

Luiz Decio - Analytics Engineer

Deel Home Task: Globepay

Part 1

Preliminary data exploration

- First it was created a table “acceptance” and a table “chargeback”, for each CSV:

The screenshot shows the Snowflake web interface. On the left, the navigation menu includes 'Create', 'Search', 'Projects', 'Data', 'Databases', 'Add Data', 'Data Products', 'AI & ML', 'Monitoring', and 'Admin'. The 'Databases' section is expanded, showing 'CITIBIKE', 'DEEL_GLOBEPAY', 'INFORMATION_SCHEMA', 'RAW', and 'SNOWFLAKE'. The 'RAW' database is selected, and the 'ACCEPTANCE' table is highlighted. The main panel displays the 'Table definition' for 'DEEL_GLOBEPAY / RAW / ACCEPTANCE'. The table was created by 'ACCOUNTADMIN' 38 minutes ago and has a size of 5.4K (421.0KB). The table definition is as follows:

```
1 create or replace TABLE DEEL_GLOBEPAY.RAW.ACCEPTANCE (  
2   EXTERNAL_REF VARCHAR(16777216),  
3   STATUS BOOLEAN,  
4   SOURCE VARCHAR(16777216),  
5   REF VARCHAR(16777216),  
6   DATE_TIME TIMESTAMP_NTZ(9),  
7   STATE VARCHAR(16777216),  
8   Cvv_PROVIDED BOOLEAN,  
9   AMOUNT NUMBER(38,2),  
10  COUNTRY VARCHAR(16777216),  
11  CURRENCY VARCHAR(16777216),  
12  RATES VARCHAR(16777216)  
13 );
```

Below the table definition, the 'Privileges' section shows that the 'ACCOUNTADMIN' role has 'OWNERSHIP' privileges on the table.

The screenshot shows the Snowflake web interface. On the left, the navigation menu is the same as in the previous screenshot. The 'Databases' section is expanded, and the 'CHARGEBACK' table is highlighted. The main panel displays the 'Table definition' for 'DEEL_GLOBEPAY / RAW / CHARGEBACK'. The table was created by 'ACCOUNTADMIN' 40 minutes ago and has a size of 5.4K (114.0KB). The table definition is as follows:

```
1 create or replace TABLE DEEL_GLOBEPAY.RAW.CHARGEBACK (  
2   EXTERNAL_REF VARCHAR(16777216),  
3   STATUS BOOLEAN,  
4   SOURCE VARCHAR(16777216),  
5   CHARGEBACK BOOLEAN  
6 );
```

Below the table definition, the 'Privileges' section shows that the 'ACCOUNTADMIN' role has 'OWNERSHIP' privileges on the table.

- For each table, it was checked:
 - The data type of each column, if was according to the API documentation
 - The consistency and validity of each column:
 - Negative amount values
 - There was 1 (external_ref **SPm_aqm_Rrer_6jxpLvO2**). This was considered a possible mistake and the column values was converted to positive in dbt staging model.
 - Dates up to the current date
 - All dates are from 2016 first quarter
 - Unique boolean values (only TRUE and FALSE)
 - Unique sources in the acceptance and chargeback table
 - There was only Globalpay as expected
 - Check the possible “status” values in both tables
 - All of them were TRUE
 - The uniqueness of the “external_ref” column, that will be later used as an unique identifier for each row:

The screenshot shows a SQL query execution interface. The query is as follows:

```
1 select count(distinct external_ref) as unique_transactions_acceptance from acceptance
2 union all
3 select count(distinct external_ref) as unique_transactions_chargeback from chargeback;
```

The results table shows two rows, both with a count of 5430.

	UNIQUE_TRANSACTIONS_ACCEPTANCE
1	5430
2	5430

Query Details:

- Query duration: 423ms
- Rows: 2
- Query ID: 01b5b1e9-0105-e981-0...
- Show more
- UNIQUE_TRANSACTIONS_ACCEPTAN... #
- 100% filled

Summary of your model architecture

The model architecture, was divided between two stages and three tables, as follows:

- **Staging Models:** Raw data ingestion from the source tables (CSV files), basic transformations (cleaning, type casting), and column renaming for clarity.
- **Mart Models:** Joining and aggregation after modeling the stages tables, so the data analysts can answer business questions.

Tables:

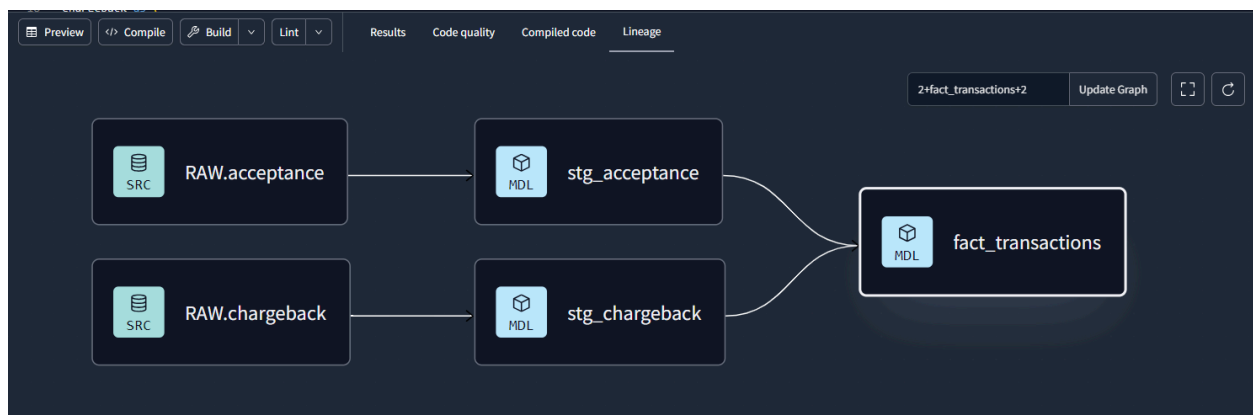
- **stg_acceptance**: Staging table for the acceptance report.
- **stg_chargeback**: Staging table for the chargeback report.
- **fact_transactions**: Fact table combining the acceptance and chargeback data.

Transformations:

- Cleaning and normalizing data.
- Handling missing or inconsistent values.
- Getting the specific rate according to the currency that it was paid
- Convert the amount to USD

Lineage graphs

The lineage graph includes both of the staging being created from the raw sources, and then joining each other into the fact table.



Tables hierarchy in Snowflake after successful “dbt build” command

The screenshot shows the Snowflake schema browser interface. On the left, a tree view displays the hierarchy: DEEL_GLOBEPAY / ANALYTICS. Under ANALYTICS, there are three main categories: Tables, Views, and RAW. The Views category is expanded, showing three views: FACT_TRANSACTIONS, STG_ACCEPTANCE, and STG_CHARGEBACK. The main panel on the right shows the details of the selected view, including its name, type, owner, and creation time. The table below lists the three views.

NAME	TYPE	OWNER	CREATED
FACT_TRANSACTIONS	View	ACCOUNTADMIN	1 minute...
STG_ACCEPTANCE	View	ACCOUNTADMIN	1 minute...
STG_CHARGEBACK	View	ACCOUNTADMIN	1 minute...

Tips around macros, data validation, and documentation

- Macros: it was create a macro called “get_rate” that used the currency code in column “currency” and searched for the correct exchange rate in the “exchange_rates” column in order to convert the amount of the transaction to USD.

Macro definition

```
{% macro get_rate(json_column, currency_column) %}  
  
    json_extract_path_text({{ json_column }}, {{ currency_column }})  
  
{% endmacro %}
```

Code snippet from fact_transaction table using the macro

```
with acceptance as (  
  
    select  
  
        transaction_id,  
        transaction_date,  
        state,  
        amount,  
        country,  
        currency,  
  
        -- Get the USD rate using get_rate macro  
        {{ get_rate('exchange_rates', 'currency') }} as currency_rate  
  
    from {{ ref('stg_acceptance') }}  
)
```

- For the data validation, it was added tests in the schema.yml file for several columns in the staging and fact tables, for example:
 - Transaction_id → unique and not_null
 - Transaction_date → not_null
 - State → accepted_values: values: ['ACCEPTED', 'DECLINED']
 - Amount → not_null

- For the documentation, dbt's documentation features create comprehensive documentation for your models, including descriptions and sources for each table and column.
 - It was added every column and table description in the schema.yml and source.yml files, to have a better comprehension of the dbt docs page.

Part 2

The fact_transactions table was modeled so the data analysts could deep dive into the data and get insight information regarding the origin of the payments that had chargeback, the amount of transactions in a chargeback, and so on.

In order to answer the 3 questions proposed in the assignment, the queries in Snowflake could look like:

1. What is the acceptance rate over time?

```
select
  to_char(date_trunc('month', transaction_date), 'yyyymm') as month,
  count(case when state = 'ACCEPTED' then 1 end) * 1.0 / count(*) as
acceptance_rate
from analytics.fact_transactions
group by 1;
```

The result will be:

↳ Results ~ Chart			
	↑ MONTH	ACCEPTANCE_RATE	
1	201901	0.695699	
2	201902	0.701190	
3	201903	0.689247	
4	201904	0.677778	
5	201905	0.693548	
6	201906	0.716667	

2. List the countries where the amount of declined transactions went over \$25M

```
select
    country,
    sum(amount) as total_declined_amount
from analytics.fact_transactions
where state = 'DECLINED'
group by country
having sum(amount) > 25000000;
```

Result:

```
45 select
46     country,
47     sum(amount) as total_declined_amount
48 from
49     analytics.fact_transactions
50 where
51     state = 'DECLINED'
52 group by
53     country
54 having
55     sum(amount) > 25000000;
```

↳ Results

~ Chart

	COUNTRY	TOTAL_DECLINED_AMOUNT	
1	AE	26335152.43	
2	US	25125669.78	
3	CA	25583266.66	

3. Which transactions are missing chargeback data?

Since the chargeback column comes from a left join between the acceptance table and the chargeback table, the transactions without chargeback data would have the column chargeback null.

```
select
    count(distinct transaction_id) as unique_transactions
from
    analytics.fact_transactions
where
    Chargeback is null;
```

Answer: **NONE**

```
18
19
20 select
21     count(distinct transaction_id) as unique_transactions
22 from
23     analytics.fact_transactions
24 where
25     chargeback is null;
26
27
```

		UNIQUE_TRANSACTIONS
1		0

The same result would be obtained by doing a double check using a subquery with the staging chargeback table.

```
27
28 select
29     count(distinct transaction_id) as unique_transactions
30 from
31     analytics.fact_transactions
32 where
33     transaction_id not in
34     (
35         select
36             transaction_id
37         from
38             analytics.stg_chargeback
39     );
40
41
42
```

		UNIQUE_TRANSACTIONS
1		0