

EV Population Data Pipeline Implementation & Demo





TOC

Overview

Pipeline Flow Architecture

Demo



Overview

As a part of the interview process for the Snowflake SI Partner Data Cloud Architect role, the hiring team would assess and review the candidate's technical hands-on skills in Spark, data processing, and data pipeline design.

This assessment requires a demo implementation exercise, which requires the candidate to:

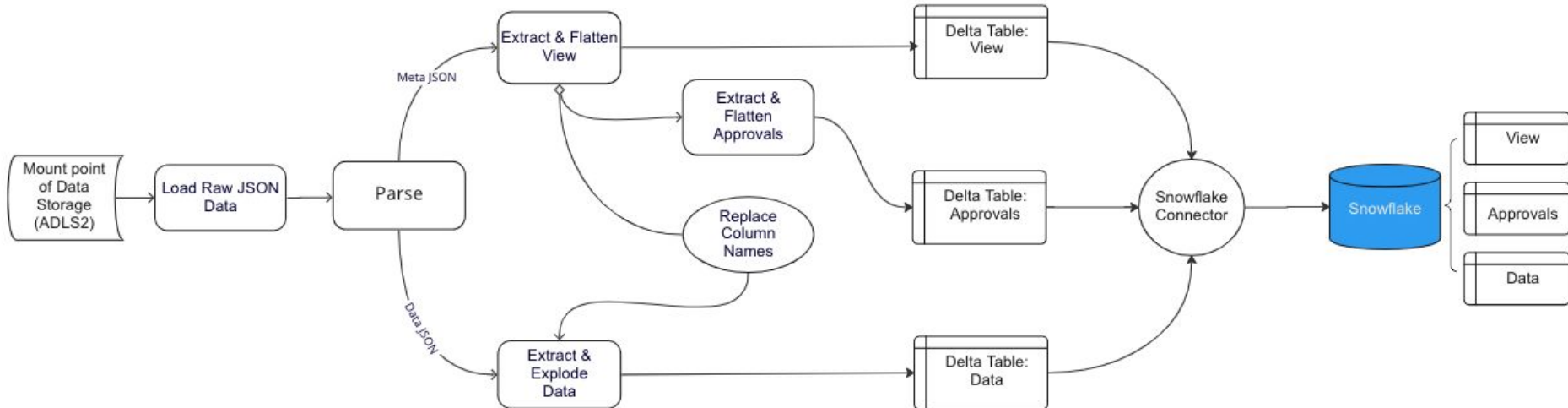
- build a data pipeline that parses the raw data from an Electric Vehicle Population Dataset (provided in a JSON file),
- extract the necessary information, and ingest it into Delta tables and then into DB tables.
- create insightful analysis based on the datasets.





Pipeline Architecture

Data Ingestion Pipeline



Demo

01: Databricks/Spark

Using Databricks Notebooks, created a Job pipeline which:

- Loads the raw data from ADLS2 storage
- Creates Spark dataframes to parse and extracts JSON objects, meta/view, meta/view/approvals, meta/view/columns and data[]
- Creates Delta Tables to flatten and store the parsed data
- Inserts the same data into Snowflake tables using Snowflake python connector

And created insights analysis example queries.

off Azure

datahacks

insight_analysis

Python

File Edit View Run Help

Last edit was yesterday

Give feedback

Run all

KEN K's Cluster - sing-...

Schedule

Share

Based on the data, which car make and model would you recommend?

```
1 #found string data type in Electric Range, fix the prob first and drop rows have any null column value and the rows that have zero-value Electric_Rnage
2 dBase = dfData.select("Model Year", "Make", "Model", F.col("Electric Range").cast("int").alias("Electric_Range")).na.drop().filter("Electric_Range > 0")
3
4 dEfficiency = dBase.groupBy("Make", "Model")\
5   .agg(F.count("Model").alias("count"), F.avg("Electric_Range").alias("Avg_Electric_Range"))\
6   .orderBy("Avg_Electric_Range", ascending=False).limit(20)
7 dEfficiency.show()
8
9 dEfficiency2 = dBase.groupBy("Model Year", "Make", "Model")\
10  .agg(F.count("Model").alias("count"), F.avg("Electric_Range").alias("Avg_Electric_Range"))\
11  .orderBy("Avg_Electric_Range", ascending=False).limit(20)
12 dEfficiency2.show()
```

(4) Spark Jobs

dBase: pyspark.sql.dataframe.DataFrame = [Model Year: string, Make: string ... 2 more fields]

dEfficiency: pyspark.sql.dataframe.DataFrame = [Make: string, Model: string ... 2 more fields]

dEfficiency2: pyspark.sql.dataframe.DataFrame = [Model Year: string, Make: string ... 3 more fields]

Make	Model	count	Avg_Electric_Range
TESLA	MODEL Y	433	291.0
HYUNDAI	KONA	43	258.0
CHEVROLET	BOLT EV	773	243.27037516170762
TESLA	MODEL X	575	241.47130434782608
TESLA	ROADSTER	7	237.85714285714286
TESLA	MODEL 3	2541	237.61668634395906
JAGUAR	I-PACE	35	234.0
POLESTAR	PS2	9	233.0
TESLA	MODEL S	1147	226.6713164777681
AUDI E-TRON	SPORTBACK	13	218.0
AUDI	E-TRON	101	208.27722722727272
PORSCHE	TAYCAN	31	198.03225806451613
KIA	NIRO	316	143.28481012658227
MINI	HARDTOP	17	110.0
VOLKSWAGEN	E-GOLF	194	108.97938144329896
TOYOTA	RAV4	8	103.0
NISSAN	LEAF	2199	102.13278763074125
THINK	CTTV	1	100.0

Command took 1.78 seconds -- by zenken@gmail.com at 3/16/2023, 12:09:22 AM on KEN K's Cluster - single node

Created a Snowpipe to process the same JSON data ingestion to compare to Databricks Job Workflow.

Used Snowpark API to perform the same Dataframe

EV_POP_DATA.DBO

```

25 -- Based on the data, which car make and model would you recommend?
26 order by count desc;
27
28 select * from popularity_phevs
29
30 --Based on the data, which car make and model would you recommend?
31 create or replace view make_model_recommend_by_efficiency as
32 select "Model Year", Make, Model, COUNT(*) as count, AVG(CAST("Electric Range" AS INTEGER)) as
33 Avg_Electric_Range
34 from DATA_TBL_FROM_DB
35 where "Electric Range" > 0 and "Electric Range" <> ''
36 group by "Model Year", Make, Model
37 order by Avg_Electric_Range desc;
38
39 select * from make_model_recommend_by_efficiency;

```

Objects

Editor

Results

Chart

	Model Year	MAKE	MODEL	COUNT	AVG_ELECTRIC_R
1	2020	TESLA	MODEL S	68	331.0
2	2020	TESLA	MODEL 3	648	297.7
3	2020	TESLA	MODEL X	120	291.5
4	2020	TESLA	MODEL Y	433	
5	2019	TESLA	MODEL X	80	
6	2019	TESLA	MODEL S	58	
7	2012	TESLA	MODEL S	35	
8	2020	CHEVROLET	BOLT EV	194	
9	2020	HYUNDAI	KONA	34	
10	2019	HYUNDAI	KONA	9	
11	2018	TESLA	MODEL S	165	
12	2010	TESLA	ROADSTER	5	
13	2018	CHEVROLET	BOLT EV	97	
14	2018	TESLA	MODEL X	151	
15	2017	CHEVROLET	BOLT EV	207	

Query Details

Query duration

317ms

Rows

282

Model Year

Aa

MAKE

Aa

BMW

34

FORD

23

TESLA

21

+ 27 more

MODEL

Aa

LEAF

10

MODEL S



Snowpark vs Databricks Spark

Snowpark and Databricks Spark are both powerful tools for data processing and analysis. However, there are several benefits to using Snowpark over Databricks Spark:

Performance: Snowpark is designed to optimize performance for Snowflake's data warehouse environment. This means that Snowpark jobs can run faster and more efficiently than Databricks Spark jobs when working with Snowflake.

Integration with Snowflake: Snowpark is tightly integrated with Snowflake, which makes it easy to write Spark code that can interact with Snowflake data warehouses. This means that you can use Snowpark to load data into Snowflake, extract data from Snowflake, and run complex analytical queries on Snowflake data.

Flexibility: Snowpark is a more flexible tool than Databricks Spark, as it can be used in a variety of environments, including on-premises data centers and public clouds. This means that you can use Snowpark to build data processing pipelines that meet your specific business needs, regardless of your organization's IT infrastructure.

Overall, Snowpark offers several benefits over Databricks Spark when working with Snowflake data warehouses. If your organization is already using Snowflake and needs a high-performance, flexible tool for data processing and analysis, Snowpark is definitely worth considering.



Thank you.

