Conditions, Subqueries, and Geo-mapping

Conditional Operators

- Write a query that scans through Labs/Apache/Access logs
    - For the last 60 minutes
    - The number of successful HTTP requests returned (where status_code=200), versus the number of error HTTP requests returned (where status_code=404).

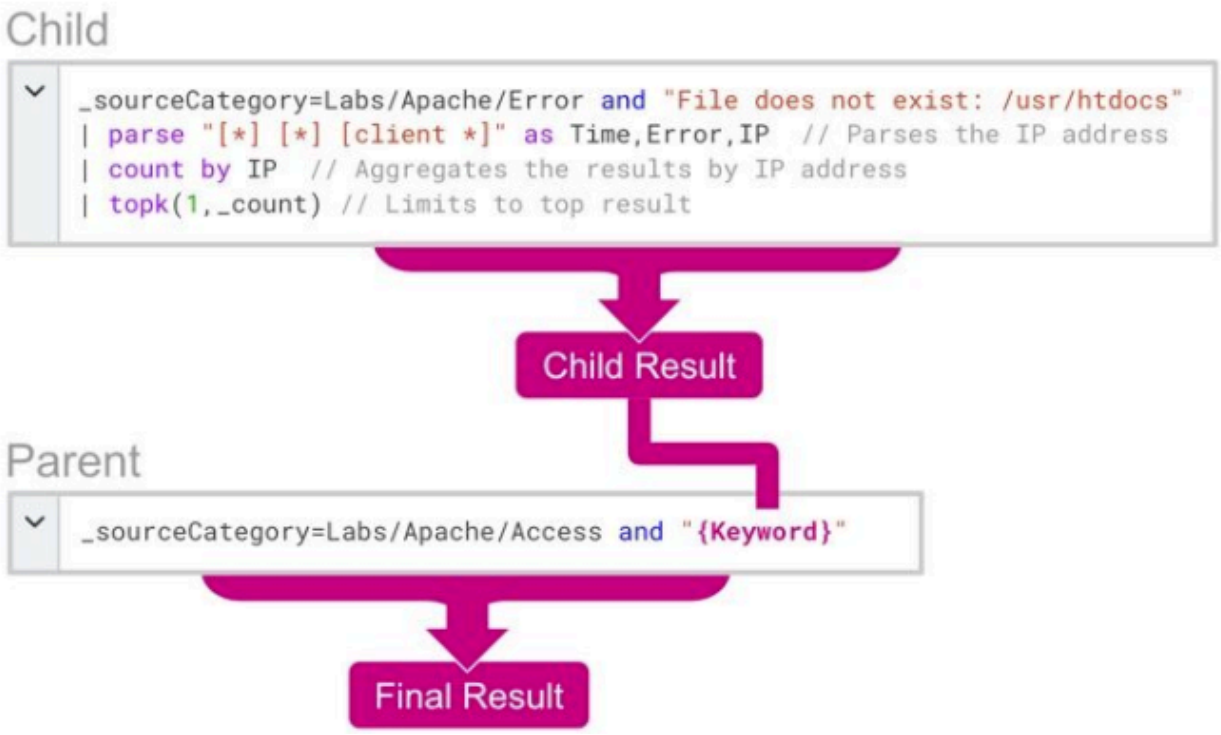
```
_sourceCategory=Labs/Apache/Access
| parse "HTTP/1.1\" * " as status_code
| if(status_code=200, 1, 0) as successes
| if(status_code=404, 1, 0) as client_errors
| sum(successes) as success_cnt, sum(client_errors) as client_errors_cnt
```

- Query to scan through Labs/Apache/Access and return the number of redirects

```
_sourceCategory=Labs/Apache/Access
| parse "HTTP/1.1\" * " as status_code
```

```
| if(status_code matches "4*", 1, 0) as client_err
| if(status_code matches "5*", 1, 0) as server_err
| if(status_code matches "3*", 1, 0) as redirects
| sum(server_err) as server_errors_cnt, sum(client_err) as client_errors_cnt, su
m(redirects) as redirects_cnt
```

Filtering using Subqueries



Subqueries

- **Filter and evaluate conditions** for a query when you may not be sure of the exact filter or condition criteria but you can write a short query to set them
- Use one query to pass results back to another query to narrow down or evaluate the set of messages that are searched in that query
- NOT available on live dashboards, in real-time alerts, or inside FERs and Scheduled Views.

Child Query

- Handles the **filtering**
- Runs first and provides intermediate input for the parent query
- Can specify a different time range than the parent query.
    - You have to factor in time for all of these queries to complete.

Parent Query

- Depends on the input from a child query or queries to **finish** its execution

```
_sourceCategory=Labs/Apache/Access
[subquery:_sourceCategory=labs/apache/error and "File does not exist: /usr/h
tdocs"
| parse "[*] [*] [client *] File does not exist: /usr/htdocs" as Time,Error,IP
| count by IP
| topk(1,_count) | compose IP keywords
]
```

Geo Lookup

- Matches a parsed IPv4 or IPv6 address to its geographical location on a map.
- To create the map the **lookup** operator matches parsed IP addresses to their physical location based on the latitude and longitude of where the addresses originated
- Need parsed ip address. Need Count to have the map to show the correct info
- Any IP address without a location (ex: internal addresses) will return null
- Query example:

```
_sourceCategory = "Labs/Apache/Access"
| parse regex "^(?<src_ip>\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3})" nodrop
| parse regex "(?<method>[A-Z]+)\s(?<url>\S+)\sHTTP/[\d\.]+\"\s(?<status_code>\d+)\s(?<size>[\d-]+)" nodrop
| parse regex "(?<method>[A-Z]+)\s(?<url>\S+)\sHTTP/[\d\.]+\"\s(?<status_code>\d+)\s(?<size>[\d-]+)\s\"(?
<referrer>.*?)\"\s\"(?<user_agent>.+?)\".*" nodrop
| lookup latitude, longitude, country_code, country_name, region, city, postal_code, area_code, metro_code from
geo://default on ip = src_ip
| where !(area_code = "0")
| where country_name = "United States"
| count by latitude, longitude, country_code, country_name, region, city, postal_code, area_code, metro_code
```