KQL : Kibana Sample Data Ecommerce

ntroduction

KQL (Kibana Query Language) est un langage de requête simple et puissant pour interroger vos données Elasticsearch dans Kibana. Il permet de filtrer rapidement vos documents sans connaître la syntaxe complexe des requêtes Elasticsearch.

NOTES :

1. **Utilisez l'autocomplétion** : Kibana suggère les champs et valeurs disponibles
2. **Préférez `.keyword` pour les valeurs exactes** : Les champs texte sont analysés, utilisez `.keyword` pour les correspondances exactes
3. **Testez progressivement** : Construisez vos requêtes étape par étape
4. **Utilisez les parenthèses** : Pour clarifier les combinaisons logiques complexes
5. **Attention à la casse** : KQL est sensible à la casse pour les valeurs

Où tester le KQL :

- **Kibana – Discover :** Le plus simple.
- **Kibana – Visualize / Lens / Dashboard**

Exercice 1 : Filtres de Base

Exemple Introductif

Les filtres de base en KQL utilisent une syntaxe simple : `champ: valeur`

```
event.dataset: "sample_ecommerce"
```

Cette requête retourne tous les documents où le champ `event.dataset` contient exactement "sample_ecommerce".

Syntaxes importantes :

- **Égalité exacte** : `customer_gender: "MALE"`
- **Préfixe** : `email: eddie*` (trouve tous les emails commençant par "eddie")
- **Existence d'un champ** : `customer_phone: *` (documents ayant ce champ)
- **Absence d'un champ** : `not customer_phone: *`

Questions

Question 1.1 : Filtrez tous les documents concernant des clientes féminines.

Question 1.2 : Trouvez toutes les commandes passées un lundi (`day_of_week`).

Question 1.3 : Affichez uniquement les documents qui ont un numéro de téléphone renseigné.

Question 1.4 : Trouvez toutes les commandes de la région "Cairo Governorate" (`geoip.region_name`).

Aides

- Le champ `customer_gender` est de type `keyword` : les valeurs sont `MALE` ou `FEMALE`
- `day_of_week` contient des valeurs comme "Monday", "Sunday", etc.
- Pour vérifier l'existence d'un champ, utilisez `*` comme valeur
- Les champs imbriqués utilisent la notation pointée : `geoip.region_name`

Erreurs Fréquentes

- ❌ `customer_gender: FEMALE` → Fonctionne, mais mieux avec guillemets
- -> `customer_gender: "FEMALE"` → Recommandé pour éviter les ambiguïtés
- ❌ `day_of_week: monday` → Erreur de casse
- -> `day_of_week: "Monday"` → Correct
- ❌ `customer_phone: null` → N'existe pas en KQL
- -> `not customer_phone: *` → Correct pour vérifier l'absence

Exercice 2 : Combinaisons Logiques

Exemple Introductif

KQL permet de combiner plusieurs conditions avec les opérateurs logiques `and`, `or` et `not`.

```
customer_gender: "FEMALE" and day_of_week: "Monday"
```

Cette requête retourne les commandes de clientes passées un lundi.

Opérateurs logiques :

- **AND** : Les deux conditions doivent être vraies
- **OR** : Au moins une condition doit être vraie
- **NOT** : Exclut les documents correspondants
- **Parenthèses** : Pour grouper les conditions

Comparaisons numériques :

- `>` : supérieur
- `>=` : supérieur ou égal
- `<` : inférieur
- `<=` : inférieur ou égal

Questions

Question 2.1 : Trouvez toutes les commandes de la catégorie "Women's Shoes" passées un dimanche.

Question 2.2 : Affichez les commandes de la catégorie "Men's Clothing" OU "Women's Clothing".

Question 2.3 : Trouvez les commandes dont le montant total TTC (`taxful_total_price`) est supérieur à 100 EUR et en devise EUR (`currency`).

Question 2.4 : Affichez les commandes qui ne sont ni du lundi ni du dimanche, et dont le montant est inférieur à 50.

Aides

- `category` est un champ de type `text` avec un sous-champ `.keyword` pour les correspondances exactes
- Utilisez des guillemets pour les valeurs contenant des espaces
- Les comparaisons numériques ne nécessitent pas de guillemets
- Pour plusieurs jours, pensez à utiliser `not (day_of_week: "Monday" or day_of_week: "Sunday")`

Erreurs Fréquentes

- ❌ `category: Women's Shoes` → Erreur : l'apostrophe casse la syntaxe
- -> `category: "Women's Shoes"` → Correct
- ❌ `taxful_total_price > "100"` → Les guillemets sont inutiles pour les nombres
- -> `taxful_total_price > 100` → Correct
- ❌ `category: "Men's Clothing" or "Women's Clothing"` → Syntaxe incorrecte
- -> `category: "Men's Clothing" or category: "Women's Clothing"` → Correct
- Sans parenthèses, `not day_of_week: "Monday" and taxful_total_price < 50` peut être ambigu
- -> `not day_of_week: "Monday" and taxful_total_price < 50` → Utilisez des parenthèses si nécessaire

Exercice 3 : Texte, Exact vs Analysé

Exemple Introductif

Elasticsearch analyse les champs de type `text` en les découpant en tokens (mots). Pour rechercher une valeur exacte, utilisez le sous-champ `.keyword`.

Différence text vs keyword :

```
manufacturer: elitelligence
```

Trouve "Elitelligence" car le texte est analysé (insensible à la casse, tokenisé).

```
manufacturer.keyword: "Elitelligence"
```

Trouve uniquement la valeur exacte "Elitelligence" (sensible à la casse et aux espaces).

Wildcards :

- `*` : zéro ou plusieurs caractères
- `?` : exactement un caractère

```
email: *@butler-family.zzz
```

Trouve tous les emails du domaine butler-family.zzz.

Questions

Question 3.1 : Trouvez tous les produits dont le nom (`products.product_name`) contient le mot "jacket".

Question 3.2 : Trouvez tous les produits du fabricant "Low Tide Media" avec une correspondance exacte.

Question 3.3 : Affichez toutes les commandes des emails se terminant par "@baker-family.zzz".

Question 3.4 : Trouvez tous les produits dont le nom contient exactement la phrase "dark blue" (dans cet ordre).

Aides

- `products.product_name` est de type `text` avec un sous-champ `.keyword`
- Pour rechercher dans le texte analysé (insensible à la casse), utilisez `products.product_name`
- Pour une correspondance exacte, utilisez `products.product_name.keyword`
- Les wildcards fonctionnent avec les deux types, mais le comportement diffère
- Pour une phrase exacte dans du texte analysé, placez-la entre guillemets

Erreurs Fréquentes

- ❌ `manufacturer.keyword: elitelligence` → Casse incorrecte
- -> `manufacturer.keyword: "Elitelligence"` → Correct
- ❌ `email: *butler-family.zzz` → Manque le @
- -> `email: *@butler-family.zzz` → Correct
- `products.product_name: "dark blue"` → Trouve les documents contenant "dark" ET "blue" mais pas nécessairement cette phrase exacte
- -> `products.product_name: "dark blue"` → En KQL, les guillemets indiquent une recherche de phrase
- ❌ `manufacturer.keyword: "Low Tide*"` → Le wildcard ne fonctionne pas bien avec .keyword dans certains contextes
- -> `manufacturer: "Low Tide*"` → Mieux pour les wildcards

Exercice 4 : Champs Imbriqués et Agrégations Conceptuelles

Exemple Introductif

Les documents peuvent contenir des objets imbriqués. Le champ `products` est un tableau d'objets avec plusieurs propriétés.

Accès aux champs imbriqués :

```
products.category: "Women's Shoes"
```

Cette requête trouve les commandes contenant au moins un produit de la catégorie "Women's Shoes".

Requêtes sur plusieurs niveaux :

```
geoip.continent_name: "Europe" and products.price > 50
```

Trouve les commandes européennes avec au moins un produit coûtant plus de 50 EUR.

Note importante : KQL recherche dans tous les éléments du tableau `products`. Si une commande contient plusieurs produits, la condition s'applique à l'ensemble.

Questions

Question 4.1 : Trouvez les commandes contenant des produits avec une réduction (`products.discount_percentage` supérieur à 0).

Question 4.2 : Affichez les commandes d'Afrique (`geoip.continent_name`) contenant des chaussures (catégorie contient "Shoes").

Question 4.3 : Trouvez les commandes où la quantité totale de produits (`total_quantity`) est supérieure ou égale à 4.

Question 4.4 : Affichez les commandes d'Asie ou d'Europe contenant des produits "Champion Arts" ou "Elitelligence".

Aides

- Les champs imbriqués utilisent la notation pointée : `products.discount_percentage`
- Pour les objets géographiques : `geoip.continent_name` , `geoip.city_name`
- Les wildcards fonctionnent aussi sur les champs imbriqués : `products.category: *Shoes*`
- N'oubliez pas les parenthèses pour grouper les conditions OR dans un contexte AND

- `total_quantity` est un champ racine (non imbriqué)

Erreurs Fréquentes

- ❌ `products.discount > 0` → Le champ s'appelle `discount_percentage`
- -> `products.discount_percentage > 0` → Correct
- ❌ `geoip.continent: "Africa"` → Le champ s'appelle `continent_name`
- -> `geoip.continent_name: "Africa"` → Correct
- ❌ `products.category: Shoes` → Trouve seulement "Shoes" exact
- -> `products.category: *Shoes*` → Trouve "Women's Shoes", "Men's Shoes", etc.
- ❌ `geoip.continent_name: "Asia" or "Europe"` → Syntaxe incorrecte
- -> `geoip.continent_name: "Asia" or geoip.continent_name: "Europe"` → Correct

Exercice 5 : Requêtes Complexes et Cas Réels

Exemple Introductif

Les requêtes réelles combinent souvent plusieurs concepts : filtres multiples, champs imbriqués, exclusions et plages de valeurs.

Exemple de requête complexe :

```
geoip.continent_name: "Europe" and
customer_gender: "FEMALE" and
taxful_total_price >= 50 and taxful_total_price <= 150 and
not day_of_week: "Saturday" and
products.category: (*Clothing* or *Shoes*)
```

Cette requête trouve les commandes européennes de clientes féminines, avec un montant entre 50 et 150 EUR, passées hors samedi, contenant des vêtements ou des chaussures.

Techniques avancées :

- **Plages de valeurs** : Combinez `>=` et `<=` pour définir un intervalle
- **Parenthèses** : Essentielles pour les combinaisons OR dans un contexte AND
- **Wildcards stratégiques** : `*Clothing*` trouve "Men's Clothing", "Women's Clothing", etc.
- **Négations multiples** : `not (A or B or C)` exclut plusieurs valeurs

Questions

Question 5.1 : Trouvez les commandes de clients masculins d'Amérique du Nord, avec un montant entre 100 et 200 EUR, contenant des produits "Oceanavigations" ou "Pyramidustries".

Question 5.2 : Affichez les commandes du weekend (samedi ou dimanche) hors Europe, avec au moins 2 produits uniques (`total_unique_products >= 2`).

Question 5.3 : Trouvez les commandes de la ville "Dubai" ou "Cairo", passées en semaine (lundi à vendredi), avec des produits dont le prix unitaire de base (`products.base_unit_price`) dépasse 50 EUR.

Question 5.4 : Affichez les commandes contenant des bottes (produit contenant "boot") ou des vestes (contenant "jacket"), passées par des femmes, avec un montant supérieur à 50 EUR.

Question 5.5 : Créez une requête pour identifier les commandes suspectes : montant total supérieur à 150 EUR et email du domaine contenant "zzz".

Récapitulatif des Bonnes Pratiques

À Retenir

1. **Filtres de base**
   - Utilisez des guillemets pour les valeurs avec espaces ou caractères spéciaux
   - `*` pour vérifier l'existence d'un champ
   - Attention à la casse des valeurs
2. **Combinaisons logiques**
   - `and` a priorité sur `or` , utilisez des parenthèses pour clarifier
   - Regroupez les OR entre parenthèses dans un contexte AND
   - Les comparaisons numériques : `>` , `>=` , `<` , `<=`
3. **Texte vs Keyword**
   - Champ text : analysé, insensible à la casse, tokenisé
   - Champ .keyword : exact, sensible à la casse, non tokenisé
   - Wildcards : `*` (plusieurs caractères), `?` (un caractère)
   - Guillemets pour rechercher des phrases
4. **Champs imbriqués**
   - Notation pointée : `geoip.city_name` , `products.price`
   - Les conditions sur tableaux s'appliquent à n'importe quel élément
5. **Requêtes complexes**
   - Construisez progressivement

- Testez chaque partie séparément
- Utilisez l'autocomplétion de Kibana
- Documentez vos requêtes complexes

Astuces de Performance

- Privilégiez les champs keyword pour les filtres exacts
- Évitez les wildcards en début de chaîne (`*something`) quand possible
- Combinez les filtres les plus sélectifs en premier
- Utilisez les suggestions de Kibana pour vérifier les types de champs

Pièges à Éviter

- Ne confondez pas `field: value1 or value2` (incorrect) avec `field: value1 or field: value2`
- N'utilisez pas `==` , `!=` : utilisez `:` et `not`
- Les dates doivent être au format ISO : `"2025-10-20"`
- Les wildcards sur .keyword peuvent avoir un comportement inattendu