Project Overview: Implementing Lakehouse and Data Processing

**1. Creating Lakehouse and Warehouse**

**Screenshot: Creation of Lakehouse and Warehouse**

A screenshot of a computer

Description automatically generated

**2. Loading Data into Tables in Lakehouse from Files**

**Convert JSON data into Table Format with** [**PySpark Notebook**](https://github.com/DevSumit913/MS-Fabric-Merilytics/blob/main/DF_NB_API_TO%20RAW.ipynb)

spark.conf.set("spark.microsoft.delta.optimizeWrite.enabled", "true")

spark.conf.set("spark.microsoft.delta.optimizeWrite.binSize", "1073741824")

spark.conf.set("spark.sql.parquet.int96RebaseModeInRead", "LEGACY")

spark.conf.set("spark.sql.parquet.int96RebaseModeInWrite", "LEGACY")

spark.conf.set("spark.sql.legacy.timeParserPolicy", "CORRECTED")

from pyspark.sql.functions import col, when

from pyspark.sql import functions as F

from pyspark.sql.types import DataType

from datetime import datetime

import re

import os

import pandas as pd

*# Define the path to your JSON file*

file\_path = "/lakehouse/default/Files/api\_data.json"

*# Load JSON data into a Pandas DataFrame*

df = pd.read\_json(file\_path)

spark\_df = spark.createDataFrame(df)

*# Add "nav\_" prefix and convert to lowercase*

final\_table\_name = f"bronze\_api\_data"

*# Save the DataFrame as a Delta table with column mapping mode 'name' and schema merge enabled*

spark\_df.write.format("delta").mode("overwrite")\

.option("delta.columnMapping.mode", "name")\

.option("mergeSchema", "true")\

.saveAsTable(final\_table\_name)

**Screenshot: Loading Data into Lakehouse**

* **File Upload Interface:**
* A screenshot of a computer

  Description automatically generated
* **Table Creation:**
  + Display the creation of tables in the Lakehouse, including schema definition and table configurations.

*Note: Screenshots should show the file selection and table creation process.*

**3. Data Cleaning with PySpark Notebook**

**Screenshot: Data Cleaning in** [**PySpark Notebook**](https://github.com/DevSumit913/MS-Fabric-Merilytics/blob/main/DE_NB_BONZE_TO_SILVER.ipynb)

* **PySpark Notebook:**

from pyspark.sql.functions import col, when

from pyspark.sql import functions as F

from pyspark.sql.types import DataType

from datetime import datetime

import re

import os

df = spark.sql("SELECT \* FROM DE\_LH\_RAW.erp\_data")

*# Replace nulls in integer/float columns with 0*

df\_cleaned = df.select([when(col(c).isNull(), 0).otherwise(col(c)).alias(c)

if t in ["int", "double", "float"] else col(c) for c, t in df.dtypes])

*# Replace nulls in string columns with 'UNKNOWN'*

df\_cleaned = df\_cleaned.select([when(col(c).isNull(), "UNKNOWN").otherwise(col(c)).alias(c)

if t == "string" else col(c) for c, t in df\_cleaned.dtypes])

*# Capitalize column names*

df\_cleaned = df\_cleaned.select([F.col(c).alias(c.upper()) for c in df\_cleaned.columns])

*# Add "nav\_" prefix and convert to lowercase*

final\_table\_name = f"silver\_erp\_data"

*# Add a special suffix for the Delta table*

delta\_table\_name = f"{final\_table\_name}\_delta"

*# Save the DataFrame as a Delta table with column mapping mode 'name' and schema merge enabled*

df\_cleaned.write.format("delta").mode("overwrite")\

.option("delta.columnMapping.mode", "name")\

.option("mergeSchema", "true")\

.saveAsTable(delta\_table\_name)

**4. Previewing Data for api\_data in Silver**

**Screenshot: Preview of Data for api\_data in Silver Layer**

* **Table Preview Interface:**

A screenshot of a computer

Description automatically generated

* **Data Preview Interface:**

A screenshot of a computer

Description automatically generated

**5. Following the Medallion Architecture**

**Screenshot: Medallion Architecture Implementation**

* **Implementation Interface:**

A screenshot of a computer

Description automatically generated