Etude de cas SQL

Introduction

Le but de ce test technique est d’évaluer votre raisonnement face à des problématiques data tout en présentant la variété des tâches effectuées chez Dacker. Ainsi :

* Les **4 parties sont** **indépendantes** - bien que liées - il n’est pas nécessaire de finir la précédente pour passer à la suivante.
* Il y a rarement une réponse unique à chaque question. La méthode et le raisonnement sont plus importants que le résultat.
* Le format de vos réponses est libre, mais elles devront être **transmises par email en amont de la restitution**.
* Nous restons à votre disposition au besoin pour toute question.

Setup

Ce test a été conçu pour utiliser le langage PostgreSQL ([documentation](https://www.postgresql.org/docs/)).

Pour tester son code, il est possible et recommandé d’utiliser un environnement “sandbox” en ligne, comme DB Fiddle ([lien](https://www.db-fiddle.com/)).

Pour la partie visualisation, nous recommandons d’utiliser Looker Studio ([lien](https://lookerstudio.google.com/)), mais il est tout à fait possible d’utiliser un autre outil.

# 

# 

# Contexte général

Dans cette étude de cas, nous allons nous intéresser aux données de ventes d’un site e-commerce. On dispose donc de commandes (***orders***), du contenu des commandes (***order\_items***) et de transactions (***charges***).

# Ingestion & Préparation de la table *orders*

Les commandes effectuées par des clients finaux sur le site sont enregistrées et traitées de manière opérationnelle par l’entreprise en utilisant un système de gestion de commandes (exemple : Shopify).

Les informations concernant ces commandes sont récupérées par connexion API, qui nous fournit les 2 fichiers suivants au format CSV :

* raw\_orders\_2022-07-06.csv
* raw\_orders\_2022-07-20.csv

Le code permettant l’*ingestion du 1er fichier* dans une base de données (BDD) PostgreSQL, dans une table qu’on appellera ***raw\_orders***, est fourni dans le fichier *raw\_orders.sql*

Ces 3 fichiers se trouvent dans le dossier “1 - Ingestion” fourni.

**Questions**

1. Modifier *raw\_orders.sql* afin d’ingérer également dans la BDD les données contenues dans le fichier csv datant du 20/07/2022.
2. La colonne **id** est-elle alors une clé primaire de la table raw\_orders ? Expliquer
3. Ecrire une requête SQL permettant d’avoir 1 ligne par **id**. Justifier la méthode employée.

**Dans la suite du test technique, on n’utilisera plus la table *raw\_orders* pour l’analyse des commandes du site, mais la table *orders* issue de la question 3.**

**Nous considérons également qu’un traitement similaire a été effectué pour créer les tables *order\_items* & *charges***

# Transformation & Etude des données

Le fichier **data\_model.sql** fourni contient un sample de données permettant de créer les tables *orders*, *order\_items* & *charges*. Ces tables sont aussi fournies en format csv à titre purement indicatif.

Chaque ligne de chaque table dispose d’un id qui est une clé primaire. Les foreign keys sont désignées par <lenomdelautretable>\_id.

*Par exemple order\_id désigne l’id de l’order associée dans la table orders*

La table *customers* existe mais n’est pas représentée.

## Etude des produits

1. Ecrire une requête SQL qui permet d’obtenir, pour chaque produit, la quantité

totale vendue, le revenu total, le prix moyen de vente, le nombre de clients ayant

acheté le produit, la date de première vente du produit.

1. Ecrire une requête SQL permettant d’obtenir, pour chaque produit, le pays où il a

été le plus vendu.

## Réconciliation paiements/montant de commande

1. Ecrire une requête SQL renvoyant une ligne par order, avec le montant de la

commande, et le montant total des charges (transactions bancaires) associées à cette order, ainsi que la différence entre les 2, en valeur et en pourcentage.

1. En appliquant cette requête aux données fournies, quel commentaire peut-on

faire ? Est ce que tout semble cohérent ?

## Cohorts LTV

**Définitions :** On appelle Customer Lifetime Value (LTV) le montant total des dépenses effectuées par un client depuis son premier achat sur le site.   
Dans cette partie, on va chercher à suivre cette LTV au cours du temps selon des cohortes mensuelles de clients, définies selon le mois de leur premier achat.

1. Ecrire une requête SQL associant à chaque commande la date de première

commande du client ayant passé la commande.

1. Ecrire une requête permettant de suivre la LTV de chaque cohort de première achat. Le résultat devra avoir cette forme (les données ne sont pas cohérentes avec celles en haut, c’est un exemple):

| Mois d’achat | Premier mois d’achat | LTV |
| --- | --- | --- |
| 0 | Janvier 2022 | 1000 |
| 1 | Janvier 2022 | 1200 |
| 0 | Février 2022 | 900 |
| 1 | Février 2022 | 1500 |
| … | … | … |

Le tableau ci-dessus se lit ainsi:

“Les clients ayant passé leur première commande en Janvier 2022 ont dépensé 1000 au mois 0 (donc en Jan 2022) et 200 (1200-1000) au mois 1 (donc en Févr 2022) ”

“Les clients ayant passé leur première commande en Février 2022 ont dépensé 900 au mois 0 (donc en Févr 2022) et 600 (1500-900) au mois 1 (donc en Mars 2022) ”

1. A quoi sert ce type d’analyse selon vous ? Avec quels autres éléments faut-il la

mettre en perspective ?

**On supposera par la suite que la table *ltv\_cohorts*, résultat de la question 2**, **a été créée.**

# Visualisation

Créer un dashboard permettant de représenter de façon pertinente les KPIs calculés dans la partie 2. Fournir un accès au dashboard ou bien des captures d’écran.

**NB:** Les fichiers CSV de la partie 2 peuvent être utilisés ici dans Looker Studio - ou l’outil de visualisation de votre choix -, mais on considérera qu’en tant normal, ce sont plutôt les tables de la base de données qui sont connectées à la place.

# Architecture (bonus)

1. Résumer les différentes étapes successives nécessaires à l’affichage des données de la table **ltv\_cohorts** dans un dashboard, en commençant par le système de gestion de commandes du site e-commerce. Un schéma explicatif peut être utilisé.
2. Proposer une méthode pour automatiser ces étapes afin que les données soient mises à jour quotidiennement.