEB (Store and Sale velocity) Applications Pipeline will be ADF pipeline in the ADF Instance 1 and will have Event based trigger, so any online event on above mentioned WEB APPS will be captured by the ADF pipeline in Real time.

- Inventory Data (Terradata / Netezza) from DWH can be pulled in regular intervals using Look up Activity.

- Pipeline run interval Frequency and trigger timing can be amended according to Data load and amount full load runs

- Planogram Data as JSON file can copied and load to Data lake through pipeline which are Triggered based on event (arrival of JSON file into the database)

- Though all source data oriented pipeline are triggered based on Event , Master pipeline will be triggered every 30 mins and interval can be increased or decreased as and when required.

- Master pipeline(ADF Instance 1) will be triggered in Tumbling window to maintain data load and transformation balance at equal intervals of time.

- ADF instance 2 is connected with ADF MASTER Pipeline Instance 1 and hence would not require any individual triggers.

- Since the Data Parameter pre-check and Data Transformation are happening in the Databricks , we can deploy or amend necessary changes time to time in the Databricks , without disturbing the Master pipeline or it’s Triggers.

Generic JSON Extraction Code

```
## code written in Spark
## files will be. Read from the location directly to execute
## Json can be flatten using the below.
## PySpark is being used since we are using Databricks, can will support PySpark code.

from pyspark.sql Import SparkSession

   Spark = SparkSession.builder.appName(“JSON_Handler”).getOrCreate()

   Json_path = ‘A/B/C’

   RDD_JS= spark.sparkContext.parallelize(json_path)

   Df_json= spark.read.option(“multiline”,”True”).json(RDD_JS)

   df_flatten = flatten_json(df_JSON)

   df_flatten.show(20)
```

Time Line Required

Phases of the project and Impact of the Project:

- Requirement Gathering and Data understanding (minimum of 3 days)
- **IDS** - identification of Sources and **Analytical Modelling** of Data are important. (min of 2 week)
- In Parallel, Data Analysis of and Logic Building minimum of week's time(5 days)
- Infra structure set up - [as most of the resources and services are on cloud - this would not take more time]
- Azure Data factory **ETL** Pipeline Development - Will take minimum of 3 days, as we need to design and test run the parameters at each stage.
- Building the Algorithm based on the business logic will take considerable amount of time with minimum of 10 days - as we need write Pre check code and Transformation
- We need to build a reporting Dashboard using Power BI or Kibana as per requirement to test data feed , before WEB Application for Assortment Planning is Ready - [min 4 days - max 8 days]
- In Total, a minimum of 35 days is required to complete majority of the integration works
- In Terms of **Service Level Agreement** and Cost this Project is not a complex but seems to be Prime project , as it is used by Store managers/corporate teams to get overall Product ranks.

Challenges

- The **Number of tables and columns** from the Schema Inventory, clearance, Item Store and Item hierarchy should be understood in details.

- **Missing Data or data gaps**, should be analysed with proper querying on the DWH

- Incase if **Older or historical data** from sources [Since all the data sources are existing and sitting in **production environment** will hold humongous amount of data] needs to be process , we would need runs the Data pipelines before GO Live Release.

- Above point, if considered real, then it would add few more weeks in planning and loading the existing data feeds along with real time data.

- **Frequency of Data ingestion** from Sales velocity and Store Web applications should be studied carefully.

- Pipeline Design provided in the above slides are considering the, Web Application will provide data based on the key value pair or JSON format and **uses GET & POST API** methods through ADF pipelines.

- Some Web applications will also have capacity to send data as CSV or PDF files and hence, that can be read using proper Machine Learning Techniques.

- On the incoming data, **ML algorithm** should be effectively applied to leverage the best use of Data Lake otherwise, the pipelines will end up loading duplicate or corrupt data which will again become intolerant.