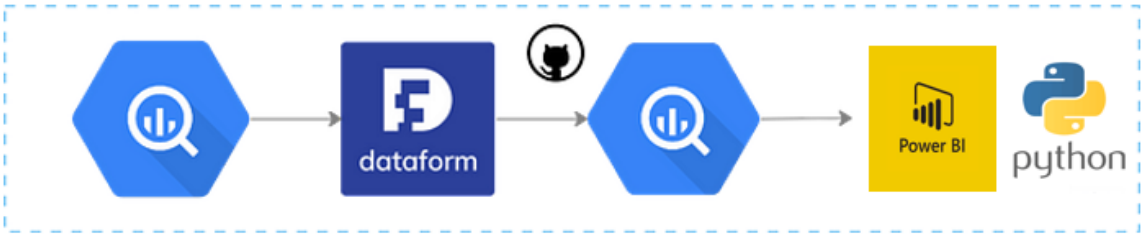
3.3. Conhecendo o Dataform

O Dataform é uma ferramenta de gerenciamento de dados projetada para ajudar equipes de análise de dados a colaborar, organizar e gerenciar seus processos ETL (Extract, Transform, Load) em um data warehouse.



Na Mottu vamos usar o Dataform para ler os dados brutos e volumosos, processando e gerando tabelas prontas e com dados agrupados para serem utilizadas nas ferramentas de analise de dado. Centralizando as regras de negócio e conceito dos dados para todas as análises.

Dataform em um Data Warehouse:

1. **Organização de Projetos ETL:**
    - **O que é:** O Dataform permite que as equipes organizem seus projetos ETL em um formato mais compreensível e controlado. Ele fornece uma camada de abstração sobre o SQL, permitindo que você estruture suas transformações e consultas de maneira mais simples e modular.
    - **Como é utilizado:** Equipes podem criar, versionar e gerenciar seus scripts SQL em projetos, facilitando a colaboração, rastreabilidade e auditoria. Isso é especialmente útil em ambientes complexos de data warehouse.
2. **Integração com o GitHub e Controle de Versão:**
    - **O que é:** Dataform oferece uma integração nativa com o GitHub, uma plataforma de controle de versão. Isso significa que os scripts SQL, projetos e transformações criados no Dataform podem ser versionados diretamente no GitHub. Isso significa que você pode controlar as alterações feitas em seus scripts ao longo do tempo, facilitando o rastreamento de quem fez o quê e quando. Isso é crucial para a integridade e a qualidade dos dados,

proporcionando uma maneira eficaz de gerenciar alterações, reverter a versões anteriores e colaborar de maneira mais eficiente.

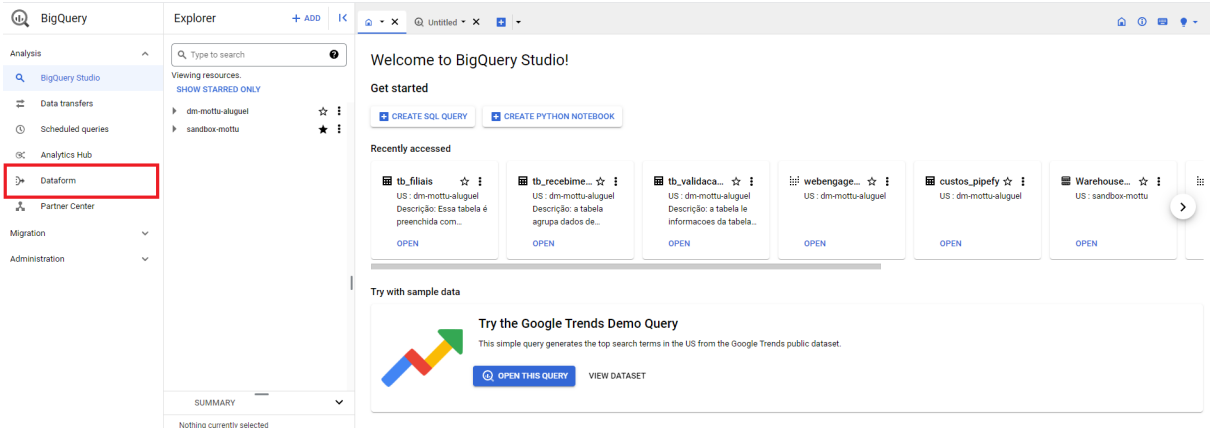
- **Como é utilizado:** A integração com o GitHub permite que as equipes aproveitem as funcionalidades robustas de controle de versão, como pull requests, branchs e histórico de alterações. Isso facilita a colaboração entre membros da equipe, garantindo que todas as alterações sejam rastreadas e revisadas. Além de permitir controle sobre as regras de aprovação das alterações sugeridas por membros da equipe.

3. **Agendamento e Orquestração:**

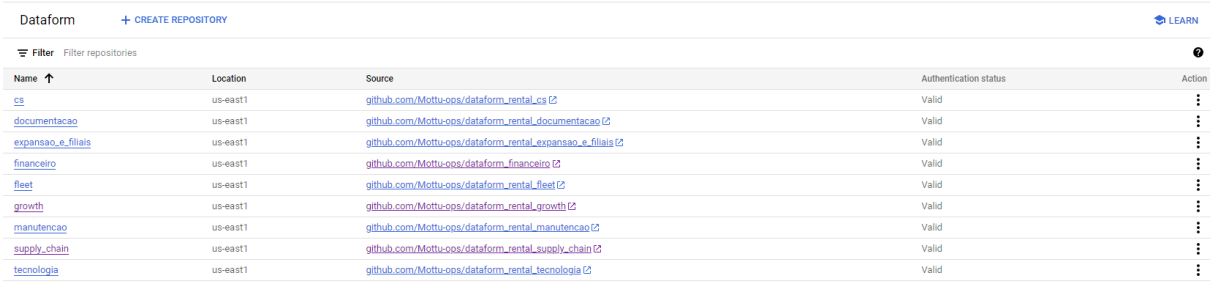
- **O que é:** Dataform oferece funcionalidades de agendamento e orquestração para automatizar a execução de scripts ETL em horários específicos. Controlando a dependência entre entre rotinas e gerando logs de cada execução dos jobs.
- **Como é utilizado:** Isso é essencial para manter os dados atualizados e garantir que as transformações sejam executadas de acordo com uma programação pré-definida.

Como acessar o Dataform:

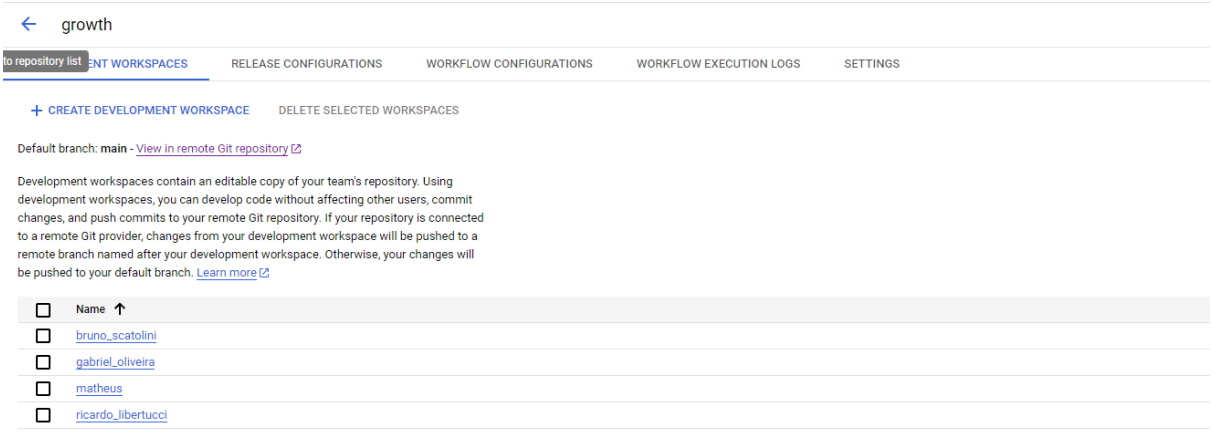
O Dataform está integrado ao BigQuery e pode ser acessado pelo mesmo console.



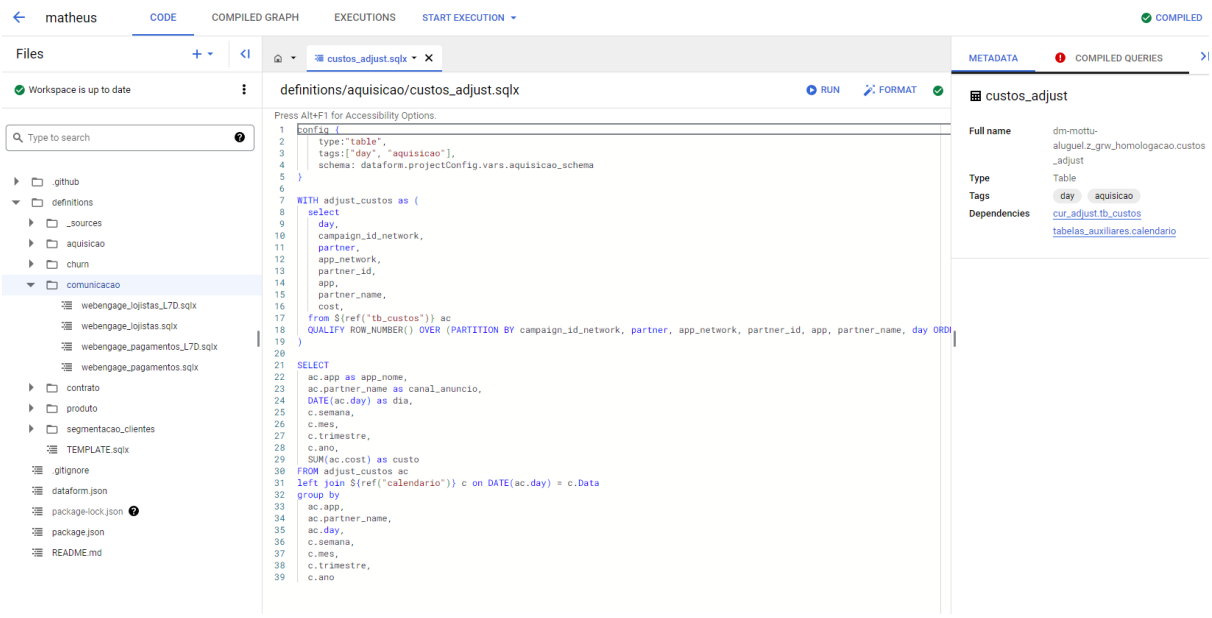
Dentro do Dataform você verá os repositórios que armazenam os códigos SQL de cada departamento

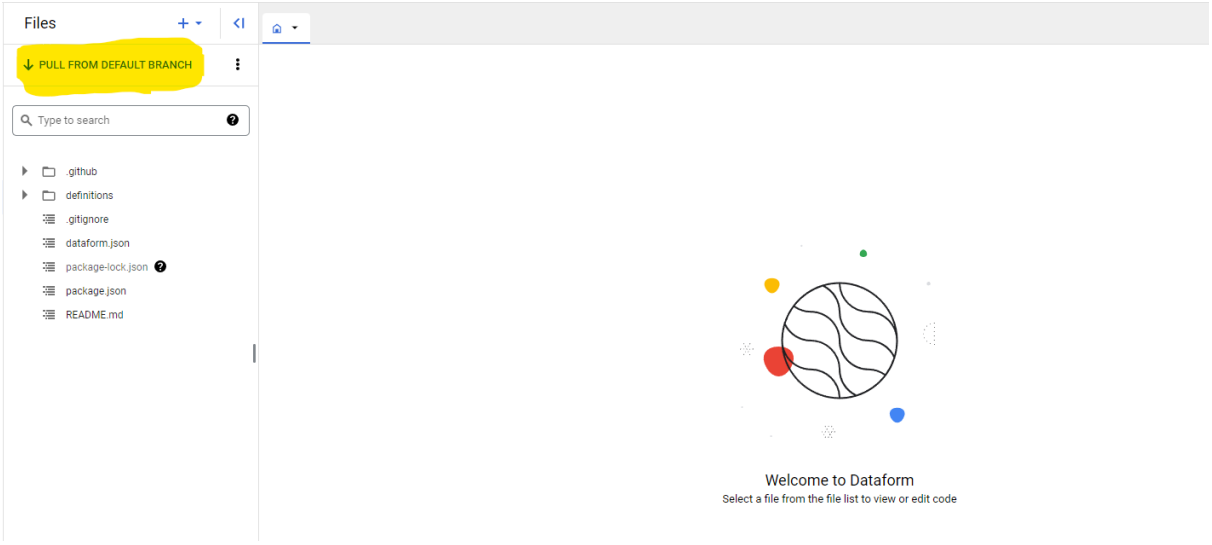


E dentro de cada departamento há um ambiente de trabalho (branch) para cada um dos Data Owners.



No seu ambiente de trabalho você terá acesso à uma cópia dos códigos que estão no ambiente de produção do repositório da sua área. Caso o seu repositório esteja desatualizado aparecerá uma opção para você efetuar o PULL (sincronização) com a versão em produção do seu repositório.





Ao criar um código no Dataform é usada a extensão .sqlx, que é uma mistura de SQL com JavaScrip. As configurações e declarações são feitas em JavaScript que passam os parâmetros para criar as rotinas que executaram o código SQL.

O formato SQLX do dataform possui 2 diferenças significativas versus o console do BigQuery em que é utilizado um SQL comum:

1. Necessidade de declaração explicita das fontes de dados utilizadas no select.
2. Necessidade de passar o JSON de configuração da rotina a ser executada.

Declaração das fontes de dados utilizadas no select.

A declaração das fontes de dados permite que o dataform faça o mapeamento as dependências de cada um dos códigos e saiba planejar a ordem de execução correta das rotinas.

1. No BigQuery faríamos o select desta forma:

```
    SELECT DISTINCT UserId, UserName, Orderid
FROM `sandbox-mottu.cur_supply_chain.OrderEvent`
WHERE UserName IS NOT NULL
```

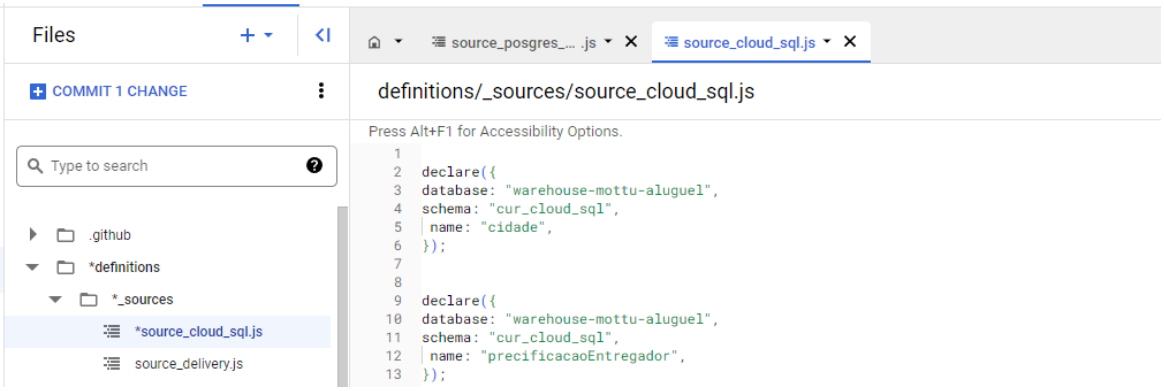
2. Enquanto que no dataform o select é feito assim:

```
SELECT DISTINCT UserId, UserName, Orderid
    FROM ${ref("OrderEvent")}
    WHERE UserName IS NOT NULL
```

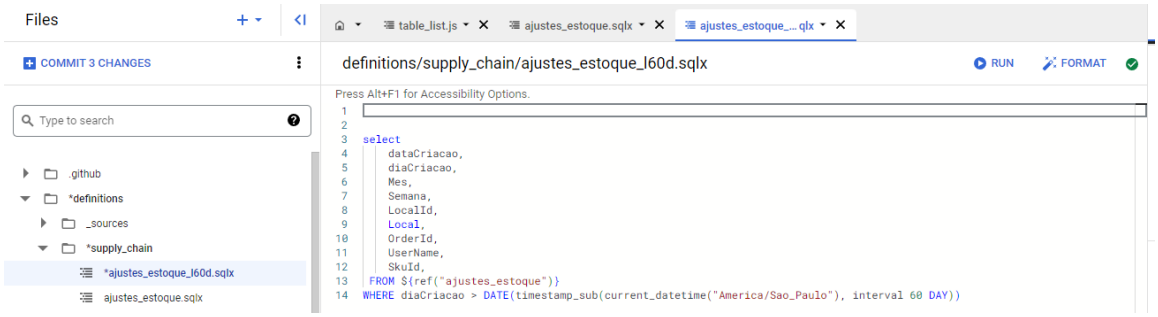
As referências do Dataform podem ser feitas em objetos externos (tabelas de do data warehouse) ou internos ao repositório (tabelas criadas no próprio repositório).

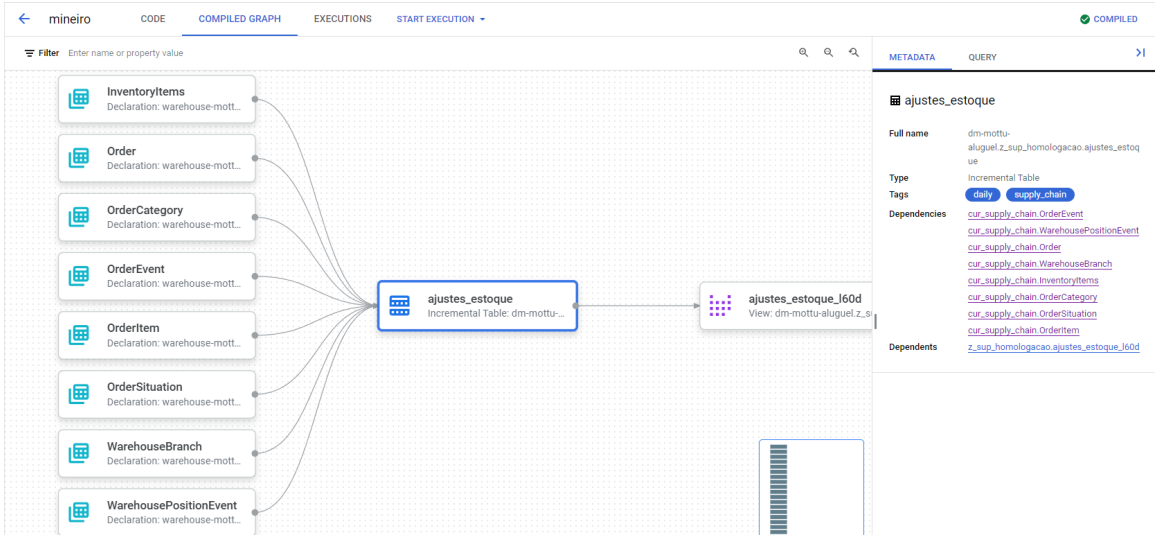
- Para fazer uma referencia a uma tabela externa ao repositório é necessário adicionar a declaração em um arquivo JavaScript (js) na pasta source como no exemplo abaixo:

```
declare({
database: "warehouse-mottu-aluguel",
schema: "cur_cloud_sql",
 name: "cidade",
});
```



- Já para referencias a objetos criados dentro do dataform não é necessário adicionar no arquivo (js) basta apenas referenciar o nome do arquivo sqlx. Dessa forma o Dataform entende que primeiro precisa processar as instruções do arquivo “ajustes_estoque.sqlx” para depois processar o código do arquivo “ajustes_estoque_l60d.sqlx”.





Configuração do job via JSON “config”.

O bloco “config” abstrai grande parte da complexidade de rotinas de ETL.

Quando você simplesmente declara uma view como no exemplo abaixo o dataform executa uma procedure de banco bem mais complexa, de forma a garantir a execução correta da rotina.

```
config {
  type:"view",
  tags:["day","supply_chain"],
  schema:  dataform.projectConfig.vars.supply_chain_schema
}
```

Na execução o Dataform transforma essa simples configuração em algo bem mais complexo:

```
BEGIN
  CREATE SCHEMA IF NOT EXISTS `dm-mottu-aluguel.sup_supply_chair
EXCEPTION WHEN ERROR THEN
  IF NOT CONTAINS_SUBSTR(@@error.message, "already exists: datas
    NOT CONTAINS_SUBSTR(@@error.message, "too many dataset metad
    NOT CONTAINS_SUBSTR(@@error.message, "User does not have big
  THEN
    RAISE USING MESSAGE = @@error.message;
  END IF;
END;
    BEGIN
      DECLARE dataform_table_type DEFAULT (
  SELECT ANY_VALUE(table_type)
  FROM `dm-mottu-aluguel.sup_supply_chain.INFORMATION_SCHEMA.TAB
  WHERE table_name = 'ajustes_estoque_l60d'
);
      IF dataform_table_type IS NOT NULL AND dataform_table_type
  IF dataform_table_type = 'BASE TABLE' THEN
    DROP TABLE IF EXISTS `dm-mottu-aluguel.sup_supply_chain.ajus
```

```
  ELSEIF dataform_table_type = "VIEW" THEN
    DROP VIEW IF EXISTS `dm-mottu-aluguel.sup_supply_chain.ajust
  ELSEIF dataform_table_type = 'MATERIALIZED VIEW' THEN
    DROP MATERIALIZED VIEW IF EXISTS `dm-mottu-aluguel.sup_suppl
  END IF;
END IF;
      BEGIN

            CREATE OR REPLACE VIEW `dm-mottu-aluguel.sup_supply_
```

Possíveis parâmetros do bloco “config”:

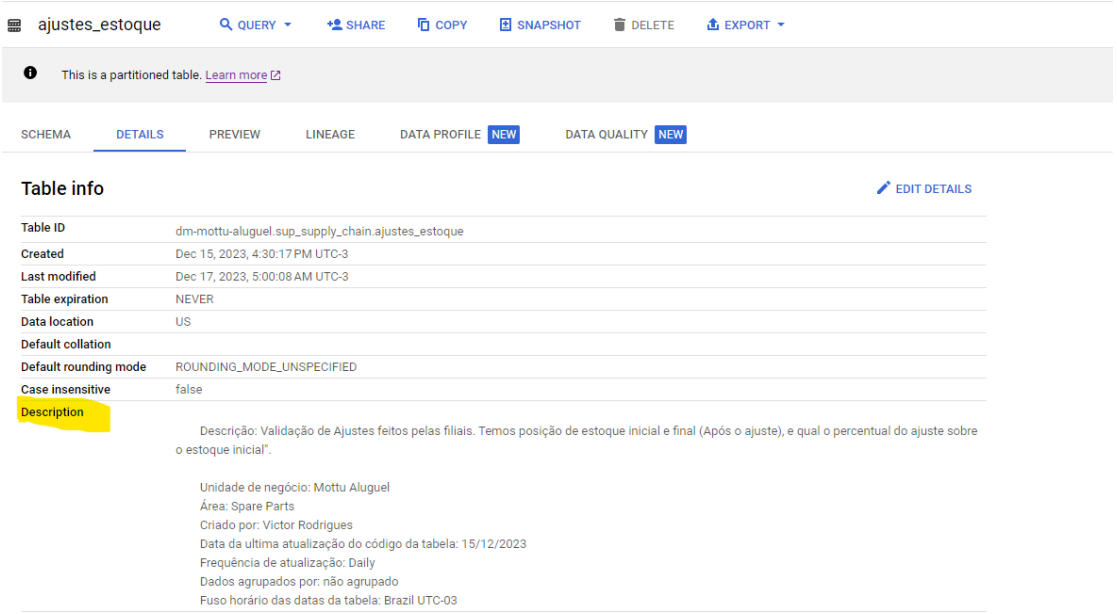
```
config {
//OBRIGATÓRIO
    type: "table",
    tags: ["daily", "supply_chain"],
    schema:  dataform.projectConfig.vars.supply_chain_schema,
    disabled: false,
    description: `
        Descrição:
                    Validação de Ajustes feitos pelas filiais.
                    Temos posição de estoque inicial e final (Ap
                    e qual o percentual do ajuste sobre o estoqu

        Unidade de negócio: Mottu Aluguel
        Área: Spare Parts
        Criado por: Victor Rodrigues
        Data da ultima atualização do código da tabela: 15/12/26
        Frequência de atualização: Daily
        Dados agrupados por: não agrupado
        Fuso horário das datas da tabela: Brazil UTC-03 `,

//OPCIONAL
    bigquery: {
      partitionBy: "diaCriacao",
      requirePartitionFilter: true,
    },
```

```
        protected: false
}
```

1. **Type:** Define o tipo de rotina que o dataform executará
    a. **view:** Deve ser usado apenas em rotinas de consumo de dados de baixa frequência <1x dia. Também pode ser usado para consultas leves para atender demanda de atualização de hora em hora. Porém não deve ser usado para consultas grandes de hora em hora devido ao custo desse processamento.
    b. **table:** Usado para relatórios de frequência diária e deveria ser o type padrão a ser utilizado.
    c. **incremental:** Usado para rotinas com alto volume de processamento.
2. **tags:** Define a frequência de atualização e o assunto do arquivo, na execução das rotinas é possível executar todas as rotinas que possuem a mesma tag. Na Mottu usaremos o padrão “daily” para agendar rotinas diárias e a tag com o nome do Dataset de destino, caso seja necessário reprocessar todo o dataset em algum momento.
3. **schema:** Define o dataset (ou seja, o schema) de destino em que será criado o objeto definido no código. Na Mottu usaremos o seguinte padrão:
    a. dataform.projectConfig.vars.”nome_do_dataset_sem_prefixo_da_area”_schema
4. **disabled:** Indica se a rotina está ativa ou inativa, sendo true para inativa e false para ativa.
5. **description: Adiciona a descrição/documentação do objeto criado no BigQuery**



6. **bigquery:** Determina parâmetros adicionais para criação de tabela permitindo definir a criação de uma tabela particionada.
    a. **partitionby:** Campo de referencia para o particionamento da tabela criada que ser o campo referencia de data para filtro da tabela ex: criacaoData.
    b. **requirePartitionFilter:** Torna obrigatório o uso do filtro de partição em consultas na tabela. Ideal para casos de tabelas grandes em que um select * errado pode custar caro.
7. **protected:** Em casos mais raros em que o dado é perecível, isto é, não possuímos backup. Esse parâmetro se torna importante pois evita que a tabela seja sobrescrita/deletada erroneamente.