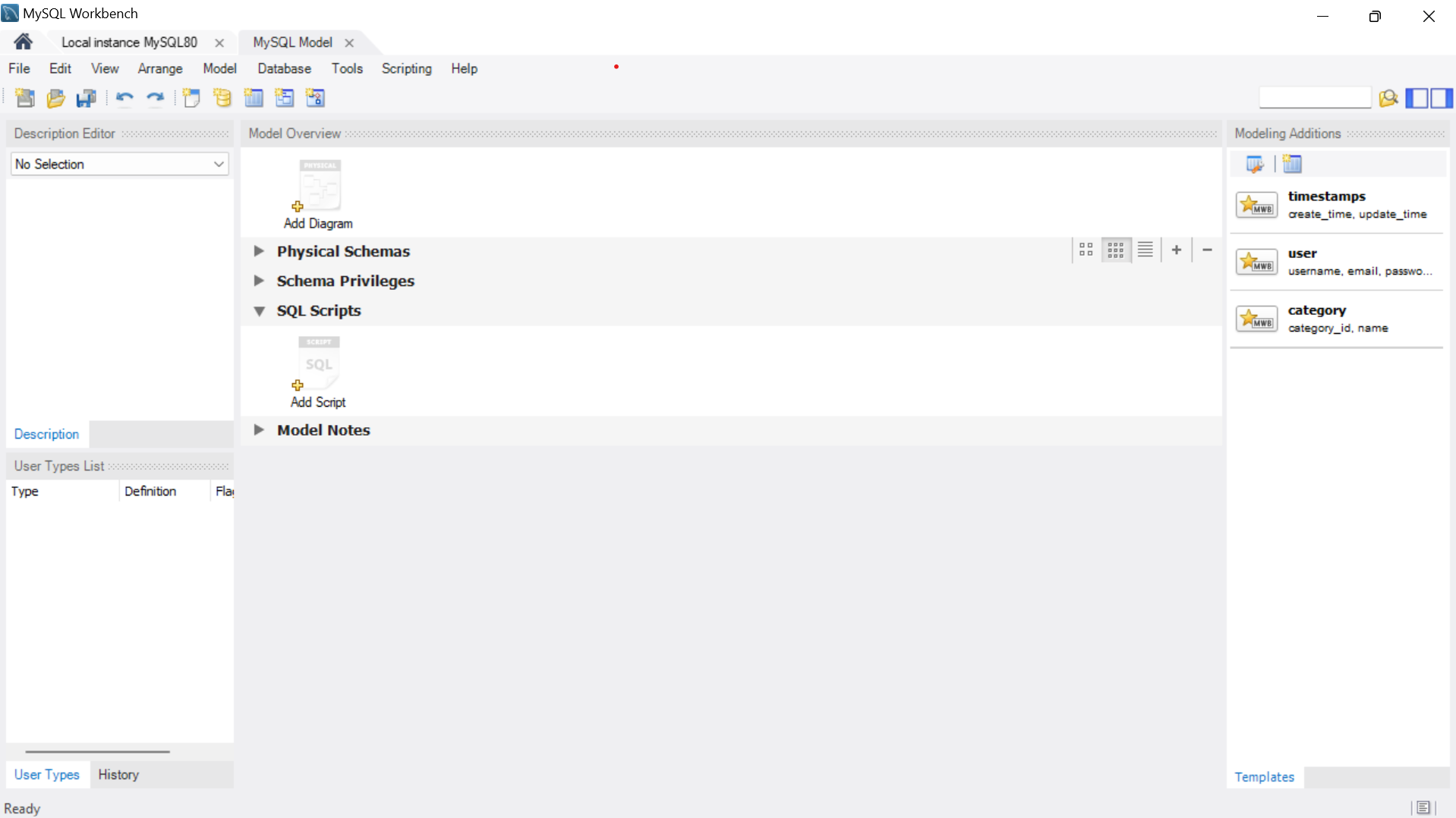
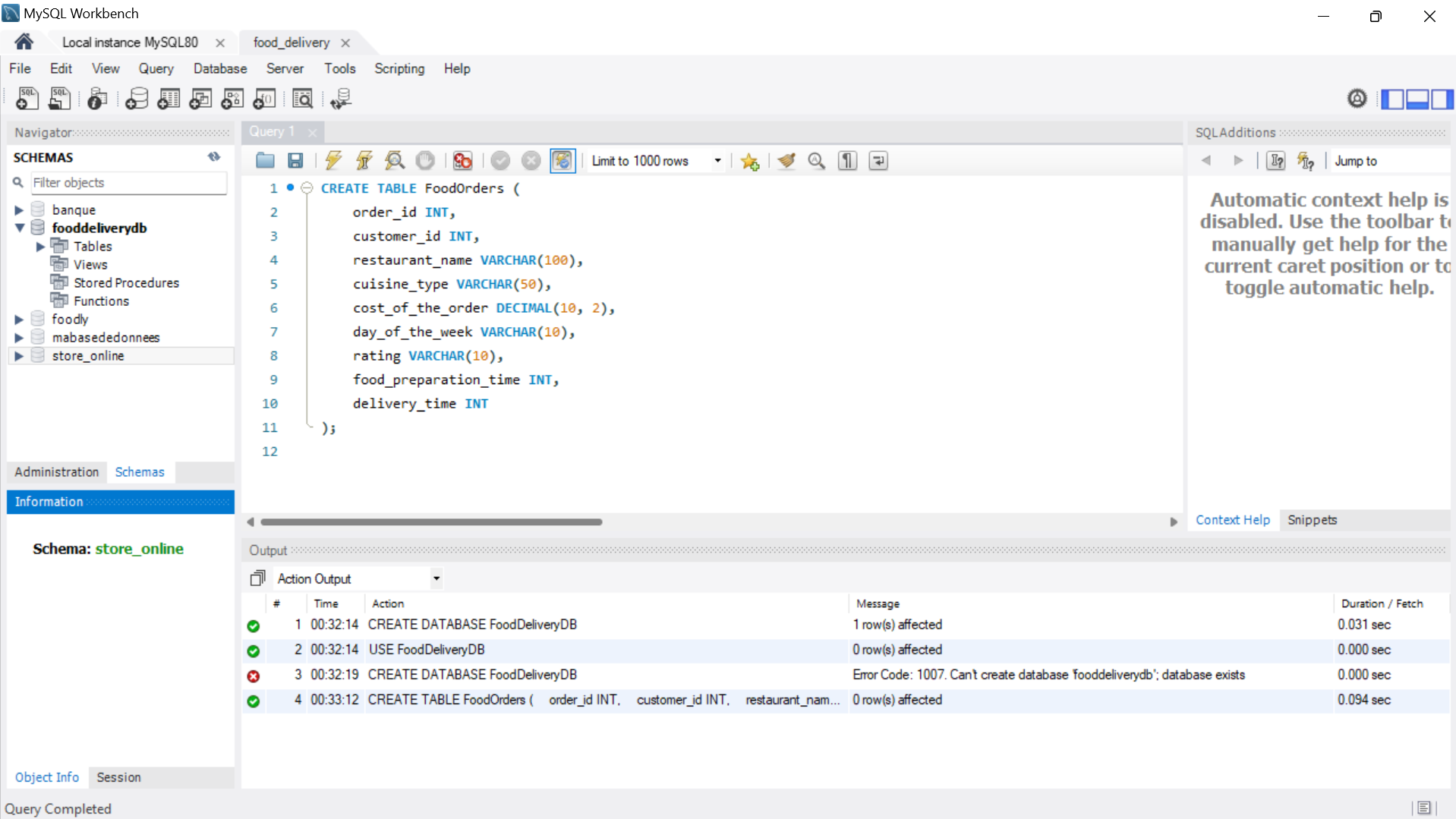
Analyse de la base de donnée Food\_delivery en SQL :

**Créer une nouvelle base de données** dans MySQL Workbench pour organiser les données :



**Créer une table** pour accueillir les données. Voici un exemple de script SQL basé sur les colonnes que tu as fournies :



**1. Compter les commandes par jour (Weekday vs Weekend)**

Cette requête te permettra de voir combien de commandes sont passées les jours de semaine et les weekends :

Une image contenant texte, logiciel, Icône d’ordinateur, nombre

Description générée automatiquement

**2. Temps moyen de préparation et de livraison par jour de la semaine**

Cette analyse aidera à voir si le temps de préparation ou de livraison est plus long en semaine ou le weekend :

Une image contenant texte, logiciel, Icône d’ordinateur, Page web

Description générée automatiquement

**3. Coût moyen par type de cuisine**

Identifier si certains types de cuisines sont plus coûteux que d'autres :

Une image contenant texte, logiciel, Icône d’ordinateur, Page web

Description générée automatiquement

**4. Relation entre le temps de livraison et la satisfaction des clients**

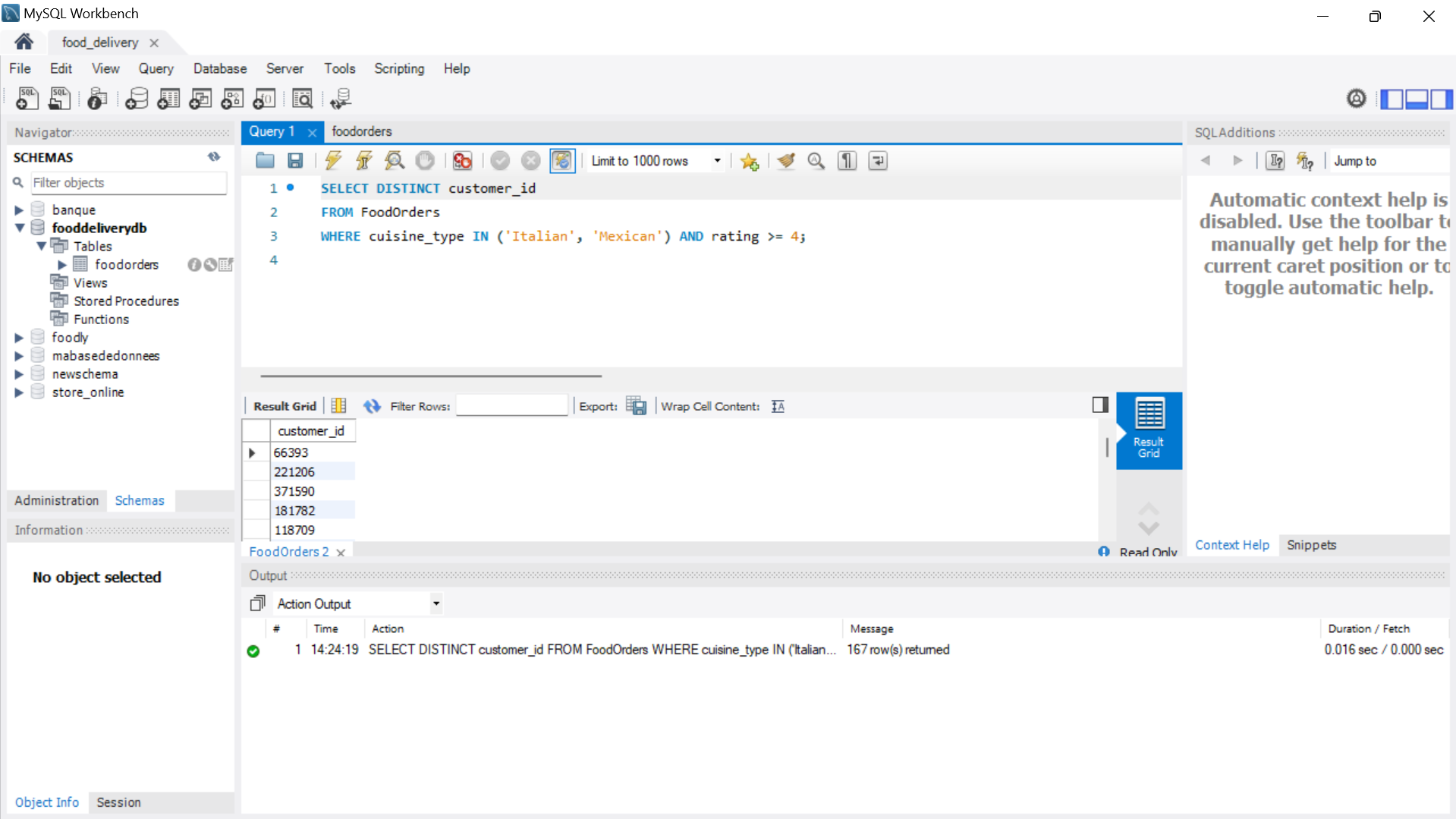
Vérifions si le temps de livraison impacte les notes des clients. Cette requête moyenne les notes par rapport au temps de livraison :

Une image contenant texte, capture d’écran, logiciel, Icône d’ordinateur

Description générée automatiquement

**Scénario** : Trouver tous les **clients** ayant passé une commande dans des restaurants de cuisine "Italian" ou "Mexican" avec une note supérieure ou égale à 4.

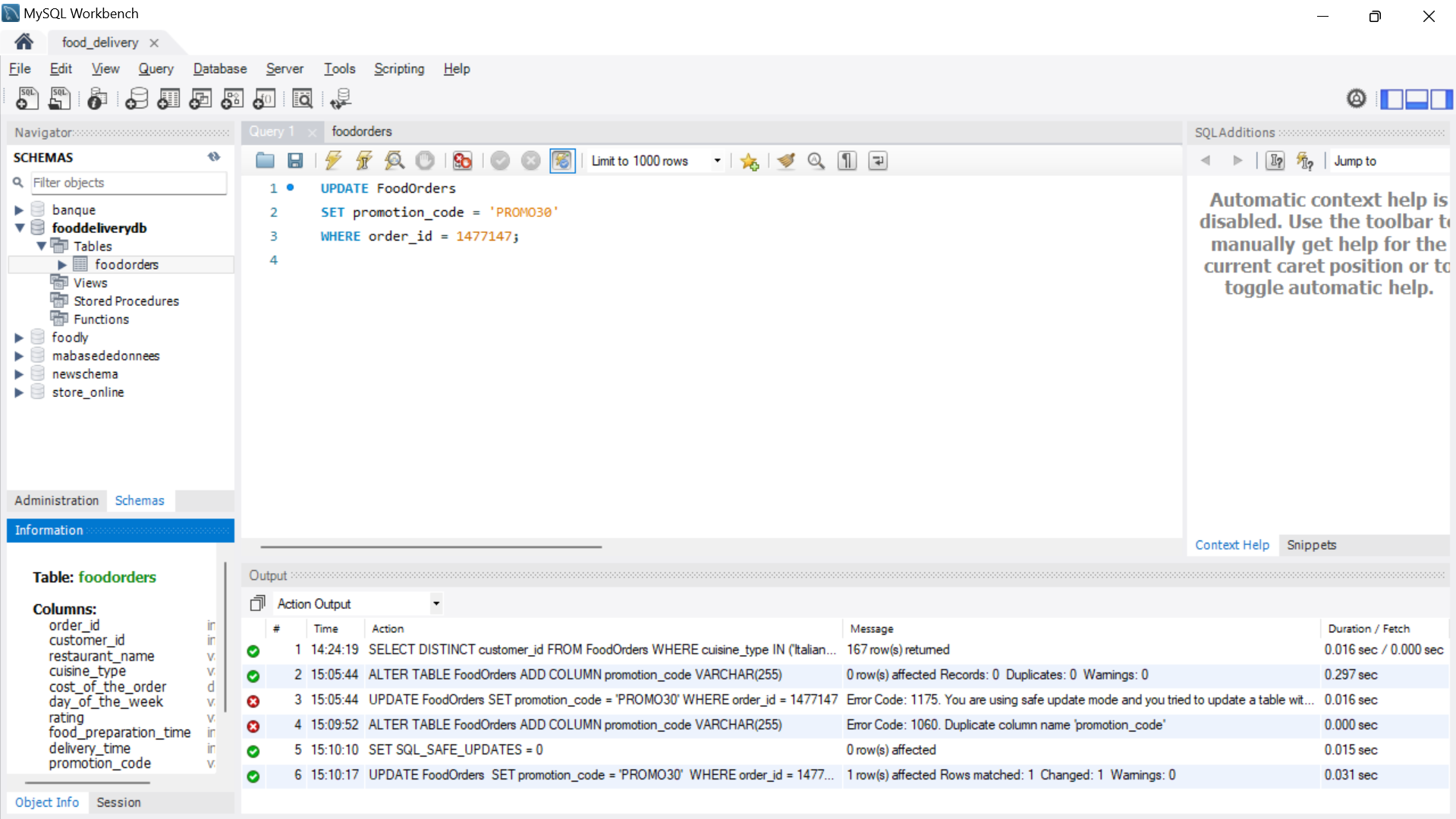
**Requête SQL :**



* 👉 **Pourquoi SQL excelle ici :** Les relations complexes ou les jointures sont bien plus naturelles en SQL.  
  👉 **Pourquoi NoSQL est limité ici :** MongoDB n’a pas de mécanisme direct de jointure, ce qui complique les requêtes avec des relations.

**Cas 2 : Données non structurées et flexibilité (Point fort NoSQL)**

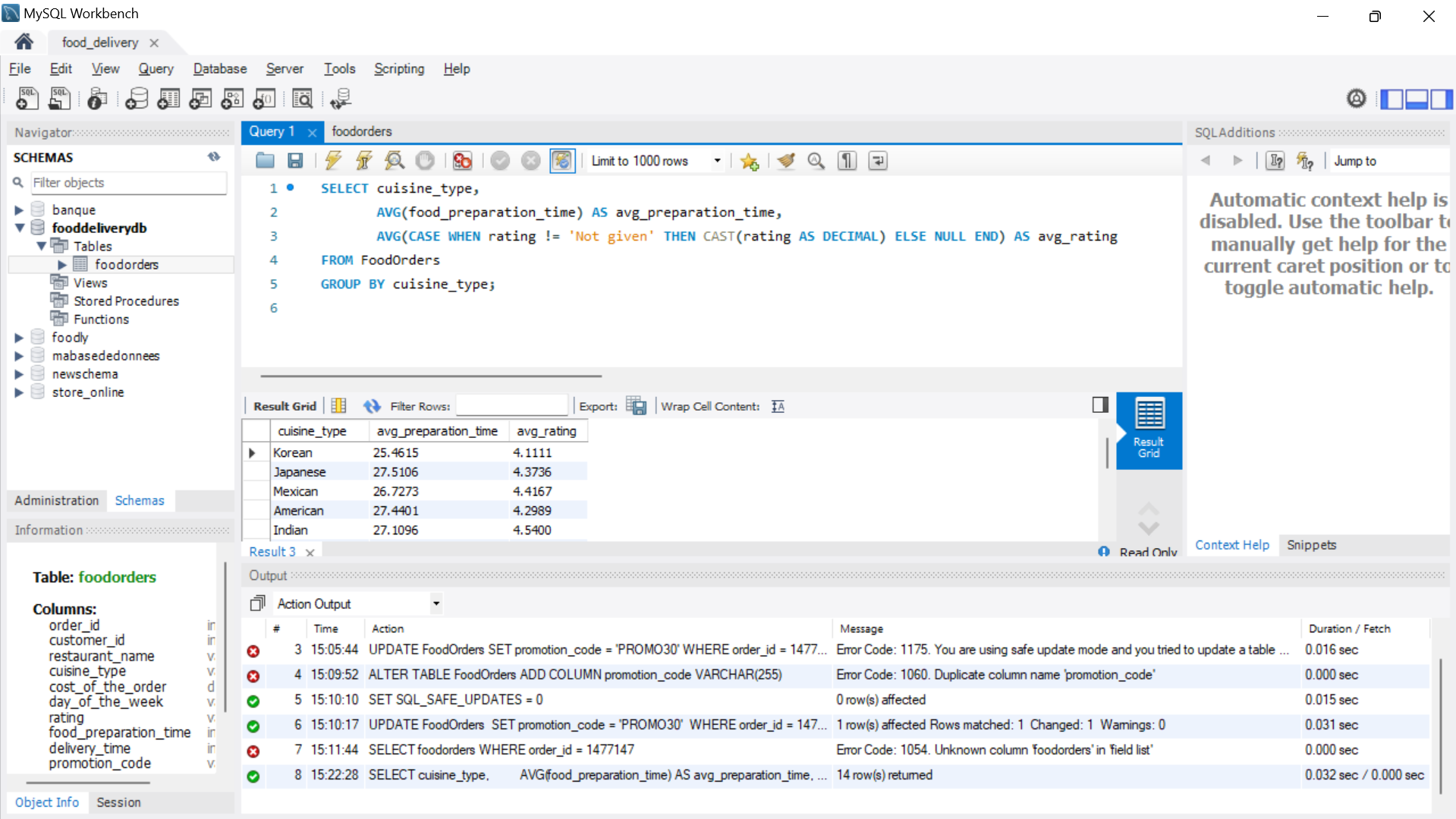
**Scénario** : Ajouter une promotion spéciale pour certaines commandes uniquement (champ dynamique), puis récupérer toutes les commandes ayant une promotion.



👉 **Pourquoi NoSQL excelle ici :** Tu peux ajouter dynamiquement des champs sans modifier la structure générale.  
👉 **Pourquoi SQL est limité ici :** Toute modification structurelle nécessite des changements dans le schéma, ce qui est moins flexible.

**Cas 3 : Agrégations complexes (Forces partagées)**

**Scénario** : Trouver le temps de préparation moyen et la note moyenne pour chaque type de cuisine.

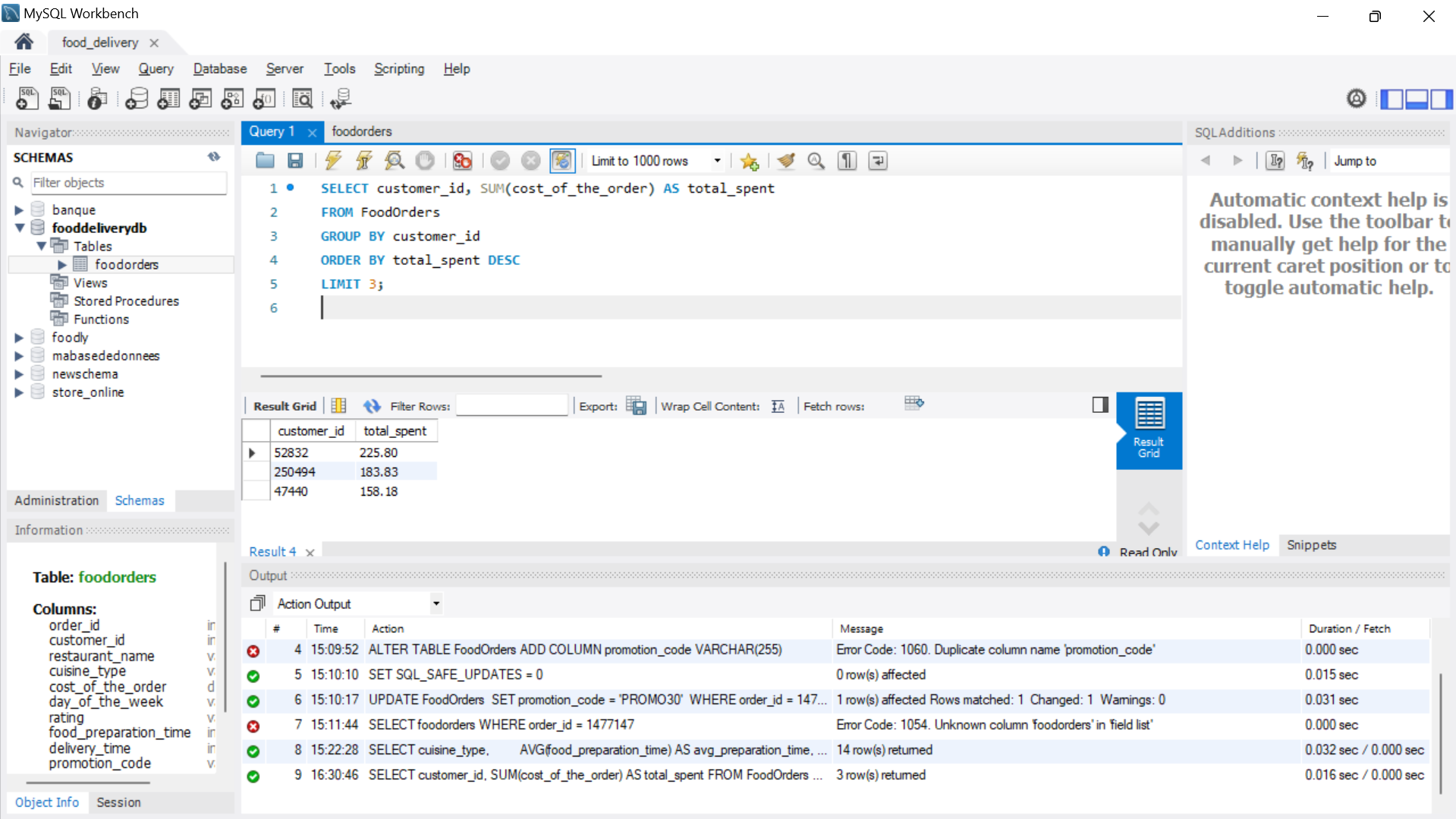


👉 **Points à noter** :

* SQL est **plus intuitif** ici grâce à la syntaxe GROUP BY.
* MongoDB utilise un **pipeline d’agrégation** qui peut paraître plus complexe pour une tâche similaire.

**Cas 4 : Volume massif de données (Point fort NoSQL)**

**Scénario** : Trouver les 3 clients ayant dépensé le plus.



👉 **Pourquoi NoSQL excelle ici :** MongoDB est conçu pour traiter efficacement les requêtes sur des datasets massifs, répartis sur plusieurs serveurs (scalabilité horizontale).  
👉 **Pourquoi SQL peut avoir des limites :** SQL peut nécessiter un index ou des optimisations supplémentaires pour traiter des millions de lignes.

**1. Comparaison SQL vs NoSQL sur le cas d’usage "Food Delivery"**

Pour ton dataset food\_delivery, nous allons structurer la comparaison autour de **trois critères clés** :

* **Facilité d’utilisation et flexibilité des requêtes**.
* **Performance et scalabilité**.
* **Gestion de la structure des données**.

**Facilité d’utilisation et flexibilité des requêtes**

| **Critère** | **SQL** | **NoSQL** |
| --- | --- | --- |
| **Requêtes complexes** | SQL excelle pour les jointures et agrégations avancées, grâce à sa syntaxe structurée (GROUP BY, etc.). | NoSQL est plus limité sans relations natives (jointures), nécessitant un pipeline plus complexe. |
| **Flexibilité des données** | Moins flexible : toute modification de structure (ajout d'une colonne) demande une mise à jour globale. | Très flexible : on peut ajouter dynamiquement des champs à certains documents, sans impacter les autres. |
| **Apprentissage & adoption** | Syntaxe SQL standardisée, facile à apprendre. | MongoDB utilise un paradigme différent (JSON-like), qui peut être déroutant pour les débutants SQL. |

**Performance et scalabilité**

| **Critère** | **SQL** | **NoSQL** |
| --- | --- | --- |
| **Traitement de petits volumes** | SQL est performant pour des bases de taille modeste (jusqu’à quelques millions de lignes). | MongoDB est légèrement moins performant pour de petites bases (overhead de JSON et de l’indexation). |
| **Scalabilité horizontale** | SQL est limité : nécessite un scaling vertical (ajouter plus de puissance à un serveur unique). | MongoDB est conçu pour le scaling horizontal : partage les données entre plusieurs serveurs. |
| **Temps de réponse des requêtes** | SQL est très optimisé pour des requêtes complexes grâce aux index et optimisations de jointure. | MongoDB peut être plus rapide pour des requêtes simples sur des documents volumineux. |

**Gestion de la structure des données**

| **Critère** | **SQL** | **NoSQL** |
| --- | --- | --- |
| **Modèle des données** | Modèle structuré et fixe (basé sur un schéma défini à l’avance). | Modèle flexible, schema-less, parfait pour des données non uniformes. |
| **Migration des données** | Demande un effort important lors de la modification des colonnes ou du schéma. | Plus simple : chaque document peut avoir une structure différente, adapté aux évolutions rapides. |
| **Stockage des données non structurées** | SQL n’est pas adapté pour des données comme JSON ou XML sans extensions supplémentaires (e.g., JSONB de PostgreSQL). | MongoDB est nativement conçu pour des documents JSON-like. |