**Getting the Most Out of Azure Synapse: A User-Friendly Guide**

Azure Synapse Analytics is a powerful tool from Microsoft that combines big data and data warehousing into a single, integrated service. It allows you to easily manage, develop, and analyze data from various sources. This blog will guide you through the main components of Azure Synapse Analytics — Data, Develop, Integrate, Monitor, and Manage.

**Data:-**  
This hub allows you to interact with your data, stored in Azure Data Lake Storage (ADLS) Gen2, SQL databases, or other data sources.

A screenshot of a computer

Description automatically generated

1. In the **Linked tab**, within **Azure Data Lake Storage Gen 2,** the storage account **synapsedemo777** contains a container named **task(primary)**.

2. Inside this container, there’s a folder called **synapsedemo**, which is part of the organized structure within ADLS Gen2.

3. The file ‘**GreenTaxiTripData\_201812-sample.csv’** is stored here and is easily accessible within the Synapse workspace for further processing or analysis.

**Develop:-**In the Develop hub, perform an ETL (Extract, Transform, Load) process by creating a notebook named **TaxiDataTransformation**. Use the Apache Spark pool named **samplesparkpool** to execute the notebook.

a. Read the data from Azure DataLake Gen 2

b. Clean the data

c. Transform the data

d. Load the data into Spark managed table

A screenshot of a computer

Description automatically generated**TaxiDataTransformation Notebook**

**Code:-**

from pyspark.sql import SparkSession  
from pyspark.sql.types import \*  
from pyspark.sql.functions import col, year, month, dayofmonth, unix\_timestamp, round, when  
from datetime import datetime  
  
# Read the Data from Azure DataLake Gen 2  
greentaxidf = spark.read.load('abfss://users@datatechdemostorage.dfs.core.windows.net/taxidata/GreenTaxiTripData\_201812.csv', format='csv', header=True)  
display(greentaxidf.limit(10))  
  
#Filtering the rows  
greentaxidf = greentaxidf.where(col('passenger\_count') > 0) \  
 .filter(col('trip\_distance') > 0.0) \  
 .dropDuplicates()   
  
# renaming the old columns and adding new columns  
greentaxidf = greentaxidf.select( col("VendorID"),  
 col("passenger\_count").alias("PassengerCount"),  
 col("trip\_distance").alias("TripDistance"),  
 col("lpep\_pickup\_datetime").alias("PickupTime"),   
 col("lpep\_dropoff\_datetime").alias("DropTime"),   
 col("PUlocationID").alias("PickupLocationId"),   
 col("DOlocationID").alias("DropLocationId"),   
 col("RatecodeID"),   
 col("total\_amount").alias("TotalAmount"),  
 col("payment\_type").alias("PaymentType")  
 ) \  
 .withColumn("TripYear", year("PickupTime")) \  
 .withColumn("TripMonth", month("PickupTime")) \  
 .withColumn("TripDay", dayofmonth("PickupTime")) \  
 .withColumn("TripTimeInMinutes", round((unix\_timestamp(col("DropTime")) - unix\_timestamp(col("PickupTime")) / 60))) \  
 .withColumn("TripType", when(col("RatecodeID") == 6,"SharedTrip").otherwise("SoloTrip")) \  
 .drop("RatecodeID")  
# To print the updated schema of the DataFrame.  
greentaxidf.printSchema()   
  
# Store the DataFrame as a managed spark Table  
spark.sql("CREATE DATABASE IF NOT EXISTS nyctaxi")  
greentaxidf.repartition(4).write.mode("overwrite").saveAsTable("nyctaxi.greentrip")

This notebook appears to be part of your ETL (Extract, Transform, Load) process for transforming **‘Green Taxi Trip data’** stored in Azure Data Lake Storage. Used PySpark code to clean, transform, and select relevant data fields.

A screenshot of a computer

Description automatically generated**UpdatePaymentType\_sp SQL Script**

The **UpdatePaymentType\_sp** procedure is a simple SQL script that updates the payment type for all records in the **greentaxi** table. This could be useful for updating the default payment method or applying a new payment policy.

**Integrate:-**In **Integrate** hub, is where you manage all your data pipelines within Azure Synapse Analytics. It shows a list of the pipelines;

1. **Copy\_spark\_SQL**: This pipeline handles copying data from the Delta table into an SQL Database.
2. **ETLPipeline**: This is the pipeline currently selected and displayed in the workspace.

A screenshot of a computer

Description automatically generated

**Copy\_spark\_SQL**

A screenshot of a computer

Description automatically generated

**ETL (Extract, Transform, Load) Pipeline**

**Pipeline Activities in ETLPipeline:-**

**1. Run PySpark ETL:-**Using notebook activity we run the PySpark ETL process, performing the extract, transform, and load operations.

**2. Wait:-**After triggering the ETL job, the pipeline waits 10 seconds for the PySpark job to complete.

**3. Copy the Spark SQL data to the Azure SQL Database:-**Copy data activity used to transfer data from the Spark-managed table into an Azure SQL Database.

**4. Run a stored procedure on the Azure SQL table to update records:-**It triggers a stored procedure in the Azure SQL Database, such as updating records, performed on the data after it has been copied to the SQL database.

**Monitor:-**

A screenshot of a computer

Description automatically generated

This part of Azure Synapse Analytics gives you a clear view of when your pipelines were run, how long they took, and whether they were successful. It helps you easily keep track of and troubleshoot your ETL processes.

**Manage:-**

A screenshot of a computer

Description automatically generated

In the **Manage** hub, there’s an option for creating different types of pools. A dedicated SQL pool named ‘**datademoSQL’** has been created. Additionally, an Apache Spark pool called ‘**samplesparkpool’** was set up, using a "Memory Optimized" node size. The configuration for this Spark pool is "Small," with 4 vCores, and 32 GB of memory, and it operates with 3 nodes.

**Main Difference between Azure Data Factory and Azure Synapse Analytics:-**

A table with text on it

Description automatically generated

**Limitation of Azure Synapse Analytics:-**

1. **Complexity:-** Synapse combines many different services and tools into one platform, which can make it complex to learn and use effectively.

2. **Cost Management:-** It can be expensive, particularly when using dedicated SQL pools, Spark pools, or running large-scale data processing jobs.

3. **Concurrency Limitations:-**There’s a limit on how many queries can run at the same time.  
Concurrency Limits by Service Level:

* **DW100c to DW500c**: Up to 32 concurrent queries.
* **DW1000c to DW3000c**: Up to 48 concurrent queries.
* **DW6000c and above**: Up to 128 concurrent queries.

4. **No Real-Time Processing**: It’s designed for batch processing, not real-time data analytics.

5. **Service Availability in Regions:-**Azure Synapse Analytics may not be available in all Azure regions, and certain features might be region-specific.

**Conclusion:-**

Azure Synapse Analytics is a powerful tool that brings together big data and data warehousing into a single platform. It simplifies how organizations store, process, and analyze data, making it easier to manage everything in one place.   
However, its many features can be overwhelming, especially for those new to the platform, and the costs can add up quickly. It’s great for handling large amounts of data, but there are limits on how many tasks you can run at the same time, and not all features are available everywhere. Despite these challenges, Azure Synapse offers a strong solution for businesses looking to get the most out of their data.