

HANDS-ON DATA SCIENCE TOOLS

TP - Analytics Engineering Exercise with DBT, PostgreSQL, and Metabase

3. Creating staging models

3.1. Staging sales data

Visualisation de toutes les données du fichier stg_sales.sql préalablement créé

The screenshot shows the Metabase interface. On the left, a code editor displays the SQL for stg_sales.sql:

```
models > staging > stg_sales.sql
1  {{ config(materialized='view') }}
2
3  select
4      sale_id,
5      product_id,
6      quantity::integer as quantity,
7      sale_date::date as sale_date
8  from {{ ref('raw_sales_data') }}
9
10 select * from stg_sales
```

On the right, a 'Nouvelle question' (New question) window is open, showing a table visualization of the stg_sales data. The table has columns: sale_id, product_id, quantity, and sale_date. The data is as follows:

sale_id	product_id	quantity	sale_date
1	1	1	mars 23, 2023
2	2	2	février 5, 2023
3	1	5	janvier 18, 2023
4	4	1	janvier 12, 2023
5	1	2	janvier 4, 2023
6	1	5	janvier 30, 2023
7	4	2	janvier 26, 2023

3.2. Staging product data

Création du fichier stg_products.sql

```
models > staging > stg_products.sql
1  {{ config(materialized='view') }}
2
3  select
4      product_id,
5      product_name::varchar(255) as product_name,
6      category::varchar(50) as category,
7      price::numeric(10, 2) as price
8  from {{ ref('raw_product_data') }}
9
```

Visualisation de toutes les données du fichier

The screenshot shows the Metabase interface. On the left, a code editor displays the SQL for stg_products.sql:

```
models > staging > stg_products.sql
1  {{ config(materialized='view') }}
2
3  select
4      product_id,
5      product_name::varchar(255) as product_name,
6      category::varchar(50) as category,
7      price::numeric(10, 2) as price
8  from {{ ref('raw_product_data') }}
9
```

On the right, a 'Nouvelle question' (New question) window is open, showing a table visualization of the stg_products data. The table has columns: product_id, product_name, category, and price. The data is as follows:

product_id	product_name	category	price
1	Widget A	Widgets	19.99
2	Widget B	Widgets	25.99
3	Gadget A	Gadgets	29.99
4	Gadget B	Gadgets	15.99

3.3 Testing staging models

Création des tests d'unicité et de contraintes non nulles sur `sale_id` du fichier `stg_sales` et sur `product_id` du fichier `stg_products`

```
models > staging > ! models.yml
1  version: 2
2
3  models:
4    - name: stg_sales
5      description: "A starter dbt model"
6      columns:
7        - name: sale_id
8          description: "The primary key for this table"
9          tests:
10             - unique
11             - not_null
12
13    - name: stg_products
14      description: "A starter dbt model"
15      columns:
16        - name: product_id
17          description: "The primary key for this table"
18          tests:
19             - unique
20             - not_null
21
```

4. Marts models

4.2 Example

Création du fichier `daily_sales_volume.sql`

```
models > marts > daily_sales_volume.sql
1  {{ config(materialized='table') }}
2
3  select
4    sale_date,
5    count(*) as total_sales
6  from {{ ref('stg_sales') }}
7  group by sale_date
```

4.3 Exercice


Création du fichier `daily_sales_revenue_by_category.sql` dans lequel nous devons définir la requête SQL pour afficher la recette quotidienne triée par catégorie de produit.

```
models > marts > daily_sales_revenue_by_category.sql
1  {{ config(materialized='table') }}
2
3  select
4    s.sale_date,
5    p.category,
6    sum(s.quantity * p.price) AS daily_revenue
7  from {{ ref('stg_sales') }} s
8  join {{ ref('stg_products') }} p on s.product_id = p.product_id
9  group by p.category, sale_date
10 order by s.sale_date ASC
```

5. Dashboard creation with Metabase


- **Daily Sales Trend: A line chart showing sales over time.**

Afin d'afficher une courbe du nombre de ventes réalisées hebdomadairement, nous utilisons le fichier `daily_sales_volume.sql`.

```
models > marts >  daily_sales_volume.sql
1  {{ config(materialized='table') }}
2
3  select
4      sale_date,
5      count(*) as total_sales
6  from {{ ref('stg_sales') }}
7  group by sale_date
```


- **Monthly Sales Trend: A line chart showing sales by month.**

Création du fichier `monthly_sales_volume.sql` avec la requête SQL permettant d'afficher le nombre de ventes par mois


```
models > marts >  monthly_sales_volume.sql
1  {{ config(materialized='table') }}
2
3  select
4      EXTRACT(MONTH FROM s.sale_date) AS sale_month,
5      count(*) as total_sales
6  from {{ ref('stg_sales') }}
7  group by EXTRACT(MONTH FROM s.sale_date)
8
```

- **Sales by Category: A pie chart showing the revenue contribution by category.**

Création du fichier `contribution_revenue_by_category.sql` avec la requête SQL permettant d'afficher la contribution aux revenus par catégorie de produit

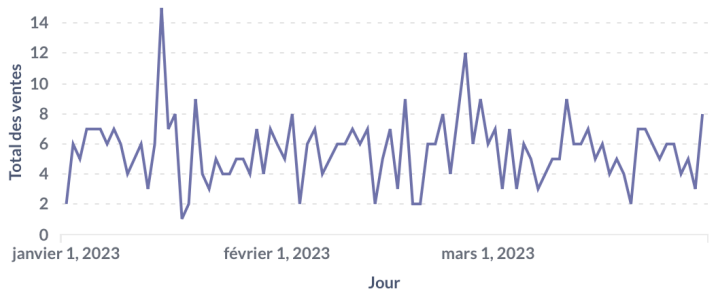
```
models > marts >  contribution_revenue_by_category.sql
1  {{ config(materialized='table') }}
2
3  select
4      p.category,
5      sum(s.quantity * p.price) AS daily_revenue
6  from {{ ref('stg_sales') }} s
7  join {{ ref('stg_products') }} p on s.product_id= p.product_id
8  group by p.category
9
```

- **Category Performance Over Time: Line charts for each product category's trends.**

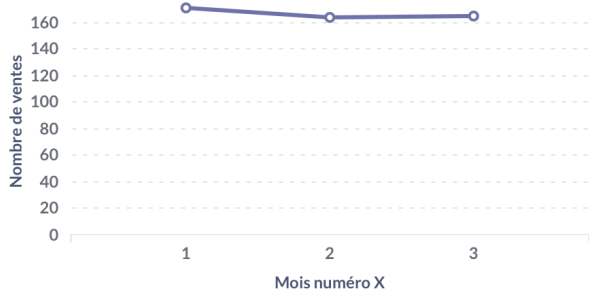
```
models > marts >  category_performance.sql
1  {{ config(materialized='table') }}
2
3  select
4      p.category,
5      s.sale_date,
6      count(*) as total_sales
7  from {{ ref('stg_sales') }} s
8  join {{ ref('stg_products') }} p on s.product_id= p.product_id
9  group by p.category, s.sale_date
```

Dashboard final

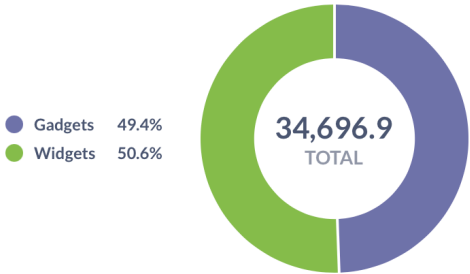
Daily Sales Trend



Monthly Sales Trend



Sales by category



Category Performance Over Time

