

GROUP 3: STORE AND ORGANIZE DATA

Wyatt Priddy

Suraj Kunthu

Mohammad Atif Siddiqui

DS 5110 | Spring 2024



SCHOOL *of* DATA SCIENCE

Introduction

Use Case #2: Store and Organize Data

- Data Engineering focused project
 - Setting up ETL pipeline for data scientists
 - Compare query processes for optimal performance when working with Big Data sets



Importance of Storage And Organization

Improved Performance:

- **Faster query execution:** Organized data with proper partitioning and indexing allows data processing tools to locate relevant information quickly, leading to faster query execution times.
- **Reduced processing overhead:** Compressed data files require less storage space, which translates to less data to transfer and process during data pipelines. This reduces the overall processing overhead and improves efficiency.

Reduced Storage Costs:

- **Efficient storage utilization:** Organized data often eliminates redundancy and allows for optimized storage formats, significantly reducing the amount of physical storage needed.
- **Compression benefits:** Compressed data files occupy less space, leading to lower storage costs on AWS services like S3. This is especially beneficial for large datasets.

Enhanced Scalability:

- **Easier data management:** Organized data with clear structure facilitates adding new data or scaling existing datasets efficiently. This makes it easier to handle growing data volumes without performance degradation.
- **Cost-effective scaling:** Lower storage requirements from compression allow for scaling your data processing pipelines with minimal impact on storage costs.

Improved Data Quality:

- **Reduced errors:** Organized data structures help mitigate errors during data ingestion and processing due to inconsistencies or ambiguity.
 - **Data validation:** Organizing data allows for easier implementation of data validation checks to ensure the integrity and accuracy of your data before processing.
-

The Data

- Court records data available from across Virginia
- Formed from existing real world court data:
 - Hope to predict outcome of misdemeanor and felony court cases
 - General district court
 - Utilizing case type and demographics surrounding the defendant
- subsection of the data 2010 – 2020
- ~93 different files Merged together used for feature Engineering .

The Data

- 7,196,322 data points
- 11 variables
- CSV format

Variable	Description	Data Type
Final Disposition	The outcome of the court case.	str
Court	The court system within Virginia where the case was heard.	str
Complanaint	The arrest officers in the case.	str
Public Defender	Whether a public defender was assigned to the case	bool
Gender	Gender of the defendant	str
Race	Race of the defendant	str
Case Type	Indication of whether the case was a mideameanor or felony	str
Class	The level of misdemeanor or felony committed	str
CodeSection	A reference to the law that was violated — can be state or local law	str
ChargeAmended	Whether the original charge was amended	bool
SentenceTime	The duration of a jail or prison sentence from the outcome of the case	int

Data Preprocessing

Feature Engineering Highlights:

Public Defender Variable:

Manipulation of free-form text entry of the *Defense Attorney* field in the raw data to discern whether a defendant was assigned a public defender.

Example Entries: "*Public Defender*", "*Public Def*", "*P Def*", "*Pub Def*", "*PD*"

Court Variable:

Distinction by court was given in the Federal Information Processing Standard (FIPS) format. [Supplemental data](#) from the Federal Communication Commission was used to map FIPS code to court name

Example Conversion: "*51540*" -> "*Charlottesville City*"

Data Preprocessing

Data Cleaning Highlights:

Final Disposition:

The response variable for a potential classification model had ~89k records with null values representing 1.2% of the dataset. These values were removed from the dataset.

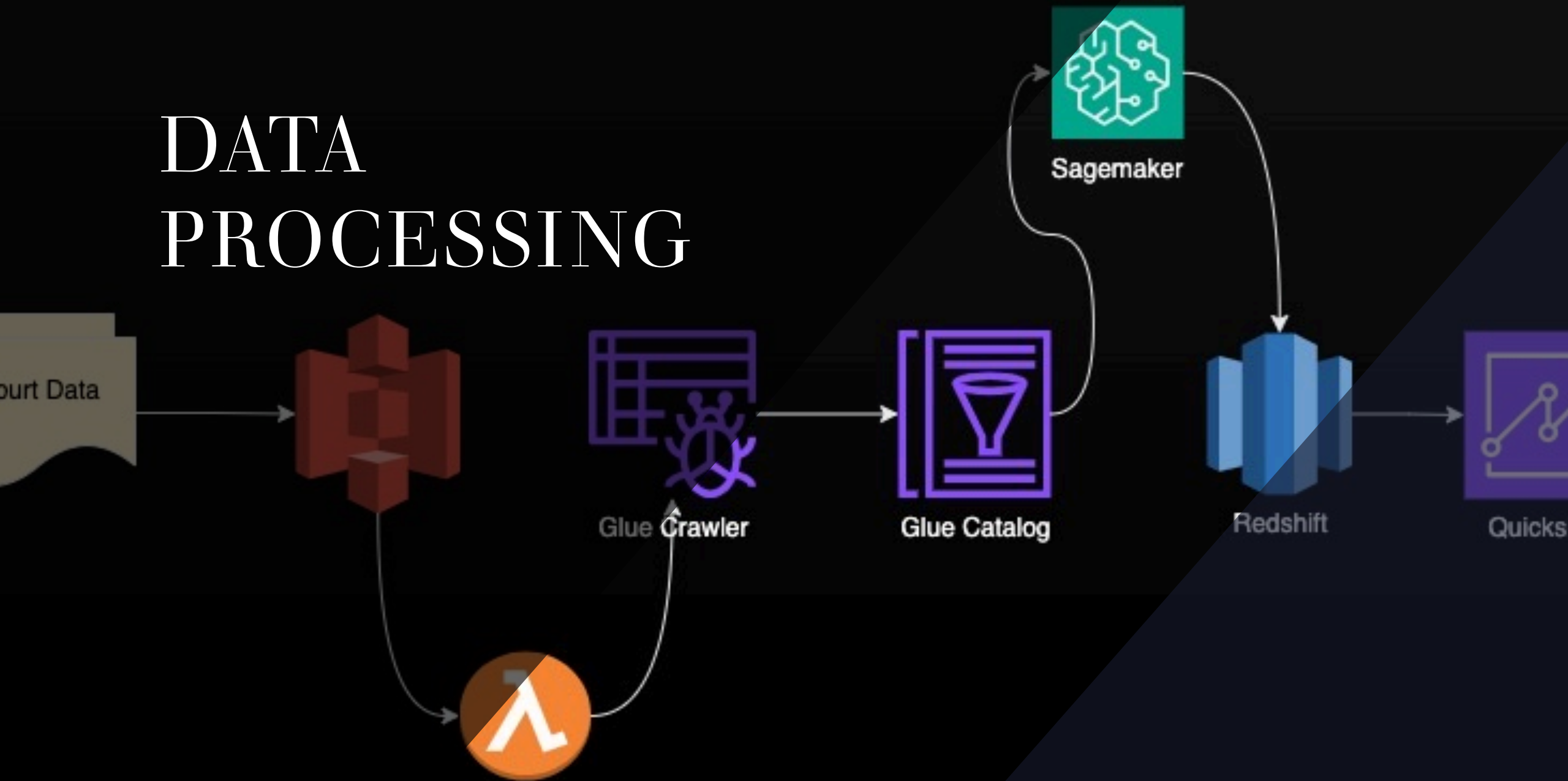
SentenceTime:

The response variable for a regression model had ~447k records with null values representing 6.2% of the dataset. 0 values were input to represent no sentence time for the case.

Charge Class:

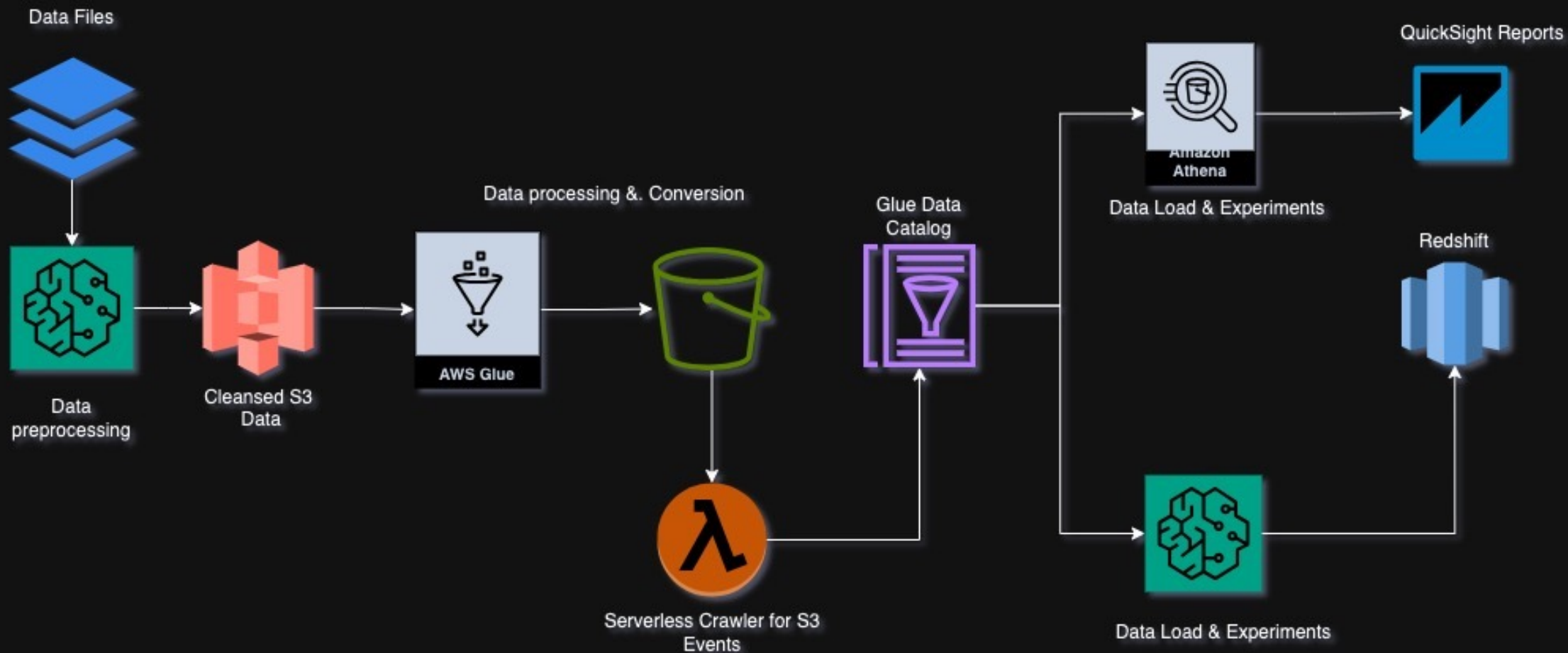
The *Class* feature had numerous entries (~2.4 million) with null values. It is not always applicable that a charge has a class therefore a value of "*None*" was inputted for these values.

DATA PROCESSING



Data Processing

Team 3 - High Level Architecture



Processing

Data Preprocessing	SageMaker / Notebooks	Clean and Merge source files
Data Processing	Glue	Job to Compress and Load data into different File formats
	Glue Crawlers	Schema definition for all different formats
	Lambda	Execute Automated crawlers whenever S3 event is created
	Athena	Read data from S3 and Glue Catalog
	Redshift Spectrum	Read data from S3 as external objects
	Redshift	Load data into different formats for experimentation
	SageMaker / Notebooks	Experimental Analysis
	Quicksight	Dashboard for Result visualization

Experimental Design

- Measure Performance of big data reports based on the different file and compression formats supported by AWS

Format	File extension (optional)	Extensions for compressed files
Comma-separated values	.csv	.gz .snappy .lz4 .bz2 .deflate
Microsoft Excel workbook	.xlsx	No compression support
JSON (JSON document and JSON lines)	.json, .jsonl	.gz .snappy .lz4 .bz2 .deflate
Apache ORC	.orc	.zlib .snappy
Apache Parquet	.parquet	.gz .snappy .lz4

Experimentation

- For tables created, execution time was retrieved from CloudWatch Logs
- Leveraged Athena to query different compression formats in S3 buckets
 - 3 READ queries performed of varying sizes, 3 times each:
 - 10 records <-- (ex: `head` or `tail` of data)
 - 10,000 records <-- (ex: sampling)
 - All records <-- (ex: full data query)
 - Query times (seconds) and Data Scanned (MB) was recorded
- Data was then averaged, exported as a .csv, and loaded onto QuickSight for visualization



Results Compression – S3

Source Data Size	Data Format	Compression	Comp S3 Size	Redshift Size
after Data cleaning and preparation 1 GB	csv	gzip	63 MB	388 MB
	csv	snappy	129 MB	360MB
	json	gzip	78 MB	224MB
	json	snappy	80 MB	224MB
	orc	snappy	54.5 MB	210MB
	orc	zlib	41.1. MB	224MB
	parquet	Gzip	38.7 MB	224MB
	parquet	lz4	50.4 MB	--
	parquet	lzo	49.9 MB	--
	parquet	snappy	50.3. MB	220MB
	parquet	zstd	39.1 MB	210MB

*** Lz compression is not supported with Athena for Csv and Json*

*** Redshift Deflates the data and then stores it in its cluster – Until Unless column level compression is defined.*



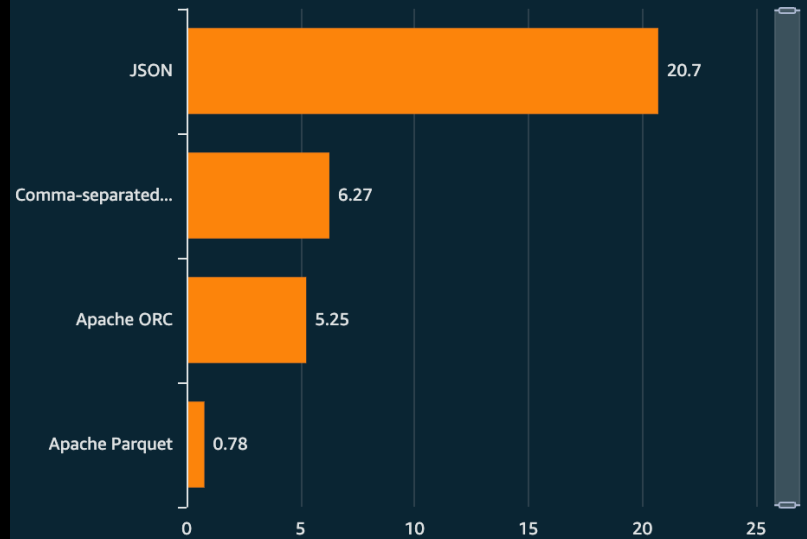
Results

- Table Creation

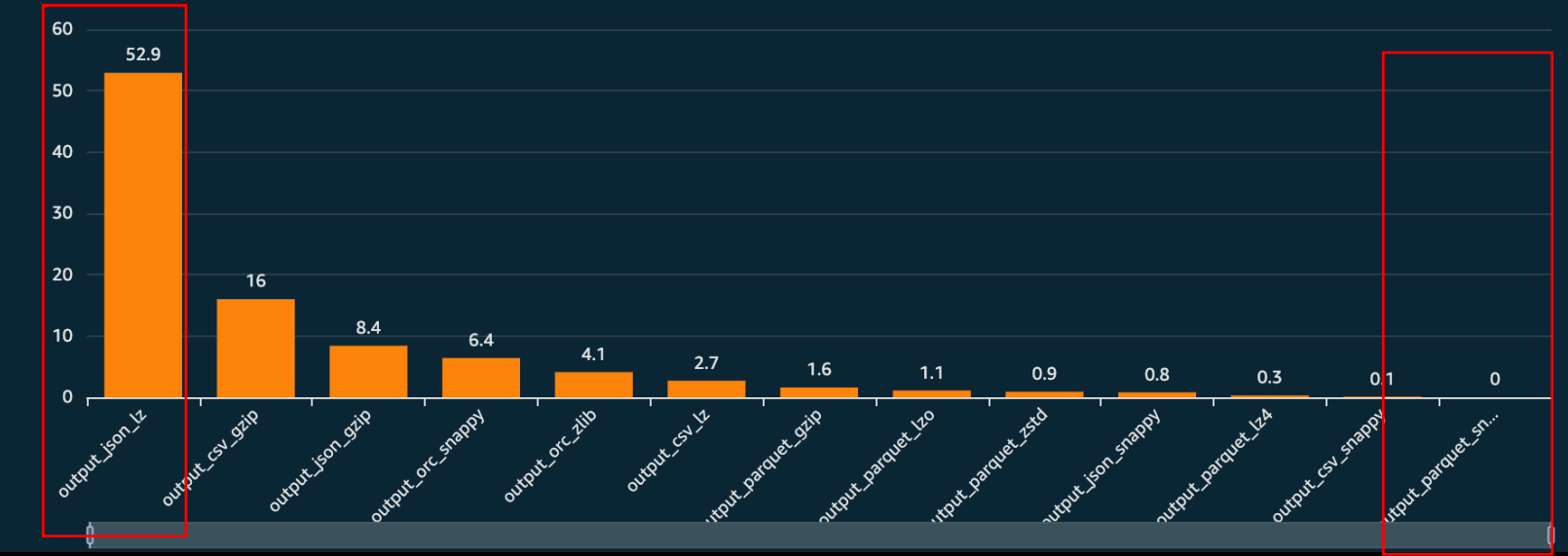
Count of Records by Format and Name

Rows	Count
<input type="checkbox"/> Apache ORC	2
<input type="checkbox"/> Apache Parquet	5
<input type="checkbox"/> Comma-separated values	3
<input type="checkbox"/> JSON	3

Average Table Creation Time (s) by Format



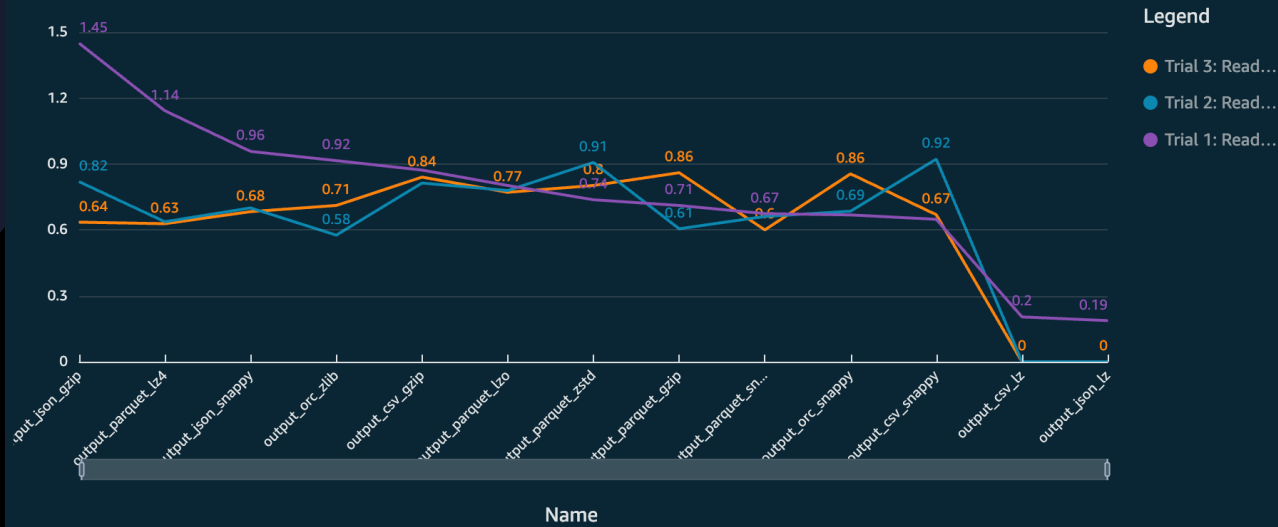
Average Table Creation Time (s) by Format & Extension



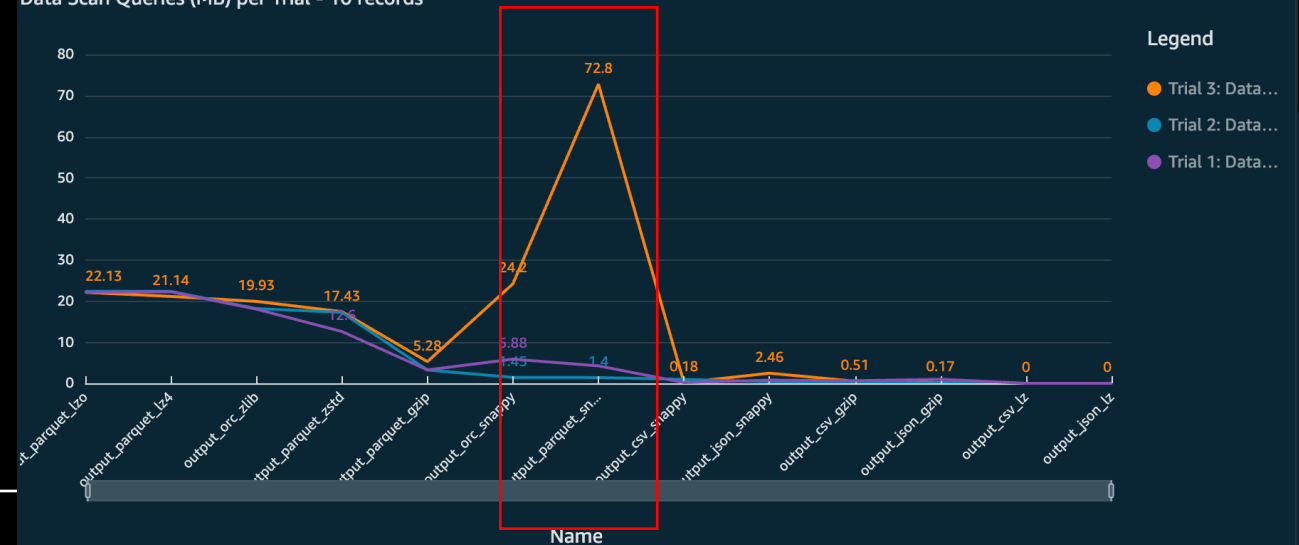


Results – 10 records

Read Query Times (s) per Trial - 10 records



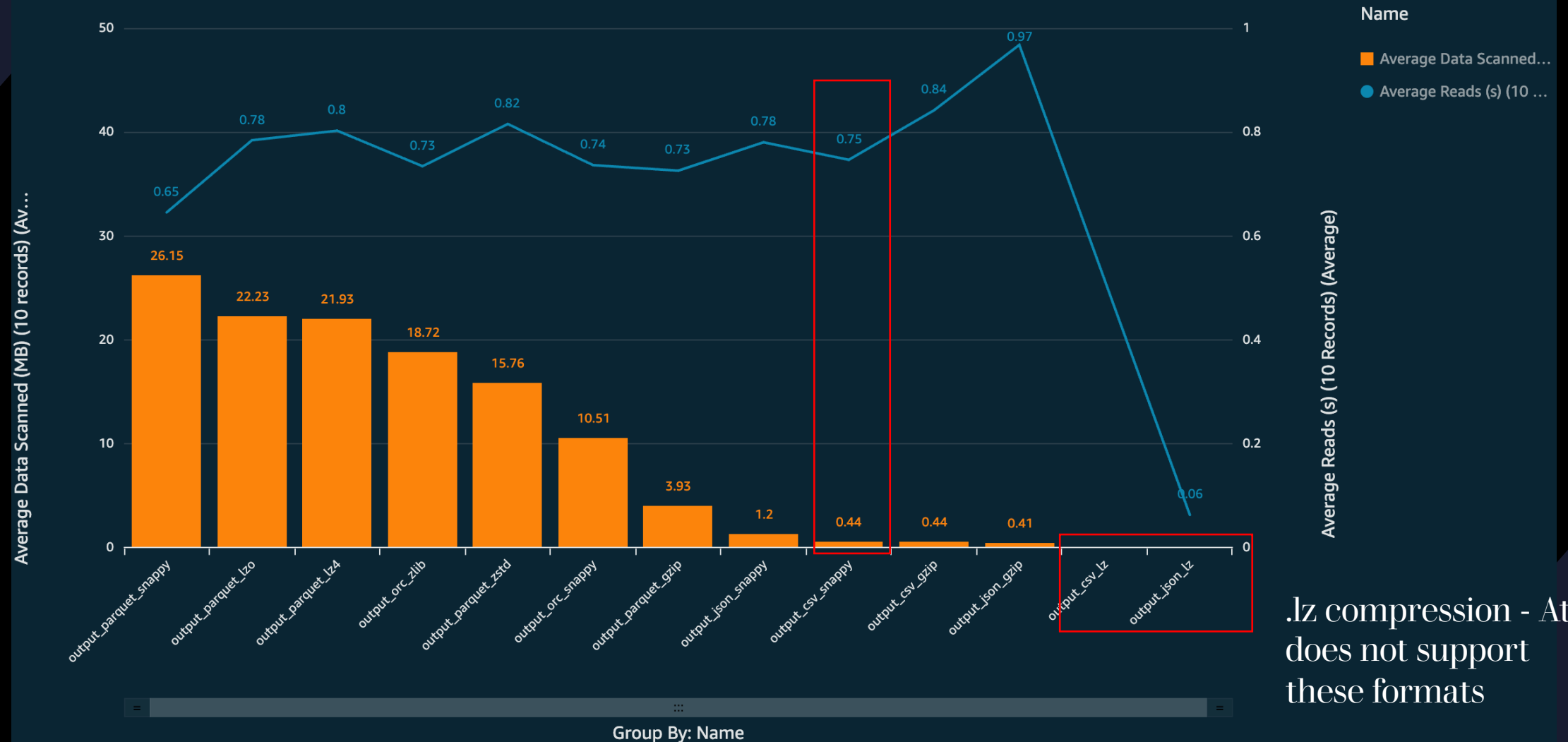
Data Scan Queries (MB) per Trial - 10 records





Results – 10 records

Average of Data Scanned (MB) and Read Time (s) - 10 records

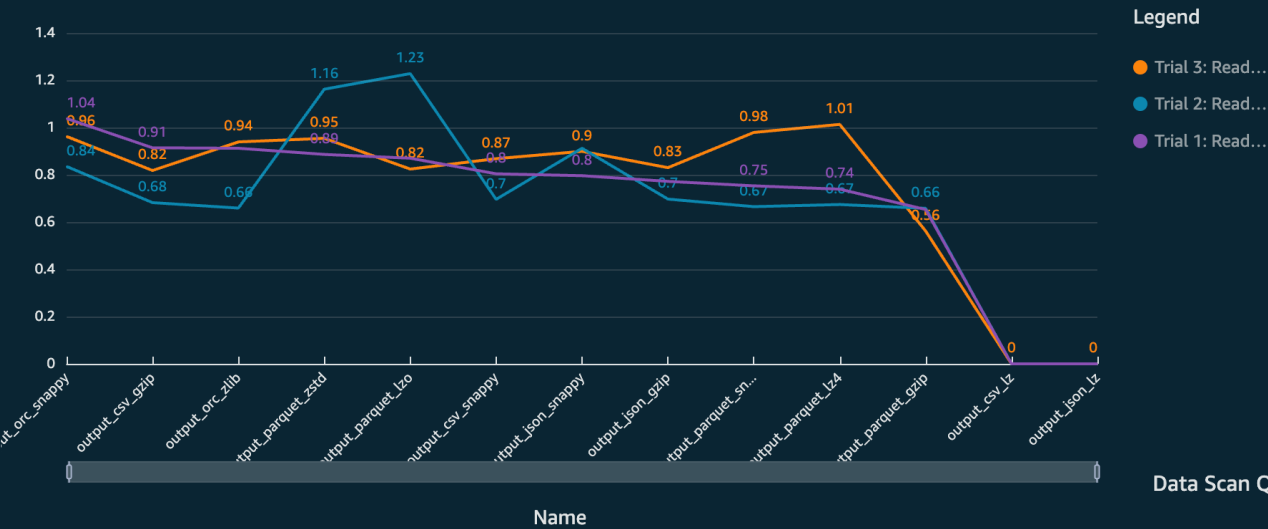


.lz compression - Athena does not support these formats

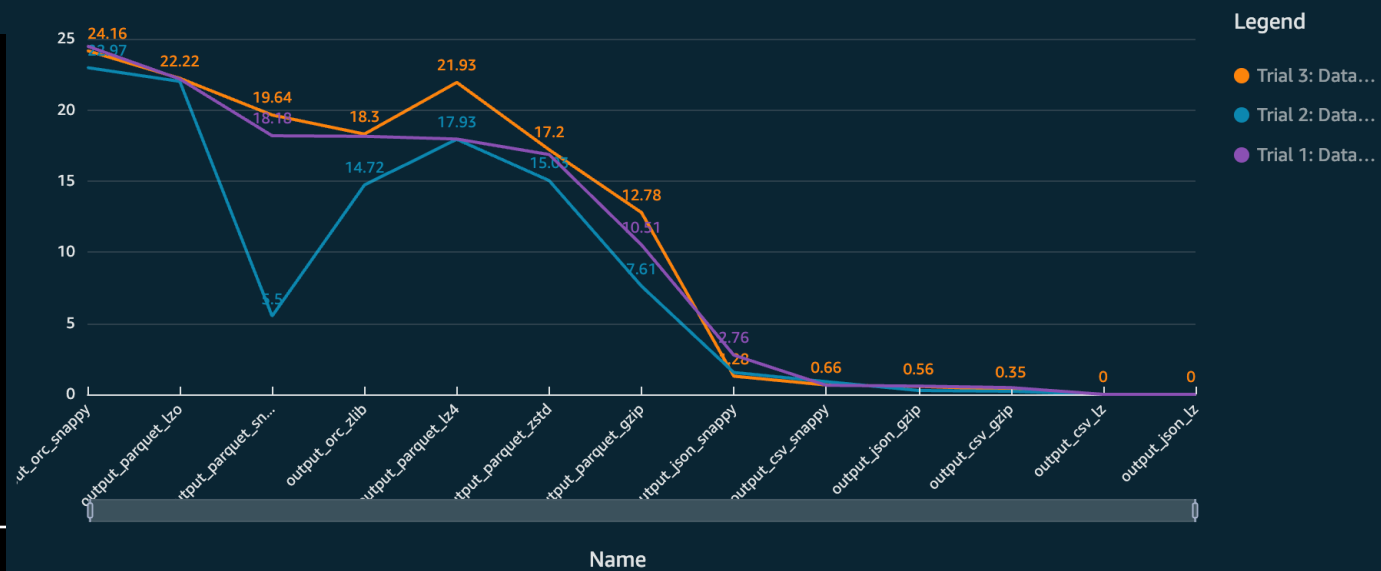


Results – 10000 records

Read Query Times (s) per Trial - 10K records



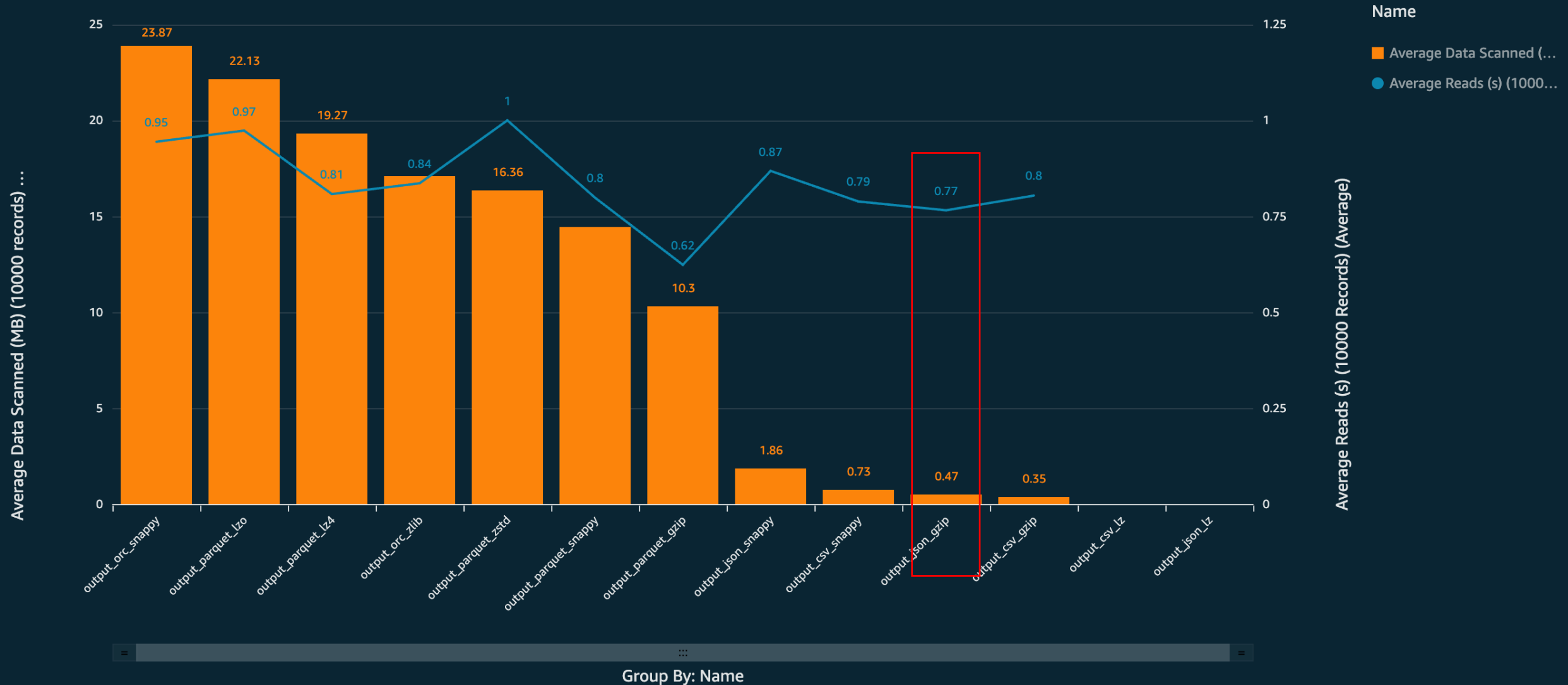
Data Scan Queries (MB) per Trial - 10K records





Results – 10000 records

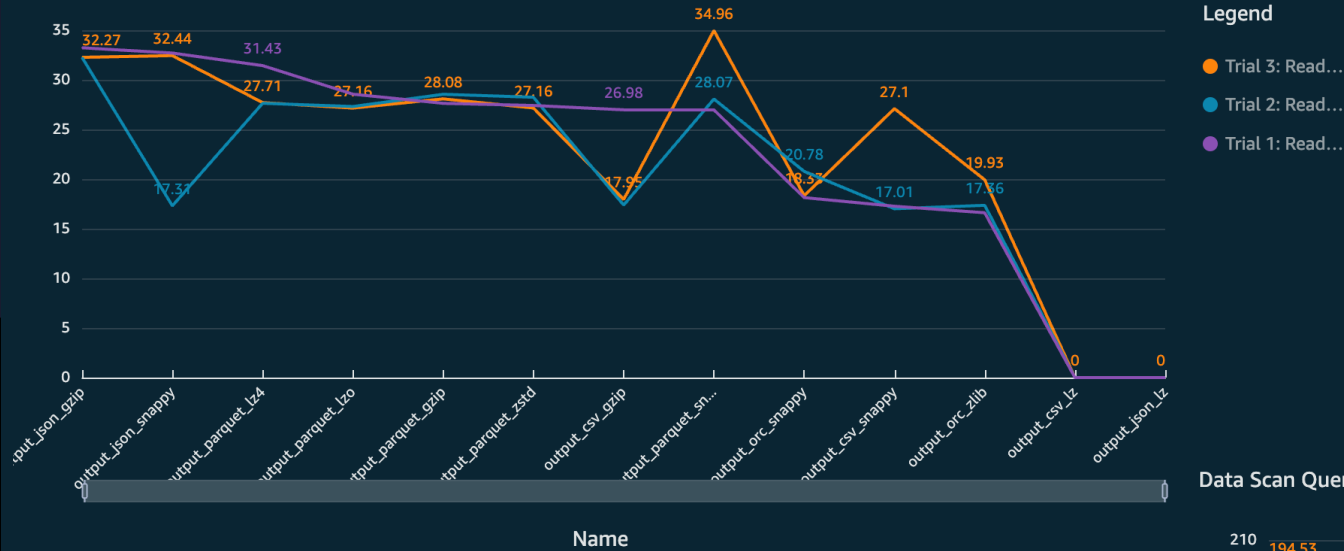
Average of Data Scanned (MB) and Read Time (s) - 10K records



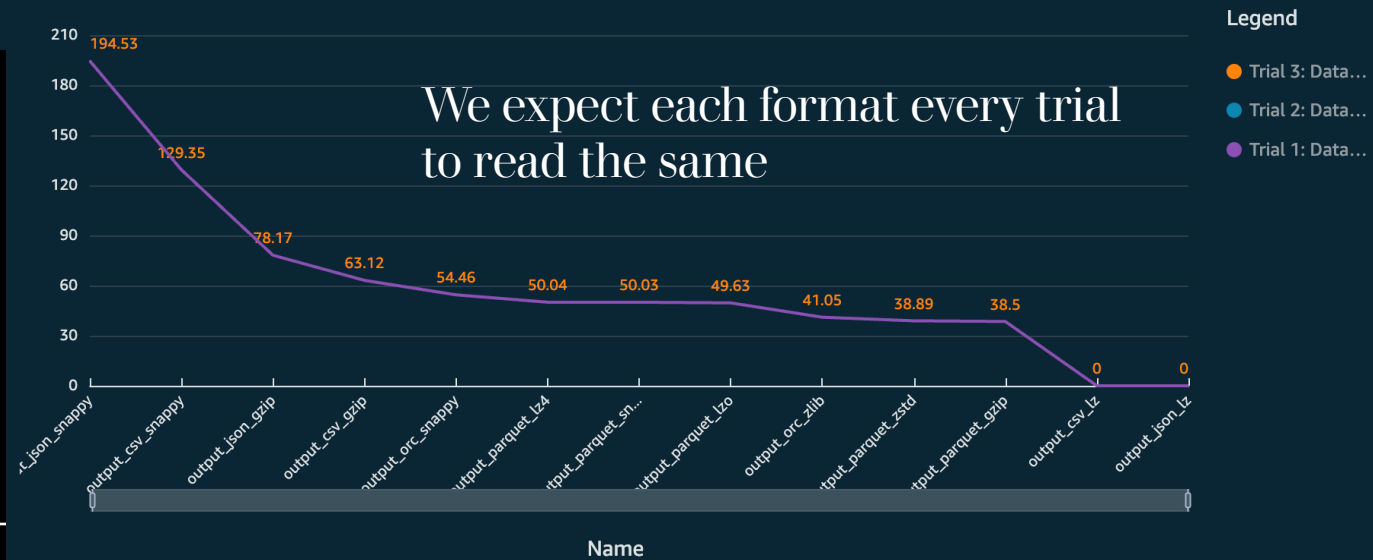


Results – All records

Read Query Times (s) per Trial - All records



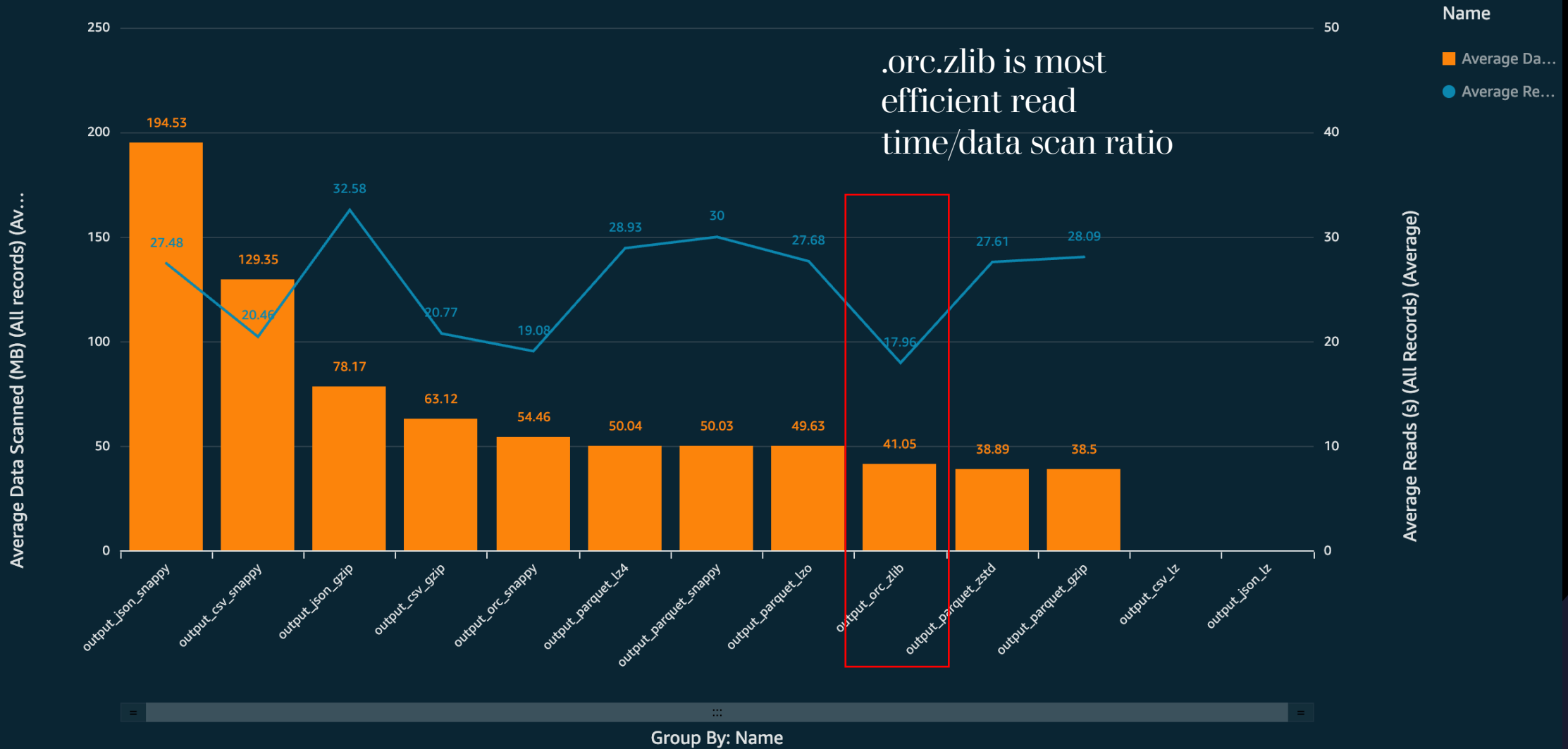
Data Scan Queries (MB) per Trial - All records





Results – All records

Average of Data Scanned (MB) and Read Time (s) - All records

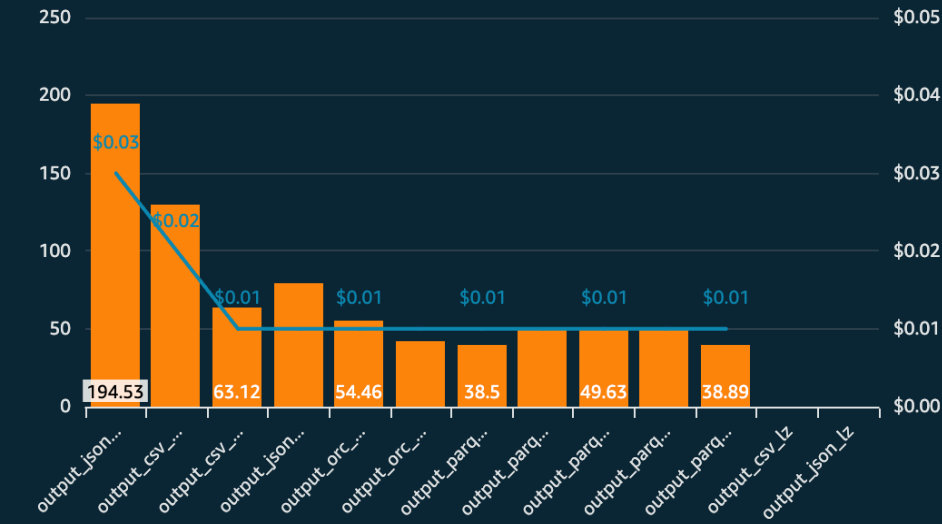




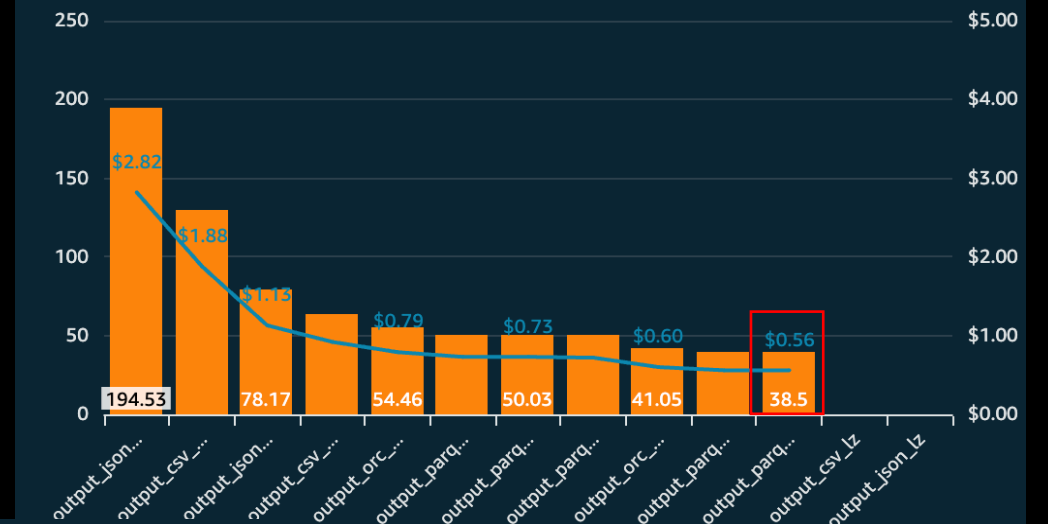
Results – Cost Estimation

Obtained from AWS Cost Estimator

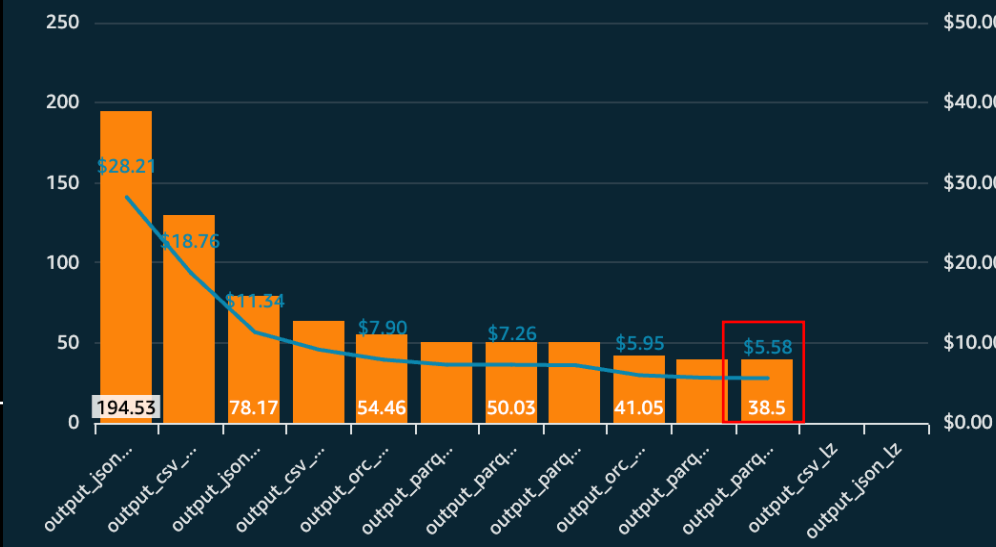
Estimated Cost (\$) from Data Scanned - 1 query per day



Estimated Cost (\$) from Data Scanned - 100 queries per day



Estimated Cost (\$) from Data Scanned - 1000 queries per day

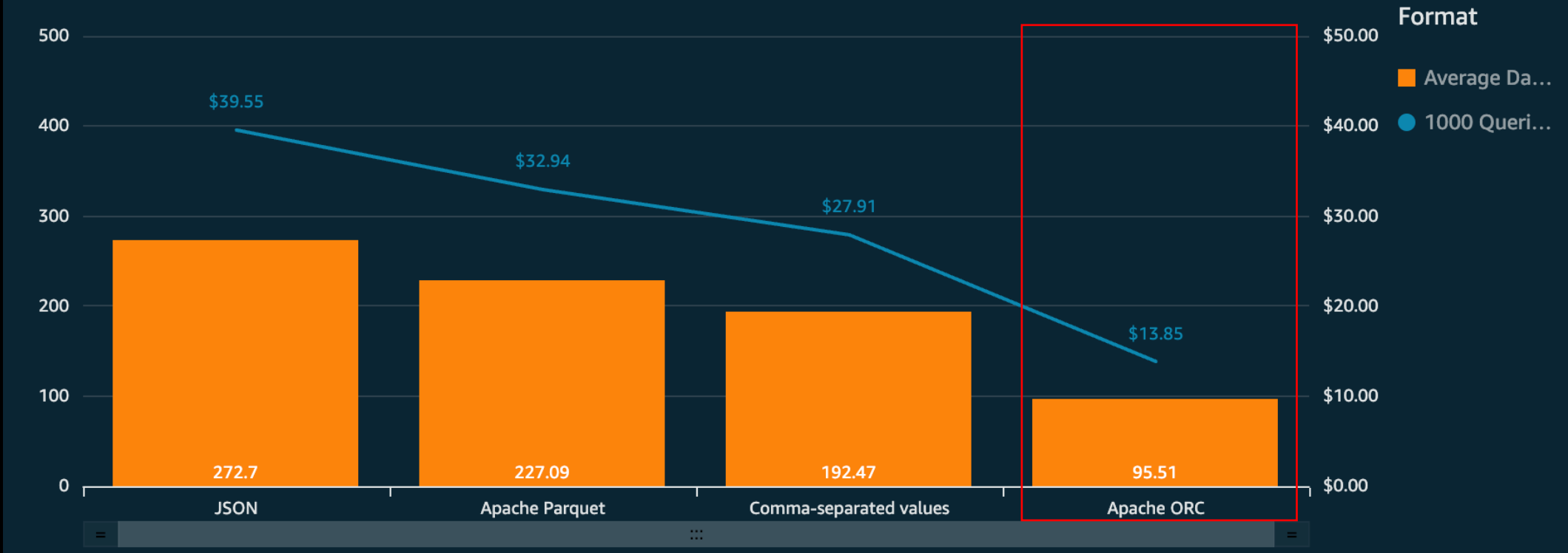


Parquet.gzip is the most cost effective format for Athena READs



Results – Cost Estimation

Estimated Cost (\$) from Data Scanned (MB) by Format - 1000 queries per day



Link to Quicksight dashboard: https://us-east-1.quicksight.aws.amazon.com/sn/accounts/211125778552/dashboards/df177a27-eeaf-4861-ab23-9bd5c4da0aacc/sheets/df177a27-eeaf-4861-ab23-9bd5c4da0aacc_9cee886f-53d0-4863-9995-c78eaa190ebc

Cost Estimation - Project

ⓘ Successfully added Amazon SageMaker estimate.

[AWS Pricing Calculator](#) > [My Estimate](#)

My Estimate

Edit ↗

Export ▾

Share

Estimate summary [Info](#)

Upfront cost

0.00 USD

Monthly cost

18.78 USD

Total 12 months cost

225.36 USD

Includes upfront cost

Getting Started with AWS

Get started for free

Contact Sales

My Estimate

Find resources

Duplicate

Delete

Move to

Create group

Add support

Add service

< 1 > ⓘ

<input type="checkbox"/>	Service Name ▾	Status ▾	Upfront cost ▾	Monthly cost ▾	Description ▾	Region ▾	Config Summary ▾
<input type="checkbox"/>	Amazon Simpl... ↗	-	0.00 USD	0.26 USD	-	US East (N. Vir...	S3 Standard storage (10 GB per month), Data returned by S3 Select (10 GB per month), Data scanned by S3 Select (10 GB per month) DT Inbound: Internet (1 TB per month), DT Inbound: Not s...
<input type="checkbox"/>	Amazon Athena ↗	-	0.00 USD	0.87 USD	-	US East (N. Vir...	Total number of queries (100 per day), Amount of data scanned per query (60 MB)
<input type="checkbox"/>	AWS Glue ↗	-	0.00 USD	5.57 USD	-	US East (N. Vir...	Number of DPU's for Apache Spark job (50), Number of DPU's for Python Shell job (0.0625) Number of Objects stored (0.0001 million per month), Number of access requests (0.001 million per ...
<input type="checkbox"/>	Amazon Redshift ↗	-	0.00 USD	7.18 USD	-	US East (N. Vir...	Nodes (1), Instance type (dc2.large), Utilization (On-Demand only) (25 Hours/Month), Pricing strategy (OnDemand), Data scanned (100 GB), Managed storage size (10 GB), Data Transfer In To (...
<input type="checkbox"/>	AWS Lambda ↗	-	0.00 USD	0.00 USD	-	US East (N. Vir...	Architecture (x86), Architecture (x86), Invoke Mode (Buffered), Amount of ephemeral storage allocated (512 MB), Number of requests (10 per month)
<input type="checkbox"/>	Amazon Sage... ↗	-	0.00 USD	4.90 USD	-	US East (N. Vir...	Instance name (ml.c5.2xlarge), Number of data scientist(s) (3), Number of Studio Notebook instances per data scientist (1), Studio Notebook hour(s) per day (1), Studio Notebook day(s) per mo...

Acknowledgement

AWS Pricing Calculator provides only an estimate of your AWS fees and doesn't include any taxes that might apply. Your actual fees depend on a variety of factors, including your actual usage of AWS services. [Learn more](#) ⓘ

Conclusion – Athena & S3



Data format and Compression

Select appropriate data formats (e.g., Parquet, ORC) and compression types (e.g., Snappy, Zstandard) based on your data characteristics and query patterns.



Organize Data in S3

Organize data in Amazon S3 using partitioning and bucketing techniques to improve query performance and reduce query costs.



Define Schema

Define the schema for your data in Athena either through AWS Glue Data Catalog or by creating tables using CREATE TABLE AS queries.



Optimize Query Performance

Optimize query performance by leveraging partition pruning, predicate pushdown, and column projection to reduce data scanned by queries.



Optimize Data Access Patterns

Optimize data access patterns by using AWS Glue Crawlers, updating partitions, and leveraging the AWS Glue Data Catalog for metadata management.



Monitor and Analyze

Monitor and analyze query performance using AWS CloudWatch metrics and Athena query execution plans to identify bottlenecks and areas for optimization.

Conclusion – Redshift



Data format and Compression

Choose Data Format and Compression: Select appropriate data formats (e.g., CSV, Parquet) and compression types (e.g., GZIP, ZSTD) based on your data characteristics and query patterns.



Load Data (Glue , COPY , Python)

Load Data to Amazon Redshift: Load data into Redshift using preferred methods such as COPY command, INSERT statements, or Glue from S3 for large datasets.



Optimize Data Distribution & sorting

Optimize Data Distribution & Sorting: Utilize appropriate distribution and sort keys (DISTKEY, SORTKEY) to optimize data distribution and storage on Redshift.



Analyze and Regular Vacuum

Perform Regular Vacuum and Analyze: Schedule regular Vacuum and Analyze operations to reclaim disk space and update statistics for query optimization.



Optimize Query Performance (Tuning)

Optimize Query Performance: Fine-tune and optimize queries for better performance, including indexing, query structure, and data distribution strategies.



Monitor and Analyze

Monitor and Analyze Query Performance: Continuously monitor query performance metrics and analyze query execution plans to identify bottlenecks and areas for improvement.

APPENDIX



S3 – Compressed Data Snapshot

The screenshot displays the Amazon S3 console interface for a bucket named 'team-3-project-data' in the 'us-east-1' region. The breadcrumb navigation shows the path: Amazon S3 > Buckets > team-3-project-data > convrted-data/ > redshift-ingestion/. The page title is 'redshift-ingestion/'.

Below the breadcrumb, there are tabs for 'Objects' (selected) and 'Properties'. A 'Copy S3 URI' button is visible in the top right corner.

The 'Objects' section shows 15 objects. A search bar is present with the placeholder text 'Find objects by prefix'. Below the search bar, a table lists the objects with columns: Name, Type, Last modified, Size, and Storage class.

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	model_data.csv	csv	March 26, 2024, 21:07:20 (UTC-04:00)	770.0 MB	Standard
<input type="checkbox"/>	model_data.csv.gz	gz	March 26, 2024, 21:07:21 (UTC-04:00)	60.7 MB	Standard
<input type="checkbox"/>	output_csv_gzip/	Folder	-	-	-
<input type="checkbox"/>	output_csv_lz/	Folder	-	-	-
<input type="checkbox"/>	output_csv_snappy/	Folder	-	-	-
<input type="checkbox"/>	output_json_gzip/	Folder	-	-	-
<input type="checkbox"/>	output_json_lz/	Folder	-	-	-
<input type="checkbox"/>	output_json_snappy/	Folder	-	-	-
<input type="checkbox"/>	output_orc_snappy/	Folder	-	-	-
<input type="checkbox"/>	output_orc_zlib/	Folder	-	-	-
<input type="checkbox"/>	output_parquet_gzip/	Folder	-	-	-
<input type="checkbox"/>	output_parquet_lz4/	Folder	-	-	-
<input type="checkbox"/>	output_parquet_lzo/	Folder	-	-	-
<input type="checkbox"/>	output_parquet_snappy/	Folder	-	-	-
<input type="checkbox"/>	output_parquet_zstd/	Folder	-	-	-

At the bottom of the console, there are links for 'CloudShell' and 'Feedback'. The footer contains the copyright notice '© 2024, Amazon Web Services, Inc. or its affiliates.' and links for 'Privacy', 'Terms', and 'Cookie preferences'.

[illegible]

Redshift Database

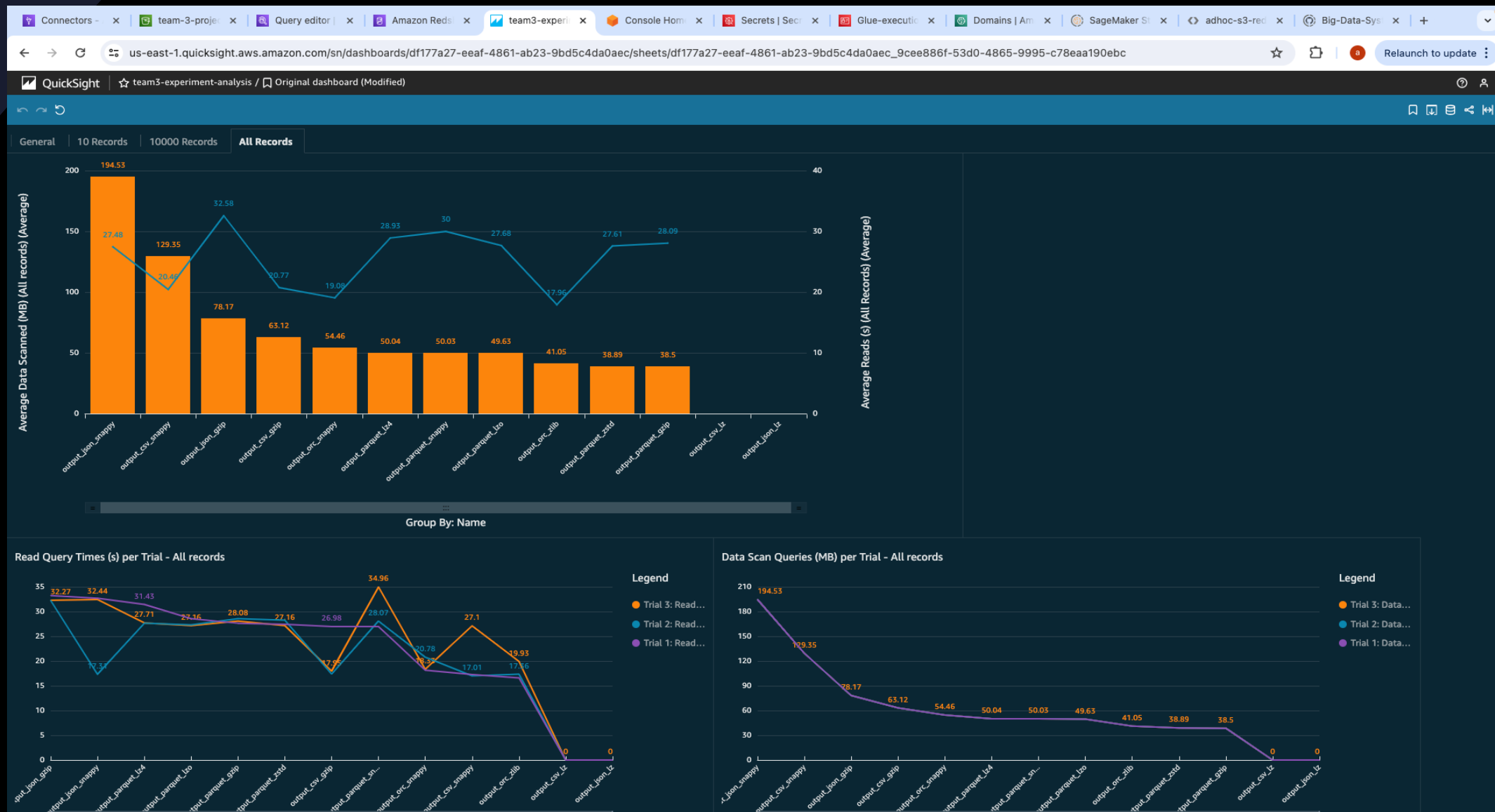
The screenshot displays the Amazon Redshift Query Editor v2 interface in a web browser. The browser's address bar shows the URL: `us-east-1.console.aws.amazon.com/redshiftv2/home?region=us-east-1#query-editor:`. The interface includes a top navigation bar with the AWS logo, a search bar, and a list of services. Below this, a blue banner provides information about Query Editor v2 features. The main workspace is divided into several sections:

- Left Panel (Resources):** Contains a 'Select database' dropdown set to 'dev', a 'Select schema' dropdown set to 'team3', and a list of tables including `output_csv_gzip`, `output_csv_snappy`, `output_json_gzip`, `output_json_lz`, `output_json_snappy`, `output_orc_snappy`, `output_orc_zlib`, `output_parquet_gzip`, `output_parquet_lz4`, `output_parquet_lzo`, `output_parquet_snappy`, and `output_parquet_zstd`.
- Top Right:** Shows the connection status as 'Connected' and the current database, schema, and user as 'database', 'dev', and 'team3' respectively. A 'Change connection' button is also present.
- Query Editor:** Displays a SQL query in the 'Query 4' tab. The query is as follows:

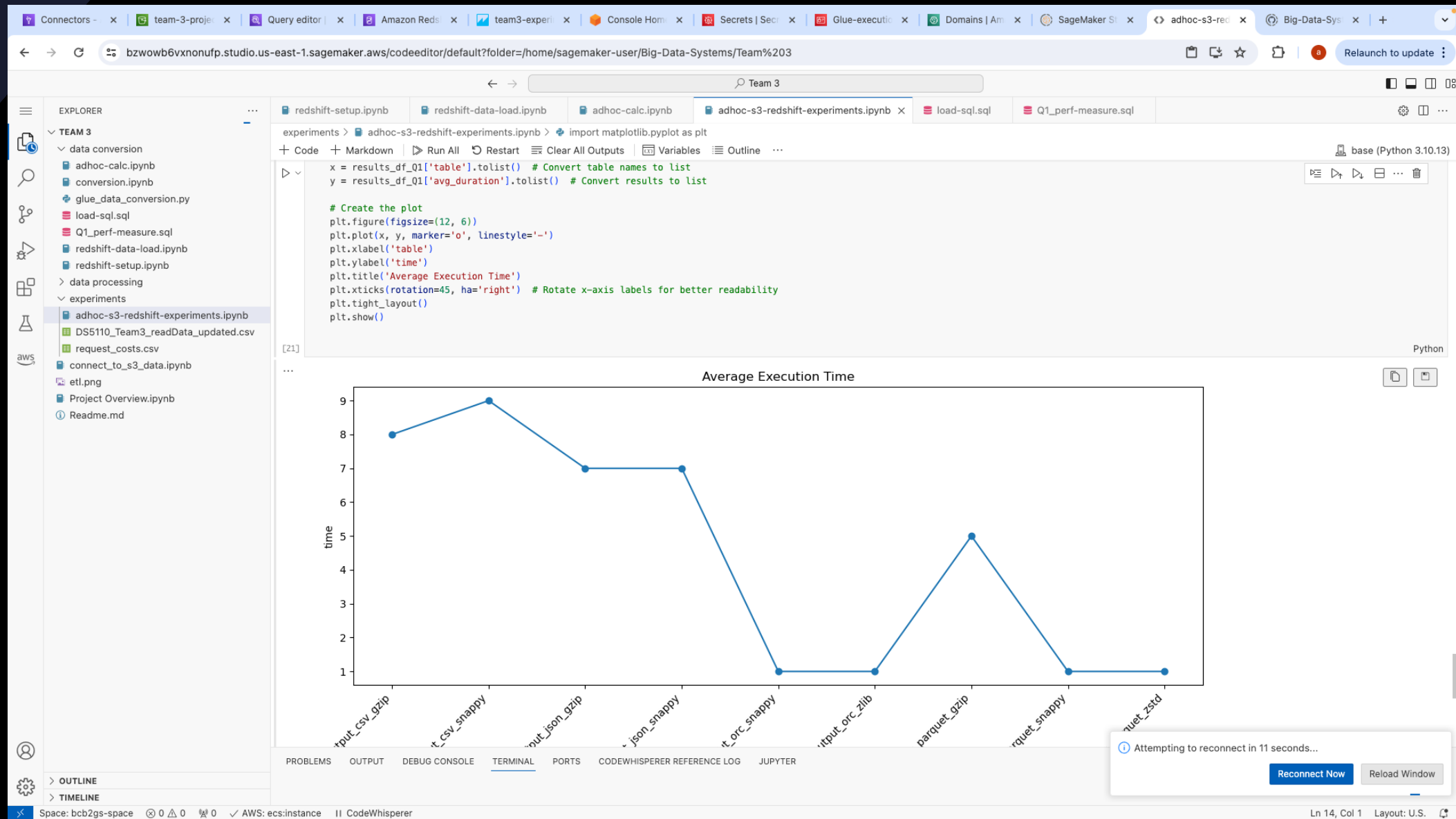
```
42
43 SELECT *
44 FROM stl_query
45 where querytxt like '%output_parquet_gzip%'
46 ORDER BY starttime DESC
47
48
49 SELECT query,querytxt,
50 --COUNT(*) as num_queries,
51 AVG(DATEDIFF(sec,starttime,endtime)) avg_duration,
52 MIN(starttime) as oldest_ts,
53 MAX(endtime) as latest_ts
54 FROM stl_query
```
- Bottom:** Includes 'Run', 'Save', 'Schedule', and 'Clear' buttons. Below these are tabs for 'Query results' and 'Table details'. The 'Table details' tab is currently selected, showing a message: 'No resources. To view details, choose data from navigator.'

The footer of the interface includes a 'CloudShell' button, a 'Feedback' link, and copyright information for Amazon Web Services, Inc. or its affiliates, along with links to 'Privacy', 'Terms', and 'Cookie preferences'.

QuickSight Reports



Sagemaker notebooks - Experiments



Thank you!
