**AI POWER POC**

**Problem Statement**

The current process involves running Python scripts locally on files stored in the local file system, which is time-consuming and not scalable. The goal is to migrate the project to a cloud-based infrastructure that can handle large volumes of data and automate the process of data cleaning and output generation.

**Proposed Solution**

Data Extraction: The first step is to extract the data from the Gen2 Lake Storage linked in Synapse. This involves setting up the appropriate authentication and access permissions to the storage account, and then reading the data into the Synapse notebook environment.

Data Transformation: Once the data has been extracted, it needs to be transformed using PySpark. The data should be loaded into a PySpark DataFrame, and then cleaned and transformed as necessary. This may involve filtering, aggregating, or joining data from multiple sources.

Data Loading: Once the data has been transformed, it needs to be loaded into an Azure PostgreSQL database. This involves creating a connection to the database, and then inserting the cleaned data into the appropriate tables.

Monitoring and Management: Finally, the solution needs to be monitored and managed to ensure that it is running smoothly. This may involve setting up alerts and notifications to detect and address any issues that arise.

**Tech Stacks**

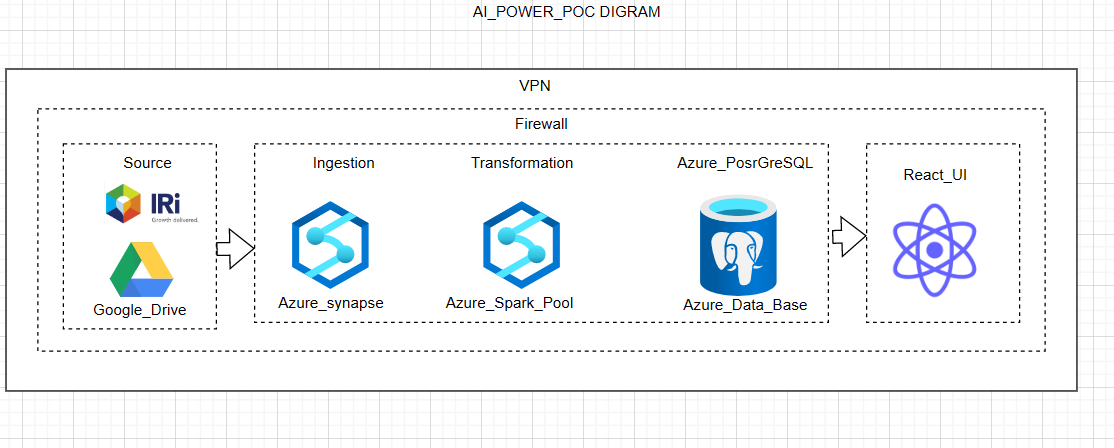
Azure Blob Storage for data storage

Azure Gen2 Lake Storage

Azure Postgres Database

Azure Synapse Analytics for data processing and cleaning

**Pipeline Architecture**





**Synapse Notebook Script**

| **from** pyspark **import** SparkConf, SparkContext **from** pyspark.sql **import** SparkSession **import** pyspark.sql.functions **as** F **import** pandas **as** pd # Optional as synapse spark configuration not working properly spark = SparkSession.builder.appName("MyApp").getOrCreate() conf = SparkConf().set("spark.executor.cores",'7') print('hello') # load excel file into synapse notebook  pdf=pd.read\_excel('abfss://data@datalgen22.dfs.core.windows.net/all\_id\_cont.xlsx') pdf = pdf.drop('intercept', axis=1) # Create a DataFrame from pandas frame df = spark.createDataFrame(pdf) # display(df) df.printSchema()    # Define the JDBC connection properties jdbcHostname = "de-postgres-server.postgres.database.azure.com" jdbcPort = 5432 jdbcDatabase = "postgres" jdbcUsername = "<azure\_postgres\_username>" jdbcPassword = "<azure\_postgres\_password>" jdbcUrl = f"jdbc:postgresql://{jdbcHostname}:{jdbcPort}/{jdbcDatabase}"  # Define the JDBC driver class jdbcDriverClass = "org.postgresql.Driver"  # Write the DataFrame to the Azure PostgreSQL database df.write \  .format("jdbc") \  .option("url", jdbcUrl) \  .option("user", jdbcUsername) \  .option("password", jdbcPassword) \  .option("driver", jdbcDriverClass) \  .option("dbtable", "all\_id\_cont") \  .mode("overwrite") \  .save() |
| --- |