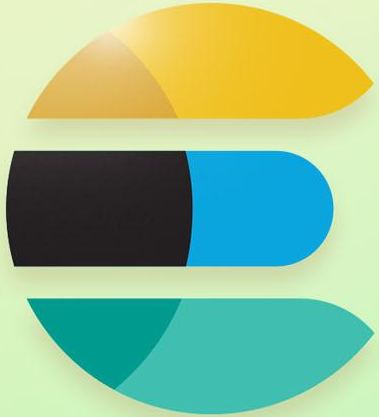
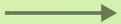
Prefixes, wildcards & regular expressions



Introduction

- Term level queries are used for exact matching
  - Query non-analyzed values with queries that are not analyzed
- There are a few exceptions; we'll cover three of them now
  - Querying by prefix, wildcards, and regular expressions
  - Remember to still query `keyword` fields

```
GET /products/_search
{
  "query": {
    "prefix": {
      "name.keyword": {
        "value": "Past"
      }
    }
  }
}
```



Match?	Indexed term
✓	"Pasta - Linguini Dry"
✓	"Paste - Black Olive"
✓	"Pastry - Baked Scones - Mini"
✗	"Linguini Pasta"

CodingExplained
CODE. SLEEP. REPEAT.

Wildcard examples

Pattern	Terms
`Past?`	✅ `"Pasta"` ✅ `"Paste"`
`Bee?`	❌ `"Bee"` ✅ `"Beer"` ❌ `"Beets"` ✅ `"Beef"`
`Bee*`	✅ `"Bee"` ✅ `"Beer"` ✅ `"Beets"` ✅ `"Beef"` ❌ `"Beverage"`
⚠️ `*Beer`	✅ `"Beer"` ✅ `"Root Beer"`

⚠️ Avoid placing wildcards at the beginning of a pattern!

Regular expressions

- The `regexp` query matches terms that match a regular expression
- Regular expressions are patterns used for matching strings
- Allows more complex queries than the `wildcard` query

Regular expressions examples

Pattern	Terms
`Bee(f|r)+`	✓ "Beef" ✓ "Beer" ✗ "Beers" ✗ "Beet" ✗ "Beets"
`Bee[a-zA-Z]+`	✓ "Beef" ✓ "Beer" ✓ "Beers" ✓ "Beet" ✓ "Beets"

Regular expressions examples

Pattern	Terms
`Bee(r|t){1}`	✅ "Beet" ❌ "Beetroot"
`Bee[a-zA-Z]+`	✅ "Beet" ✅ "Beetroot"
`Beer`	❌ "Heineken (Beer)" ❌ "Beer - Heineken"
`Beer.*`	❌ "Heineken - Beer" ✅ "Beer - Heineken"
⚠️ `.*Beer`	✅ "Heineken - Beer" ❌ "Beer - Heineken"
⚠️ `.*Beer.*`	✅ "Heineken - Beer" ✅ "Beer - Heineken"

⚠️ Avoid placing wildcards at the beginning of a pattern!

CodingExplained
CODE. SLEEP. REPEAT.

Engine comparison

Other engines	Apache Lucene
`^Beer`	`Beer.*`
`Beer$`	`.*Beer`
`Bee[a-zA-Z]+$`	`Bee[a-zA-Z]+.*`

Case insensitivity

Prefix

```
GET /products/_search
{
  "query": {
    "prefix": {
      "name.keyword": {
        "value": "Past",
        "case_insensitive": true
      }
    }
  }
}
```

Regexp

```
GET /products/_search
{
  "query": {
    "regexp": {
      "tags.keyword": {
        "value": "Bee(f|r)+",
        "case_insensitive": true
      }
    }
  }
}
```

Wildcard

```
GET /products/_search
{
  "query": {
    "wildcard": {
      "tags.keyword": {
        "value": "Past?",
        "case_insensitive": true
      }
    }
  }
}
```

Lecture summary

- The `prefix` query matches terms that *begin* with a prefix
- The `wildcard` query enables us to use wildcards
  - `?` to match any single character
  - `*` to match any number of characters (`0-N`)
- The `regexp` query matches terms based on regular expressions
  - More flexible than the `wildcard` query
  - The whole term must be matched
  - Uses Apache Lucene regex engine (`^` and `$` anchors not supported)
- Avoid placing wildcards at the beginning of patterns if at all possible
- Use the `case_insensitive` parameter to ignore letter casing