Real-Time Scenario: Athena Slow Queries on Partitioned S3 Data

Problem Scenario:

At **CloudInsights Inc.**, a fintech company focused on compliance and fraud analytics, the data engineering team is responsible for maintaining the performance of analytical queries running on Amazon Athena. These queries process large-scale customer transaction data stored in Amazon S3.

Reported Issue:

The Analytics team reported that:

> *"Athena queries are getting slower and slower. The query plan is taking longer to compile, even before any data is scanned. We suspect it's due to too many partitions in S3."*

After investigation, the following was confirmed:

- The transaction dataset was partitioned by year/month/day/hour resulting in **millions of small partitions**.
- The data was stored in **CSV format** – not optimized for Athena.

Goals:

- ✅ Reduce query planning time.
- ✅ Improve query execution performance.
- ✅ Avoid repeated Glue crawler runs for updating partitions.

Step-by-Step Solution:

1 Convert CSV to Apache Parquet Format

CSV files are row-based and slow to scan. Parquet is a **columnar format** that enables:

- Faster scans
- Efficient compression
- Predicate pushdown (scanning only needed columns)

Glue ETL Job:

Action:

- Read data from the CSV S3 bucket (`s3://cloudinsights-raw-transactions/`)
- Write it in Parquet format partitioned by `year`, `month`, and `day`
- Store it at: `s3://cloudinsights/transactions_parquet/`

```
# AWS Glue PySpark Job Script
import sys
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job
from awsglue.utils import getResolvedOptions

args = getResolvedOptions(sys.argv, ['JOB_NAME'])
sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)
job.init(args['JOB_NAME'], args)

# Read CSV
df = spark.read.option("header", "true").csv("s3://cloudinsights-raw-
transactions/")

# Write as Parquet with partitions
df.write.mode("overwrite").partitionBy("year", "month", "day") \
    .parquet("s3://cloudinsights/transactions_parquet/")

job.commit()
```

2 Enable Partition Projection in Athena

Each time Athena queries the partitioned table, it fetches metadata for all partitions from Glue — which becomes **extremely slow** when millions of partitions exist.

Partition Projection allows Athena to **generate metadata on-the-fly** during query execution — no need to crawl and register partitions!

🔧 *Create Athena Table with Partition Projection:*

```
CREATE EXTERNAL TABLE transactions (
  transaction_id string,
  amount double,
  status string
)
PARTITIONED BY (year string, month string, day string)
STORED AS PARQUET
LOCATION 's3://cloudinsights/transactions_parquet/'
TBLPROPERTIES (
  'projection.enabled' = 'true',
```

```
  'projection.year.type' = 'integer',
  'projection.year.range' = '2021,2025',
  'projection.month.type' = 'integer',
  'projection.month.range' = '1,12',
  'projection.day.type' = 'integer',
  'projection.day.range' = '1,31',
  'storage.location.template' =
's3://cloudinsights/transactions_parquet/year=${year}/month=${month}/day=${da
y}/'
);
```

Now, Athena **automatically infers partitions** based on query input.

🧠 Why This Approach Works:

Alternative	Why It Was Rejected
AWS Glue Crawlers	Too slow and not scalable for millions of partitions
DML Partition Re-org	Requires constant maintenance and manual updates
Merge Small Files	Only helps with execution speed, not planning latency

🏆 Results (Reported by Analytics Team):

- ⚡ **Query planning time** reduced by over 60%
- 💸 Athena compute costs lowered due to faster scans
- 💬 *"Queries that took 10+ seconds to plan now start almost instantly!"*

📂 Workspace Output (Paste Your Results Below)

📝 Paste screenshots, query outputs, S3 structure, job logs, etc. here:

```
[S3 Screenshot with folder structure]
[Glue job run screenshot or job logs]
[Athena query outputs before & after optimization]
[Performance comparison table/chart]
```

📌 Summary

You successfully simulated a real-time data engineering challenge involving:

- AWS S3
- AWS Glue (ETL)

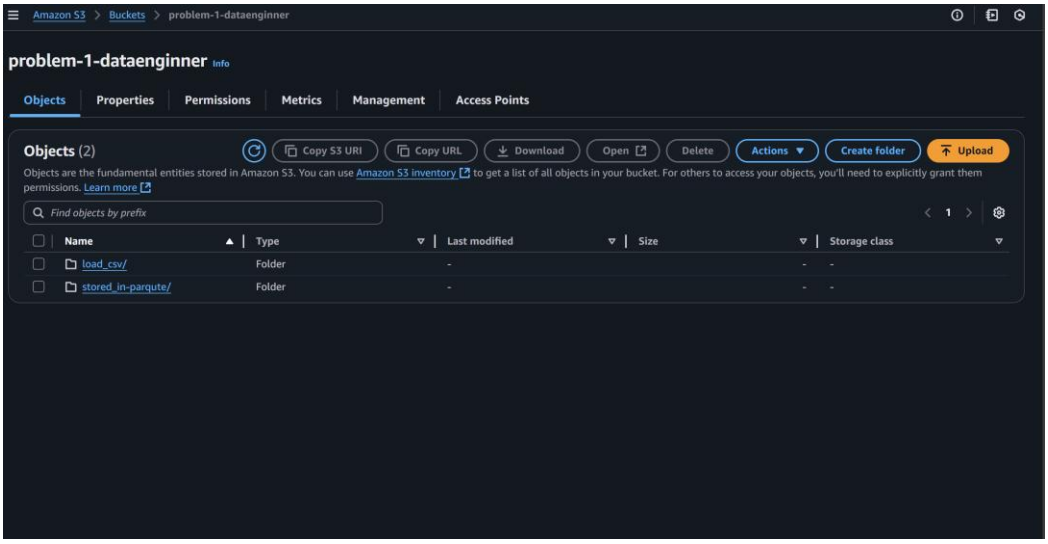
- Apache Parquet
- Athena Partition Projection

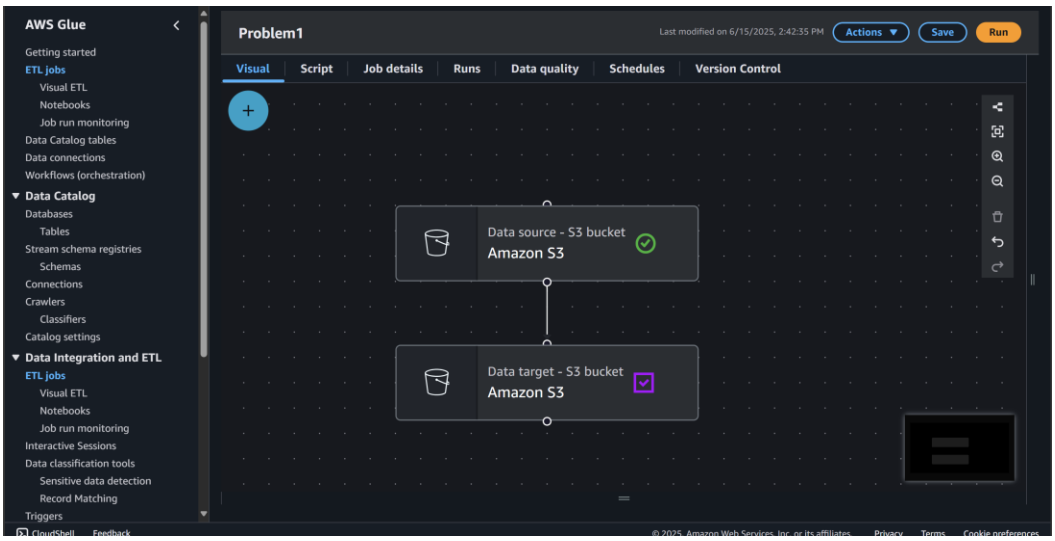
This is **production-level architecture** that scales. Add this to your GitHub and resume as a real case study!

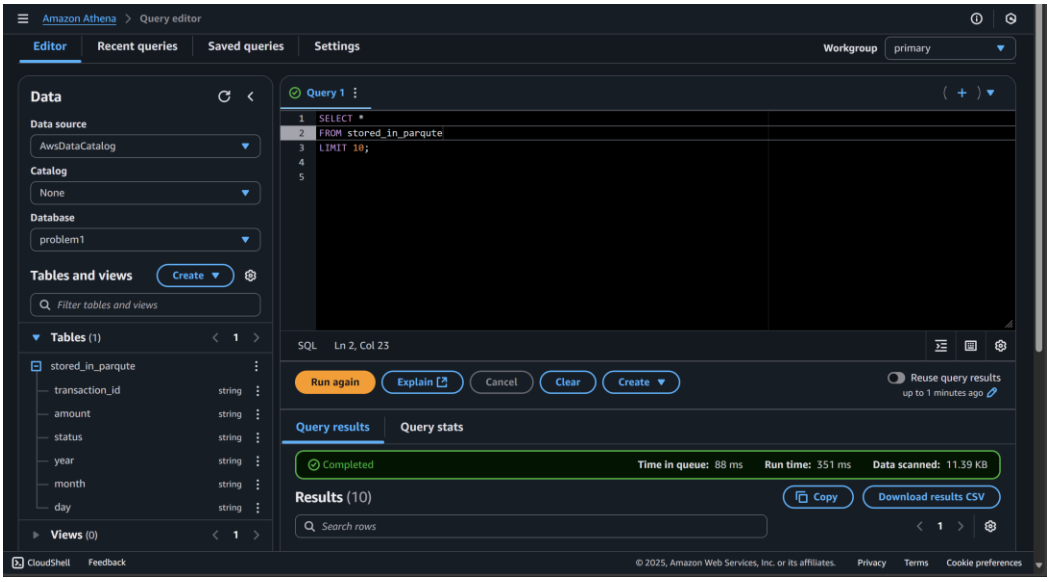
Tags

#AWS #AthenaOptimization #GlueETL #PartitionProjection #ParquetConversion #FintechAnalytics

📁 Project Output & Artifacts:






Amazon Athena
Query editor
Editor
Recent queries
Saved queries
Settings
Workgroup
primary
Data
Data source
AwsDataCatalog
Catalog
None
Database
problem1
Tables and views
Create
Filter tables and views
Tables (1)
stored_in_parqute
transaction_id string
amount string
status string
year string
month string
day string
Views (0)
Query 1
SELECT *
FROM stored_in_parqute
LIMIT 10;
SQL Ln 2, Col 23
Run again
Explain
Cancel
Clear
Create
Reuse query results
up to 1 minutes ago
Query results
Query stats
Completed
Time in queue: 88 ms
Run time: 351 ms
Data scanned: 11.39 KB
Results (10)
Copy
Download results CSV
Search rows
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences