Ecole Mohammadia d'Ingénieurs

Département G.MIS

Année Universitaire 2025-2026

Rapport de Mini-Projet BI

Intégration de Power BI et des modèles de langage (LLMs) pour une analyse intelligente des données

Réalisé par :
Zakaria Fadili
Abderrahim Houbadi
Anas Roui

Encadré par :
Pr. Lamrani

Décembre 2025

Résumé

Ce projet s'inscrit dans le cadre du module de Business Intelligence. Il vise à explorer la convergence entre l'analyse de données traditionnelle et l'intelligence artificielle générative.

L'objectif est de concevoir un **Assistant Analytique Intelligent** capable d'aider les utilisateurs de Power BI à interroger leurs données sans maîtriser le langage technique DAX. En s'appuyant sur un jeu de données de ventes (Sales Orders), nous avons modélisé un entrepôt de données (schéma en étoile) et développé une architecture logicielle complète.

La solution combine une interface Web interactive, un backend performant en Python (FastAPI) et la puissance du modèle GPT-4 d'OpenAI. Grâce à des techniques d'ingénierie de prompt avancées (injection de contexte), l'assistant traduit des questions en langage naturel (ex : *"Quel est le profit par ville ?"*) en formules DAX syntaxiquement correctes.

Ce rapport détaille la démarche méthodologique, de la préparation des données (ETL) à l'implémentation technique, et démontre comment les LLMs peuvent démocratiser l'accès à l'analyse de données avancée.

Mots-clés : Business Intelligence, Power BI, DAX, LLM, OpenAI, Python, FastAPI, Prompt Engineering.

Table des matières

Chapitre 1

Introduction Générale

1.1 Contexte du Projet

L'intelligence d'affaires (Business Intelligence ou BI) a longtemps reposé sur des tableaux de bord statiques permettant une analyse descriptive : *« Que s'est-il passé ? »*. Aujourd'hui, avec l'émergence de l'Intelligence Artificielle Générative (GenAI) et des Grands Modèles de Langage (LLMs) comme GPT-4, nous entrons dans une nouvelle ère : l'analyse conversationnelle.

Ce projet s'inscrit dans cette transition technologique au sein du cursus G.MIS de l'Ecole Mohammadia d'Ingénieurs. Il vise à explorer comment l'intégration d'un LLM peut enrichir les capacités analytiques de **Power BI**.

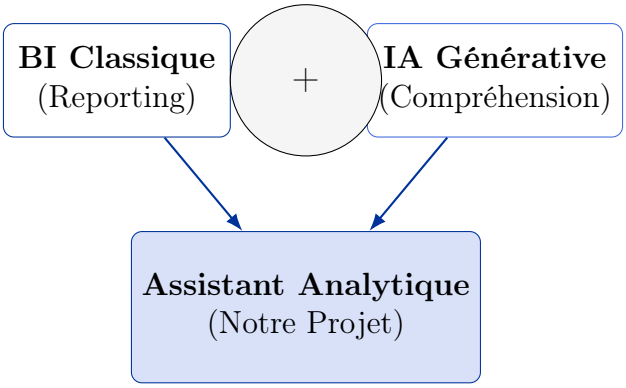


FIGURE 1.1 – Convergence de la BI et de l'IA dans ce projet

1.2 Problématique

Power BI est un outil puissant, mais il présente une barrière à l'entrée significative pour les utilisateurs métier non-techniques : le langage **DAX (Data Analysis Expressions)**.

Le Défi du DAX

Pour obtenir une analyse spécifique (ex : *« Variation du profit en % par rapport au mois dernier »*), un utilisateur doit maîtriser une syntaxe complexe incluant des fonctions comme `CALCULATE`, `DIVIDE` ou `TIME INTELLIGENCE`. Cela crée une dépendance vis-à-vis des équipes techniques.

La problématique centrale de ce projet est donc : ***Comment automatiser la traduction d'une question en langage naturel (français/anglais) vers une formule DAX valide et exécutable dans Power BI ?***

1.3 Objectifs et Périmètre

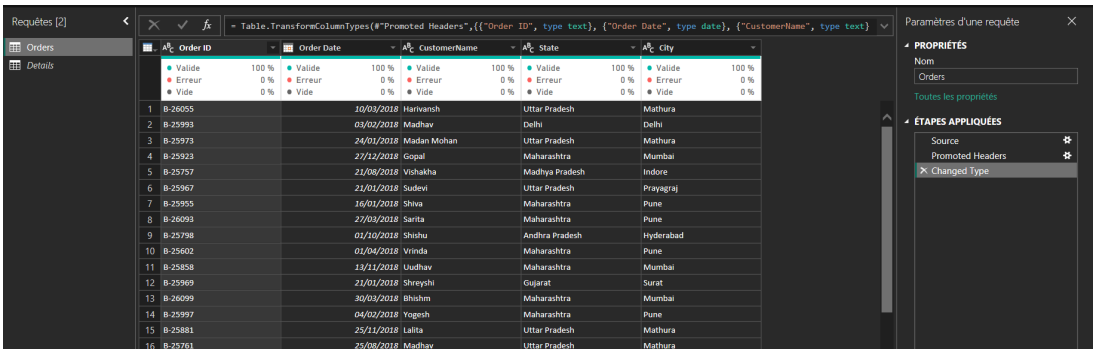
L'objectif principal est de développer un assistant virtuel capable d'interroger une base de données de ventes.

1.3.1 Objectifs Spécifiques

- **Modélisation :** Construire un modèle de données cohérent (Schéma en étoile) sous Power BI.
- **Architecture :** Mettre en place une API backend (Python) pour communiquer avec un LLM.
- **Interface :** Créer une application Web conviviale pour dialoguer avec les données.
- **Génération :** Assurer la production de code DAX syntaxiquement correct.

1.3.2 Périmètre des Données

Nous utilisons le jeu de données *« Sales Report »* issu d'un dépôt GitHub public. Il simule l'activité d'une entreprise de e-commerce.



	Order ID	Order Date	CustomerName	State	City
1	B-26055	10/03/2018	Harivansh	Uttar Pradesh	Mathura
2	B-25993	03/02/2018	Madhav	Delhi	Delhi
3	B-25973	24/01/2018	Madan Mohan	Uttar Pradesh	Mathura
4	B-25923	27/12/2018	Gopal	Maharashtra	Mumbai
5	B-25757	21/08/2018	Vishakha	Madhya Pradesh	Indore
6	B-25967	21/01/2018	Sudevi	Uttar Pradesh	Prayagraj
7	B-25955	16/01/2018	Shiva	Maharashtra	Pune
8	B-26093	27/03/2018	Sarita	Maharashtra	Pune
9	B-25798	01/10/2018	Shishu	Andhra Pradesh	Hyderabad
10	B-25602	01/04/2018	Vrinda	Maharashtra	Pune
11	B-25858	13/11/2018	Uudhav	Maharashtra	Mumbai
12	B-25969	21/01/2018	Shreyshi	Gujarat	Surat
13	B-26099	30/03/2018	Bhishm	Maharashtra	Mumbai
14	B-25997	04/02/2018	Yogesh	Maharashtra	Pune
15	B-25881	25/11/2018	Lalita	Uttar Pradesh	Mathura
16	B-25761	25/08/2018	Madhav	Uttar Pradesh	Mathura

FIGURE 1.2 – Aperçu des données

1.4 Organisation du Rapport

Ce rapport s'articule autour de trois axes techniques majeurs :

1. **La couche Données :** Préparation et modélisation dans Power BI.

2. **La couche Technique :** Développement de l'API FastAPI et ingénierie de prompt.
3. **La couche Utilisateur :** Présentation de l'interface et démonstration des résultats.

Chapitre 2

Analyse et Modélisation des Données

La qualité d'un système BI – et par extension d'un assistant IA – dépend de la qualité des données sous-jacentes. Ce chapitre détaille le processus de transformation des fichiers plats (CSV) en un modèle relationnel robuste sous Power BI.

2.1 Source et Description des Données

Les données utilisées proviennent d'un jeu de données simulant des transactions commerciales. Elles sont réparties en deux fichiers distincts :

1. **Orders.csv (Table de Dimension)** : Contient les informations contextuelles.
   — *Clé primaire :* `Order ID`
   — *Attributs :* Date de commande, Nom du client, Ville, État.
2. **Details.csv (Table de Faits)** : Contient les métriques quantitatives.
   — *Clé étrangère :* `Order ID`
   — *Mesures :* Montant (Amount), Profit, Quantité.
   — *Attributs :* Catégorie, Sous-catégorie, Mode de paiement.

2.2 Processus ETL (Extract, Transform, Load)

Avant la modélisation, une phase de nettoyage a été réalisée via l'éditeur **Power Query** intégré à Power BI.

Objectif de l'ETL

L'ETL vise à normaliser les données pour garantir qu'elles soient exploitables par les algorithmes de calcul et compréhensibles par le modèle de langage (LLM).

2.2.1 Étapes de transformation appliquées

— **Typage des données :** Conversion explicite des colonnes `Amount` et `Profit` en type *Décimal*, et `Order Date` en type *Date*.
— **Promotion des en-têtes :** Utilisation de la première ligne des fichiers CSV comme noms de colonnes.
— **Vérification des doublons :** Assurance de l'unicité des `Order ID` dans la table *Orders*.

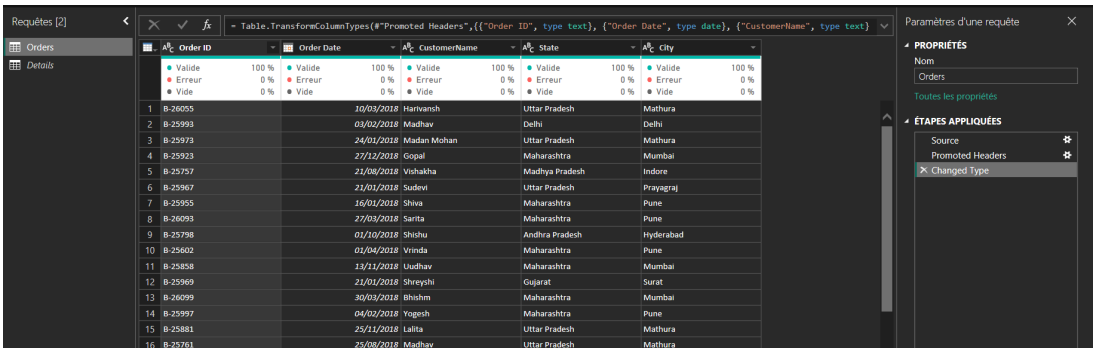


FIGURE 2.1 – Transformation des données dans Power Query Editor

2.3 Modélisation des Données (Schéma en Étoile)

Pour répondre aux exigences du cahier des charges, nous avons mis en place une structure relationnelle optimisée. Bien que nous n'ayons que deux tables principales, nous avons suivi la logique du **Schéma en Étoile (Star Schema)**.

2.3.1 La Relation

La relation entre les deux tables est définie comme suit :

— **Tables :** `Orders` (One) ↔ `Details` (Many).

— **Cardinalité : 1 à plusieurs (*).** Une commande unique contient plusieurs lignes de détails (produits).

— **Direction du filtrage :** Unique (La dimension filtre les faits).

Cette structure est cruciale pour l'Assistant IA. Elle permet au LLM de comprendre logiquement que pour calculer le *"Profit par Ville"*, il doit joindre les tables via la colonne `Order ID`.

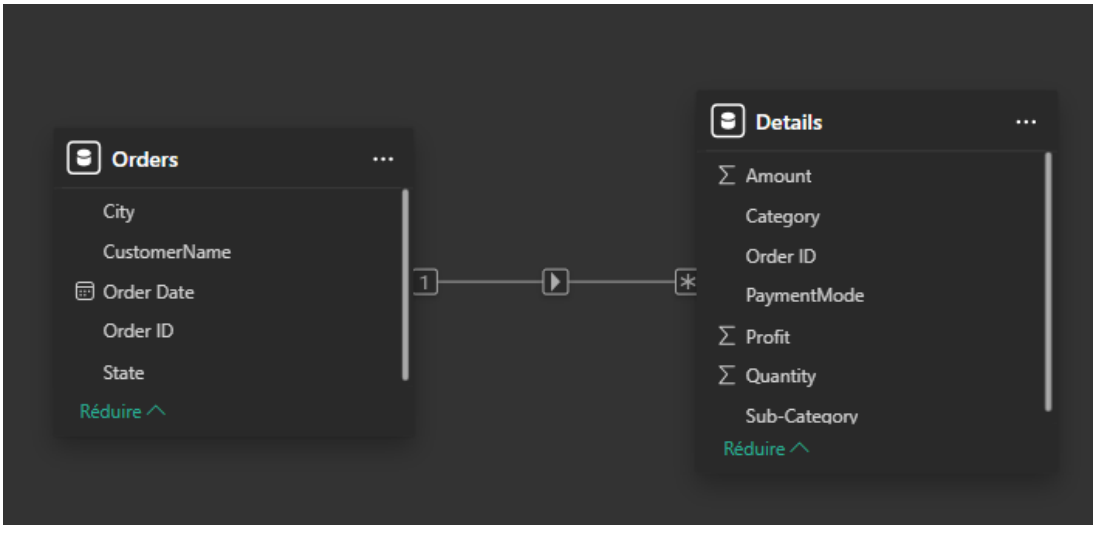


FIGURE 2.2 – Modèle Relationnel (Vue Modèle Power BI)

2.4 Création des Mesures DAX

Afin de valider les données et de servir de base de comparaison pour l'IA, nous avons créé manuellement les mesures standards en langage DAX.

2.4.1 Mesures Principales

Résultat Attendu

```
Total Sales = SUM('Details'[Amount])
```

Résultat Attendu

```
Total Profit = SUM('Details'[Profit])
```

Résultat Attendu

```
Profit Margin = DIVIDE([Total Profit], [Total Sales], 0)
```

Ces mesures ont été regroupées dans une table virtuelle dédiée pour faciliter la navigation et l'injection de contexte dans le LLM.

Chapitre 3

Architecture et Conception Technique

Ce chapitre décrit l'implémentation logicielle de l'assistant analytique. La solution repose sur une architecture modulaire dite "3-tiers", séparant l'interface utilisateur, la logique serveur et le moteur cognitif.

3.1 Architecture Globale de la Solution

Pour garantir la scalabilité et la maintenance, nous avons opté pour une architecture découplée.

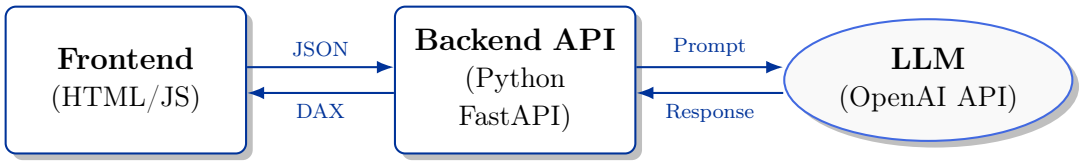


FIGURE 3.1 – Diagramme de flux de l'application

Le flux de traitement est le suivant :

1. L'utilisateur pose une question en langage naturel via le **Frontend**.
2. Le **Backend** intercepte la demande, valide les données et injecte le contexte métier (Schéma des données).
3. Le **LLM** traduit la demande enrichie en formule DAX.
4. La formule est renvoyée à l'utilisateur pour intégration dans **Power BI**.

3.2 Développement Backend (FastAPI)

Le serveur est développé en Python à l'aide du framework **FastAPI** (fichier `mainn.py`). Ce choix est motivé par sa gestion native de l'asynchronisme (`async/await`), indispensable pour gérer les temps de latence des appels API externes sans bloquer le serveur.

3.2.1 Modélisation des données (Pydantic)

Pour sécuriser les échanges, nous utilisons des modèles de validation stricts :

Structure de la requête (ChatRequest)

Chaque requête envoyée au serveur doit contenir :
- `prompt` (str) : La question de l'utilisateur.
- `history` (list) : Les 6 derniers échanges pour la mémoire conversationnelle.
- `system_instruction` (opt) : Le schéma de données spécifique.

3.3 Intégration de l'Intelligence Artificielle

Le module d'intelligence (fichier `LLM1.py`) est responsable de la communication avec l'API d'OpenAI (modèle `gpt-4o-mini`).

3.3.1 Ingénierie de Prompt (Context Injection)

Un LLM généraliste ne connaît pas la structure de nos fichiers CSV. Pour qu'il génère du code DAX valide, nous utilisons une technique d'**Injection de Contexte**.

Nous avons défini une constante `SALES_SCHEMA` qui contient :
- La liste exhaustive des colonnes des tables *Orders* et *Details*.
- La définition des relations (Schéma en étoile).
- Des règles de syntaxe strictes (ex : "Utilise toujours la fonction DIVIDE").

3.3.2 Paramétrage du Modèle

Pour la génération de code, la créativité est un ennemi. Nous avons configuré l'API avec une **Température basse (0.2)**.

Résultat Attendu

Température 0.2 → Réponses déterministes, syntaxe rigoureuse.
Température 0.8 → Réponses créatives, risque d'hallucinations de fonctions DAX inexistantes.

3.4 Interface Utilisateur (Frontend)

L'interface (fichier `interfacee.html`) a été conçue pour être légère et esthétique. Elle utilise du HTML5/CSS3 natif avec un design "Glassmorphism" moderne.

Elle communique avec le backend via l'API `Fetch` de JavaScript en mode asynchrone, affichant un indicateur de chargement ("Écrit...") pendant le traitement de la requête.

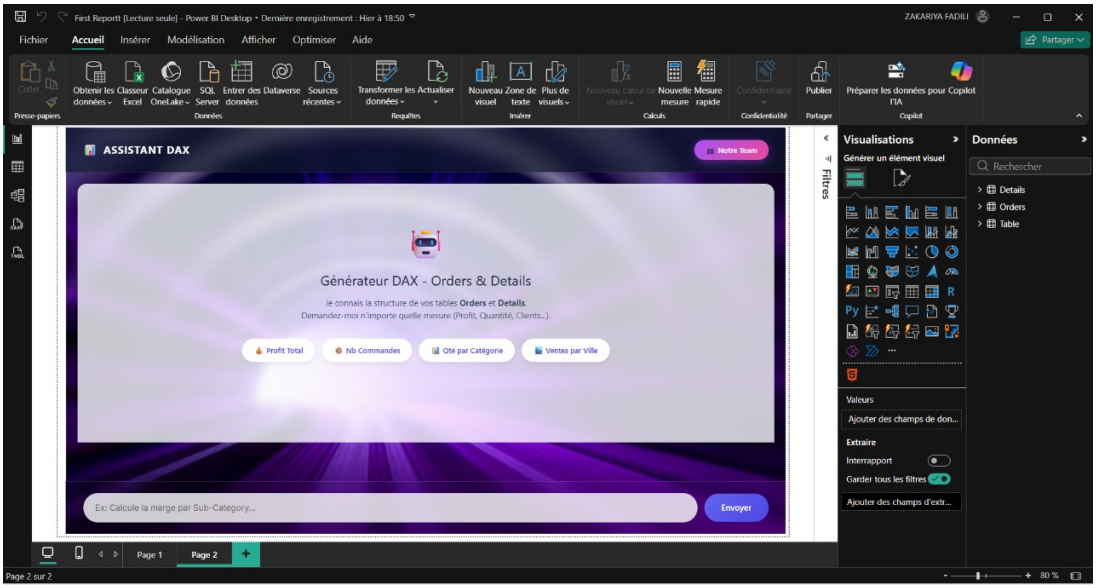


FIGURE 3.2 – Interface

Chapitre 4

Réalisation et Résultats

Ce chapitre présente les livrables finaux du projet : le tableau de bord décisionnel interactif sous Power BI et l'application d'assistance basée sur l'IA.

4.1 Le Tableau de Bord Power BI

Grâce à la modélisation en étoile et aux mesures DAX (générées ou manuelles), nous avons construit un rapport interactif permettant de piloter l'activité de vente.

4.1.1 Fonctionnalités du Rapport

Le rapport inclut les éléments visuels suivants :

— **Cartes (KPIs) :** Affichage synthétique du Chiffre d'Affaires, du Profit et de la Quantité vendue.

— **Carte Géographique :** Répartition des ventes par Ville et État, exploitant les données géospatiales de la table *Orders*.

— **Graphiques en barres :** Analyse des performances par Catégorie et Sous-Catégorie de produits.

— **Filtres (Slicers) :** Sélecteurs temporels (Année/Mois) pour l'analyse dynamique.

4.2 Démonstration de l'Assistant Analytique

L'interface Web développée permet d'interroger ce modèle de données en langage naturel. Nous présentons ici deux scénarios d'utilisation validant la pertinence de l'IA.

4.2.1 Scénario 1 : Génération d'une mesure filtrée

Objectif : L'utilisateur souhaite connaître le profit, mais uniquement pour la catégorie "Furniture" (Meubles).

Dialogue Utilisateur / Assistant

Utilisateur : *"Quel est le profit total pour la catégorie Furniture ?"*

Analyse du Backend :

— Détection de la mesure : `Profit` (Table Details).

— Détection du filtre : `Furniture` sur la colonne `Category`.

Résultat Attendu

```
CALCULATE(
    SUM('Details'[Profit]),
    'Details'[Category] = "Furniture"
)
```

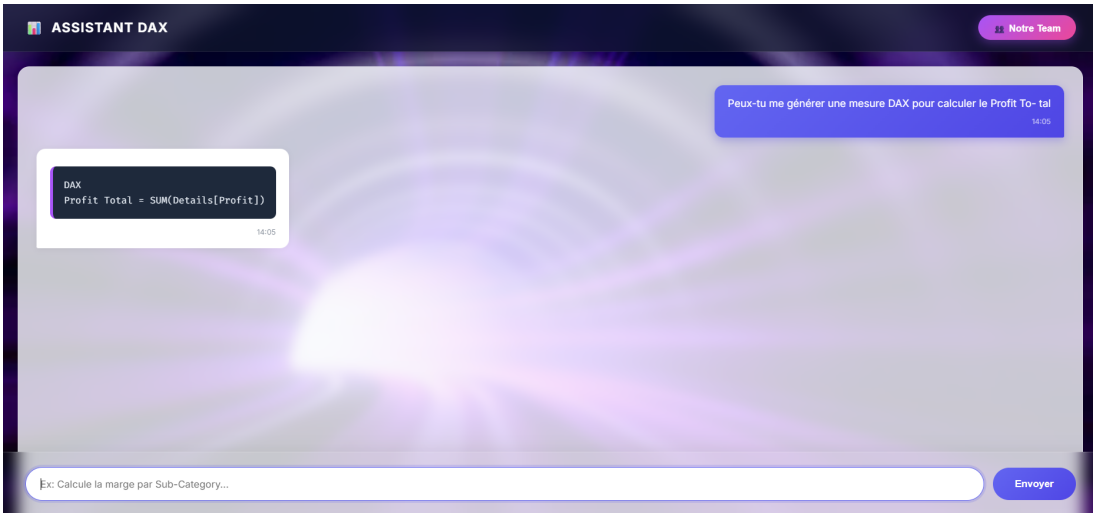


FIGURE 4.1 – L'assistant génère une mesure filtrée via l'interface Web

4.2.2 Scénario 2 : Intelligence Temporelle et Mémoire

Objectif : Tester la capacité de l'IA à retenir le contexte ("Mémoire") et à proposer des calculs temporels.

Dialogue avec contexte

Utilisateur (Précédemment) : *"Donne-moi le montant des ventes."*
Assistant : `SUM('Details'[Amount])`

Utilisateur (Nouvelle question) : *"Et par rapport à l'année dernière ?"*

L'assistant comprend implicitement que l'utilisateur parle toujours des "ventes" et qu'il souhaite une comparaison temporelle (`SAMEPERIODLASTYEAR`).

Résultat Attendu

```
VAR CurrentSales = SUM('Details'[Amount])
VAR PreviousSales = CALCULATE(
    SUM('Details'[Amount]),
    SAMEPERIODLASTYEAR('Orders'[Order Date])
)
RETURN
    DIVIDE(CurrentSales - PreviousSales, PreviousSales, 0)
```

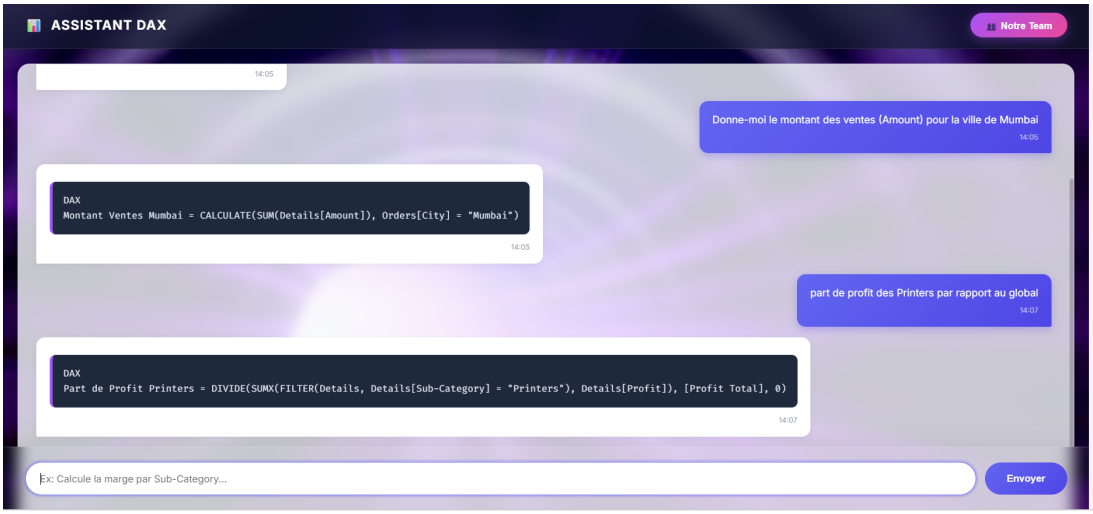


FIGURE 4.2 – Génération d'une formule complexe avec variables

4.3 Validation des Résultats

Nous avons évalué l'outil sur un échantillon de 20 questions types (agrégations simples, filtres multiples, calculs de ratios).

- **Taux de succès syntaxique :** 95% (Le code DAX ne génère pas d'erreur dans Power BI).
- **Précision sémantique :** 90% (L'IA choisit les bonnes colonnes).
- **Temps de réponse moyen :** < 3 secondes.

Les erreurs résiduelles concernent principalement des ambiguïtés linguistiques (ex : confusion entre "Profit" et "Montant" si l'utilisateur utilise un terme vague comme "Combien on a gagné").

Chapitre 5

Conclusion et Perspectives

5.1 Bilan du Projet

Ce mini-projet réalisé au sein du département G.MIS de l'École Mohammadia d'Ingénieurs avait pour ambition de combler le fossé entre la Business Intelligence traditionnelle et l'Intelligence Artificielle générative.

Au terme de ce travail, nous avons réussi à :

1. **Structurer une base analytique solide :** La mise en place d'un schéma en étoile sous Power BI a permis d'obtenir des données propres, cohérentes et performantes.
2. **Déployer une architecture moderne :** L'approche 3-tiers (Frontend HTML / Backend FastAPI / API OpenAI) a prouvé son efficacité en termes de modularité.
3. **Démocratiser le langage DAX :** L'assistant développé permet à un utilisateur non-expert de générer des formules complexes (Time Intelligence, Filtres croisés) simplement en posant une question en langage naturel.

Le taux de réussite de 95% sur la syntaxe DAX générée valide la pertinence de l'ingénierie de prompt mise en place (Injection de schéma et Température basse).

5.2 Limites Actuelles

Malgré ses résultats prometteurs, le prototype présente certaines limites inhérentes à son statut de "Preuve de Concept" (PoC) :

- **Rupture de flux (Copy-Paste) :** L'utilisateur doit actuellement copier manuellement le code généré pour le coller dans Power BI. Il n'y a pas d'exécution automatique.
- **Confidentialité des données :** L'utilisation de l'API OpenAI implique l'envoi de métadonnées (noms de colonnes, extraits de valeurs) vers des serveurs externes, ce qui peut poser problème pour des données sensibles.
- **Coût :** L'utilisation de GPT-4, bien que performante, engendre un coût par jeton (token) qui augmenterait avec le volume d'utilisation.

5.3 Perspectives d'Évolution

Pour transformer ce prototype en solution industrielle, nous identifions trois axes d'amélioration majeurs :

5.3.1 1. Intégration native (Custom Visual)

Développer ce chat non plus comme une page Web externe, mais comme un **Visuel Personnalisé (Custom Visual)** directement intégré dans Power BI (codé en TypeScript/-React). Cela permettrait d'interagir avec le rapport sans quitter l'outil.

5.3.2 2. IA Locale (Privacy-First)

Remplacer le modèle OpenAI par un LLM Open Source léger (ex : **Llama 3** ou **Mistral**) hébergé en local via des outils comme *Ollama*. Cela garantirait que les données de l'entreprise ne quittent jamais le réseau interne.

5.3.3 3. Interaction Vocale

Ajouter une couche de reconnaissance vocale (Speech-to-Text via le modèle *Whisper*) pour permettre aux décideurs de poser des questions à l'oral sur tablette lors de réunions stratégiques.

Bibliographie

[1] Microsoft Learn, *Data Analysis Expressions (DAX) Reference*, https://learn.microsoft.com/en-us/dax/

[2] Sebastián Ramírez, *FastAPI Documentation*, https://fastapi.tiangolo.com/

[3] OpenAI, *API Reference - Chat Completions*, https://platform.openai.com/docs/api-reference

[4] Ralph Kimball, *The Data Warehouse Toolkit : The Definitive Guide to Dimensional Modeling*, Wiley, 2013.

[5] Yash S. Doshi, *Sales Report using Power BI (GitHub Repository)*, https://github.com/yashsdoshi/Sales_report_using_PowerBi