



Dennis GetGround Challenge

🕒 Created	@October 22, 2022 7:35 AM
⋮ Tags	
≡ Property	
≡ Description	GetGround's Data Task
👤 Owner	
⋮ Status	<div>Applying PR Feedback</div> <div>Completed</div>

Goal 🎯

Exploration of Get Ground input files/data 🔍

Relationship Findings 📊

Example queries to discover my findings:

Data Quality Checks 🚀

MVP Scope and Design 📐

Scope

Design

Purpose of each Model Type

Critical Model Considerations ☀️

Future controls:

Findings ✅ via viewer

Future and improvements 🧑

Goal 🎯

Design and build a dbt data model for Get Ground's referral data

This will allow analysts and stakeholders to answer questions regarding GetGround's partners such as:

- How many successful referrals does each partner?
- Which industry/partner_type has the most referrals?

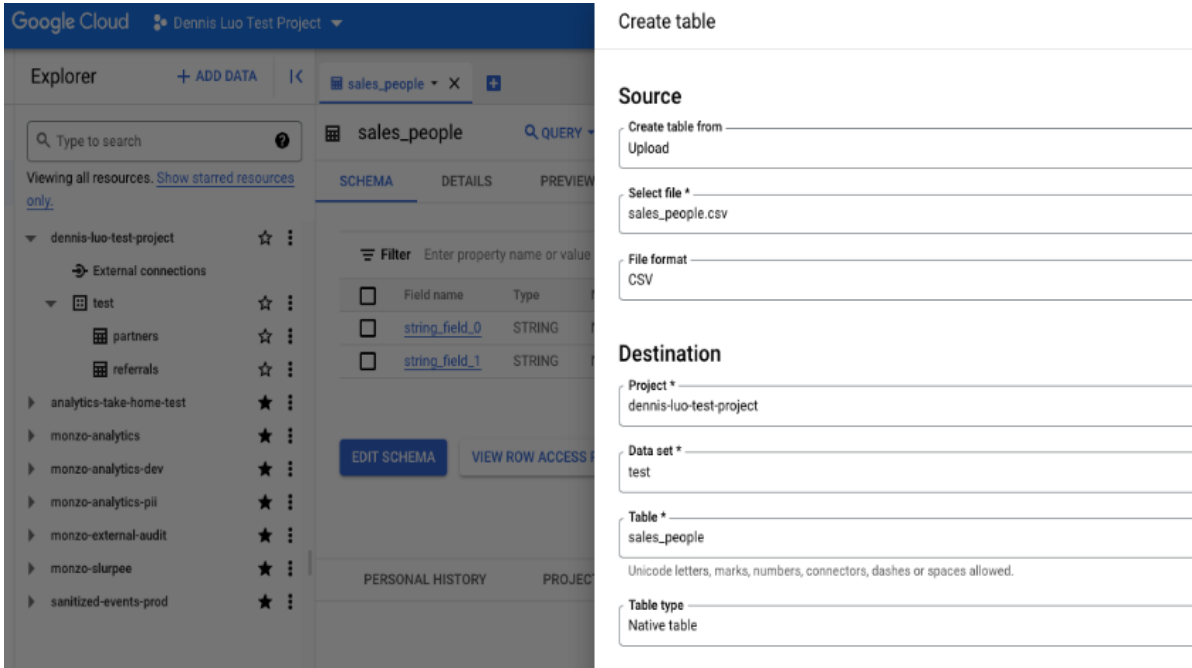
- Which partner brings the most referrals?
- Which partner brings the most successful referrals?
- Which sales men has the most successful referrals?
- Which country brings the most referrals?
- Most recent successful referral for each partner?

Exploration of Get Ground input files/data

▼ The 3 csv files represent the sales (sales.csv), referrals (referrals.csv) and partners (partners.csv) data for getground.

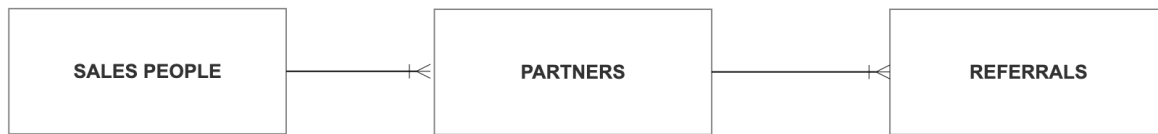
All 3 files will act as our 'events' and are uploaded to Bigquery

Example below:



The screenshot shows the Google Cloud BigQuery console interface. On the left, the 'Explorer' pane displays the project hierarchy: 'dennis-luo-test-project' > 'test' > 'partners', 'referrals', and 'sales_people'. The 'sales_people' table is selected, and its schema is visible in the center pane, showing three fields: 'Field name', 'string_field_0', and 'string_field_1', all of type 'STRING'. On the right, the 'Create table' wizard is open, showing the 'Source' and 'Destination' sections. The 'Source' section is set to 'Upload' with the file 'sales_people.csv' selected. The 'Destination' section is set to 'Project *' with the project 'dennis-luo-test-project', data set 'test', and table 'sales_people'.

Relationship Findings



▼ The design of my data model will be based on the findings and assumptions below:

- Every time a customer is reference this becomes a row in the **referrals** table and has a unique referral id
- Sales people and partners have a one-to-many relationship
- Partners and referrals have a one-to-many relationship
- A company can have multiple customers
- Each referral can only belong to one company and one partner
- A consultant can have multiple companies
- A consultant can only work for one partner
- One salesmen can have many partners
- A partner has up to one sales person
- Each sales person name is unique and belongs to one country
- The lead salesmen can be a salesmen that is not on GetGround sales_people data i.e. Potato.
- Each row where `is_outbound` is `true` acts as a successful referral for GetGround and hence a successful commission

Example queries to discover my findings:

```
-- Assert and check assumptions:
-- Check each partner is only represented by one sales contact
SELECT *
FROM `dennis-luo-test-project.test.partners`
WHERE id IN (
  SELECT id
  FROM `dennis-luo-test-project.test.partners`
  GROUP BY id
```

```

        HAVING COUNT(distinct lead_sales_contact) > 1
    )
    -- Each sales contact can represent multiple partners i.e.
    SELECT *
    FROM `dennis-luo-test-project.test.partners`
    WHERE lead_sales_contact IN (
        SELECT lead_sales_contact
        FROM `dennis-luo-test-project.test.partners`
        GROUP BY lead_sales_contact
        HAVING COUNT(distinct id) > 1
    )

    -- A consultant belongs to one partner
    SELECT *
    FROM `dennis-luo-test-project.test.referrals`
    WHERE consultant_id IN (
        SELECT consultant_id
        FROM `dennis-luo-test-project.test.referrals`
        GROUP BY consultant_id
        HAVING COUNT(distinct partner_id) > 1
    )

    -- A company can belong to many partners
    SELECT *
    FROM `dennis-luo-test-project.test.referrals`
    WHERE company_id IN (
        SELECT company_id
        FROM `dennis-luo-test-project.test.referrals`
        GROUP BY company_id
        HAVING COUNT(distinct partner_id) > 1
    )
    order by company_id

```

Data Quality Checks

The base table acts as the gateway and ingests the input files as cleansed sources of data.

These checks cover all data relevant to the model.

They will indicate what cleansing needs to be done in the model and the tests that need to be added to the staging and entity tables.

Data quality checks are done for all models via the **yml** files.

MVP Scope and Design

Scope

The MVP will include a final product level entity table: `partner_stats`

`partner_stats`

This model will provide all partners and their associated customer reference stats.

It is an aggregation table which allows us to conduct the necessary analysis.

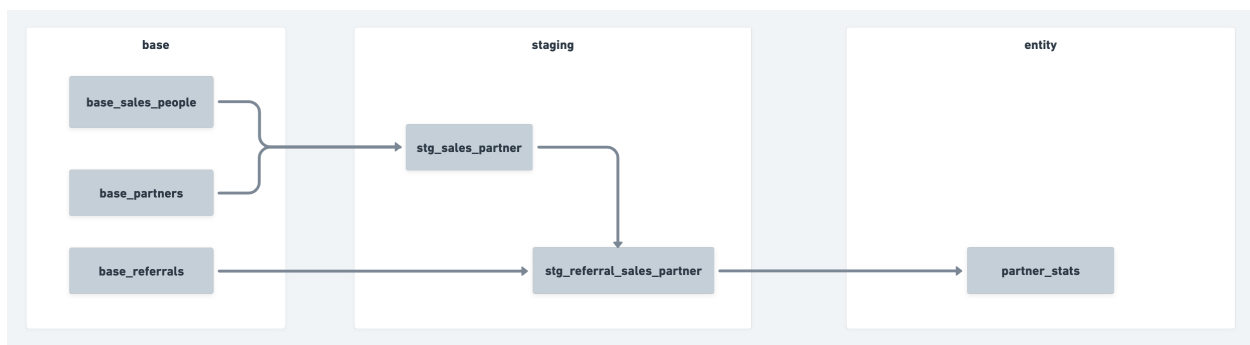
This will allow data users to answer questions such as:

- How many successful referrals does each partner?
- Which industry/partner_type has the most referrals?
- Which partner brings the most referrals?
- Which partner brings the most successful referrals?
- Which sales men has the most successful referrals?
- Which country brings the most referrals?
- Most recent successful referral for each partner?

As the referral creation date is an original attribute and will never change, this model can also be used to analyse sign-up rates over time as a future enhancement

The model will include the most important key attributes for the partner and sales stage tables.

Design



Purpose of each Model Type

▼ Base

- Deduplicate
 - There should be only one partner, sales and referral id and therefore logic is added to deduplicate and ensure each row is unique
- Cleanse
 - All data should be consistent format and so any names shall be upper case
 - Data types such as timestamp and integer should be casted to ensure data is readable and used for filtering later on

▼ Staging

- Combine/join base models and add business logic
 - Having this stage stops the code in the entity models from being too long and complex from several joins
- Referrals can't directly join to sales so `stg_sales_partner` is used by joining the base tables `sales` and `partner` . This can be used for a future entity table for sales (i.e. sales level)
- `stg_referral_sales_partner` can be created from the above step to create a bespoke partner level entity
- `stg_sales_partner` can be joined back to `stg_referral_sales_partner` to give full partner level table

▼ Entity

- Combine staging tables to give a high level aggregated view
- Add further business logic to get aggregated view
- In our case our aggregated views is partners
- Created by merging and grouping `stg_sales_partner` and `stg_referral_sales_partner`

NOTE: The `ref` command in dbt creates our dependencies and allows us to run base>stage>entity in the correct order

Critical Model Considerations

As the final entity partner level data is so important, these models will be classified as **critical**.

This means we have to add extra controls so we can guarantee their accuracy and reliability.

Here are the controls and how they have been considered in the above model designs:

▼ Reliable

This means the models can only be created using other critical models.

The final product entity model depend on base models or staging models that are specific to them. This means there aren't any dependencies on non-critical data.

▼ Tested

The models include tests on the PKs and business logic.

▼ Isolated

This means the models run upstream of less urgent data.

That's the case as the final product entity model only depend on staging models that are specific to them.

▼ Documented

There are model descriptions above and the YML files will contain field descriptions for the staging and entity tables as well as the [dictionary.md](#) docs file.

Future controls:

The final control is **ownership**.

As a future enhancement and improvement to GetGround data these models will be owned by the Analytics Engineering team and shared access via Github for version control.

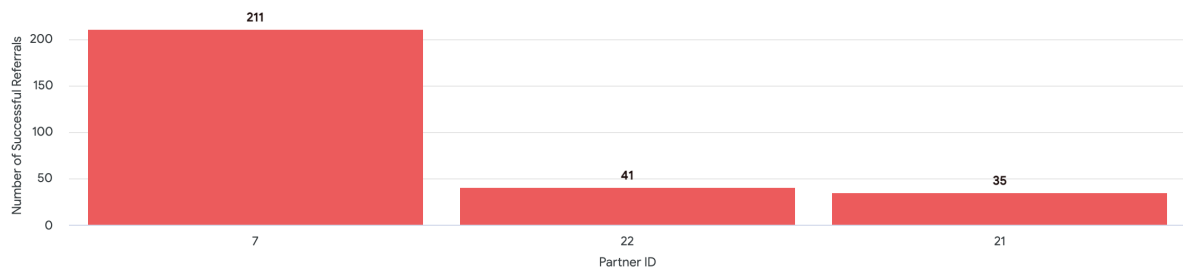
All PRs will need to be approved by a member of the team.

Findings via looker

Now we have our partner level model entity we can answer the below questions: 🙏

For this I have created a looker ml script ingesting our entity table and then using looker to visualise our results.

▼ How many successful referrals does each partner?/Which partner brings the most referrals?

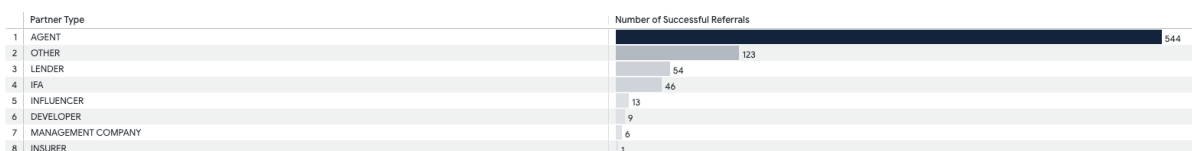


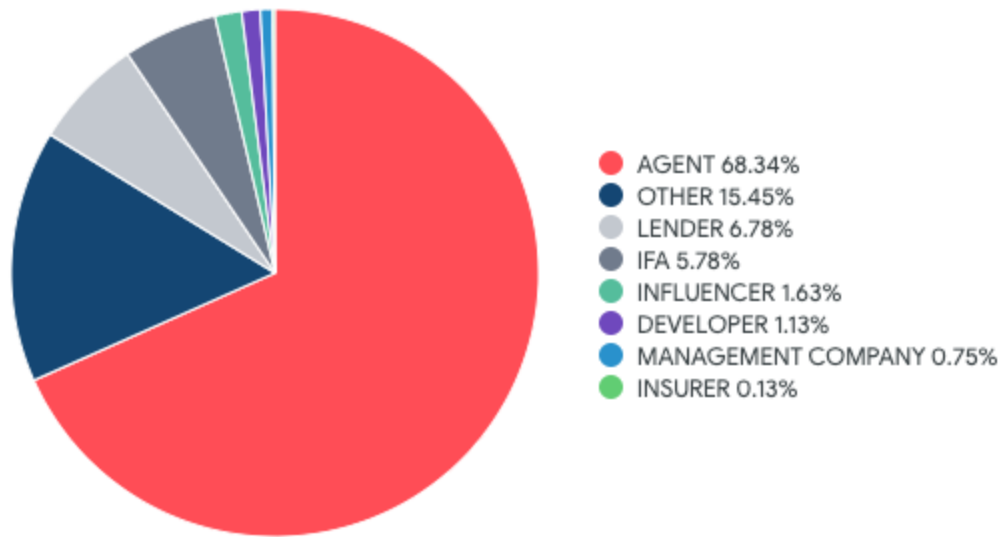
Filtering for the top 3 partners we can see that partners 7, 22 and 21 have the most referrals.

Partner 7 has the most referrals with 211.

Note: For this we are basing it on the rule that a successful referral is when the customers sign up for individual companies under that partner (i.e. number_of_successful_referrals column). If several customers sign up for one company then it counts as one referral for that partner (i.e. number_of_signups column).

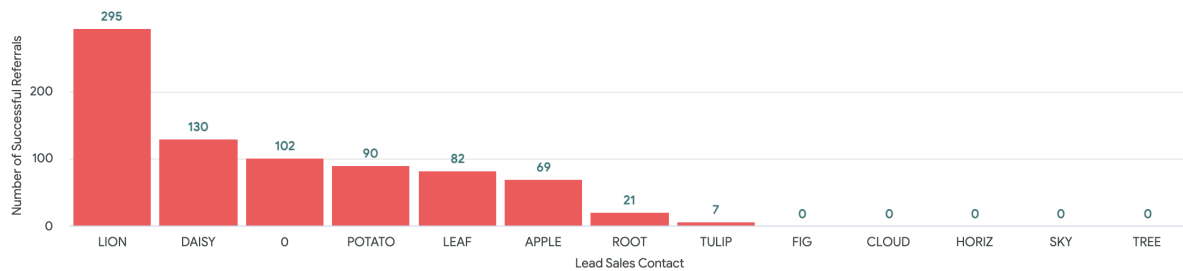
▼ Which industry/partner_type has the most referrals?





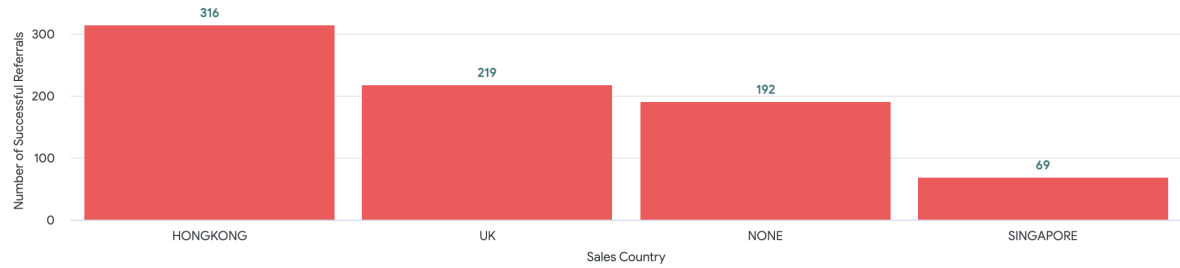
Agents provide the most successful referrals with a huge 68% (544 referrals).

▼ Which sales men has the most successful referrals?

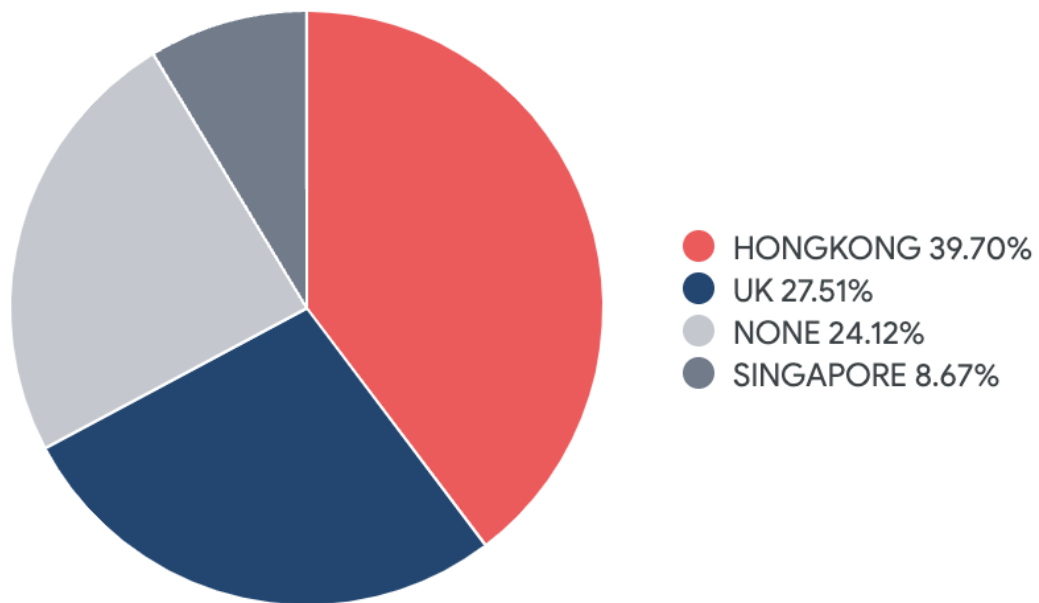


Lion is the most successful salesmen with a whopping 295 of the total!

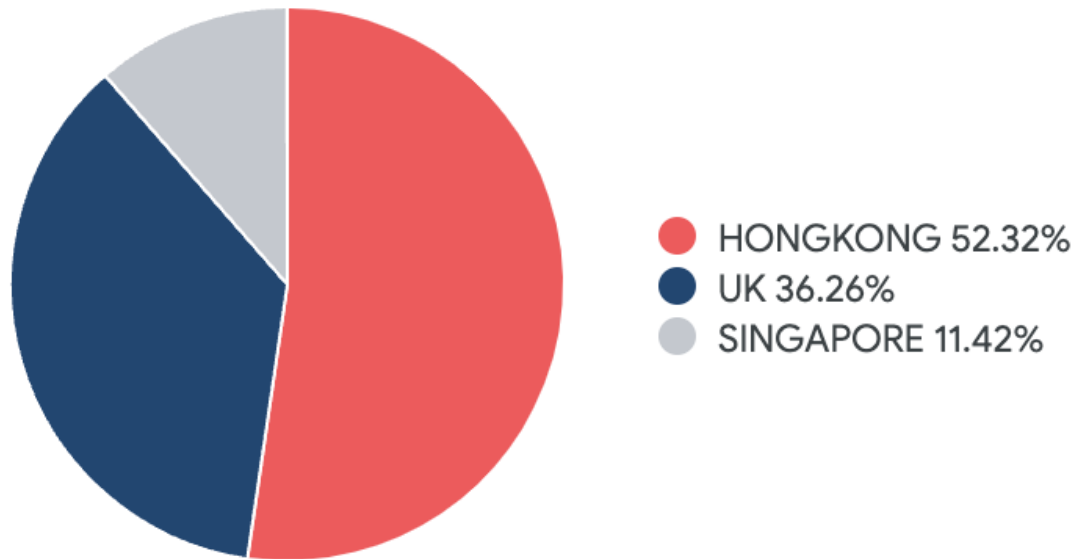
▼ Which country brings the most referrals?



Successful referrals by country:



Successful referrals by country: (Excluding salesmen who are not registered on our system)



Hong Kong brings the most successful referrals with 316 *(40%). We can also see that a 1/4 of the successful referrals belong to salesmen who are not registered on our system (e.g. Potato). When we exclude them Hong Kong takes 1/2 of all successful sales at 52%.

▼ Most recent successful referral for each partner?

	Partner ID	Latest Successful Referral
1	22	2021-05-05
2	519	2021-05-05
3	67	2021-05-05



Partners 22, 519 and 67 completed the most recent successful referrals on 5th May 2021.

Future and improvements

Based on the data and exploration there are several ways we could improve the level of the data.

- There should be a daily immutable referral table incremented on a daily basis where each referral id only has one record per date showing the status update per day

rather than constantly update the entire referral table every time there is a change which lacks historical records and can cause unexpected consequences to other account data.

- There should be a daily table for partners so we can keep a track record of there lead sales person across time
- We should split the outbound key to be used for one atomic purpose (i.e. for referral money). Another referral key and column should be made to represent upsells.
- We should create a referral_status_history table. This is a daily history table containing the status of each referral from the date the account was created up until yesterday.

Future: ✨

Include snapshots for failsafes if data is changed

Use staging table stg_sales_partner to create sales level entity table for stats.