Method of Django’s ORM (Object Relational Mapping) :

filter(): Filtre les objets en fonction des conditions spécifiées.

exclude(): Exclut les objets correspondant aux conditions spécifiées.

get(): Récupère un seul objet correspondant aux conditions spécifiées.

all(): Récupère tous les objets d'un modèle particulier.

count(): Retourne le nombre d'objets correspondant aux conditions spécifiées.

order_by(): Trie les objets en fonction d'un ou plusieurs champs.

create(): Crée un nouvel objet et l'enregistre dans la base de données.

update(): Met à jour les champs des objets correspondant aux conditions spécifiées.

delete(): Supprime les objets correspondant aux conditions spécifiées.

values(): Retourne un QuerySet de dictionnaires contenant les valeurs des champs spécifiés.

annotate(): Effectue des agrégations sur les objets, telles que le comptage, la somme, la moyenne, etc.

distinct(): Supprime les objets en double du QuerySet.

exists(): Vérifie si au moins un objet correspond aux conditions spécifiées.

select_related(): Récupère les objets liés dans une seule requête pour éviter le problème de la requête N+1.

prefetch_related(): Récupère efficacement les objets liés en utilisant le prefetching.

values_list(): Retourne un QuerySet de tuples contenant les valeurs des champs spécifiés.

first(): Récupère le premier objet du QuerySet.

last(): Récupère le dernier objet du QuerySet.

aggregate(): Effectue des agrégations et retourne un dictionnaire avec les résultats.

distinct(): Retourne un QuerySet avec des objets uniques basés sur les champs spécifiés.

Le prefetching avec Django est une technique qui permet d'optimiser les performances lors du chargement de données liées (related data) à l'aide de requêtes supplémentaires anticipées.

Lorsque vous avez une relation entre plusieurs modèles dans Django, comme une relation ForeignKey ou ManyToManyField, il est courant de devoir accéder aux données liées lors du traitement des données principales. Par exemple, si vous avez un modèle "Article" avec une clé étrangère vers le modèle "Auteur", lorsque vous récupérez une liste d'articles, vous souhaiterez généralement accéder aux auteurs de chaque article.

Sans prefetching, Django effectuerait une requête de base de données distincte pour chaque article afin de récupérer les données de l'auteur correspondant. Cela peut entraîner de nombreuses requêtes de base de données supplémentaires et ralentir considérablement les performances de votre application, surtout lorsque vous traitez de grandes quantités de données.

Le prefetching résout ce problème en utilisant la méthode prefetch_related() de l'ORM de Django. Cette méthode vous permet de spécifier les relations que vous souhaitez pré-charger lors du chargement des objets principaux. Django effectue alors une requête optimisée pour pré-charger les données liées en une seule requête, au lieu d'effectuer une requête distincte pour chaque objet.