		AWS Athena + S3	Google Big Query	SnowFlake
Overview	Highlights	- Athena has no storage costs/server set up - Pay per query: amount of data scanned - Connectivity to S3; data queried as is stored - Loose error handling	- User-friendly, detailed error messaging, strict error handling - Pay per query: amount of data scanned - Native tables stored in columnar format - Connectivity with Google stack	- Traditional data warehouse in the cloud - Pay per query: time query takes to run - (Auto-) Scalability on-demand solves concurrency issues - Metadata layer + optimized storage (partitioned, columnar, compressed) - Sits on AWS hence data transfer from S3 is easy
	Purpose	Storage (S3) + Querying (Athena)	Storage + Querying	Storage + Querying
Specs	Serverless Compute?	Yes	Yes	Yes - insofar as Warehouse can be set to be automatically suspended and resumed just prior to querying
	Distributed?	Yes	Yes	Yes - Can specify and change warehouse size
	Pricing - Storage	**S3** $0.023/GB/month	**Native Table (BigQuery)** - Long term, i.e. table not edited for 90 consecutive days: $0.01 /GB/month - Short term, e.g. a growing table: $0.02/GB/month **External Table (Cloud Storage)** - Pay for cost of storage in regional/multi-regional bucket - Multi-regional bucket* : $0.026/GB * We encountered the error: "cannot read data in location: US-WEST-1" when using a regional instead of multi-regional bucket)	**Pre-Pay** - Dollar committment; lower + long-term price guarantee - $23/TB/month; once past capacity, billed on-demand **On-Demand** - Minimum $25 monthly charge (maintained by either storage or compute); otherwise billed ad-hoc - Storage: $40/TB/month
	Pricing - Compute	**Athena** Priced per query: $5/TB scanned, 10MB minimum	Priced per query: $5/TB scanned, 10MB minimum	Billed for, not data scanned - per second with a one minute per query minimum. Different warehouse sizes consume credits at different rates. **Pre-Pay** -Billed per second; $2/credit/hour with discount by volume **On-Demand** - $2/credit/hour; number of credits used depends on size of warehouse: small = 2/hr; medium = 4/hr; large = 8/hr
	Pricing - In/Out	Cost of transferring *out of* S3 into Cloud Storage using gsutil: - Tier 1 Request (PUT, COPY, POST, or LIST): $0.005/1,000 requests - Tier 2 Request (GET): $0.004/10,000 requests ***Tentative**	Cost of transferring *out of* Cloud Storage into S3: ~$50/ 2.5TB	-TBD-
	SQL Syntax	Presto SQL: less pre-built text parsing syntax available	Standard SQL (needs to be selected), or Legacy SQL (native BigQuery syntax)	Transactional SQL
	Web UI	Athena Query Editor	BigQuery Compose Query - Can be fiddly to run multiple queries simultaneously (via Compose Query), cancel a query, etc - Need to manually Show/Hide Options check to use standard SQL and write results to table - Browser notifications for when query completes is available	Snowflake Worksheet - Looks more like a traditional SQL client; can do most things if not all via scripts, including warehouse provisioning - Shows number of rows scanned (in addition to amount of data scanned)

		AWS Athena + S3	Google Big Query	SnowFlake
	CLI/Connectivity	aws - SQL: aws athena start-query-execution - Queries Athena table and saves output to S3 location, e.g. aws athena start-query-execution --query-string "select * from table;" --result-configuration "OutputLocation=s3://output_buckett/" - No interactive mode - https://sysadmins.co.za/using-the-aws-cli-tools-to-interact-with-amazons-athena-service/ - Python: several packages (eg. boto) to acces S3 bucket; but as a data lake, you can't read data directly, need to retrieve/send content	- gsutil: Cloud Storage - bq: Big Query - SQL: can run using bash (bq query --use_legacy_sql=false "SELECT STATEMENT") or in interactive mode (bq shell) - Python: connect via API: https://cloud.google.com/bigquery/create-simple-app-api#bigquery-simple-app-build-service-python	- SQL: use snowsql CLI client (interactive mode) - Python: Snowflake connector available via pip - Spark: Snowflake connector
Process	Supported Data Sources	S3	Local files, Cloud Storage, Google Drive, Google Cloud BigTable	Local files (to be staged), S3
	Supported Input Formats	Anything structured in S3	CSV, JSON, AVRO	CSV, JSON, AVRO, ORC, PARQUET, XML
	Data Transfer (In)	CLI: aws s3 cp	CLI: gsutil cp/rsync E.g. Copy from one folder to another: gsutil -m rsync -R gs://source_folder/ gs://target_folder/ Copy from Cloud Storage to S3: gsutil -m rsync -R gs://source_folder/ s3://target_folder/ Several files failed to transfer from Cloud Storage to S3; unclear if due to gsutil or yens server	CLI: snowsql COPY INTO If transferring from S3: data moving within AWS environment and hence very fast. (10min for Events/Mentions; 90 min for GKG)
	Data Load	"Bulk load" from bucket (data not actually stored)	"Bulk load" by using wildcard * to point to all files in a bucket External Table: BQ external table simply points to the external data source; does not load anything inline Native Table: BQ native tables are backed by BigQuery storage and creating one initiates a load job to load the data to BigQuery Has option to automatically detect file schema	"Bulk load" from bucket Need to stage files either on S3 or Snowflake platform and load from there; data in S3 an be copied directly using COPY INTO
	Data Storage Format	S3: a data lake Athena: does not store data, it simply points to data in S3; output is stored in S3 Note that this means that to work with columnar data, conversion to a columnar format has to be done outside Athena (e.g. via EMR)	**External Table**: data is as is externally formatted **Native Table**: data stored in columnar format	When loaded to Snowflake, data is automatically split into micro-partitions, meta-data is extracted to enable efficient query processing, and micro-partitions are columnar-compressed. As Snowflake is built on S3, all Snowflake data (input and query results) is fundamentally stored in S3.
	Error Handling	By default, Athena loads everything with loose error handling: allows jagged rows (i.e. rows that don't match schema), UTF-encoding issues	**On "Load":** – Provides load error and points to file that caused error – By default, strict with error handling; via UI, can choose to allow (1) quoted newlines, (2) jagged rows, (3) ignore unknown values, (4) x number of errors (for all other issues) **On Query:** - Extremely detailed error messages (e.g. "Project name needs to be separated by dot from dataset name, not by colon in table name "gsb-circlerss:GDELT.gkg_native")	**On Load** - More options (via copy options and file format options) available for specific error handling (e.g. re. UTF-encoding) as opposed to blanket allow error clause

		AWS Athena + S3	Google Big Query	SnowFlake
	Performance - Pros	Queries potentially faster on all-string datasets, even if casting is required (we tested same queries on dataset wth ints and floats and dataset with all strings)	**External Table**: queries may exhibit slower performance due to the non-columnar nature of external storage **Native Table**: queries are optimized against columnar datasets Queries run faster when writing into a table vs. displaying output (e.g. 2 min vs. 4 min for Query 32)	- Data storage format + meta-data layer optimized for querying - Processing speed depends on size of warehouse selected - (Auto-) Scability on-demand: can scale up (size) for a single massive query or wide (more machines) for multiple concurrent users
	Performance - Cons	- Default query timeout limit of 30 minutes - Exhausted resources error	Potential issue with mulitple concurrent users querying the same table (Snowflake sales)	
	Output	- All Athena query results are automatically saved to S3 as a single file (no size limit) in pre-specified format - To query this output, need to locate and repoint Athena to the S3 file	- If size of output is >128MB compressed, need to save output to native table - If size of output is <= 128MB: with the option to save as a native table or temporarily save the result in BQ - Can export BQ tables to flat file where each file is limited to 1GB (can export to multiple files using wildcard: e.g. gs://output_bucket/results-*.csv)	- Use 'create table as' syntax to write query into a table, or - Export results to CSV/TSV
	Built On	Athena: HIVE, Presto	-TBD-	AWS
	Maintenance	-TBD-	-TBD-	-TBD-
	Security	-TBD-	-TBD-	-TBD-