

Day2

データベースとデータの参照

技術部データ基盤チーム 財津大夏 / GMO PEPABO inc.

2022.07.13 データエンジニアリング研修 基礎編 Day2

GMOペパボ

カリキュラム目標と概要

- **Day1: 扱いやすいデータの集合の形を理解できる**
 - データを構造化するための知識の導入
- **Day2: 初歩的な SQL を使ってデータベースからデータを参照できる**
 - データを参照するために必要な基礎的な知識の導入
- **Day3: 複数テーブルのデータを組み合わせて参照できる**
 - リレーショナルデータベースからデータを参照するための知識の導入
- **Day4: データを要約・可視化して情報や知識を取り出すことができる**
 - データを実際の施策や判断に利用するために必要な知識の導入

➡ 各日のハンズオンを通して手を動かしながら知識の解釈を高める

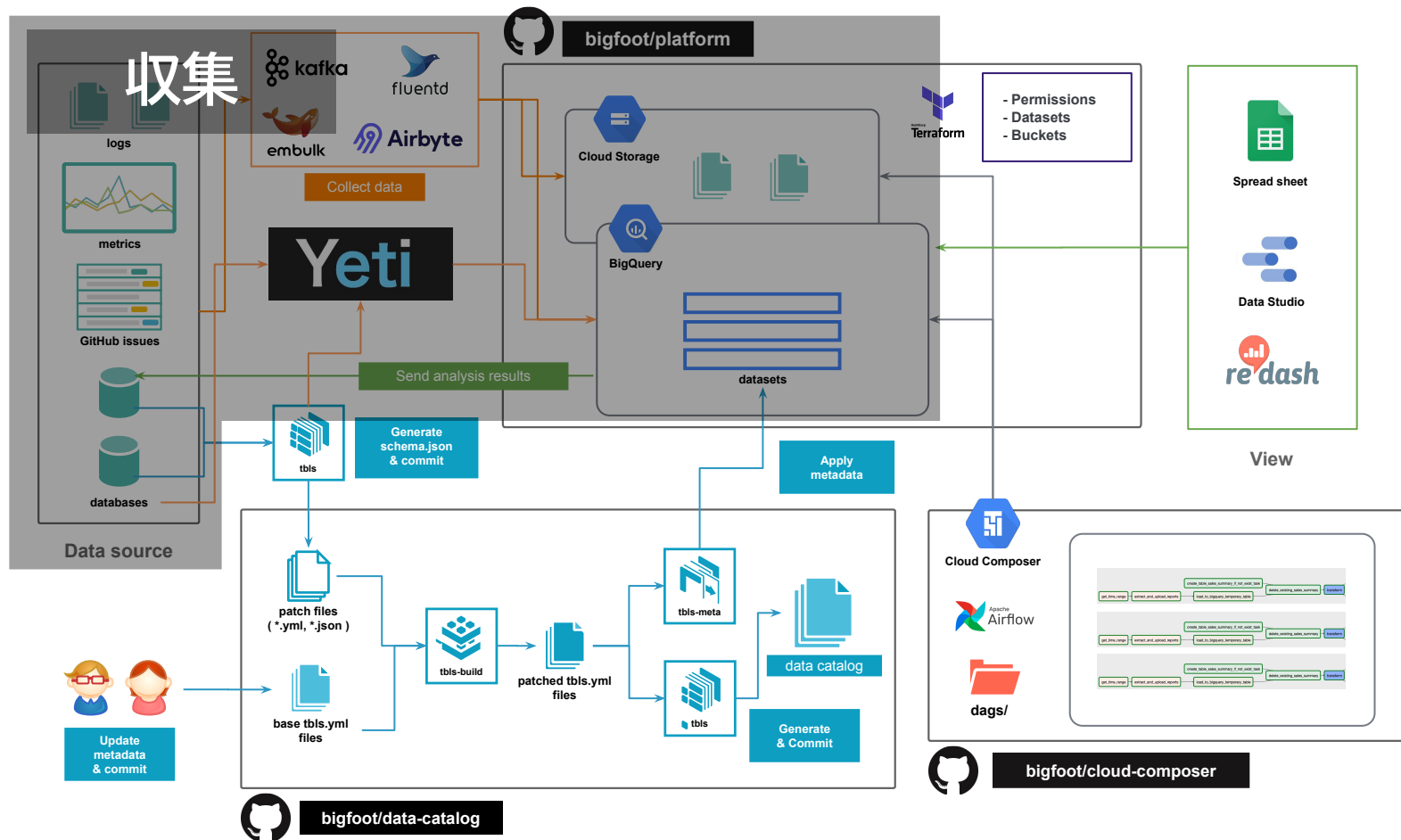
データベースとクライアント

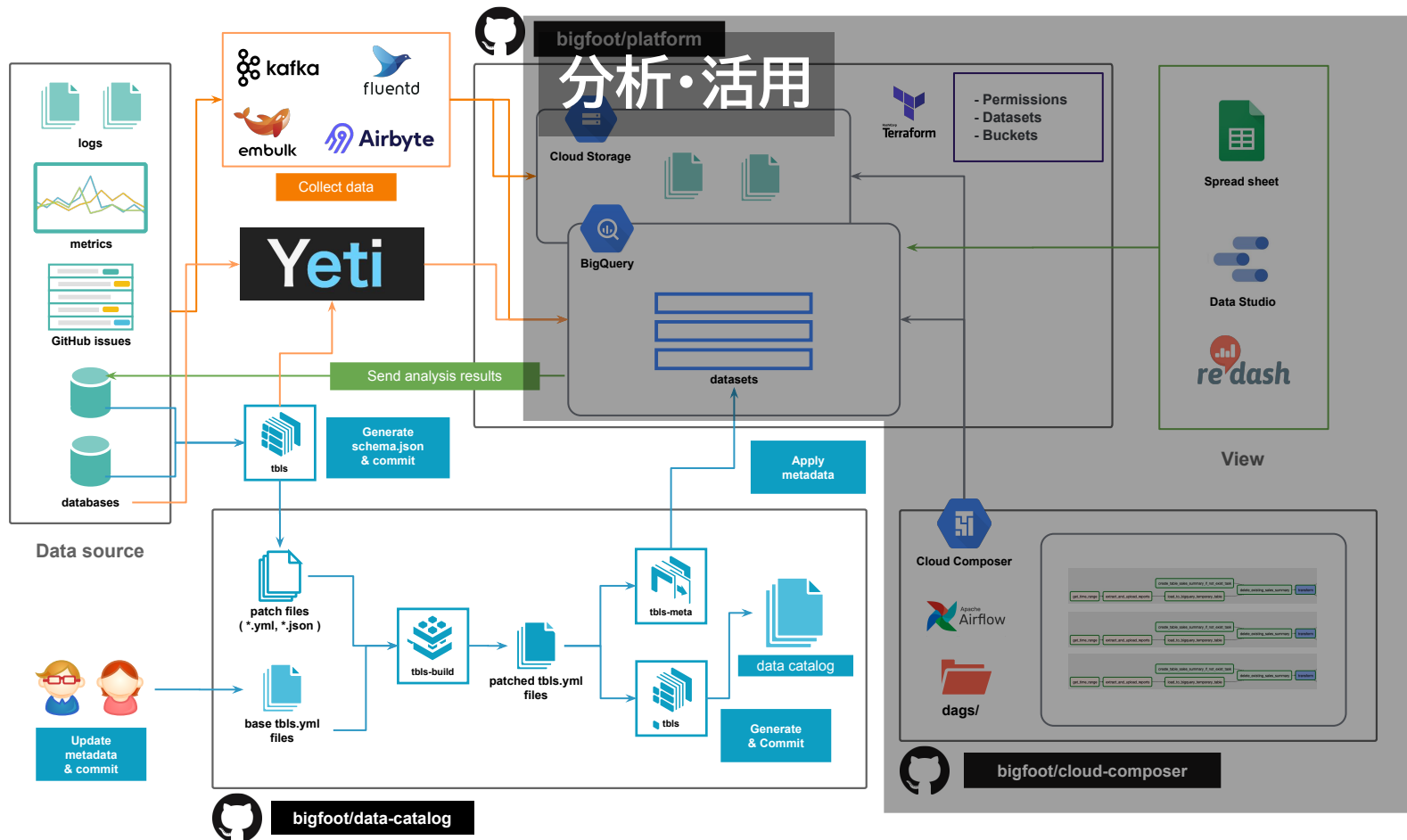
データベースとは

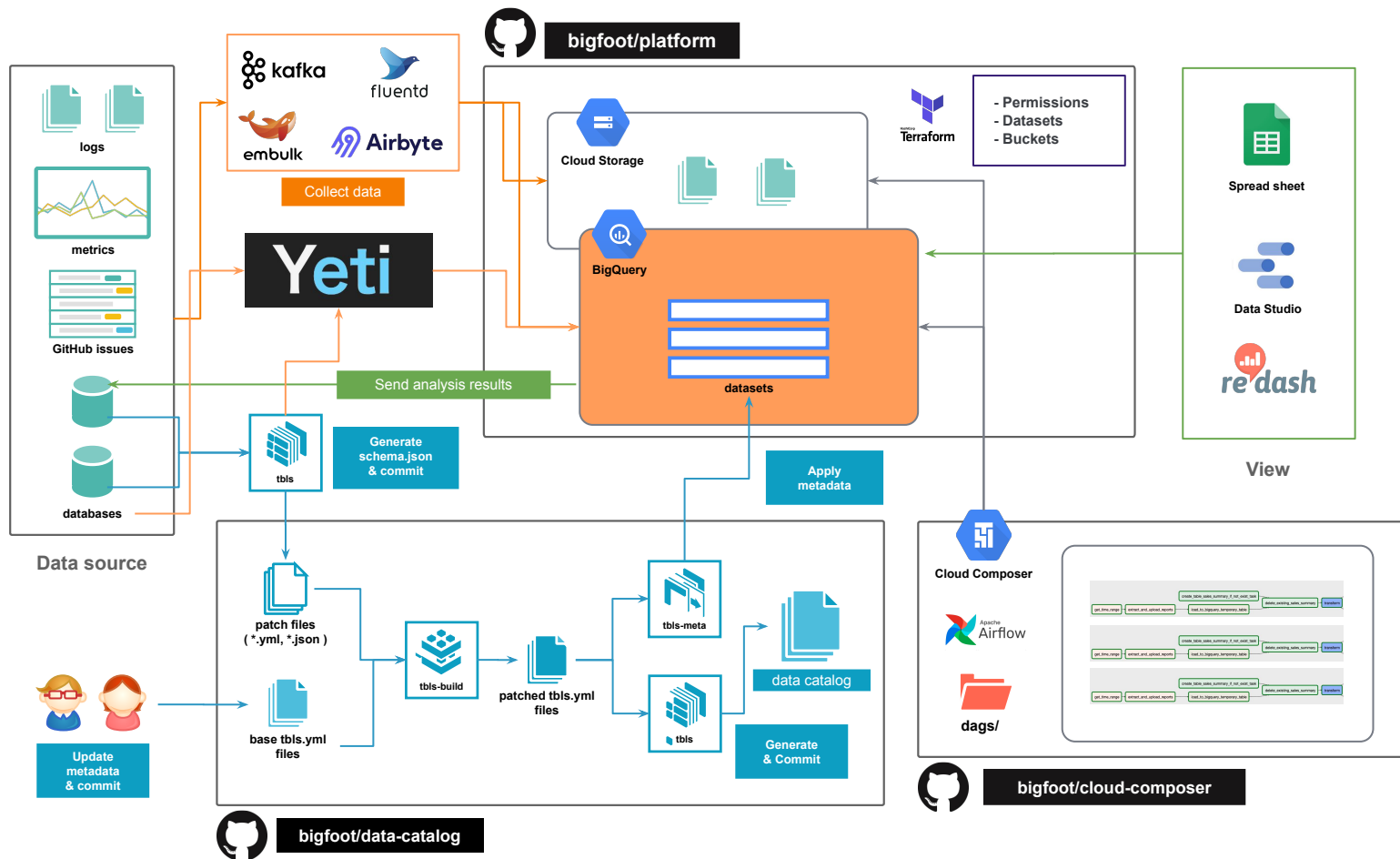
- データベースとは、構造化した情報またはデータの組織的な集合であり、通常はコンピュータ・システムに電子的に格納されています。データベースは通常、データベース管理システム(DBMS^{*1})で制御します。データと DBMS、およびそれらに関連するアプリケーションをまとめてデータベース・システムと呼びます。多くの場合は単にデータベースと呼んでいます。
- データは、処理とデータのクエリを効率化するために、一連のテーブルの行と列でモデル化されていることが普通です。

データベースの主な役割

- データを格納する
 - データベース内に複数のテーブルを格納する
 - 1 つのテーブルが 1 つの tidy data に対応する
- 問い合わせに対して応答する
 - データの抽出は構造化クエリ言語(Structured Query Language / SQL)で行う
 - SQL について詳しくは今日の研修の後半で扱います







Google Cloud BigQuery

- 大規模データ分析対応のフルマネージド型データウェアハウス*1
 - 一般的なデータベースと同様に標準SQLが利用可能
 - データの保存量が事実上無制限
 - クエリ実行速度が超速
 - タイムマシンや列レベルセキュリティなどデータ基盤を運用する上で便利な機能が色々
- ➡ ペパボではデータ基盤「Bigfoot」のデータウェアハウスとして使っています
- ➡ この研修でも BigQuery を使います

*1: データの倉庫。データ基盤において、様々なデータソースから収集したデータを保存しておく場所

クライアントとは

- データベースに対してクエリを発行するソフトウェア
 - ディレクター, CS の皆さんは GUI のものをよく使うことになりそうです
 - プログラムから利用するもの(クライアントライブラリ)もある
- データベースごとに様々なクライアントがある
 - 結果をグラフ化したりダッシュボードを作ったりする機能が付いているもの
 - 例)Redash, Metabase など
 - 分析ソフトウェアの一部にクエリを発行する機能があるもの
 - 例)Tableau など

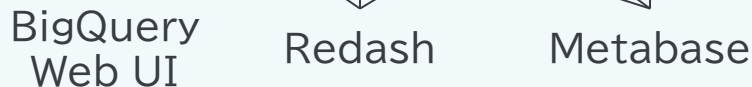
データベースとクライアントは違うもの

- ・ データソースが曖昧にならないようにデータベースとクライアントは区別しましょう
- ・ 何のクライアントを使ってどのデータベースにクエリしているかを意識する

データベース



クライアント



ロゴは <https://www.mysql.com/jp/about/legal/logos.html>, <https://wiki.postgresql.org/wiki/Logo>, <https://cloud.google.com/icons?hl=ja> から引用

余談: データウェアハウスとリレーショナルデータベースの違い

- BigQuery などのデータウェアハウスは列志向が多い
 - 特定列に対する集計処理が得意
 - 例)1億行10列のテーブルで列Aの平均を算出 → 列ごとにデータを保存しているので列Aのみ走査
- MySQL, PostgreSQLなどのリレーショナルデータベースは行志向
 - 特定行に対する操作が得意
 - 行の特定を高速に行う仕組み(インデックス)がある
 - 構造的に全列を走査するので列方向の集計処理はリソース効率的に向いていない
- 他にも...
 - Primary Key, Foreign Key(行の一意性を担保する仕組み)の有無^{*1}
 - データをどの程度正規化して保存するかの違い^{*2}

*1) *2) Day 3 で補足します。

余談: 大量のデータをスプレッドシートに貯めるのはやめましょう

- スプレッドシートは小規模なデータの表計算や可視化を簡単にできて便利
 - 数百行～数千行のデータを保存するような使い方は...?
 - 検索性が低いので目的のデータを探せない
 - ブラウザのリソースを使って処理するので集計が遅い
 - 動作が重くなってそもそもシートが開かなくなったりもする
- ➡ データは Bigfoot(BigQuery)に貯めてクエリで集計しましょう
- ➡ 基礎編ではデータベースからデータを参照して要約・可視化するところを扱います
エンジニアと合流して行う研修ではデータベースにデータを貯めるところも扱います

SQL基礎 + ハンズオン

SQL とは

- 構造化クエリ言語(Structured Query Language / SQL)
 - 構造化されたデータの集合を参照するための言語
 - 単純にそのまま取り出すだけでなく集計や分析もできる*1
- データベースのソフトウェアごとに多少の方言がある
 - BigQuery の「Standard SQL」は SQL:2011 という標準に準拠している

```
-- こういう文字列を見たことがあるかも
SELECT
  login
FROM
  `project.dataset.table`;
```

*1: この研修では扱いませんがデータの変更や削除もできます

BigQuery の Web UI で クエリを実行してみよう

BigQuery の Web UI を開く

- <https://console.cloud.google.com/bigquery>
- データを参照するのに必要な権限は既に設定してあります
 - 書き込み権限は無いので何かを壊す心配はありません
- Bigfoot の BigQuery は定額です
 - 課金が爆発することはありません
- 安心して色々触ってみてください

ここには画面キャプチャがありました
公開資料からは削除しています

Web UI の基本構成

- 左ペインにプロジェクト, データセット, テーブル一覧
- 右ペインにクエリエディタ(初期表示)
 - 一覧からテーブルをクリックするとテーブル詳細が開く
 - 右ペインはタブ機能を使える

ここには画面キャプチャがありました
公開資料からは削除しています

テーブルを眺める

- 今日は GitHub Enterprise Server^{*1} のデータを題材に SQL の基礎を学びます
- 検索フォームに「github_」と入力して Enter
- 検索結果からテーブルをクリック
 - SCHEMA が表示されます
 - 列の名前
 - 列の型
 - 列の説明 など
 - PREVIEW タブを選択すると
データをプレビューできます
- 他のテーブルも開いてみましょう

ここには画面キャプチャがありました
公開資料からは削除しています

^{*1}: 以降は GHES と表記します。

クエリエディタを開く

- テーブルを開いている場合は「QUERY」ボタンからエディタを開けます
 - In new tab, In split tab はお好みで
 - タブ表示の右側の「+」ボタンを押してもエディタを開けます

ここには画面キャプチャがありました
公開資料からは削除しています

クエリを実行してみる

- 下記のクエリのうち「YOUR EMAIL ADDRESS」を自分のメールアドレスに置き換えてクエリエディタに入力します
- 「RUN」ボタンを押すとクエリが実行され GHES の ID を得られます 🎉

```
SELECT  
  login  
FROM  
  `project.dataset.table`  
WHERE  
  email = "YOUR EMAIL ADDRESS";
```

ここには画面キャプチャがありました
公開資料からは削除しています

SQL の構文

ハンズオンの進め方


- 「🔗」の絵文字はハンズオンで取り組む問題を表します
- 脚注に公式ドキュメントなどの参考 URL を入れています

問題を解く際に不明点があったら参照してみてください

解説中の補足などでも適宜参照していきます

- 余談: 今回の研修に限らず何かで困ったらまず公式ドキュメントを読むことをお勧めします
 - 多くの公式ドキュメントには網羅的な情報が書いてあります
 - ブログ記事などの二次情報は間違っていることも結構あります
 - 英語の公式ドキュメントしか無いこともありますが、
英語が苦手でも Google 翻訳や DeepL の力を借りて読んでみてください

SELECT と FROM

- SQL にはクエリを組み立てるためのキーワード(予約語)がある
- 「SELECT」: 参照する列を指定するステートメントを表す
- 「FROM」: 参照するテーブルを指定する句を表す
-  (2-1) 現在 GHES に存在するユーザー ID を取得するクエリを作成してください

SELECT と FROM / 回答例

SELECT

login

FROM

`project.dataset.table`;

-- プロジェクト名と backtick は場合によって省略可能

SELECT

login

FROM

dataset.table;

WHERE

- 「WHERE」: 列の値によって参照する行を絞り込む句を表す
- 📌 (2-2) 現在 GHES に存在するユーザーのうち「type = "Bot"」である ID の一覧を取得するクエリを作成してください
 - ヒント: 前半で使った以下のクエリは email の値によって行を絞り込んでいます

```
SELECT
  login
FROM
  `project.dataset.table`
WHERE
  email = "YOUR EMAIL ADDRESS";
```


WHERE / 回答例

```
SELECT  
  login  
FROM  
  `project.dataset.table`  
WHERE  
  type = "Bot";
```

-- プロジェクト名と backtick は場合によって省略可能

```
SELECT  
  login  
FROM  
  dataset.table  
WHERE  
  type = "Bot";
```

GROUP BY

- 「GROUP BY」: 列の値を一意にした結果を返す句を表す
-  (2-3) GHES に存在するユーザーの type を一意に取得するクエリを作成してください

[https://cloud.google.com/bigquery/docs/reference/standard-sql/query-syntax?hl=ja#group by clause](https://cloud.google.com/bigquery/docs/reference/standard-sql/query-syntax?hl=ja#group%20by%20clause)

GROUP BY / 回答例

```
SELECT
  type
FROM
  dataset.table
GROUP BY
  type;
```

-- SELECT DISTINCT ステートメントでも同じ結果が得られる

```
SELECT DISTINCT
  type
FROM
  dataset.table;
```

[https://cloud.google.com/bigquery/docs/reference/standard-sql/query-syntax?hl=ja#group by clause](https://cloud.google.com/bigquery/docs/reference/standard-sql/query-syntax?hl=ja#group%20by%20clause)
[https://cloud.google.com/bigquery/docs/reference/standard-sql/query-syntax?hl=ja#select distinct](https://cloud.google.com/bigquery/docs/reference/standard-sql/query-syntax?hl=ja#select%20distinct)

集計関数の例 COUNT()

- 集計関数: 対象の値に何らかの集計を行った結果を返す
 - COUNT(): 対象の値を数え上げる
- 📌 (2-4) GHES に存在するユーザー ID を取得するクエリと、
GHES に存在するユーザーを数えるクエリを作成して、結果を比較してみましょう

<https://cloud.google.com/bigquery/docs/reference/standard-sql/aggregate-functions?hl=ja#count>

集計関数の例 COUNT() / 回答例

-- 現在 GHES に存在するユーザー ID を取得するクエリ

```
SELECT  
  login  
FROM  
  dataset.table;
```

-- GHES に存在するユーザーを数えるクエリ

-- 「*」は全てのカラムを意味する

```
SELECT  
  COUNT(*)  
FROM  
  dataset.table;
```

<https://cloud.google.com/bigquery/docs/reference/standard-sql/aggregate-functions?hl=ja#count>

GROUP BY と集計関数の組み合わせ

- ✎ (2-5) 直近半年分の GHES の Issue のリポジトリ別の件数を取得する

SQL を作成してください


- ・ ヒント: テーブルスキーマやデータのプレビューを確認して使うべきデータを探しましょう
- ・ ヒント: Issue の作成日は TIMESTAMP 型で格納されています
- ・ ヒント: TIMESTAMP の値を加減算するための関数が存在します

<https://cloud.google.com/bigquery/docs/reference/standard-sql/timestamp-functions?hl=ja>

GROUP BY と集計関数の組み合わせ / 回答例

```
SELECT
  sdc_repository,
  COUNT(*) AS _count, -- 結果に別名をつけることができる
FROM
  dataset.table
WHERE
  created_at >= TIMESTAMP_SUB(CURRENT_TIMESTAMP(), INTERVAL 30 * 6 DAY)
  -- TIMESTAMP の減算は TIMESTAMP_SUB() でできる
  AND pull_request IS NULL
GROUP BY
  sdc_repository;
```

ORDER BY

- 「ORDER BY」: 列の値によって結果の並び替えを行う
-  (2-6) Bigfoot に格納されている GHES の Issue のうち、更新が最も古い/最も新しいものの URL と更新時刻を取得するクエリを作成してください
 - ヒント: URL は `https://git.example.com/org/repo/issues/123` の形式で取得してください

[https://cloud.google.com/bigquery/docs/reference/standard-sql/query-syntax?hl=ja#order by clause](https://cloud.google.com/bigquery/docs/reference/standard-sql/query-syntax?hl=ja#order%20by%20clause)

ORDER BY / 回答例(1)

```
-- ORDER BY で昇順に並び替える
SELECT
  html_url,
  updated_at,
FROM
  dataset.table
ORDER BY
  updated_at;
```

<https://cloud.google.com/bigquery/docs/reference/standard-sql/operators?hl=ja>

ORDER BY / 回答例(2)

```
-- ORDER BY で降順に並び替える
SELECT
  html_url,
  updated_at,
FROM
  dataset.table
WHERE
  pull_request IS NULL
ORDER BY
  updated_at DESC
LIMIT 1; -- LIMIT 句で結果の行数を制限できる
```

<https://cloud.google.com/bigquery/docs/reference/standard-sql/operators?hl=ja>

Standard SQL の型

- データの種類によっていくつかの「型」がある
 - 文字列型: あいうえお
 - 数値型(整数型): -99 0 1 12345
 - 数値型(小数型)*1: 1.23456
 - 日付型: 2022-07-21
 - 日時型: 2022-07-21 12:00:00
 - タイムスタンプ型: 2022-07-21 03:00:00 UTC
 - ...
- データベースによって異なる

<https://cloud.google.com/bigquery/docs/reference/standard-sql/data-types?hl=ja>

*1) 小数点以下の扱いによっていくつか種類があります

型が取りうる値

- それぞれの型が取りうる値(データの表現)と値の文字列表現は異なる
 - 文字列表現: SELECT すると見える文字列
 - 例)タイムスタンプ型のデータは「初期時点からのマイクロ秒数」
 - タイムゾーンを持たない(「UTC の時刻」ではない)
 - `2022-01-01 00:00:00 Asia/Tokyo` と `2021-12-31 15:00:00 UTC` は
同じ TIMESTAMP
 - 例)文字列型の **1** と整数型の **1** は別物


```
1 SELECT TIMESTAMP("2022-01-01 00:00:00", "Asia/Tokyo");
```

Query results

JOB INFORMATION		RESULTS	JSON	EXECUTION DETA
Row	f0_			
1	2021-12-31 15:00:00 UTC			

<https://cloud.google.com/bigquery/docs/reference/standard-sql/data-types?hl=ja>

BigQuery で日付時刻を扱う際のポイント

- 日付時刻に関係する型は日付型, 日付時刻型, タイムスタンプ型の 3 つ
- BigQuery に日付時刻を保存する際はタイムスタンプ型を使いましょう
 - タイムスタンプ関数^{*1}で日付時刻の操作ができる
 - 加減算や日ごと, 月ごとの変換も簡単
 - 日付型, 日付時刻型は「ローカルタイムゾーンの日付, 日付時刻」を表すのでタイムゾーンの情報が失われる
 - ローカルタイムに変換したい場合はビューで参照する際に日付型, 日付時刻型にする
 - `DATETIME(TIMESTAMP("2022-01-01 00:00:00", "Asia/Tokyo"), "Asia/Tokyo")`
 - 文字列型で日付時刻を扱うのは推奨しません  ^{*2}
 - フォーマットが一定でなく関数で日付時刻の操作がしづらい

^{*1}: <https://cloud.google.com/bigquery/docs/reference/standard-sql/timestamp-functions?hl=ja>

^{*2}: BigQueryのテーブルに保存する日付時刻を文字列にすることを推奨しないという意味であり、ビューに表示する際に必要なフォーマットは当然この限りではありません

「何もない」には 3 種類ある

- NULL: 「何もない」こと
 - BigQuery の Web UI では「null」と表示される
 - `column IS NULL` `column IS NOT NULL` で判定
- 空文字: 長さが 0 の文字列
 - `LENGTH(column) = 0` `column != ""` で判定
- スペース記号
 - `column = " "` で判定

➡ 等価ではないので注意

```
1 SELECT
2   NULL AS _null,
3   "" AS empty_string,
4   " " AS space_symbol;
```

Query results

JOB INFORMATION		RESULTS	JSON
Row	_null	empty_string	space_symbol
1	null		

まとめ

- データベースとクライアントの役割
 - ペパボのデータ基盤「Bigfoot」とデータウェアハウス「BigQuery」の紹介
- BigQuery の Web UI の開き方
- SQL の構文
 - SELECT, FROM
 - WHERE
 - GROUP BY と集計関数
 - ORDER BY
- 型の話

早く終わったら

- Google Cloud Self-Paced Labs をやってみましょう
 - Google BigQuery で SQL を使用して e コマース データセットを操作する
 - <https://www.cloudskillsboost.google/focuses/3618?locale=ja&parent=catalog>
 - BigQuery でのよくある SQL エラーのトラブルシューティング
 - <https://www.cloudskillsboost.google/focuses/3642?locale=ja&parent=catalog>