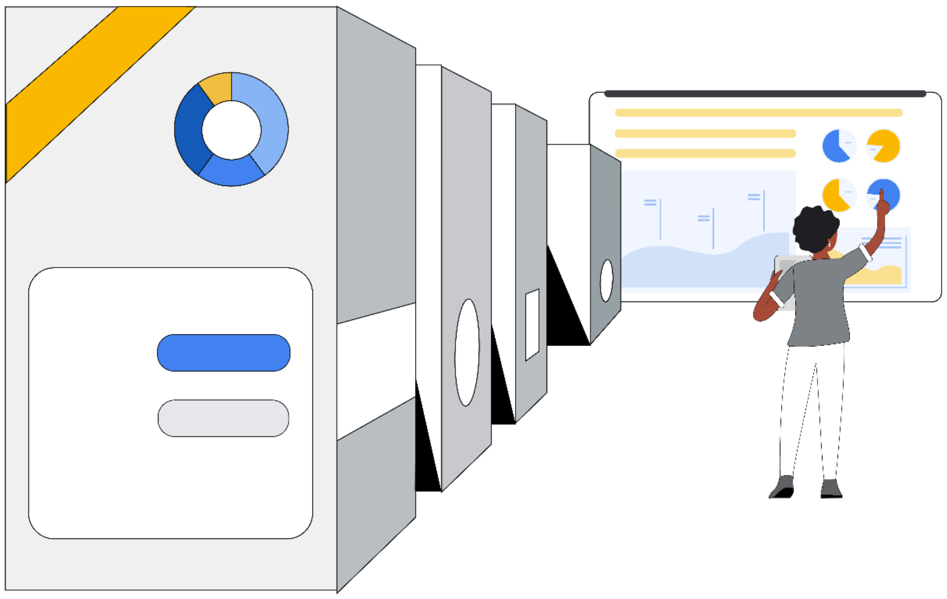
Google Cloud

BigQuery view and Looker template for FOCUS™ v1.0

Last updated: Aug 9, 2024



For more information visit **cloud.google.com**

Normalize data across Clouds

At Google Cloud, open standards are part of our DNA

As a founding member of the FinOps Foundation and a strong advocate for open standards, Google Cloud is proud to announce a new BigQuery View that leverages the recent FOCUS™ (GA) standard to help simplify cost management across clouds. The FOCUS™ standard is a living, breathing specification that is constantly being defined and improved by the Working Group consisting of FinOps practitioners, CSP leaders, Software as a Service (SaaS) providers, and more. The FOCUS™ specification v1.0 GA was launched in November 2023, helping to pave the way for efficient and transparent cloud cost management. If you'd like to read more or join the Working Group, here is a link to the FOCUS™ website: focus.finops.org.

BigQuery View for FOCUS™ v1.0 GA

Today, Google Cloud offers three Billing exports to BigQuery for cost and usage related data; Standard Billing Export, Detailed Billing Export, and Price Export. We are introducing a BigQuery view that transforms this data towards the FOCUS™v1.0 GA specified data attributes and metrics.

A BigQuery view is a virtual table that represents the results of a SQL query. We can provide a base query that maps Google Cloud data into the display names, format, and behavior of the FOCUS™ GA specified dimensions and metrics. BigQuery views are great because the queryable virtual table can only contain data from the tables and fields specified in the base query that defines the view. BigQuery views are virtual tables, so you incur no additional charges for data storage of this view, if you are already using Billing Export to BigQuery.

Note that **this View serves as directional guidance towards the v1.0 GA specification only**.

Step 1: Enable Billing Exports to BigQuery

The FOCUS™ BigQuery View acts as a virtual table that sits on top of your existing Google Cloud Billing data. To use this feature, **you need at least Detailed Billing Export and Price Exports enabled.** If you have these exports, skip this step.

Detailed Billing Export contains resource-level data (like Instance-level costs), and fields that act as keys to join with the Price Export. Price Export contains price metadata and contractual pricing information. Here is more information and schemas for the Detailed Billing Export and Price Export.

Follow these instructions to set up your billing exports to BigQuery. The BigQuery View uses a base SQL query, which we provide below, to map your cloud billing data into the FOCUS™ schema, presenting it in the standardized format. This allows you to query and analyze your data as if it were native to FOCUS™, making it easier to compare costs across different cloud providers.

Step 2: Create a BigQuery View

Permissions

Views are treated as table resources in BigQuery, so creating a view requires the same permissions as creating a table. You must also have permissions to query any tables that are referenced by the view's SQL query. To create a view, you need the `bigquery.tables.create` IAM permission. The `roles/bigquery.dataEditor` predefined IAM role includes the permissions that you need to create a view.

Additionally, if you have the `bigquery.datasets.create` permission, you can create views in the datasets that you create. To create a view for data that you don't own, you must have `bigquery.tables.getData` permission for that table.

Naming

When you create a view in BigQuery, the view name must be unique per dataset. See complete set of requirements here.

Create a View

You can create a view by composing a SQL query that is used to define the data accessible to the view. The SQL query must consist of a SELECT statement. Other statement types (such as DML statements) and multi-statement queries aren't allowed in view queries. Below are the steps to create a View using the Cloud Console. If you'd like to use command-line interface or API instead, follow these steps.

Copy the SQL query below. This is the base query for your FOCUS BigQuery View. You need to replace the green highlighted dataset paths below with your actual dataset paths for the Billing and Pricing exports.

Unset
```
WITH
 region_names AS (
 SELECT *
```

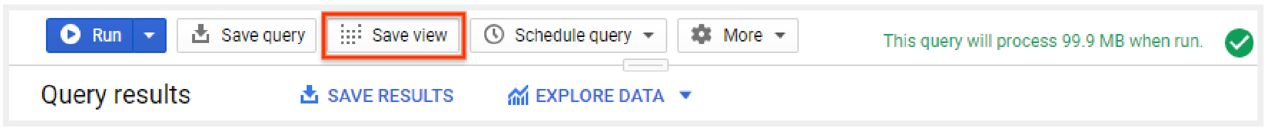
```
FROM UNNEST([
 STRUCT<id STRING, name STRING>("africa-south1", "Johannesburg"),
 ("asia-east1", "Taiwan"),
 ("asia-east2", "Hong Kong"),
 ("asia-northeast1", "Tokyo"),
 ("asia-northeast2", "Osaka"),
 ("asia-northeast3", "Seoul"),
 ("asia-southeast1", "Singapore"),
 ("australia-southeast1", "Sydney"),
 ("australia-southeast2", "Melbourne"),
 ("europe-central2", "Warsaw"),
 ("europe-north1", "Finland"),
 ("europe-southwest1", "Madrid"),
 ("europe-west1", "Belgium"),
 ("europe-west2", "London"),
 ("europe-west3", "Frankfurt"),
 ("europe-west4", "Netherlands"),
 ("europe-west6", "Zurich"),
 ("europe-west8", "Milan"),
 ("europe-west9", "Paris"),
 ("europe-west10", "Berlin"),
 ("europe-west12", "Turin"),
 ("asia-south1", "Mumbai"),
 ("asia-south2", "Delhi"),
 ("asia-southeast2", "Jakarta"),
 ("me-central1", "Doha"),
 ("me-central2", "Dammam"),
 ("me-west1", "Tel Aviv"),
 ("northamerica-northeast1", "Montréal"),
 ("northamerica-northeast2", "Toronto"),
 ("us-central1", "Iowa"),
 ("us-east1", "South Carolina"),
 ("us-east4", "Northern Virginia"),
 ("us-east5", "Columbus"),
 ("us-south1", "Dallas"),
 ("us-west1", "Oregon"),
 ("us-west2", "Los Angeles"),
 ("us-west3", "Salt Lake City"),
 ("us-west4", "Las Vegas"),
 ("southamerica-east1", "São Paulo"),
 ("southamerica-west1", "Santiago")
])
),
usage_cost_data AS (
SELECT
 *,
 (
 SELECT
  AS STRUCT type,
  id,
  full_name
 FROM
  UNNEST(credits)
 WHERE
  type IN UNNEST(["COMMITTED_USAGE_DISCOUNT", "COMMITTED_USAGE_DISCOUNT_DOLLAR_BASE"])
 LIMIT
  1) AS cud,
 ARRAY( ( (
   SELECT
    AS STRUCT key AS key,
    value AS value,
    "label" AS x_type,
    FALSE AS x_inherited,
        "n/a" AS x_namespace
      FROM
        UNNEST(labels))
    UNION ALL (
      SELECT
        AS STRUCT key AS key,
        value AS value,
        "system_label" AS x_type,
        FALSE AS x_inherited,
        "n/a" AS x_namespace
      FROM
        UNNEST(system_labels))
    UNION ALL (
      SELECT
        AS STRUCT key AS key,
        value AS value,
        "project_label" AS x_type,
        TRUE AS x_inherited,
        "n/a" AS x_namespace
      FROM
        UNNEST(project.labels))
    UNION ALL (
      SELECT
        AS STRUCT key AS key,
        value AS value,
        "tag" AS x_type,
        inherited AS x_inherited,
        namespace AS x_namespace
      FROM
        UNNEST(tags) ) )) AS focus_tags,
  FROM
   `project.dataset.gcp_billing_export_resource_v1_account`),
   -- TODO - replace with your detailed usage export table path
  prices AS (
  SELECT
   *,
   flattened_prices
  FROM
   `project.dataset.cloud_pricing_export`,
   -- TODO - replace with your pricing export table path
   UNNEST(list_price.tiered_rates) AS flattened_prices
  WHERE
   DATE(export_time) = 'YYYY-MM-DD')
   -- TODO - replace with a date after you enabled pricing export to use pricing data as of this date
SELECT
  usage_cost_data.location.zone AS AvailabilityZone,
  CAST(usage_cost_data.cost AS NUMERIC) + IFNULL((
   SELECT
    SUM(CAST(c.amount AS NUMERIC))
   FROM
    UNNEST(usage_cost_data.credits) AS c), 0) AS BilledCost,
  "TODO - replace with the billing account ID in your detailed usage export table name" AS BillingAccountId,
  usage_cost_data.currency AS BillingCurrency,
  PARSE_TIMESTAMP("%Y%m", invoice.month, "America/Los_Angeles") AS BillingPeriodStart,
  TIMESTAMP(DATE_SUB(DATE_ADD(PARSE_DATE("%Y%m", invoice.month), INTERVAL 1 MONTH), INTERVAL 1 DAY),
"America/Los_Angeles") AS BillingPeriodEnd,
  CASE LOWER(cost_type)
   WHEN "regular" THEN "usage"
   WHEN "tax" THEN "tax"
   WHEN "rounding_error" then "adjustment"
   WHEN "adjustment" then "adjustment"
   ELSE "error"
   END AS ChargeCategory,
  IF(
   COALESCE(
      usage_cost_data.adjustment_info.id,
      usage_cost_data.adjustment_info.description,
      usage_cost_data.adjustment_info.type,
      usage_cost_data.adjustment_info.mode)
    IS NOT NULL,
   "correction",
   NULL) AS ChargeClass,
  usage_cost_data.sku.description AS ChargeDescription,
  usage_cost_data.usage_start_time AS ChargePeriodStart,
  usage_cost_data.usage_end_time AS ChargePeriodEnd,
  CASE usage_cost_data.cud.type
   WHEN "COMMITTED_USAGE_DISCOUNT_DOLLAR_BASE" THEN "Spend"
   WHEN "COMMITTED_USAGE_DISCOUNT" THEN "Usage"
   END AS CommitmentDiscountCategory,
  usage_cost_data.subscription.instance_id AS CommitmentDiscountId,
  usage_cost_data.cud.full_name AS CommitmentDiscountName,
  IF(usage_cost_data.cost_type = "regular", CAST(usage_cost_data.usage.amount AS NUMERIC), NULL) AS
ConsumedQuantity,
  IF(usage_cost_data.cost_type = "regular", usage_cost_data.usage.unit, NULL) AS ConsumedUnit,
  CAST(usage_cost_data.cost AS NUMERIC) AS ContractedCost,
  CAST(usage_cost_data.price.effective_price AS NUMERIC) AS ContractedUnitPrice,
  CAST(usage_cost_data.cost AS NUMERIC) + IFNULL((
   SELECT
    SUM(CAST(c.amount AS NUMERIC))
   FROM
    UNNEST(usage_cost_data.credits) AS c), 0) AS EffectiveCost,
  CAST(usage_cost_data.cost_at_list AS NUMERIC) AS ListCost,

  IF(usage_cost_data.cost_type = "regular", CAST(prices.flattened_prices.account_currency_amount AS NUMERIC), NULL
) AS ListUnitPrice,
  IF(
   usage_cost_data.cost_type = "regular",
   IF(
    LOWER(usage_cost_data.sku.description) LIKE "commitment%" OR usage_cost_data.cud IS NOT NULL,
    "committed",
    "standard"),
   null) AS PricingCategory,
  IF(usage_cost_data.cost_type = "regular", usage_cost_data.price.pricing_unit_quantity, NULL) AS PricingQuantity,
  IF(usage_cost_data.cost_type = "regular", usage_cost_data.price.unit, NULL) AS PricingUnit,
  "Google Cloud" AS ProviderName,
  IF(usage_cost_data.transaction_type = "GOOGLE", "Google Cloud", usage_cost_data.seller_name) AS PublisherName,
  usage_cost_data.location.region AS RegionId,
  (
  SELECT
   name
  FROM
   region_names
  WHERE
   id = usage_cost_data.location.region) AS RegionName,
  usage_cost_data.resource.global_name AS ResourceId,
  usage_cost_data.resource.name AS ResourceName,
  IF(
   STARTS_WITH( usage_cost_data.resource.global_name, '//'),
   REGEXP_REPLACE(
    usage_cost_data.resource.global_name,

'(//)|(googleapis.com/)|(projects/[^/]+/)|(project_commitments/[^/]+/)|(locations/[^/]+/)|(regions/[^/]+/)|(zones/[^/]+/)|(globa
l/)|(/[^/]+)',
    ''),
   NULL) AS ResourceType,
  prices.product_taxonomy AS ServiceCategory,
  usage_cost_data.service.description AS ServiceName,
  IF(usage_cost_data.cost_type = "regular", usage_cost_data.sku.id, NULL) AS SkuId,
  IF(
   usage_cost_data.cost_type = "regular",
   CONCAT(
    "Billing Account ID:", usage_cost_data.billing_account_id,
    ", SKU ID: ", usage_cost_data.sku.id,
    ", Price Tier Start Amount: ", price.tier_start_amount),
   NULL) AS SkuPriceId,
  usage_cost_data.billing_account_id as SubAccountId,
  usage_cost_data.focus_tags AS Tags,
  ARRAY((
   SELECT
    AS STRUCT name AS Name,
    CAST(amount AS NUMERIC) AS Amount,
    full_name AS FullName,
    id AS Id,
    type AS Type
   FROM
    UNNEST(usage_cost_data.credits))) AS x_Credits,
  usage_cost_data.cost_type AS x_CostType,
  CAST(usage_cost_data.currency_conversion_rate AS NUMERIC) AS x_CurrencyConversionRate,
  usage_cost_data.export_time AS x_ExportTime,
  usage_cost_data.location.location AS x_Location,
  (
  SELECT
   AS STRUCT usage_cost_data.project.id,
   usage_cost_data.project.number,
   usage_cost_data.project.name,
   usage_cost_data.project.ancestry_numbers,
   usage_cost_data.project.ancestors) AS x_Project,
  usage_cost_data.service.id AS x_ServiceId
FROM
  usage_cost_data
LEFT JOIN
  prices
ON
  usage_cost_data.sku.id = prices.sku.id
  AND usage_cost_data.price.tier_start_amount = prices.flattened_prices.start_usage_amount;
```

This query was last updated on March 19, 2024

Paste the SQL query into the BigQuery editor and run the query. After running a query, click the Save view button above the Query Results window to save the query as a view.



Save a View

1. For **Project name**, select a project to store the view.
2. For **Dataset name**, choose a dataset to store the view. The dataset that contains your view and the dataset that contains the tables referenced by the view must be in the same location.

3. For **Table name**, enter the name of the view.
4. Click **Save**.

Note: When you create a view using Google Cloud console, you cannot add a label, description, or expiration time. You can add these optional properties when you create a view using the API or bq command-line tool. After you create a view using the Google Cloud console, you can add an expiration, description, and labels. For more information, see Updating views.

Step 3: Query the BigQuery view

After you create the view, you can query it like you query a table.

Limitations

The FOCUS spec has five possible values for ChargeCategory - “usage”, “purchase”, “tax”, “credit”, and “adjustment”. In GCP, we currently don;t distinguish between usage and purchase charges, and credits appear on the line items they’re applied to. Thus, GCP’s FOCUS view only uses “usage”, “tax”, and “adjustment” as possible values, and purchases & credits appear alongside usage. Similarly, all the cost columns in the GCP FOCUS view will include charges that the FOCUS spec would consider to be purchases (even BilledCost & ContractedCost), as we classify them under “usage”.

For commitments, GCP amortizes most commitment charges by *time*. While the commitment charges may appear on a separate SKU from their corresponding usage charges, they’re distributed over time to appear alongside those usage charges. While we believe this supports the FinOps use case of amortizing commitment fees over usage, it means our EffectiveCost column doesn’t support the FOCUS semantic of “moving” cost from the commitment fee SKU to the corresponding usage - we calculate EffectiveCost simply as the cost + credit associated with each line item.

BillingAccountId will be the same ID in the table name, while SubAccountId is the billing account that the usage was associated with. In the case of standard and resold accounts, these two accounts will be the same. For reseller accounts, the BillingAccountId is the parent/reseller account, and SubAccountId is the resold account.

Other notes:

- GCP’s FOCUS view doesn’t support the “dynamic” possible value for PricingCategory
- Resource information (ID, name, and type) is available for most, but not all usage
- ServiceCategory contains the GCP product taxonomy, which doesn’t precisely match the FOCUS set of possible values which is to be defined with FOCUS in v1.1 as a next step across all providers

- ChargeClass has a value of “correction” in the GCP FOCUS view for all billing adjustments, not only adjustments to charges from previous invoices
- CommitmentDiscountId is not always present since Subscription Instance ID today supports CUD fee breakdown; there are more features to come.

Finally, as with GCP’s FOCUS preview BQ view, fields that come from pricing export use the price as of the date manually chosen in the SQL query rather than the price as of the date of usage. This currently affects only ListUnitPrice & ServiceCategory, which are not expected to change frequently.

Changelog

A detailed changelog of the query is being kept beneath the SQL query above. We will help to provide a summary of the changes as they occur in this section.

2024/08/09 (v1.0 GA)

- Updated Looker Github Repo Link

2024/06/20 (v1.0 GA)

- Added:
  - BilledCost
  - ChargeCategory
  - ChargeClass
  - ContractedCost
  - ContractedUnitPrice
  - EffectiveCost
  - PricingCategory
  - RegionName
  - ResourceType
  - SubAccountId
- Changed:
  - Updated BillingAccountId to be filled in by the user; the view now uses the GCP billing_account_id for SubAccountId instead
  - Updated BillingPeriodStart & BillingPeriodEnd to use a USA Pacific time zone timestamp to better reflect that GCP uses pacific time to determine month end
  - Updated CommitmentDiscountId to use GCP’s new subscription ID field, and removed currently redundant CommitmentDiscountType field
  - Renamed UsageQuantity & UsageUnit to ConsumedQuantity & ConsumedUnit
  - Set ListUnitPrice, PricingCategory, PricingQuantity, PricingUnit, SkuId, and SkuPriceId to be null for non-usage line items (i.e. tax, adjustment, rounding error)

    - Updated ProviderName to use "Google Cloud" and PublisherName to use GCP's seller_name field per FOCUS spec clarification
    - Renamed Region to RegionId
    - Added Billing Account Id to the SkuPriceId string to ensure it's unique across accounts
    - Minor cosmetic changes to the SQL query
- Non-FOCUS fields:
    - Renamed "gc_" prefixed fields to use an "x_" prefix to align with the FOCUS spec
    - Renamed Tags subfields other than "key" and "value" to use the "x_" prefix
    - Added additional useful non-FOCUS fields:, x_CurrencyConversionRate, x_ExportTime, x_Location, x_Project, and x_ServiceId
    - Removed x_Cost field as it's now redundant with ContractedCost

2024/03/19 (v1.0 Preview)

- Renamed UsageAmount to UsageQuantity to align with the FOCUS spec

2024/01/29 (v1.0 Preview)

- Minor cosmetic changes to the SQL query

Looker Template for FOCUS™ v1.0 GA

Steps to enable Looker template

1. **Prepare Your Looker Environment:**
    - Ensure you have permissions to create a Looker project dedicated to this analysis.
    - Have the following information on hand:
        - The name of your database connection in Looker.
        - The date you want your analysis to start from.
        - The names of the pricing and billing tables within your database.
2. **Download and Adapt the Repository:**
    - Fork the following public github repo to another github repo: https://github.com/looker-open-source/google_cloud_focus
        - For instructions on how to fork a github repo, use the Github documentation.
    - Create a new LookML Project and select "Blank Project".

On your new LookML Project, Configure Git using SSH or HTTPS using the new repo you created in step 1.

    - Locate the manifest file (named manifest.lkml) within the downloaded repository.

   - Open the manifest file and modify the following constants to match your specific environment:
     - connection: Replace with the name of your Looker database connection.
     - date: Replace with the date you wish to start your analysis from.
     - pricing_table: Replace with the name of your pricing table.
     - billing_table: Replace with the name of your detailed billing table.
3. **Test Your Configuration:**
   - Navigate to the "Focus" Explore in Looker and verify that the data is loading correctly and as expected.
4. **Deploy to Production:**
   - Commit your modified LookML files.
   - Deploy the updated LookML model to your production environment.
5. **Customize and Share:**
   - Open the "Focus" dashboard within Looker by navigating to the LookML Dashboards folder.
   - Make a copy of the dashboard.
   - Customize the copied dashboard to your specific needs and preferences.
   - Share the customized dashboard with your team members or other stakeholders.

Important Notes:

- **Temporary Tables:** Unlike BigQuery Views, Looker utilizes temporary tables. The provided LookML code will create and manage these tables automatically, so you won't need to create them manually.
- **Data Validation:** Always double-check your data in Looker to ensure it aligns with your expectations after making changes.

FAQs

1. **I have feedback or suggested improvements. Where can I submit feedback?**

   Please email google-cloud-focus@google.com and we will help to triage as we can and see fit.

2. **Are you keeping this guide up-to-date?**

   We're continuously working to keep this SQL query and guide aligned with the evolving FOCUS specification, as we can.