Amazon Athena

Michael Lin
Senior Solutions Architect
Amazon Web Services

Agenda

- Introduction to Amazon Athena
- Athena Design Patterns
- Athena in Action
- Summary

1. Introduction to Amazon Athena

Amazon Athena

Amazon Athena is an interactive query service that makes it easy to analyze data using standard SQL. Athena is serverless, so there is no infrastructure to manage, and you pay only for the queries that you run.

- Query data in your Amazon S3 based data lake
- Analyze infrastructure, operation, and application logs
- Interactive analytics using popular BI tools
- Self-service data exploration for data scientists
- Embed analytics capabilities into your applications
- Perform ETL using SQL

aws

Demo

https://aws.amazon.com/blogs/big-data/analyzing-data-in-s3-using-amazon-athena/

Why Amazon Athena

- Decouple storage from compute
- Serverless – No infrastructure or resources to manage
- Pay only for data scanned
- Schema on read – Same data, many views
- Secure – IAM/SAMLv2 for authentication; Encryption at rest & in transit
- Standard compliant and open storage file formats
- Built on powerful community supported OSS solutions

Simple Pricing

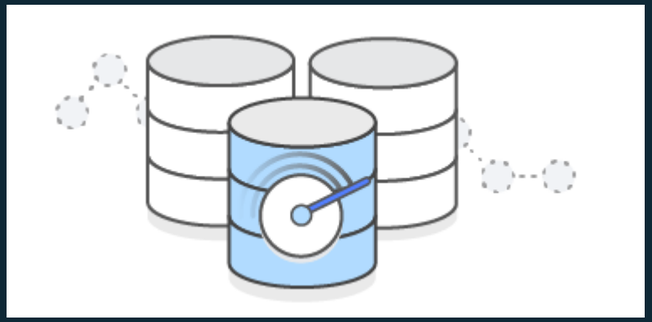
- DDL operations – FREE
- SQL operations – FREE
- Query concurrency – FREE
- Data scanned - $5 / TB
- Standard S3 rates for storage, requests, and data transfers apply

Run Standard SQL

- Uses Presto with ANSI SQL support
- Works with standard data formats
  - CSV
  - Apache Weblogs
  - JSON
  - Parquet
  - ORC
- Handles complex queries
  - Large Joins
  - Window functions
  - Arrays

Fast Performance for Large Data Sets

- Fast, ad-hoc queries
- Executes queries in parallel
- No provisioning extra resources for complex queries
- Scales automatically
- Amazon Athena Federated Query



2. Athena Design Patterns

When to use Athena vs EMR

Athena

- Ad-hoc querying
- Serverless - no setup or management of cluster
- Run queries using standard SQL
- Scales automatically based on complexity of queries

EMR

- Data processing/ETL
- Build and manage your own cluster
- Run custom applications and code
- Use big data processing frameworks (Spark, Hadoop, Presto, or Hbase)

New — Amazon Athena for Apache Spark

by Donnie Prakoso | on 30 NOV 2022 | in Amazon Athena, Analytics, Announcements, AWS Re:Invent, Launch, News | Permalink | Comments | Share

0:00 / 0:00

Voiced by Amazon Polly

When Jeff Barr first announced Amazon Athena in 2016, it changed my perspective on interacting with data. With Amazon Athena, I can interact with my data in just a few steps—starting from creating a table in Athena, loading data using connectors, and querying using the ANSI SQL standard.

Over time, various industries, such as financial services, healthcare, and retail, have needed to run more complex analyses for a variety of formats and sizes of data. To facilitate complex data analysis, organizations adopted Apache Spark. Apache Spark is a popular, open-source, distributed processing system designed to run fast analytics workloads for data of any size.

https://aws.amazon.com/blogs/aws/new-amazon-athena-for-apache-spark/

When to use Athena vs Redshift

Athena

- Ad-hoc querying
- Serverless - no setup or management of cluster
- Run queries using standard SQL
- Scales automatically based on complexity of queries

Redshift

- Data warehouse (historical analysis and reporting)
- Need to setup a cluster
- Run queries against highly structured data with many joins
- Can use same S3 data source as Athena

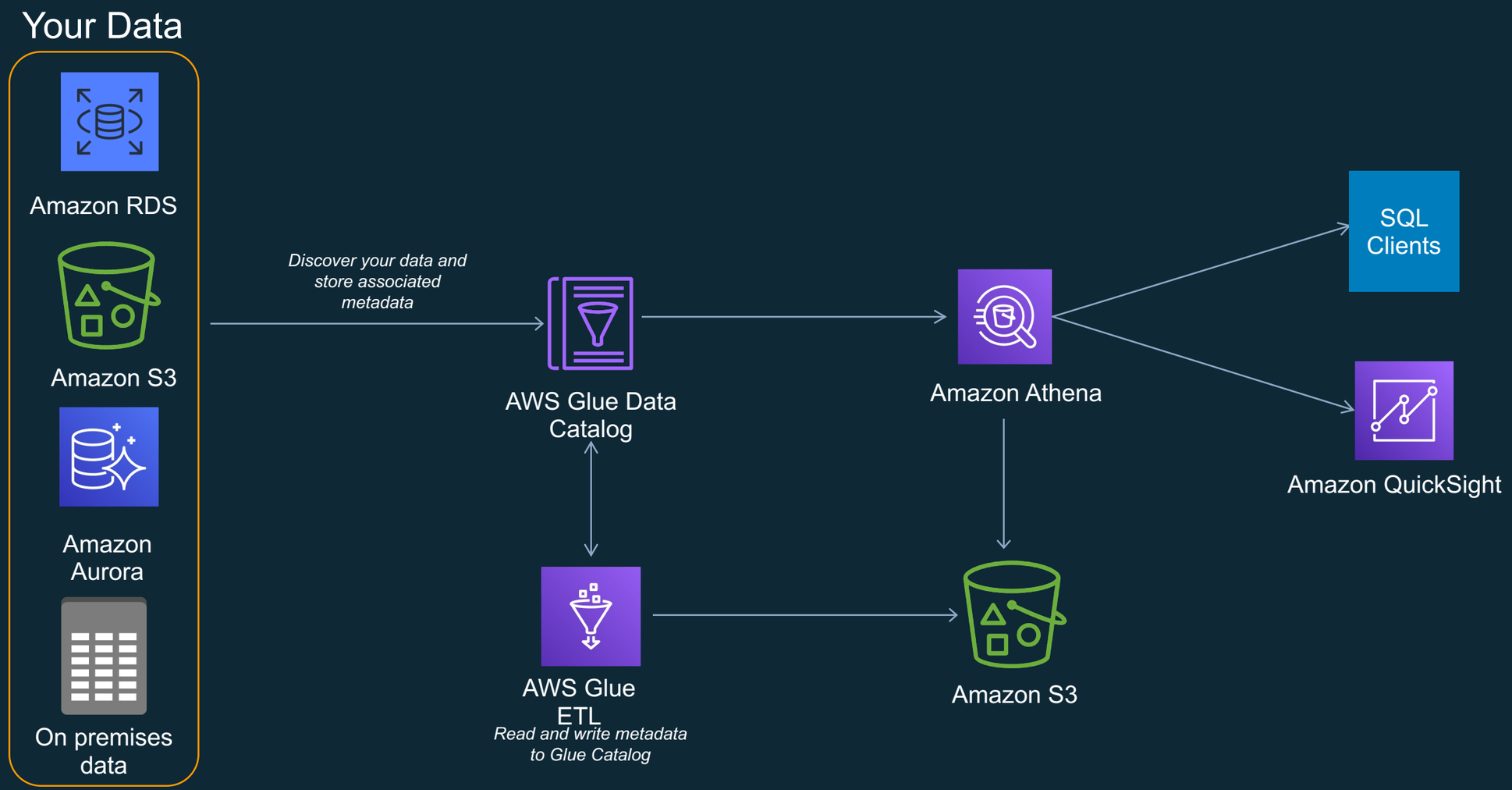
Amazon Redshift Spectrum Extends Data Warehousing Out to Exabytes—No Loading Required

by Maor Kleider | on 21 JUL 2017 | in Amazon Redshift, AWS Big Data | Permalink | Comments | Share

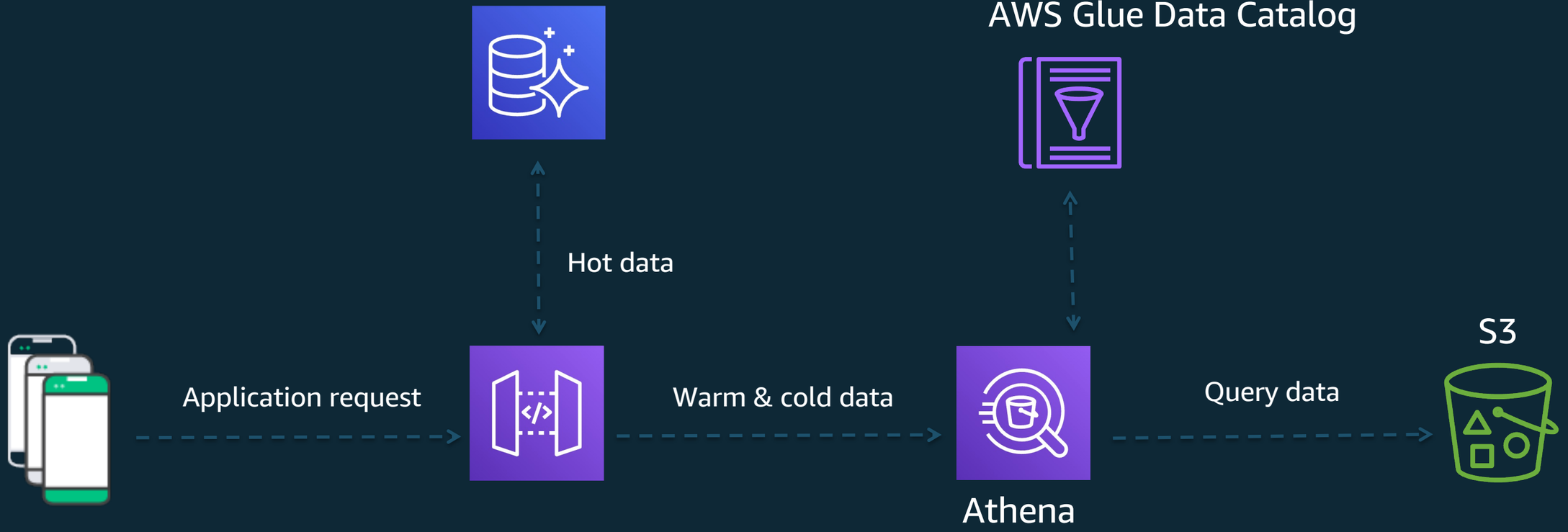
When we first looked into the possibility of building a cloud-based data warehouse many years ago, we were struck by the fact that our customers were storing ever-increasing amounts of data, and yet only a small fraction of that data ever made it into a data warehouse or Hadoop system for analysis. We saw that this wasn't just a cloud-specific anomaly. It was also true in the broader industry, where the growth rate of the enterprise storage market segment greatly surpassed that of the data warehousing market segment.

https://aws.amazon.com/blogs/big-data/amazon-redshift-spectrum-extends-data-warehousing-out-to-exabytes-no-loading-required/

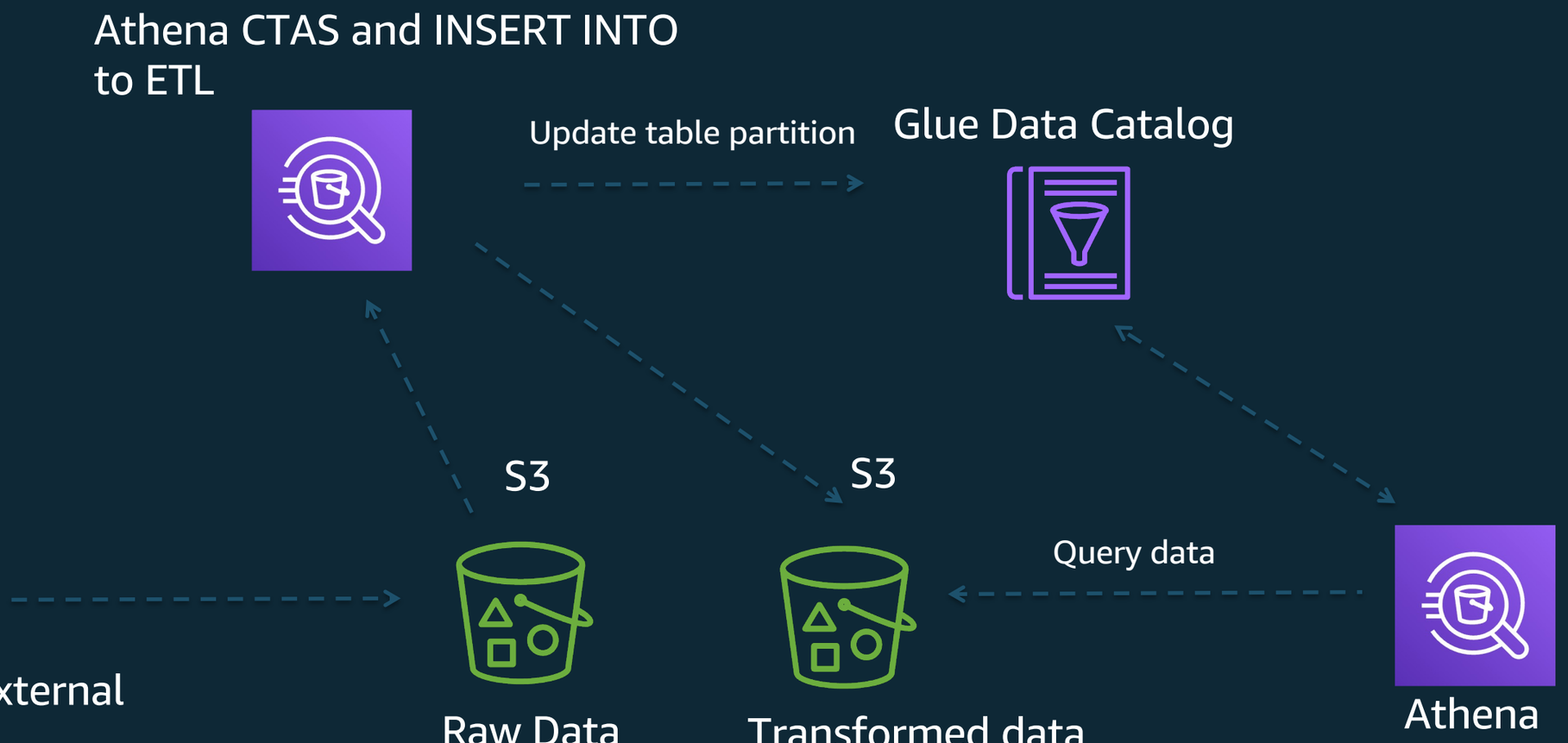
1: Data Exploration use-case



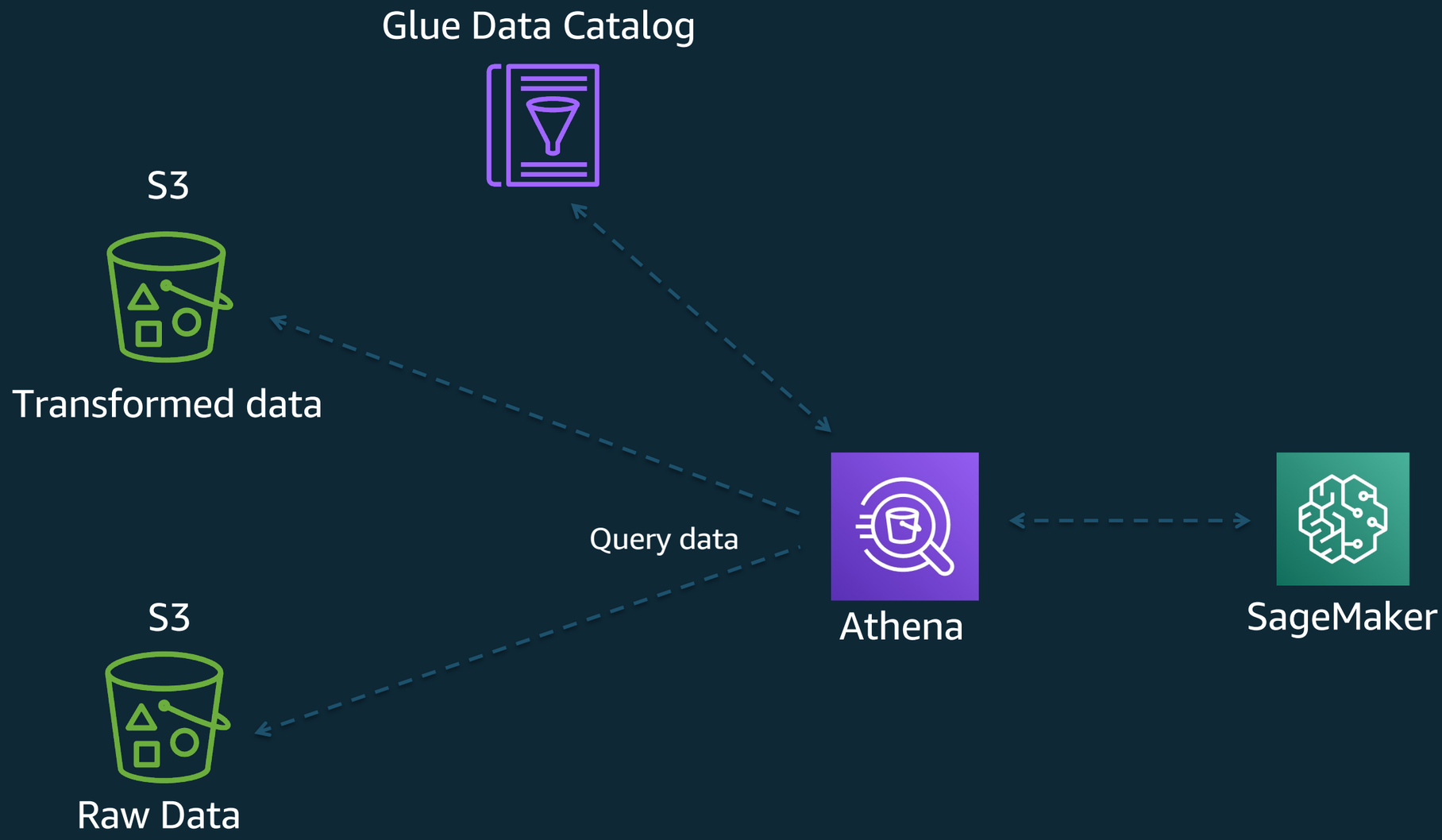
2: SaaS use-case



3: ETL and query use-case



4: Data science exploration and feature engineering



Querying AWS service logs

Topics

- Application Load Balancer
- Elastic Load Balancing
- CloudFront
- CloudTrail
- Amazon EMR
- Global Accelerator
- GuardDuty
- Network Firewall
- Network Load Balancer
- Route 53
- Amazon SES
- Amazon VPC
- AWS WAF

https://docs.aws.amazon.com/athena/latest/ug/querying-aws-service-logs.html

aws

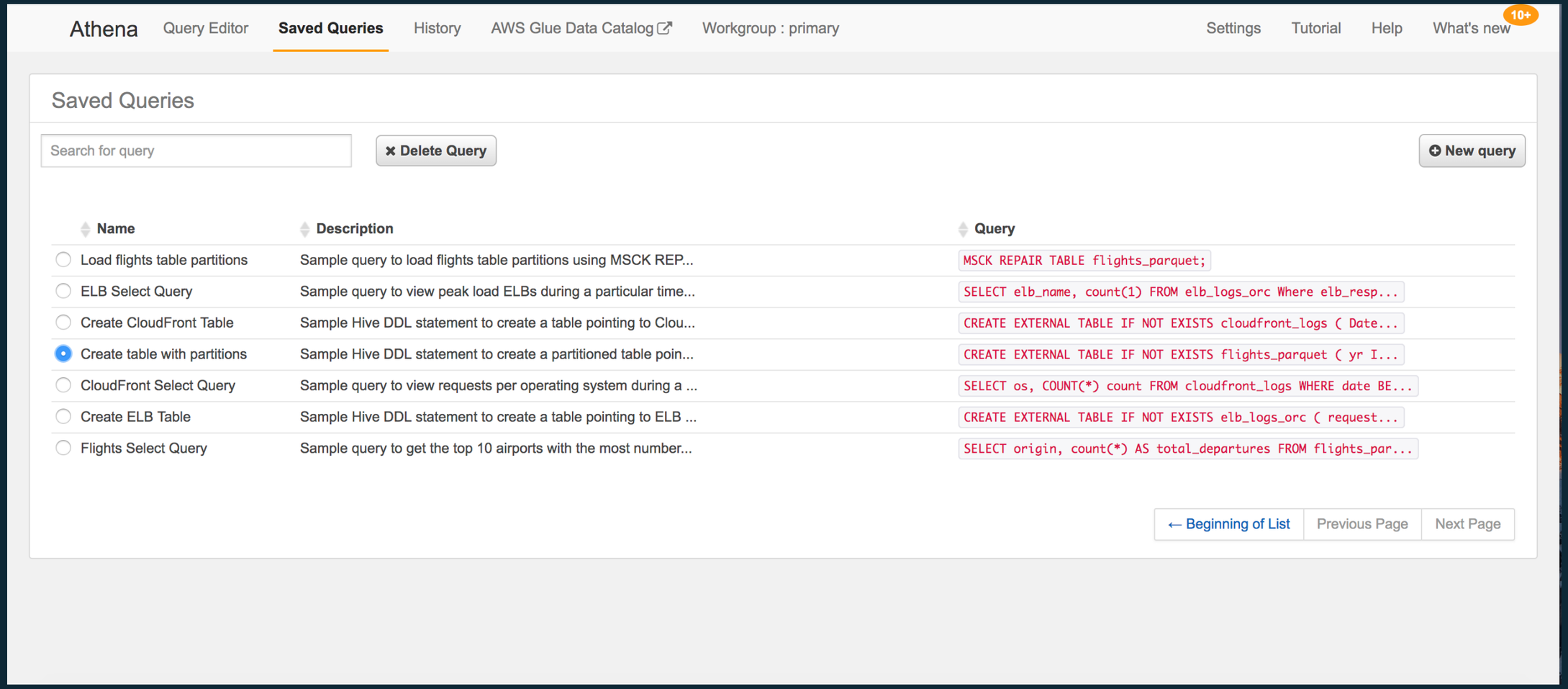
3. Athena in Action

Create External Tables

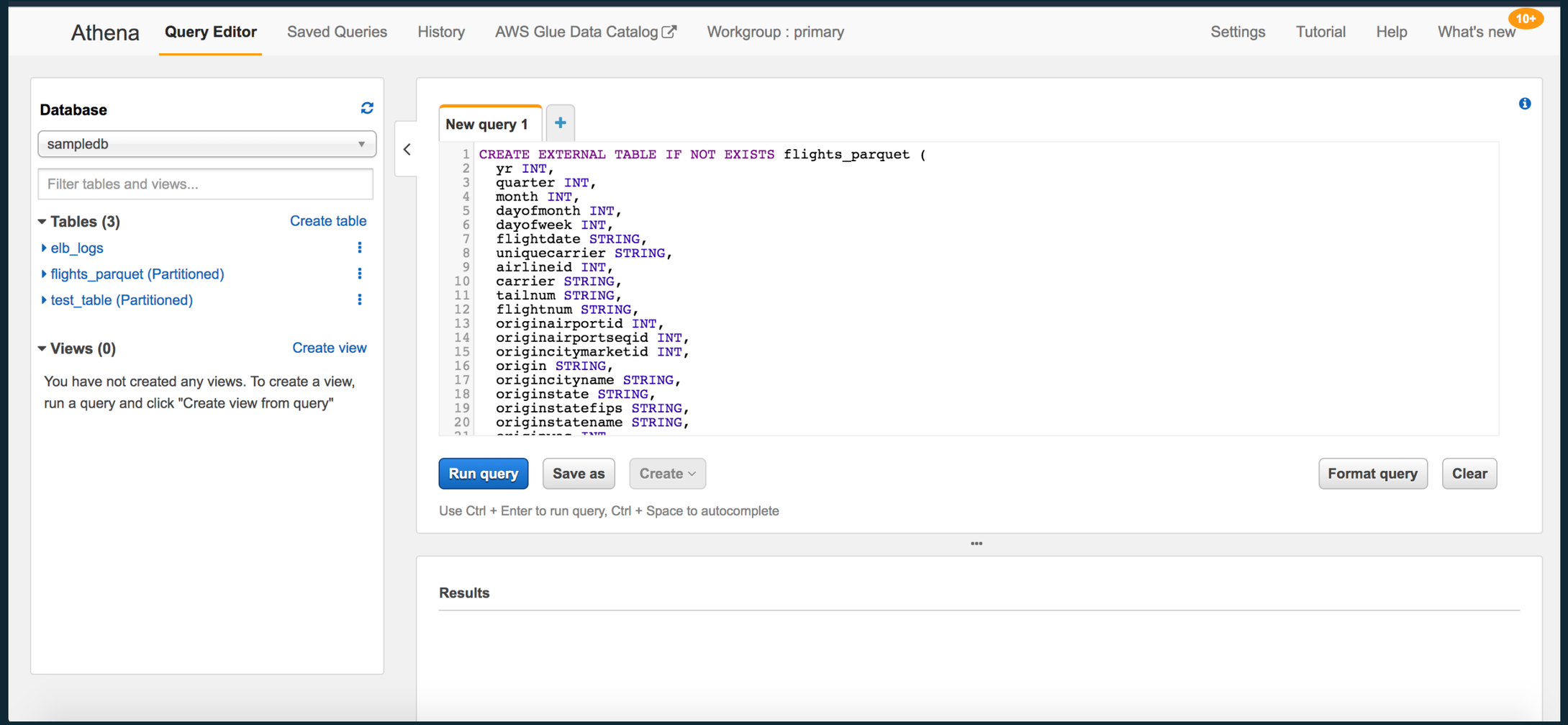
- Use Apache Hive DDL to create table
  - Run DDL statements using the Athena console
  - Via a JDBC driver, using SQL workbench
  - Using the Athena create table wizard
- Create “external” table in DDL
  - Creates a view of the data
  - Deleting table doesn’t delete the data in S3
- Schema-on-read
  - Projects your schema onto data at query execution time
  - No need for data loading or transformation

aws

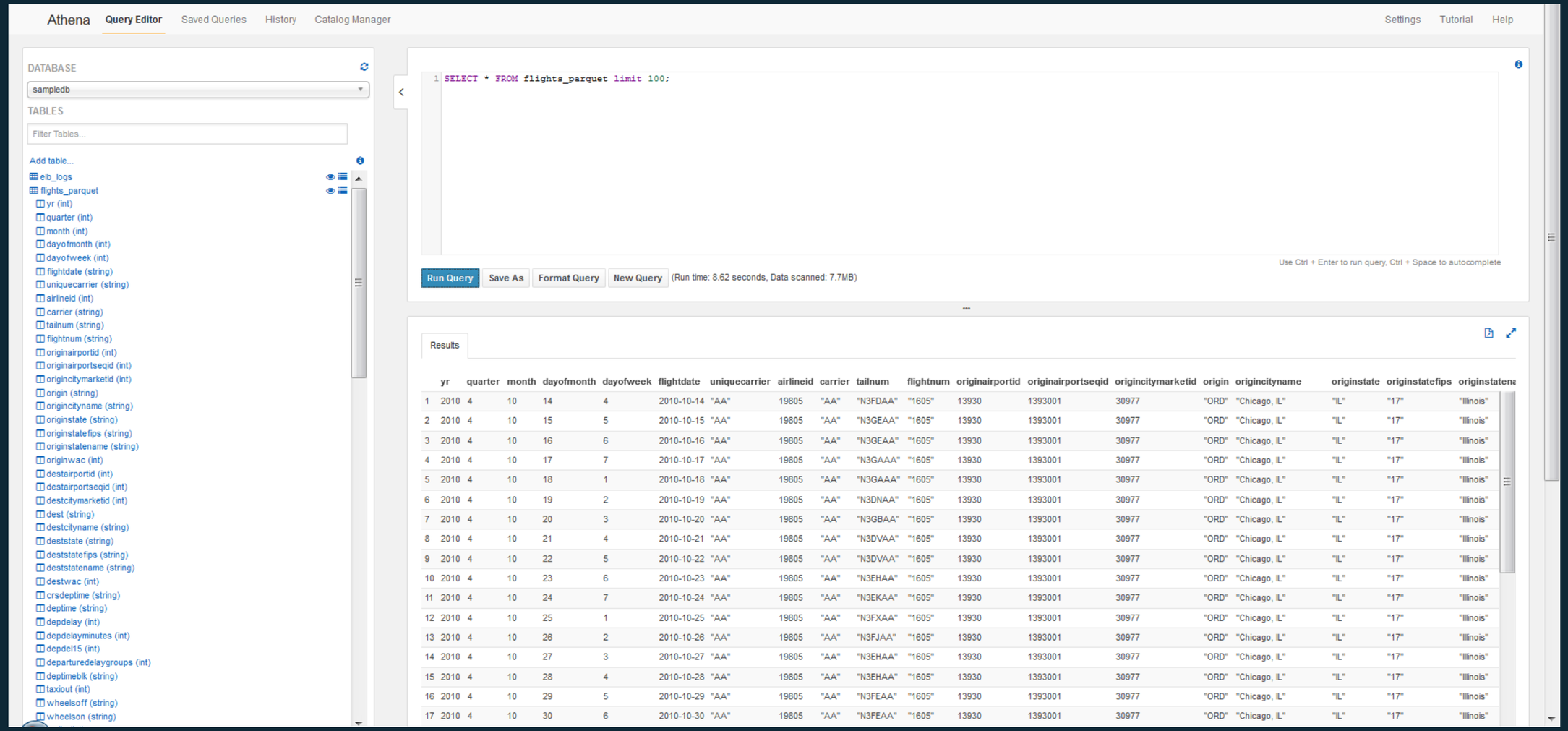
Navigate to ‘Saved Queries’ to get DDL



Create External Table using DDL



New Flights Parquet Table Created

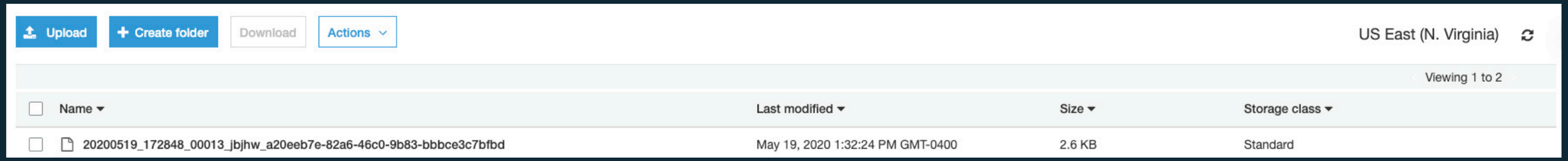


Save Query Results

- When you run an Athena query for the first time, an S3 bucket called "aws-athena-query-results-<account_id>" is created on your account, where "<account_id>" is replaced with your AWS account ID.

```
aws-athena-query-results-{account_id}-{region}/Unsaved/{year}/{month}/{day}/{query_id}.csv
```

- The results of all Athena queries are stored in this S3 bucket using the year, month, and day the query was run, the hexadecimal Athena query ID, and the region the query was run in the following format:



3.1 Tuning and Design Patterns

Data Formats and SerDes

- Current data formats and SerDes
  - Standard
    - CSV, TSV
    - JSON
    - Apache web logs
    - Custom-Delimited Files
  - Open-sourced columnar formats
    - Parquet
    - ORC

Convert to Columnar Formats-Parquet/ORC

- Convert to Parquet/ORC, use EMR
  - Create EMR cluster with Hive or Spark
  - In the step section of create cluster
    - Specify script in S3, Hive or PySpark
    - Point to the input data in text/CSV/JSON
    - Create output data in Parquet/ORC in S3

- Use Athena CTAS queries to create a new table with Parquet data from a source table in a different format.
  - CREATE TABLE new_table WITH (format = 'Parquet', parquet_compression = 'SNAPPY') AS SELECT * FROM old_table;

aws

Column Data Types

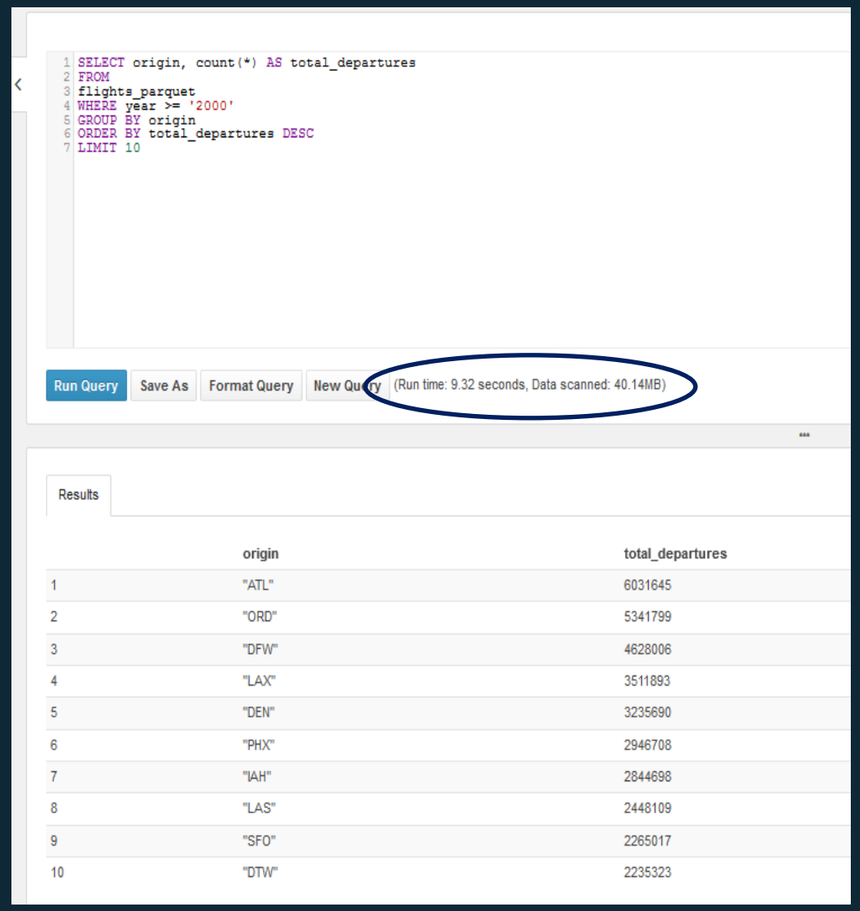
- Column Data Types
  - Primitives (Integer, Double, Date, String, Decimal, Varchar…)
  - ARRAY
  - MAP
  - STRUCT
- Nested JSON
  - Query using structures and array of structures

3.2 Partitions

Partitions

- Partition on any column in table
- Limits the amount of data each query scans
- Common practice
    - Multi-level time-based partition
- Example
    - PARTITIONED BY (year STRING, month STRING, day STRING)
    - Performance and cost savings on queries filtering by day, month and year

How does partitioning work?



```
SELECT origin, count(*) AS total_departures
FROM
flights_parquet
WHERE year >= '2000'
GROUP BY origin
ORDER BY total_departures DESC
LIMIT 10
```

Run Query | Save As | Format Query | New Query | (Run time: 9.32 seconds, Data scanned: 40.14MB)

Results

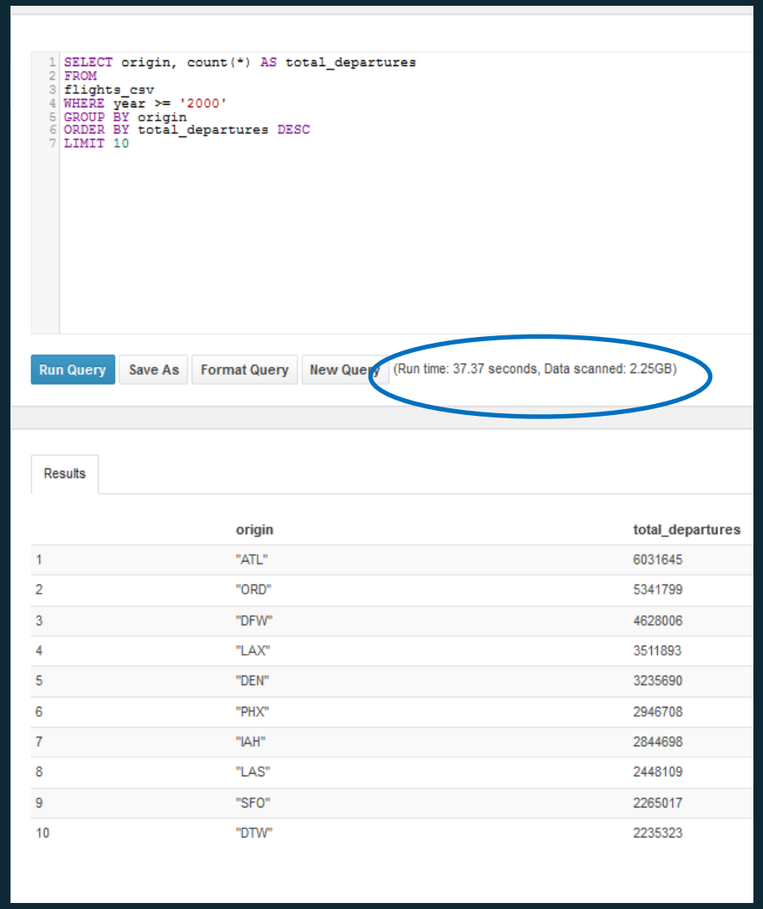
	origin	total_departures
1	"ATL"	6031645
2	"ORD"	5341799
3	"DFW"	4628006
4	"LAX"	3511893
5	"DEN"	3235690
6	"PHX"	2946708
7	"IAH"	2844698
8	"LAS"	2448109
9	"SFO"	2265017
10	"DTW"	2235323

- Flight dataset in Parquet
- Partitioned by year
- Group by query
  - Top 10 airports with the most departures since 2000
- Where year >= 2000
  - Athena doesn’t have to scan data in partitions before year 2000
- Amount of data scanned affects pricing
  - 1 min 6s, 3.11GB for csv unpartitioned
  - 37.37 s, 2.25GB for csv partitioned

aws

How does partitioning work?

With partitions:
CSV: 37.37 s, 2.25GB



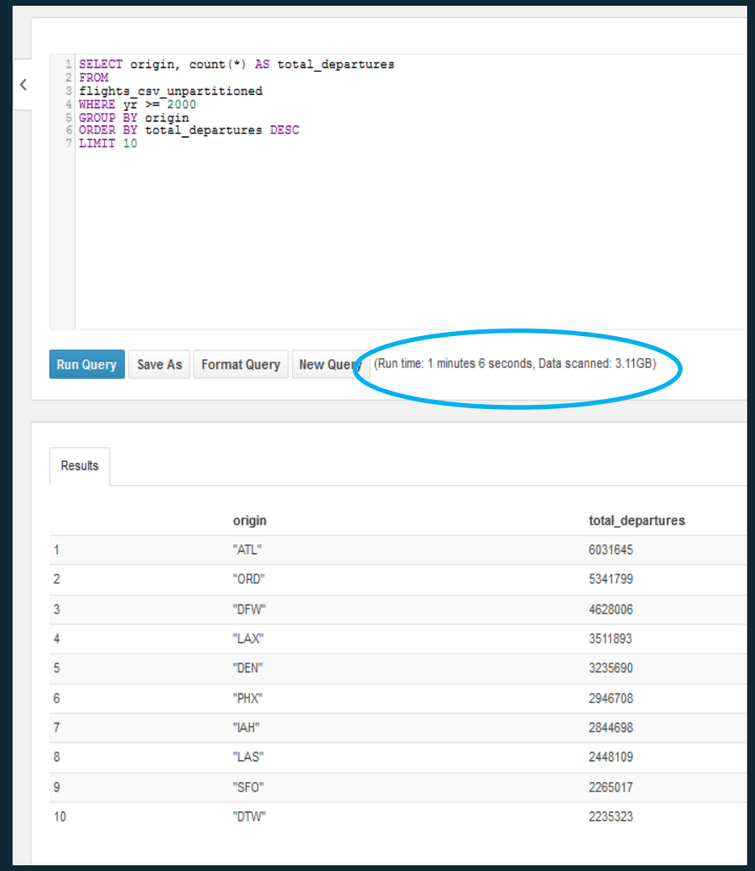
```
SELECT origin, count(*) AS total_departures
FROM
flights_csv
WHERE year >= '2000'
GROUP BY origin
ORDER BY total_departures DESC
LIMIT 10
```

Run Query | Save As | Format Query | New Query | (Run time: 37.37 seconds, Data scanned: 2.25GB)

Results

	origin	total_departures
1	"ATL"	6031645
2	"ORD"	5341799
3	"DFW"	4628006
4	"LAX"	3511893
5	"DEN"	3235690
6	"PHX"	2946708
7	"IAH"	2844698
8	"LAS"	2448109
9	"SFO"	2265017
10	"DTW"	2235323

No partitions:
1min 6s, 3.11GB



```
SELECT origin, count(*) AS total_departures
FROM
flights_csv_unpartitioned
WHERE yr >= 2000
GROUP BY origin
ORDER BY total_departures DESC
LIMIT 10
```

Run Query | Save As | Format Query | New Query | (Run time: 1 minutes 6 seconds, Data scanned: 3.11GB)

Results

	origin	total_departures
1	"ATL"	6031645
2	"ORD"	5341799
3	"DFW"	4628006
4	"LAX"	3511893
5	"DEN"	3235690
6	"PHX"	2946708
7	"IAH"	2844698
8	"LAS"	2448109
9	"SFO"	2265017
10	"DTW"	2235323

Partition by Running MSCK Repair

- If your data is already partitioned in Hive format
  - Use MSCK repair table command.
  - To sync all new data in S3 with Hive metastore
  - Example:
    - msck repair table flights_csv
    - show partitions flights_csv

Data stored on Amazon S3 in Hive format

```
aws s3 ls s3://elasticmapreduce/samples/hive-ads/tables/impressions/

    PRE dt=2009-04-12-13-00/
    PRE dt=2009-04-12-13-05/
    PRE dt=2009-04-12-13-10/
    PRE dt=2009-04-12-13-15/
    PRE dt=2009-04-12-13-20/
    PRE dt=2009-04-12-14-00/
    PRE dt=2009-04-12-14-05/
    PRE dt=2009-04-12-14-10/
    PRE dt=2009-04-12-14-15/
    PRE dt=2009-04-12-14-20/
    PRE dt=2009-04-12-15-00/
    PRE dt=2009-04-12-15-05/
```

https://docs.aws.amazon.com/athena/latest/ug/partitions.html

Partition by Manually Adding Partitions

- If data is not partitioned in Hive format
  - Cannot use MSCK repair table command
  - Use ALTER TABLE ADD PARTITION to add each partition manually
  - Look at automating adding partitions
  - Example:
    - ALTER TABLE elb_logs_raw_native_part ADD PARTITION (year='2015',month='01',day='01’) location 's3://athena-examples/elb/raw/2015/01/01/'

aws

3.3 Optimize and Secure

Access Control for Athena

- Access Control
  - AWS Identity and Access Management (IAM)
  - Access Control Lists (ACLs)
  - Amazon S3 bucket policies.
- IAM policies for S3
  - IAM is natively integrated with S3
  - Grant IAM users fine-grained control to S3
  - Restrict users from querying it using Athena

Access Control Continued

- To run queries in Athena, you must have the appropriate permissions for the following:
    - Athena API actions including additional actions for Athena workgroups
    - Amazon S3 locations where the underlying data to query is stored.
    - Metadata and resources that you store in the AWS Glue Data Catalog, such as databases and tables, including additional actions for encrypted metadata.

aws

Best Practices

- Partition your data
    - Divides your table into parts and keeps the related data together based on column values such as date, country, region, etc
- Bucket your data
    - You can specify one or more columns containing rows that you want to group together and put those rows into multiple buckets.
- Compress and split files
    - We recommend using either Apache Parquet or Apache ORC, which compress data by default and are splitable
- Optimize File Sizes
    - Ideally not less than 128 MB

aws

Best Practices Continued

- Optimize columnar data store generation
    - Apache Parquet and Apache ORC
- Optimize Queries
    - ORDER BY (with limit)
    - JOIN (with larger table on the left)
    - GROUP BY (with higher to lower cardinality)
    - LIKE (better substituted using regex)
- Only include the columns that you need

aws

Top 10 Performance Tuning Tips for Amazon Athena

by Mert Hocanin and Pathik Shah | on 24 MAR 2017 | in Amazon Athena, AWS Big Data | Permalink | Comments | Share

May 2022: This post was reviewed and updated with more details like using EXPLAIN ANALYZE, updated compression, ORDER BY and JOIN tips, using partition indexing, updated stats (with performance improvements), added bonus tips.

Amazon Athena is an interactive query service that makes it easy to analyze data stored in Amazon Simple Storage Service (Amazon S3) using standard SQL. Athena is serverless, so there is no infrastructure to manage, and you pay only for the queries that you run. Athena is easy to use. Simply point to your data in Amazon S3, define the schema, and start querying using standard SQL.

https://aws.amazon.com/blogs/big-data/top-10-performance-tuning-tips-for-amazon-athena/

3.4 Workgroups

Athena Workgroups

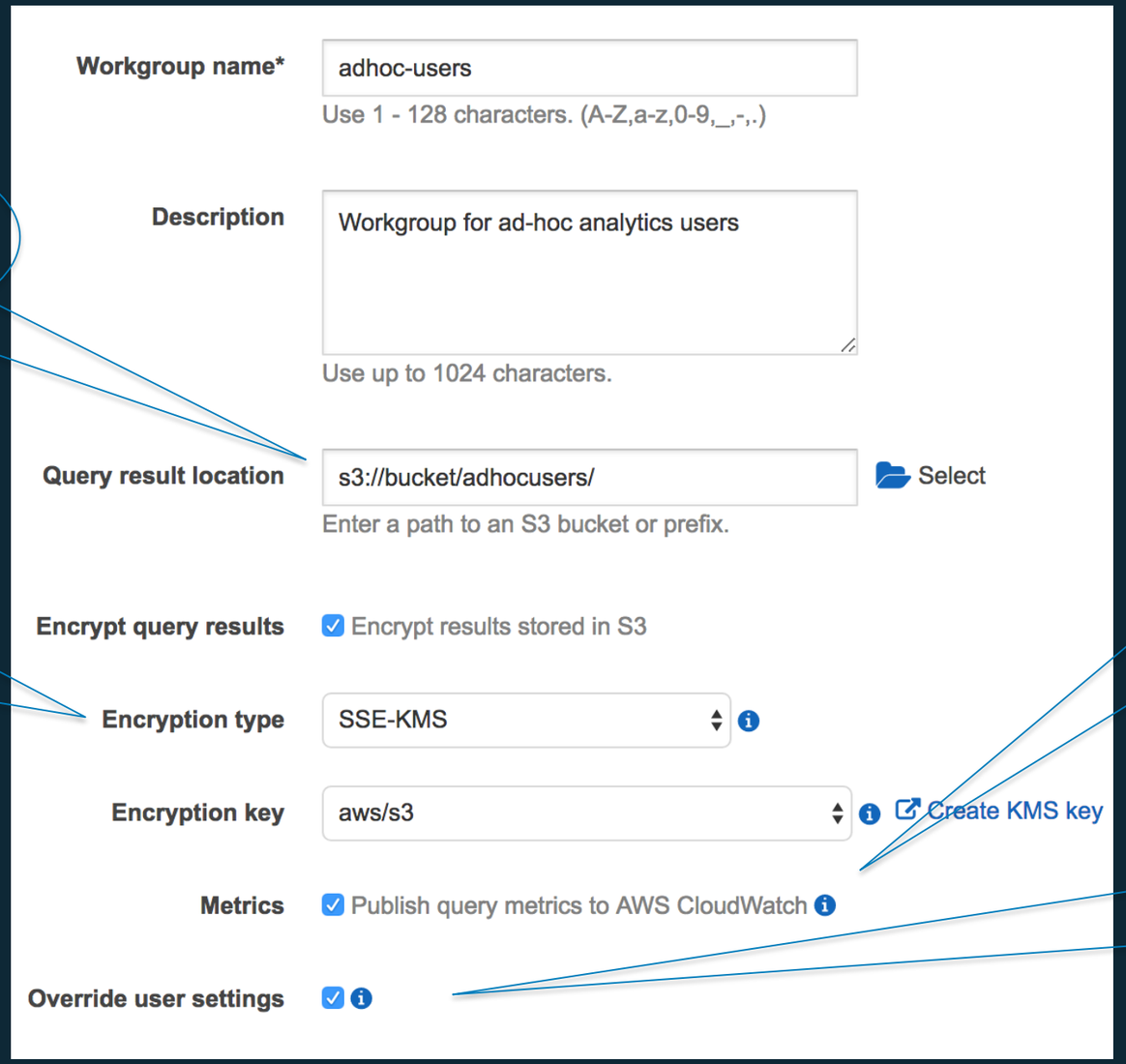
Athena Workgroups are used to isolate queries between different teams, workloads or applications, and to set limits on amount of data each query or the entire workgroup can process

Workload Isolation

Query Metrics

Cost Controls

Workgroups – Workload Isolation



Unique query output location per Workgroup

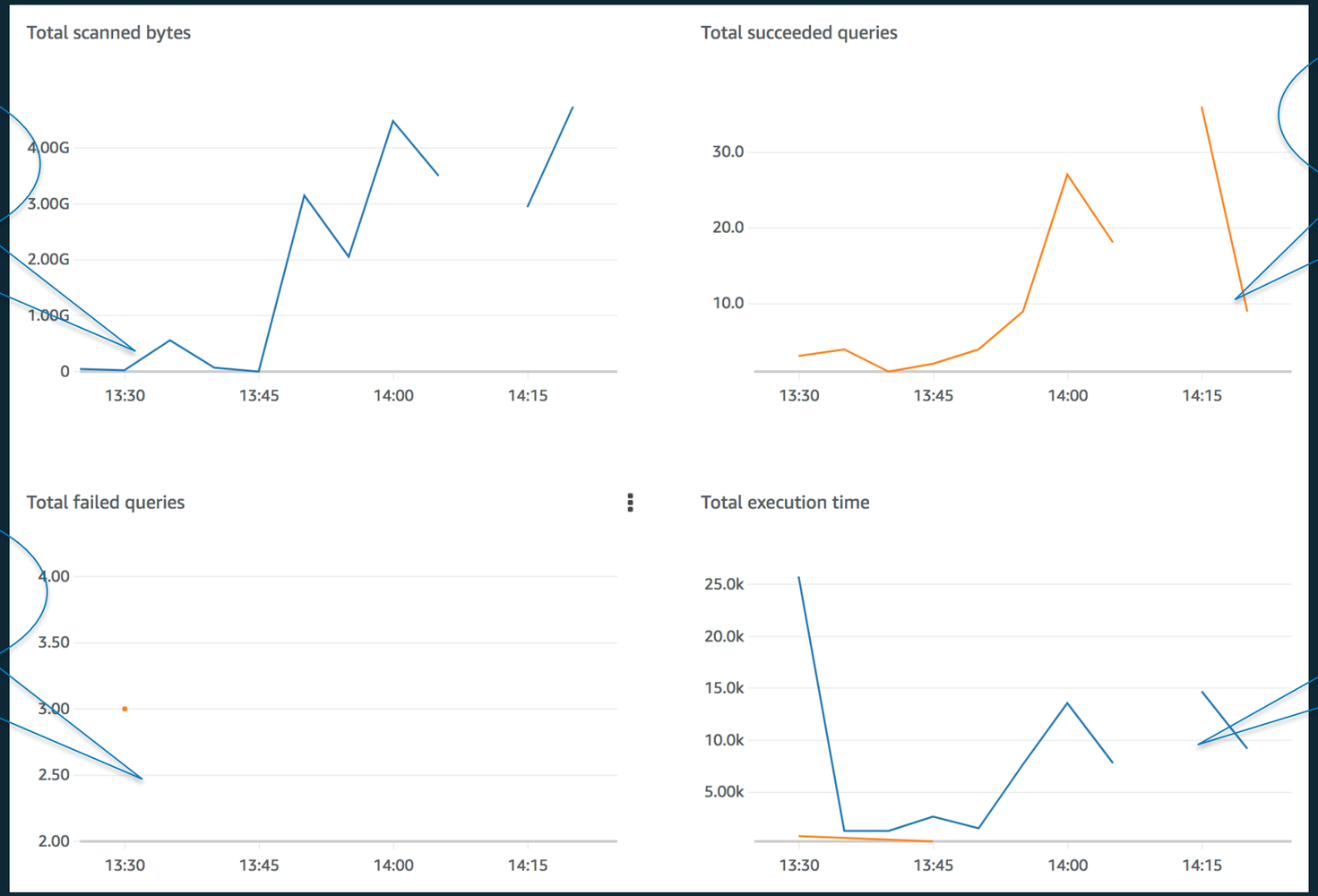
Encrypt results with unique AWS KMS key per Workgroup

Collect and publish aggregated metrics per Workgroup to AWS CloudWatch

Use Workgroup settings eliminating need to configure individual users

Workgroups – Metric Reporting

Total bytes scanned per Workgroup



Total successful queries per Workgroup

Total failed queries per Workgroup

Total query execution time per Workgroup

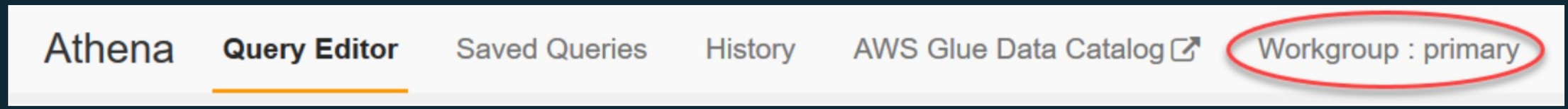
Workgroups – Cost Controls

- Per query data scanned threshold; exceeding, will cancel query
- Trigger alarms to notify of increasing usage and cost
- Disable Workgroup when all queries exceed a maximum threshold

Any Athena metric: successful/failed & total queries, query run time, etc.

Data limit	Time period	Action
10 Gigabytes	Not applicable	Query will be cancelled.
1 Terabytes	24 hours	Send notification to topic : arn:aws:sns:us-east-1:9 9:AthenaAlarm
10 Gigabytes	1 hour	Send notification to topic : arn:aws:sns:us-east-1:9 9:AthenaAlarm

Default Workgroup



By default, if you have not created any workgroups, all queries in your account run in the primary workgroup

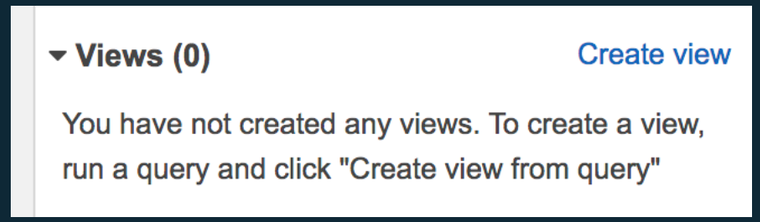
3.5 Views

View

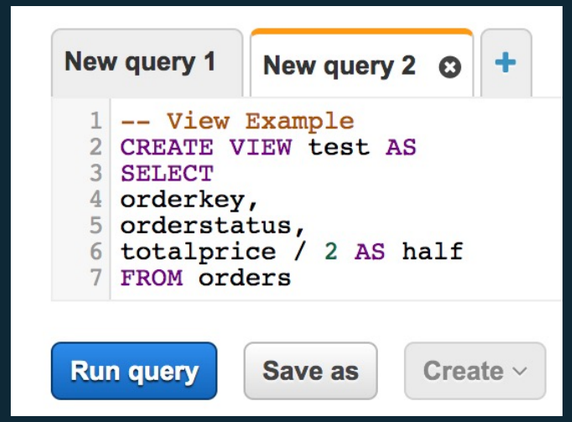
- A view in Amazon Athena is a logical, not a physical table.
- The query that defines a view runs each time the view is referenced in a query.
- Views do not contain any data and do not write data.
    - Instead, the query specified by the view runs each time you reference the view by another query.

Create a View

- Creates a new view from a specified SELECT query.
    - The view is a logical table that can be referenced by future queries.
- Flow:
    - Create View:



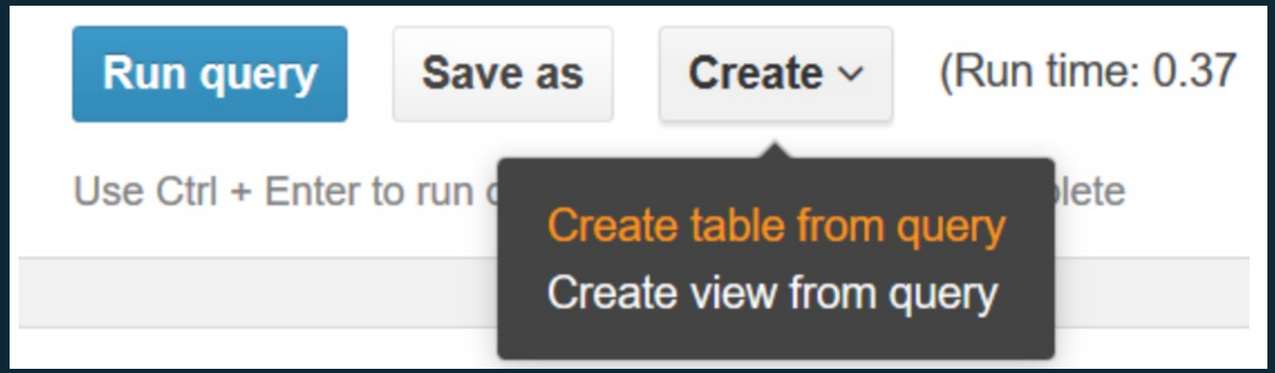
    - Write View:



3.6 CTAS

CREATE TABLE AS SELECT (CTAS)

- Creates a new table in Athena from the results of a SELECT statement from another query
- Athena stores data files created by the CTAS statement in a specified location in Amazon S3



Use CTAS queries to:

- Create tables from query results in one step, without repeatedly querying raw data sets. This makes it easier to work with raw data sets.

- Transform query results into other storage formats, such as Parquet and ORC. This improves query performance and reduces query costs in Athena.

- Create copies of existing tables that contain only the data you need.

aws

Extract, Transform and Load data into S3 data lake using CTAS and INSERT INTO statements in Amazon Athena

by Pathik Shah | on 26 NOV 2019 | in Amazon Athena, Analytics, AWS Big Data | Permalink | Comments | Share

Amazon Athena is an interactive query service that makes it easy to analyze the data stored in Amazon S3 using standard SQL. Athena is serverless, so there is no infrastructure to manage, and you pay only for the queries that you run. You can reduce your per-query costs and get better performance by compressing, partitioning, and converting your data into columnar formats. To learn more about best practices to boost query performance and reduce costs, see *Top 10 Performance Tuning Tips for Amazon Athena*.

Overview

This blog post discusses how to use Athena for extract, transform and load (ETL) jobs for data processing. This example optimizes the dataset for analytics by partitioning it and converting it to a columnar data format using Create Table as Select (CTAS) and INSERT INTO statements.

CTAS statements create new tables using standard SELECT queries to filter data as required. You can also partition the data, specify compression, and convert the data into columnar formats like Apache Parquet and Apache ORC using CTAS statements. As part of the execution, the resultant tables and partitions are added to the AWS Glue Data Catalog, making them immediately available for subsequent queries.

INSERT INTO statements insert new rows into a destination table based on a SELECT query statement that runs on a source table. If the source table's underlying data is in CSV format and destination table's data is in Parquet format, then INSERT INTO can easily transform and load data into destination table's format. CTAS and INSERT INTO statements can be used together to perform an initial batch conversion of data as well as incremental updates to the existing table.

https://aws.amazon.com/blogs/big-data/extract-transform-and-load-data-into-s3-data-lake-using-ctas-and-insert-into-statements-in-amazon-athena/

aws

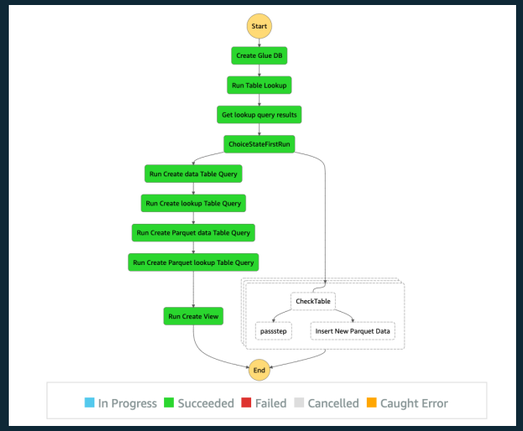
Build and orchestrate ETL pipelines using Amazon Athena and AWS Step Functions

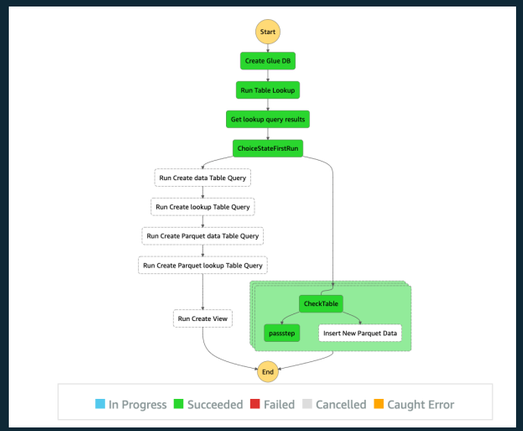
by Behram Irani, Dipankar Kushari, and Rahul Sonawane | on 12 OCT 2021 | in Amazon Athena, AWS Step Functions | Permalink | Comments | Share

Extract, transform, and load (ETL) is the process of reading source data, applying transformation rules to this data, and loading it into the target structures. ETL is performed for various reasons. Sometimes ETL helps align source data to target data structures, whereas other times ETL is done to derive business value by cleansing, standardizing, combining, aggregating, and enriching datasets. You can perform ETL in multiple ways; the most popular choices being:

- Programmatic ETL using Apache Spark. Amazon EMR and AWS Glue both support this model.
- SQL ETL using Apache Hive or PrestoDB/Trino. Amazon EMR supports both these tools.
- Third-party ETL products.

Many organizations prefer the SQL ETL option because they already have developers who understand and write SQL queries. However, these developers want to focus on writing queries and not worry about setting up and managing the underlying infrastructure.





https://aws.amazon.com/blogs/big-data/build-and-orchestrate-etl-pipelines-using-amazon-athena-and-aws-step-functions/

3.7 Monitoring & Auditing

Workgroups

- Separate users, teams, applications, or workloads
- Enable queries on Requester Pays buckets in Amazon S3
- In order to:
  - Set limits on amount of data each query or the entire workgroup can process
  - Track costs
    - Decide which workgroups to create.
    - Create workgroups as needed and add tags to them.
    - Create IAM policies for your users, groups, or roles to enable their access to workgroups.
    - Specify a location in Amazon S3 for query results and encryption settings.

Monitoring Athena Queries with CloudWatch Metrics

- Use CloudWatch Events with Athena:
  - EngineExecutionTime – in milliseconds
  - ProcessedBytes – the total amount of data scanned per DML query
  - QueryPlanningTime – in milliseconds
  - QueryQueueTime – in milliseconds
  - ServiceProcessingTime – in milliseconds
  - TotalExecutionTime – in milliseconds, for DDL and DML queries

aws

Monitoring Athena Queries with CloudWatch Events

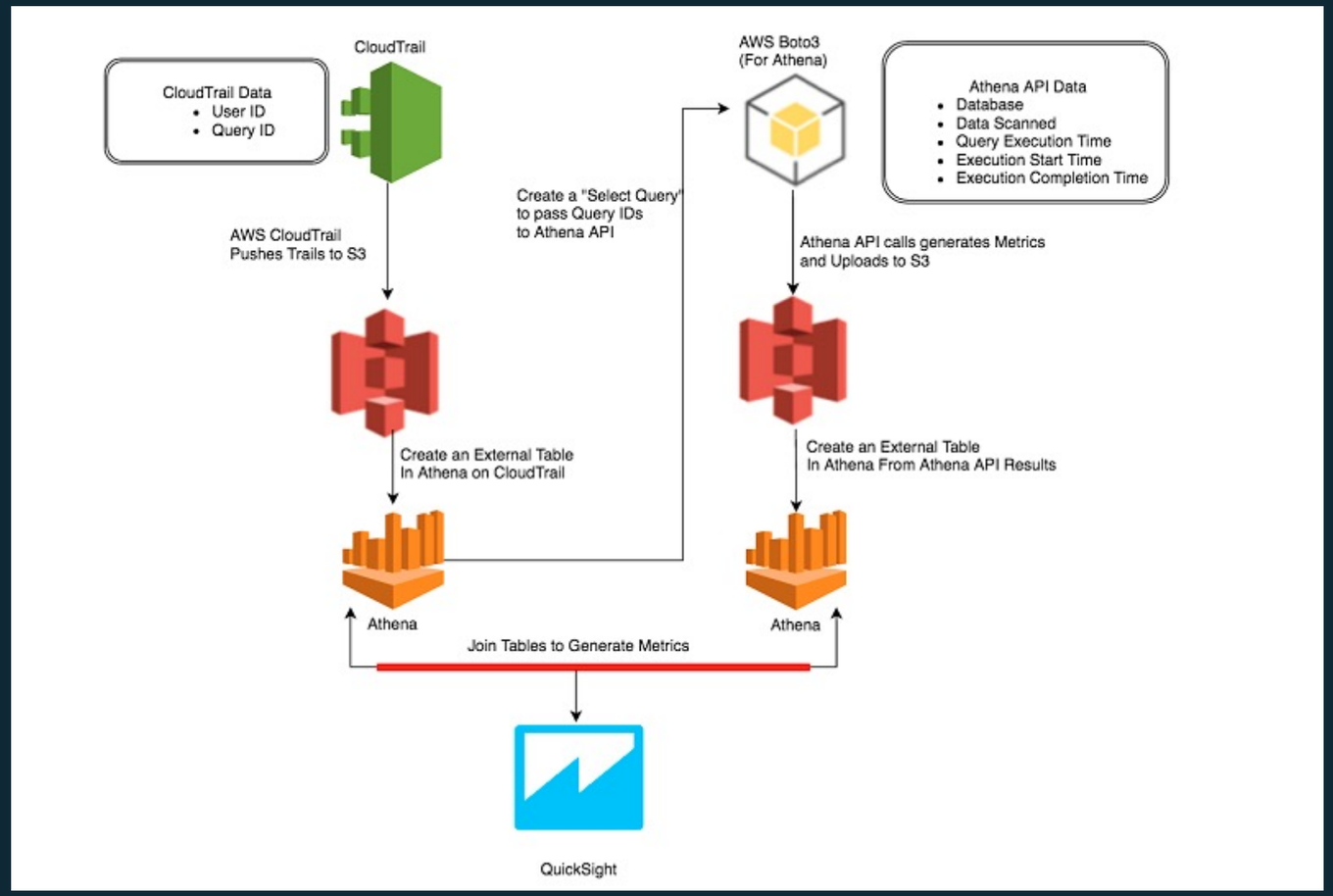
- Use simple rules to match events and route them to one or more target functions or streams.

- Respond to operational changes and takes corrective action as necessary, by
  - Sending messages to respond to the environment
  - Activating functions
  - Making changes
  - Capturing state information

```
{
    "source":[
        "aws.athena"
    ],
    "detail-type":[
        "Athena Query State Change"
    ],
    "detail":{
        "currentState":[
            "SUCCEEDED"
        ]
    }
}
```

CloudTrail

Monitor Athena with AWS CloudTrail



4. Summary

Key Benefits of Athena

- Fast ad-hoc queries
- Query directly against data in S3
- Use standard ANSI SQL
- No infrastructure to setup and manage
- Use IAM for security
- JDBC driver for BI tools and SQL clients
- Pay per query, depending on data scanned

Thank you!

Michael Lin

linmicht@amazon.com