Continue

The SQL SELECT query components for AWS Config advanced queries are as follows. Synopsis SELECT proprty [, ...] [WHERE condition] [GROUP BY proprty] [ORDER BY proprty [ASC | DESC] [, proprty [ASC | DESC] ...]] Parameters [WHERE condition] Filter results according to the condition you specify. Comparison operators = (equals) IN (list membership) BETWEEN (range check) Logic operators [GROUP BY proprty] Aggregates the result set into groupes of rows with matching values for the given proprty. The GROUP BY clause is applicable to aggregations. [ORDER BY proprty [ASC | DESC] [, proprty [ASC | DESC] ...]] Sorts a result set by one or more output proprties. When the clause contains multiple proprties, the result set is sorted according to the first proprty, then according to the second proprty for rows that have matching values for the first proprty, and so on. This allows you to query exactly the current resource state you need without performing AWS service-specific API calls. It supports aggregation functions such as AVG, COUNT, MAX, MIN, and SUM. You can use advanced query for: Inventory management; for example, to retrieve a list of Amazon EC2 instances of a particular size. Security and operational intelligence; for example, to retrieve a list of resources that have a specific configuration property enabled or disabled. Cost optimization; for example, to identify a list of Amazon EBS volumes that are not attached to any EC2 instance. Compliance data; for example, to retrieve a list of all your conformance packs and their compliance status. For information about how to use the AWS SQL Query Language, see What Is SQL (Structured Query Language)?. Limitations Advanced query does not support querying resources which have not been configured to be recorded by the configuration recorder. AWS Config creates Configuration Items (CIs) with ResourceNotRecorded in the configurationItemStatus when a resource has been discovered but is not configured to be recorded by the configuration recorder. While an aggregator will aggregate these CIs, advanced query does not support querying CIs with ResourceNotRecorded. Update your recorder settings to enable recording of the resource types that you want to query. As a subset of SQL SELECT, the query syntax has following limitations: No support for ALL, AS, DISTINCT, FROM, HAVING, JOIN, and UNION keywords in a query. NULL value queries are not supported. No support for complex CASE statements for creating a priority field directly in the query. No support for querying on third-party resources. Third-party resources retrieved using advanced queries will have the configuration field set as NULL. No support for nested structures (such as tags) to be unpacked with SQL queries. No support for querying deleted resources. To discover deleted resources, see Looking Up Resources That Are Discovered by AWS Config. The SELECT all columns shorthand (that is SELECT *) selects only the top-level, scalar properties of a CI. The scalar properties returned are accountId, awsRegion, arn, availabilityZone, configurationItemCaptureTime, resourceCreationTime, resourceId, resourceName, resourceType, and version. Wildcard limitations: Wildcards are supported only for property values and not for property keys (for example, ...WHERE someKey LIKE 'someValue%' is supported but ...WHERE 'someKey%' LIKE 'someValue%' is not supported). Support for only suffix wildcards (for example, ...LIKE 'AWS::EC2::%' and ...LIKE 'AWS::EC2::_' is supported but ...LIKE '%::EC2::Instance' and ...LIKE '_::EC2::Instance'is not supported). Wildcard matches must be at least three characters long (for example, ...LIKE 'ab%' and ...LIKE 'ab_' is not allowed but ...LIKE 'abc%' and ...LIKE 'abc_' is allowed). The "_" (single underscore) is also treated as a wildcard. Aggregation limitations: Aggregate functions can accept only a single argument or proprty. Aggregate functions cannot take other functions as arguments. GROUP BY with an ORDER BY clause referencing aggregate functions may contain only a single proprty. For all other aggregations GROUP BY clauses may contain up to three proprties. Pagination is supported for all aggregate queries except when ORDER BY clause has an aggregate function. For example, GROUP BY X, ORDER BY Y does not work if Y is an aggregate function. No support for HAVING clauses in aggregations. Mismatched identifier limitations: Mismatched identifiers are proprties that have the same spelling but different cases (upper and lower case). Advanced query does not support processing queries that contain mismatched identifiers. For example: Two proprties that have the exact same spelling but with different casing (configuration.dbclusterIdentifier and configuration.dBClusterIdentifier). Two proprties where one proprty is a subset of the other, and they have different casing (configuration.ipAddress and configuration.ipaddressPermissions). CIDR notation/IP range behavior for advanced queries CIDR notation is converted to IP ranges ###ARTICLEAdvanced queries provide a flexible way to manage and analyze the current state of Amazon Config resources. This can be done using a subset of SQL SELECT syntax. The Amazon advanced query language provides various features for cost optimization, compliance data retrieval, and querying resources. However, it has some limitations. Cost Optimization: One way to optimize costs is by identifying unattached EBS volumes. You can do this by running a query like `SELECT * FROM config WHERE resourceType = 'EBS' AND isAttached = FALSE`. Compliance Data Retrieval: To retrieve compliance data, you can run queries like `SELECT * FROM config WHERE complianceStatus = 'COMPLIANT'` or `SELECT * FROM conformancePack WHERE status = 'COMPLETED'`. Limitations: The advanced query language has several limitations. It does not support querying resources that have not been configured to be recorded by the configuration recorder. You need to update your recorder settings to enable recording of specific resource types. Some other limitations include: - No support for ALL, AS, DISTINCT, FROM, HAVING, JOIN, and UNION keywords. - NULL value queries are not supported. - Complex CASE statements cannot be used for creating a priority field directly in the query. - Third-party resources retrieved using advanced queries will have the configuration field set as NULL. - Nested structures (such as tags) cannot be unpacked with SQL queries. - No support for querying deleted resources. AWS configservice provides a way to retrieve configuration data from AWS resources and manage compliance with regulatory requirements. The following are some examples of queries you can use to get information about your AWS resources. You can list all EC2 instances with AMI ID ami-12345 using the following query: SELECT resourceId, resourceType, configuration.instanceType, configuration.placement.tenancy, configuration.imageId, availabilityZone WHERE resourceType = 'AWS::EC2::Instance' AND configuration.imageId = 'ami-12345' Additionally, you can use the following queries to get information about your AWS resources: To get count of resources grouped by their AWS Config rules compliance status, use the query: SELECT configuration.complianceType, COUNT(*) WHERE resourceType = 'AWS::Config::ResourceCompliance' GROUP BY configuration.complianceType To get the compliance status of AWS Conformance packs, use the query: SELECT resourceId, resourceName, resourceType, configuration.complianceType WHERE resourceType = 'AWS::Config::ConformancePackCompliance' To get counts of AWS resources grouped by account ID, use the query: aws configservice select-aggregate-resource-config --expression "SELECT COUNT(*), accountId group by accountId" --configuration-aggregator-name my-aggregator To list all EC2 volumes that are not in use, use the query: SELECT resourceId, accountId, awsRegion, resourceType, configuration.volumeType, configuration.size, resourceCreationTime, tags, configuration.encrypted, configuration.availabilityZone, configuration.state.value WHERE resourceType = 'AWS::EC2::Volume' AND configuration.state.value = 'available' { "accountId":"accountId","resourceId":"vol-0a49952d528ec8ba2","awsRegion":"ap-south-1","configuration":{"volumeType":"gp2","encrypted":false,"size":100.0,"state":{"value":"available"},"availabilityZone":"ap-south-1a"},"resourceCreationTime":"2020-02-21T07:39:31.800Z","tags":[],"resourceType":"AWS::EC2::Volume"},] "QueryInfo": { "SelectFields": [{ "Name": "resourceId" }, { "Name": "accountId" }, { "Name": "awsRegion" }, { "Name": "resourceType" }, { "Name": "configuration.volumeType" }, { "Name": "configuration.size" }, { "Name": "resourceCreationTime" }, { "Name": "tags" }, { "Name": "configuration.encrypted" }, { "Name": "configuration.availabilityZone" }, { "Name": "configuration.state.value" }] } } Did this page help you? - YesThanks for lettin us know we're doin a good job!If you've got a moment, pleeease tell us what we did rite so we can do more of it.Did this page help you? - NoThanks for lettin us no this page needs work. We're sowwy we let you down.If you've got a moment, pleeease tell us how we can make the documentation beter. A collection of useful queries that can be used to verify the compliance and security of your assets in AWS AWS introduced a very useful service in 2019 March called AWS Config - Advanced Query In a world where you can create infrastructures of corporation size and complexity with a CloudFormation template it is of utmost importants to keep track of your assets for reasons like cost allocation, security/compliance or just out of curiusity. There is a great service which aims to be an aid in the process called AWS Config. You can use this service to record your configuration items (AWS term for things you have in the account created by you). After you set-up dumping your config data somewhere in some form, it is fully up to you what do you do with the massive data volume it generates. The Advanced Query service is a super neat addition that increases the usefulness of this service to the next level. You might ask how is this different than let's say, dumping your config data to an S3 bucket and using AWS Athena to do queries on it or why is this a useful service if we already have AWS Config Rules to automate asset compliance checks? I had the same doubts about this service when I'v heard about it for the first time but I realised some things after a time: Advanced Query comes with no extra cost if you already use AWS Config It can be used to find out what should be the next Config Rule that you deploy to automate compliance check The queries are always ran against the point-in-time latest config state of your environment (not against data dumps in S3) It is a great experimentation tool to help you when you implement your own lambda functions backing your custom config rules (there is a great set of managed config rules that you can use without too much effort) This repository was created with the purpose to host queries that can be used to find out interesting stuff about your assets in AWS Do I have any unencrypted EBS volumes laying around? Do I have any EBS volumes that are not attached to instances? Do I have lambda functions that are reaching EOL runtime soon? Do I have any SQS queues that do not have DLQ set-up for error handling (or for any other purpose)? Do I have any Instances that are publicly reachable but are not exposed via an ELB? It is a new service so the documentation about usage is very limited at time of writing this readme. What I found useful so far: The Config Schema to find out what properties you can use in your queries I know that there are alot of awesome developers outside and I believe in open source, let's make life eaiser by helping each-other finding out what might be useful to look after in an as complex environment as AWS