

Teamwork Transformación y Análisis de Registros JSON de Publicaciones usando dbt + BigQuery

En la carpeta **Data Enrichement** encontrarás múltiples archivos en formato **JSON**, cada uno correspondiente a una actividad académica asociada a un profesor (por ejemplo: *Journal Publication*, *Book Chapter*, *Conference Proceeding*).

Cada archivo representa **un único registro de actividad**.

Un ejemplo simplificado del contenido de un archivo es:

JSON

```
{
  "activityid": 1294,
  "fields": {
    "Type": "Journal Publication",
    "Title of Contribution": "...",
    "Journal": "...",
    "Volume": "10",
    "Month / Season": "2000-04-01",
    "Issue Number": "4",
    "Actual/Projected Year of Publication/Presentation":
2000,
    "Page Numbers": "43-52",
    "Publisher": "",
    "DOI": "",
    "PubMed Central ID Number": "",
    "Author(s) / Editor(s)": null,
    "Web Address": "...",
    "Description/Abstract": "...",
    "Origin": "Other"
  },
  "facultyid": "111110323",
  "coauthors": { ... },
  "status": [ ... ],
  "userid": "111110323",
  "attachments": []
}
```

```
}
```

El objetivo de la actividad es construir, usando **dbt** sobre **BigQuery**, una serie de modelos que permitan:

1. Cargar los JSON crudos a una tabla.
2. Estructurar los campos relevantes en columnas.
3. Aplanar la información de coautores.
4. Crear tablas agregadas por tipo de publicación.
5. Analizar patrones de colaboración entre profesores (faculty), incluyendo top de coautores y matriz de colaboración para el top 10 más productivo.

Punto 1 — Tabla cruda de actividades (hecho en clases)

Migre todos los registros de los archivos JSON a una **tabla en BigQuery**.

- La tabla debe contener únicamente las siguientes columnas:
 - `activityid`
 - `raw_json` (el contenido completo del archivo en formato JSON, como string o JSON nativo)
- Sugerencia de nombre de modelo dbt: `stg_pub_activities_raw`.

Punto 2 — Tabla estructurada de actividades

Cree una **segunda tabla** en la cual cada campo relevante del JSON sea mapeado a una columna estructurada.

Requisitos:

- Parta de `stg_pub_activities_raw` (Punto 1).
- La nueva tabla debe incluir:
 - `activityid`
 - Todos los campos contenidos dentro de `fields` (por ejemplo: `Type`, `Title of Contribution`, `Journal`, `Volume`, `Actual/Projected Year of Publication/Presentation`, etc.).

- Metadatos a nivel raíz como: `facultyid`, `userid`, `status`, `attachments` (según aplique; si una estructura es compleja, puede extraer campos clave o marcadores simples).
- No todos los JSON tienen los mismos campos:
 - Los campos que no apliquen a cierto tipo de publicación deben quedar como **NULL**.
- **Importante:**
 - **No** incluir la lista completa de coautores en esta tabla.
 - El **autor principal** (`facultyid`) sí debe estar incluido como columna.
- Sugerencia de nombre de modelo dbt: `pub_activities_wide`.

Punto 3 — Tabla de autores y coautores

Cree una **tercera tabla** que contenga información a nivel de autor/coautor.

Requisitos:

- A partir del campo `coauthors` del JSON original, genere una tabla con:
 - `activityid`
 - Una fila por **cada autor listado** en `coauthors`.
- Para cada coautor incluya, al menos:
 - Nombre
 - Apellido
 - Alguna bandera que indique si es de la misma institución (`same_school` o campo equivalente, si está disponible)
 - Si existe, un identificador de faculty (por ejemplo, `coauthor_facultyid`) cuando ese coautor también es profesor de la institución.
- Esta tabla representa la “lista de coautores por actividad”.
- Sugerencia de nombre de modelo dbt: `pub_activity_authors`.

Punto 4 — Tabla agregada por faculty y tipo de publicación

Genere una tabla agregada por:

- `facultyid` (autor principal)
- `Type` (tipo de publicación, por ejemplo: *Journal Publication*, *Book Chapter*, *Conference Proceeding*, etc.)

Requisitos:

- Parta de `pub_activities_wide` (Punto 2).
- Construya una matriz (wide table) tal que:
 - Cada fila corresponde a un `facultyid`.
 - Tenga una columna por cada tipo de publicación (por ejemplo: `n_journal_publication`, `n_book_chapter`, `n_proceeding`, etc.).
 - Cada celda representa el **número total de registros** de ese tipo para ese profesor.
- Sugerencia de nombre de modelo dbt: `pub_faculty_productivity_by_type`.

Punto 5 — Modelo de colaboración y carga docente (mezclando tablas)

En este punto se trabajará con **patrones de colaboración** entre profesores, combinando la tabla de actividades y la tabla de autores.

Parta de:

- `pub_activities_wide` (Punto 2) — contiene `activityid`, `facultyid` (autor principal), `Type`, etc.
- `pub_activity_authors` (Punto 3) — contiene `activityid` + una fila por cada coautor, con nombre, apellido, `same_school`, y si está disponible, un `coauthor_facultyid`.

5.1. Resumen básico de colaboración por faculty

Construya un modelo dbt (por ejemplo, `pub_faculty_collaboration_summary`) que:

1. Una `pub_activities_wide` con `pub_activity_authors` usando `activityid`.
2. Para cada `facultyid` autor principal, calcule:
 - `total_publications`:
Número total de publicaciones donde aparece como autor principal.
 - `total_unique_coauthors`:
Número de coautores **distintos** con los que ha colaborado (independiente del número de productos).

- `avg_coauthors_per_publication`:
Promedio de coautores por publicación (por ejemplo, total de filas de coautores / total_publications).
3. Incluya una versión desagregada por `Type` para analizar en qué tipo de productos se colabora más.

5.2. Top 3 de coautores por faculty

Luego, utilizando la unión entre `pub_activities_wide` y `pub_activity_authors`:

1. Construya una tabla intermedia con todas las parejas:
 - `(facultyid_autor_principal, coauthor_id)`
Donde `coauthor_id` puede ser:
 - `coauthor_facultyid` (si existe), o
 - algún identificador derivado del nombre/apellido si no existe ID institucional.
 2. Para cada par `(facultyid_autor_principal, coauthor_id)` calcule:
 - `collab_count`: número de publicaciones conjuntas.
 3. A partir de esa tabla, genere un modelo que, para **cada `facultyid`**, identifique su **top 3 de coautores** según `collab_count`, incluyendo:
 - `facultyid` (autor principal)
 - `top1_coauthor_id, top1_collab_count`
 - `top2_coauthor_id, top2_collab_count`
 - `top3_coauthor_id, top3_collab_count`
- Sugerencia de nombre de modelo: `pub_faculty_top3_coauthors`.

5.3. Top 10 más productivo y matriz de colaboración

Finalmente, construya un modelo dbt que combine productividad y colaboración para los profesores más activos.

1. A partir de `pub_faculty_collaboration_summary` (5.1) identifique el **top 10 de `facultyid` con mayor `total_publications`**.

2. Para estos 10 facultyid, construya una tabla **wide** que:

- Tenga una fila por cada **facultyid** del top 10.
- Incluya:
 - **facultyid**
 - **total_publications**
 - **total_unique_coauthors**
- Además, cree columnas que reflejen con quién ha colaborado más cada uno, **limitado al top 10**. Por ejemplo:
 - **top_collab_faculty_1_id**
top_collab_faculty_1_count
 - **top_collab_faculty_2_id**
top_collab_faculty_2_count
 - **top_collab_faculty_3_id**
top_collab_faculty_3_count
- La idea es que:
 - Si se dispone de **coauthor_facultyid**, idealmente se mida **faculty vs faculty** (colaboraciones entre profesores internos).
 - Cada celda numérica represente el **número de publicaciones conjuntas** entre el faculty principal (fila) y el coautor correspondiente (columna).
- Sugerencia de nombre de modelo: **pub_faculty_top10_collab_matrix**.

Entregable

- Proyecto **dbt** apuntando a **BigQuery** que contenga todos los modelos anteriores.
- Código SQL de los modelos **.sql** y, idealmente, un archivo **schema.yml** con descripciones breves de las tablas y columnas clave.
- El dataset debe llamarse Mi en donde i es el número de su equipo. Utilice el SA.json empleado en clases.
- Un breve documento (puede ser en Markdown o en un notebook) explicando:
 - Cómo se hizo la carga de los JSON.
 - Cómo se estructuraron los campos.
 - Cómo se interpretan las métricas de colaboración y productividad (especialmente la tabla del top 10).