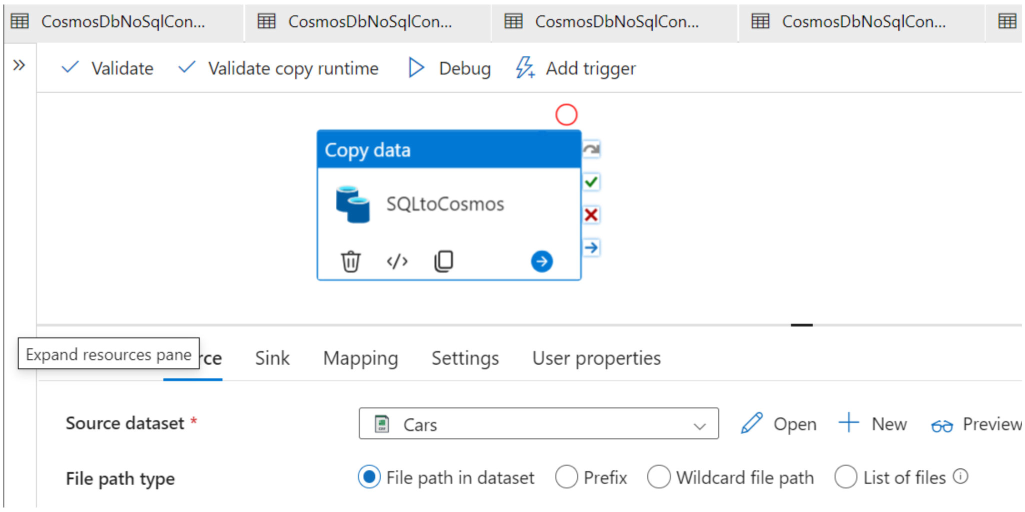
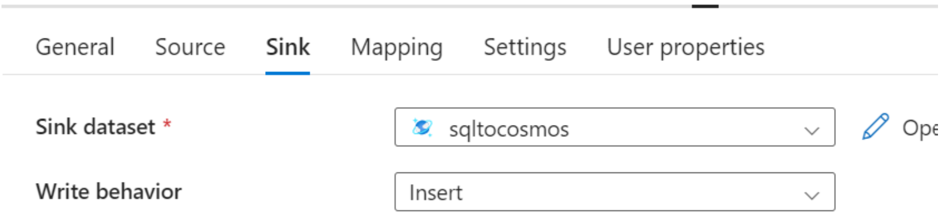
Assignment 10 - Azure Data Factory

1. Design an ADF pipeline to copy data from an on-premise Azure SQL database to Azure Cosmos DB, ensuring data consistency and performance optimization. Pick correct options of partitioning for better performance.



Source dataset * Cars

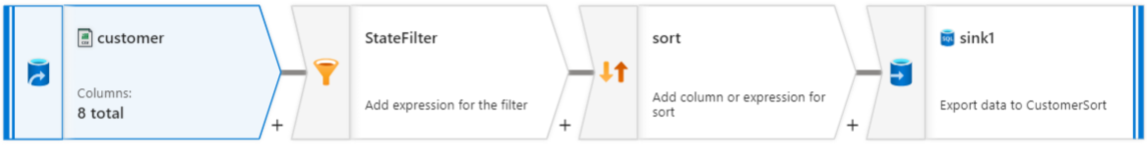
File path type ● File path in dataset ○ Prefix ○ Wildcard file path ○ List of files

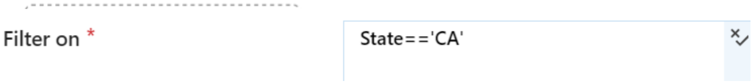


Sink dataset * sqltocosmos

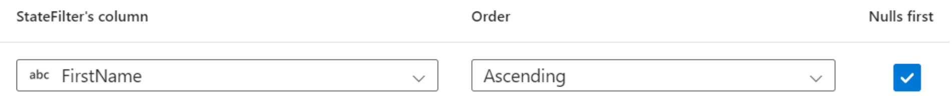
Write behavior Insert

2. Create Pipeline using Azure Data Flow in Azure Data Factory to apply Filter and Sort transformations on datasets.


customer
Columns:
8 total
StateFilter
Add expression for the filter
sort
Add column or expression for sort
sink1
Export data to CustomerSort

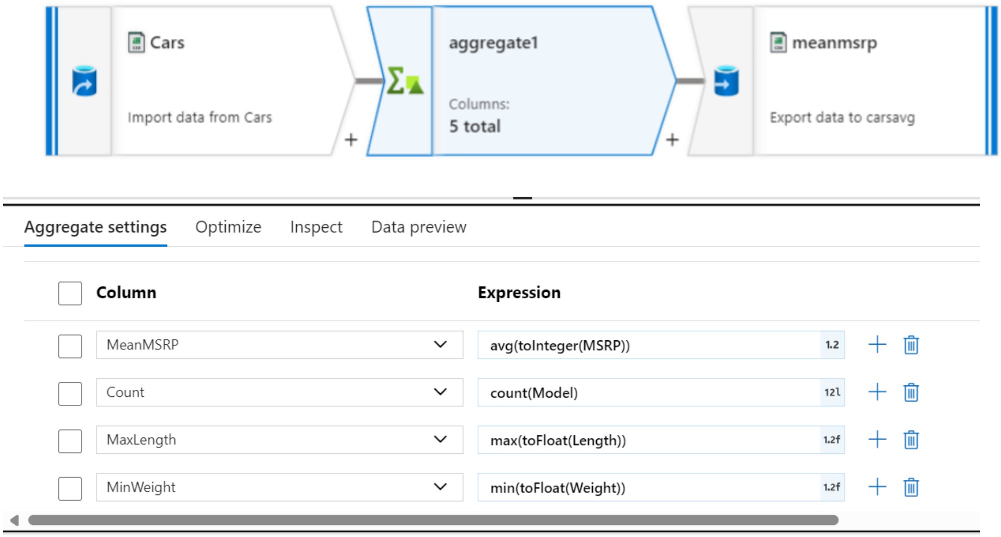


Filter on * State=='CA'



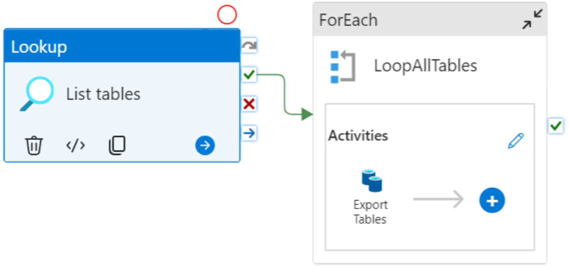
StateFilter's column	Order	Nulls first
abc FirstName	Ascending	☑

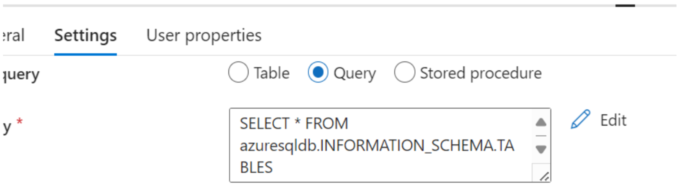
3. Design an ADF pipeline to implement aggregate operations, such as sum, average, max, min and count, within an Azure Data Flow.



□ Column	Expression
□ MeanMSRP	avg(toInteger(MSRP))
□ Count	count(Model)
□ MaxLength	max(toFloat(Length))
□ MinWeight	min(toFloat(Weight))

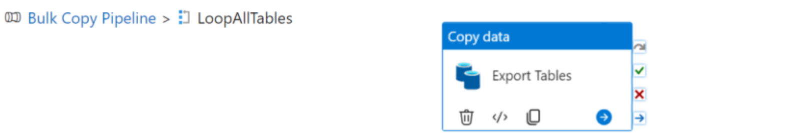
4. Create best approach to bulk copy data from multiple homogenous sources into Azure SQL Database using ADF pipelines. Show usage of Lookup, For Each Loop and Expressions in Azure Data Factory.

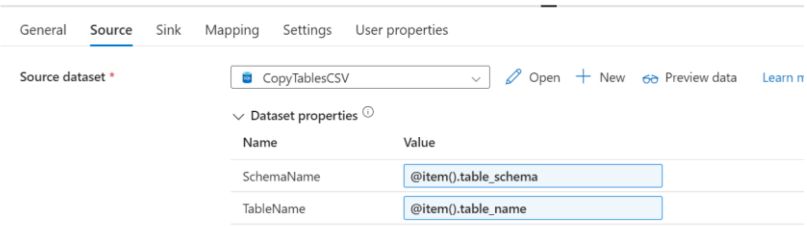




uery ○ Table ● Query ○ Stored procedure

y * SELECT * FROM azuresqldb.INFORMATION_SCHEMA.TABLES ✎ Edit

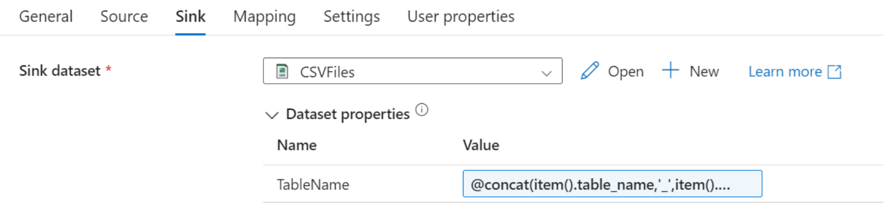




Source dataset * CopyTablesCSV

Dataset properties

Name	Value
SchemaName	@item().table_schema
TableName	@item().table_name

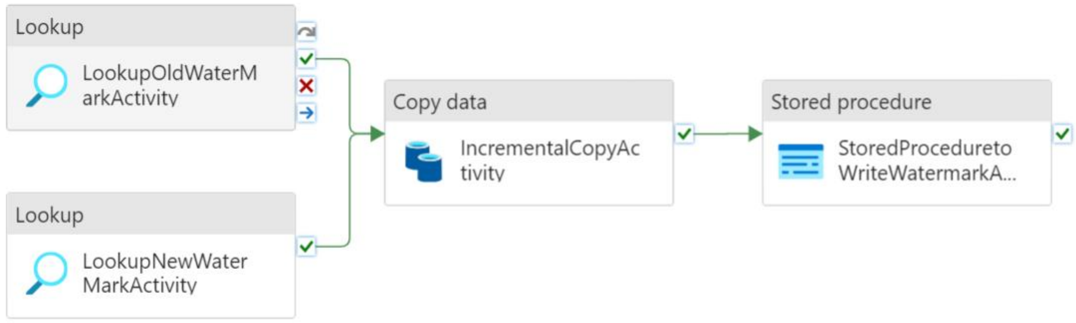


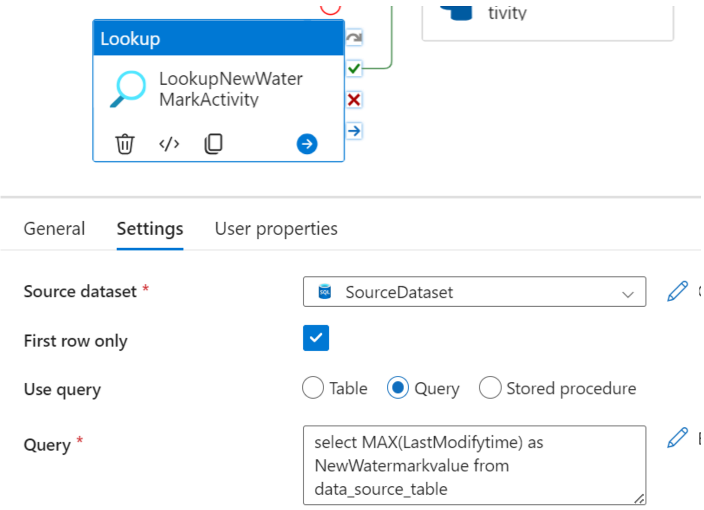
Sink dataset * CSVFiles

Dataset properties

Name	Value
TableName	@concat(item().table_name,'_',item()....

5. Implement incremental load Pipeline in Azure Data Factory for handling datasets, ensuring efficient insert/upsert/updates to the target storage without re-inserting the entire dataset?


Lookup
LookupOldWaterMarkActivity
Lookup
LookupNewWaterMarkActivity
Copy data
IncrementalCopyActivity
Stored procedure
StoredProceduretoWriteWatermarkA...



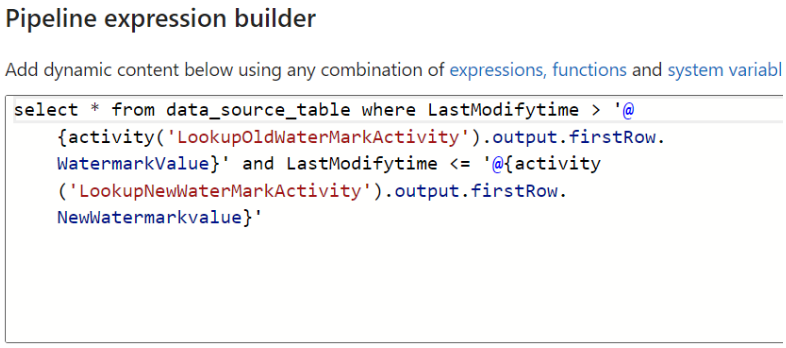
Source dataset * SourceDataset

First row only ☑

Use query ○ Table ● Query ○ Stored procedure

Query *

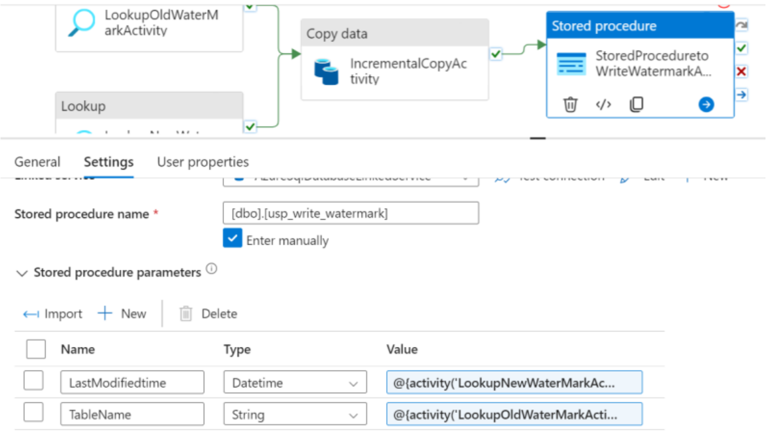
```
select MAX(LastModifytime) as
NewWatermarkvalue from
data_source_table
```



Pipeline expression builder

Add dynamic content below using any combination of expressions, functions and system variabl

```
select * from data_source_table where LastModifytime > '@
    {activity('LookupOldWaterMarkActivity').output.firstRow.
    WatermarkValue}' and LastModifytime <= '@{activity
    ('LookupNewWaterMarkActivity').output.firstRow.
    NewWatermarkvalue}'
```



Stored procedure name * [dbo].[usp_write_watermark]

☑ Enter manually

Stored procedure parameters

Import + New Delete

□	Name	Type	Value
□	LastModifiedtime	Datetime	@{activity('LookupNewWaterMarkAc...
□	TableName	String	@{activity('LookupOldWaterMarkActi...

6. What are the key steps to connect Azure Databricks to Cosmos DB for real-time analytics and data transformation using spark and Databricks.

   - Create a Databricks Workspace in Azure.
   - Set up a Cluster in the Databricks workspace.
   - Install the Cosmos DB Spark Connector on the cluster:
     com.azure.cosmos.spark:azure-cosmos-spark_3-3_2-12:4.22.0

   - Create a Cosmos DB Account in Azure.
   - Generate Cosmos DB Connection Details:
     Get the URI, Primary Key, and Database Name from your Cosmos DB account.

```
Endpoint = "https://swiggycosmos.documents.azure.com:443/"
Masterkey =
"jAWfOozBwIzueCRU87cIeVr3UVO6tPg0zere4QZP9PAVjIEFf3gBcWak9wFFpcmN7n4MNsvhg2fLACDblNk3Rg=="
Database = "SwiggyDatabase"
RestaurantContainer="restaurantswiggy"
CityContainer = "city"
MenuContainer="menu"
restaurant_df = spark.read.format("cosmos.oltp") \
    .option("spark.cosmos.accountEndpoint", Endpoint) \
    .option("spark.cosmos.accountKey", Masterkey) \
    .option("spark.cosmos.database",Database ) \
    .option("spark.cosmos.container", RestaurantContainer) \
    .load()
```

   - Write Connection Code in Databricks:
     Use Spark with the Cosmos DB connector to establish the connection:

```
restaurant_df = spark.read.format("cosmos.oltp") \
  .option("spark.cosmos.accountEndpoint", Endpoint) \
  .option("spark.cosmos.accountKey", Masterkey) \
  .option("spark.cosmos.database",Database ) \
  .option("spark.cosmos.container", RestaurantContainer) \
  .load()
```

   - Perform Data Transformations in Databricks using Spark on the Cosmos DB data.
   - Write Data Back to Cosmos DB (optional):
     Use .write() method to send processed data back to Cosmos DB.

```
storage_account_name = "azureblob285496"
storage_account_key =
"JN3rN+UOMu0dLq45XTT8K0vrkKeKl09vZaJartsX+Z8Ur5oIJWgRA1qVE5MX7jJ51Hq3NZ2fChad+ASt1n4Sqw=="
container_name = "cleanswiggydata"
spark.conf.set(f"fs.azure.account.key.{storage_account_name}.blob.core.windows.net",
storage_account_key)
output_path =
f"wasbs://{container_name}@{storage_account_name}.blob.core.windows.net/swiggytransformedda
ta/"
final_df.write.mode("overwrite").parquet(output_path)
```