Adevinta

A Federated Information Infrastructure that Works

Xavier Gumara Rigol
@xgumara

October 3rd, 2019

Barcelona

multitenancy *(noun)* mode of operation of software where multiple independent instances operate in a shared environment.

What are the challenges of building a multi tenant information architecture for business insights?

How we solved them at Adevinta?

About me

Xavier Gumara Rigol

@xgumara

Data Engineering Manager at Adevinta (former Schibsted) since 2016.

Consultant Professor at the Open University of Catalonia (UOC) since 2016.

Between 2013 and 2016 I worked as a **Business Intelligence Engineer** at Schibsted, and previously as a **Business Intelligence Consultant** at Stratebi for almost 3 years.

About Adevinta

Adevinta is a **marketplaces specialist**. We are an international family of local digital brands.

Our marketplaces **create perfect matches on the world's most trusted marketplaces**.

Thanks to our second hand effect our users potentially save every year:

1.1 million
tonnes of plastic

20.5 million tons of
greenhouse gases

About Adevinta

More than 30 brands in 16 countries in Europe, Latin America and North Africa:

+ a global services organization located between Barcelona and Paris.

Framing the problem

Problems we are trying to solve

- Easy access to key facts about our marketplaces (tenants)
- Eliminate data-quality discussions, establish trust in the facts
- Reduce impact on manual data requests to each tenant
- Minimize regional effort needed for global data collection
- Provide a framework and infrastructure that can be extended locally

The lowest common denominator for successful information architecture initiatives

- Executive support
- Provide results sooner than later and iterate
- It is not a project but an initiative
- Fix data quality at the source
- Invest in solving technical debt

The challenges of a multi tenant information architecture

1. Finding the right level of authority
2. Governance of the data sets
3. Building common infrastructure as a platform

01 Finding the right level of authority

Finding the right level of authority

Decentralization
Authority delegated

Centralization
Authority not delegated

Silos of unreachable data

Pros: speed of execution (locally) and market customization

Cons: difficult to have a global view, duplication of efforts

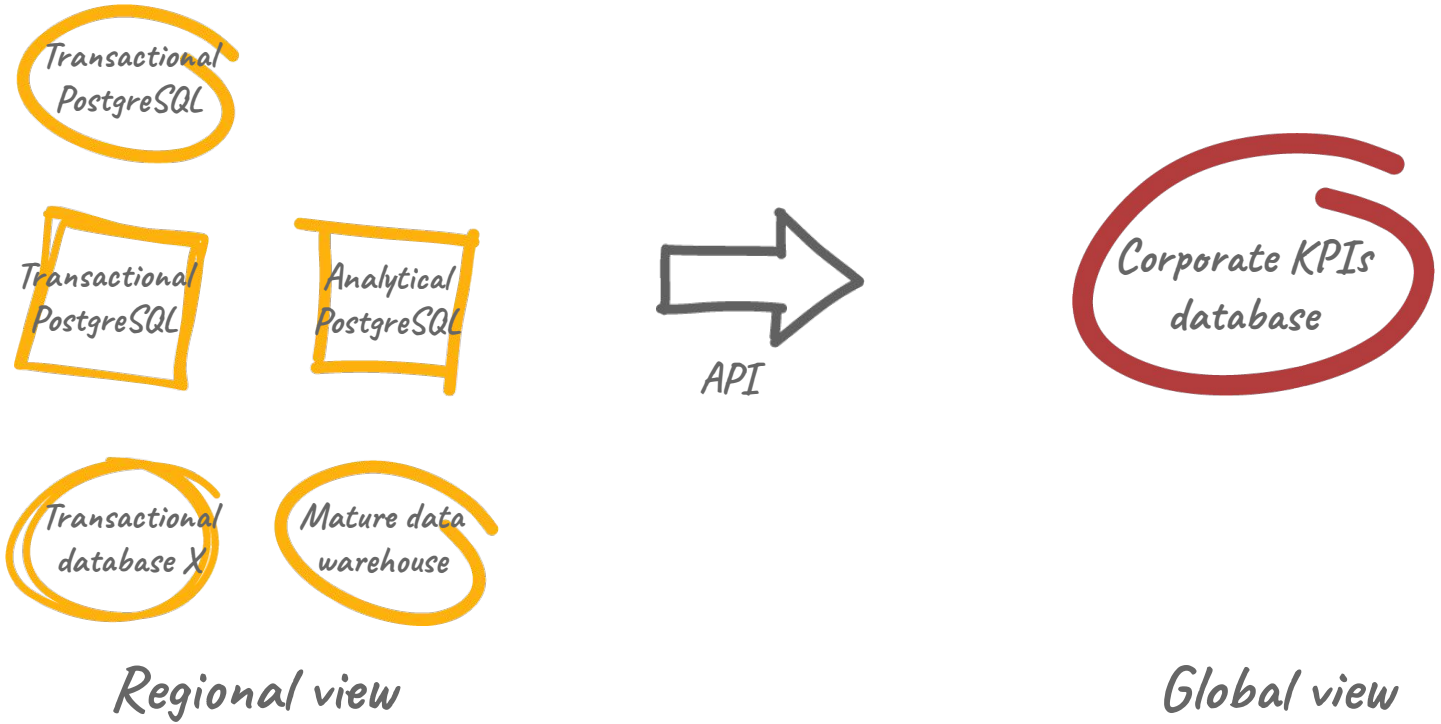
Monolithic data platform bottleneck

Pros: can work at small scale

Cons: long response times, difficult to harmonise

Finding the right level of authority

Decentralization

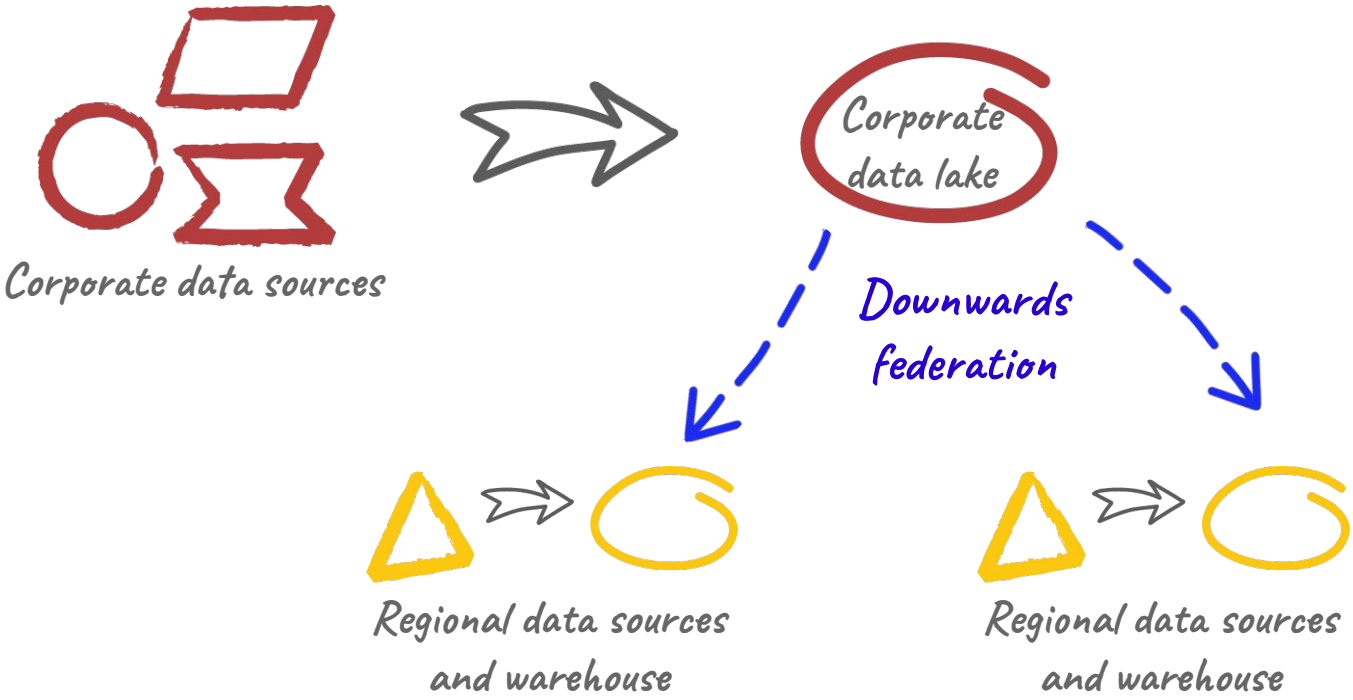


Finding the right level of authority

Centralization

Finding the right level of authority

Current solution: federation



Finding the right level of authority

Current solution: federation

- Each regional data warehouse is a different **Redshift** instance
- Physical storage is **S3** and can be accessed:
  - Via **Athena** for global/central analysts
  - Via **Redshift Spectrum** for global teams (downwards federation)

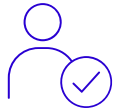
02 Governance of data sets

Governance of data sets

Embrace the concept of “data set as a product” that defines the basic qualities of a data set as:

Discoverable

Addressable



Trustworthy

Self-describing

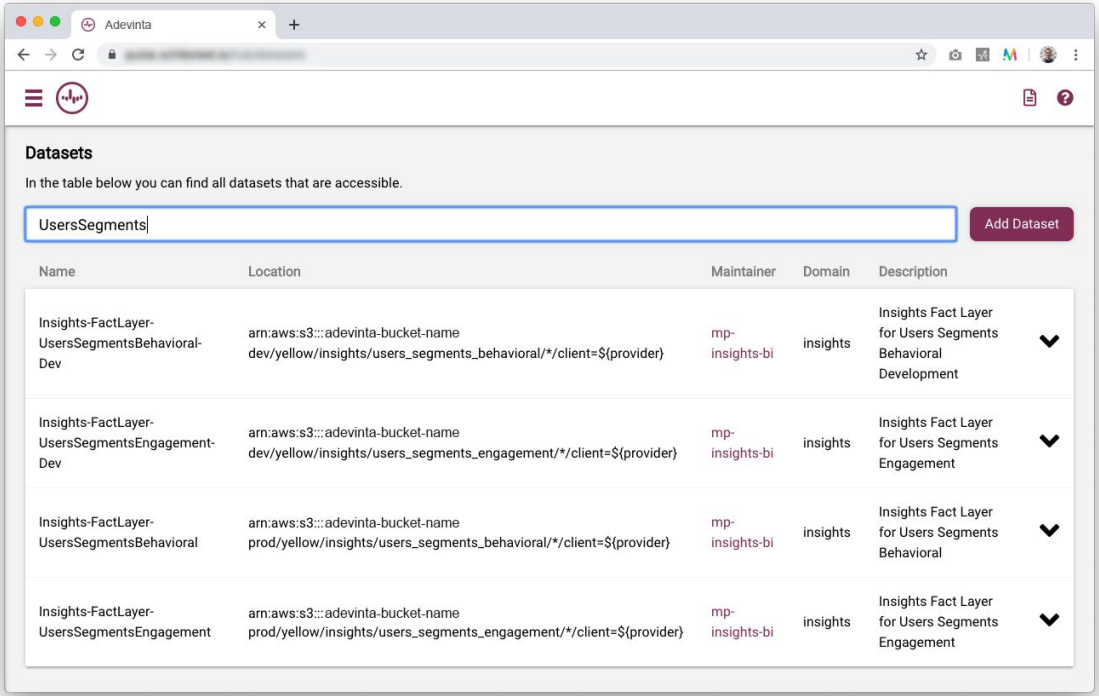
Inter operable

Secure

Source: “How to Move Beyond a Monolithic Data Lake to a Distributed Data Mesh” by Zhamak Dehghani
https://martinfowler.com/articles/data-monolith-to-mesh.html

Governance of data sets

“Data set as a product”: Discoverable



Governance of data sets

“Data set as a product”: Addressable

Data Location

The master location of events is S3, accessed through Athena or Jupyter. See the data model of the entity in the section below: Events Behavioral schema.

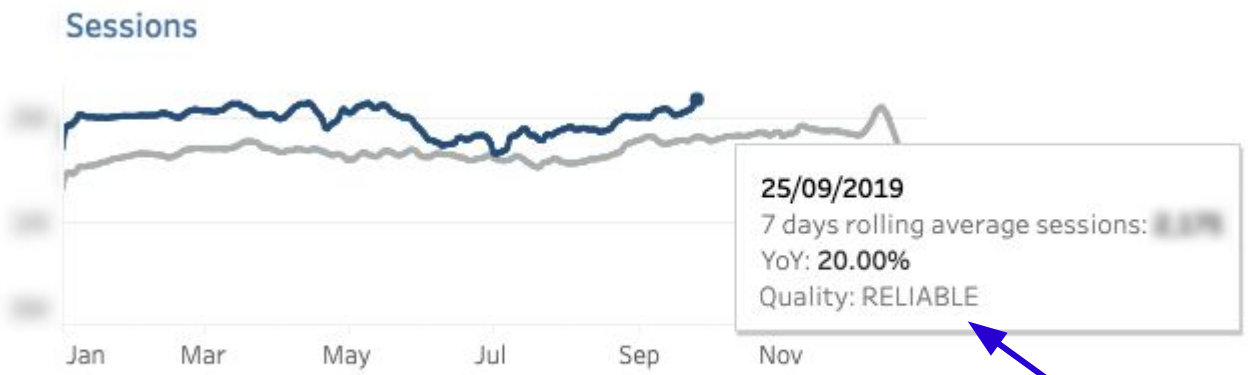
1.- **Dataset** name to request access: Insights-FactLayer-EventsBehavioral

2.- **Athena** (SQLaaS): `XXXXXX_databox.insights_events_behavioral_fact_layer_365d` where XXXXXX is your client_id.

3.- **S3** path `s3://adevinta-bucket-name`
`prod/yellow/insights/events/source=pulse/version=4/year=$year/month=$month/day=$day/`
`gen=0/client=$client/`

Governance of data sets

“Data set as a product”: Trustworthy



Governance of data sets

“Data set as a product”: Self-describing

All data set documentation includes:

- Data location
- Data provenance and data mapping
- Example data
- Execution time and freshness
- Input preconditions
- Example Jupyter notebook using the data set

Governance of data sets

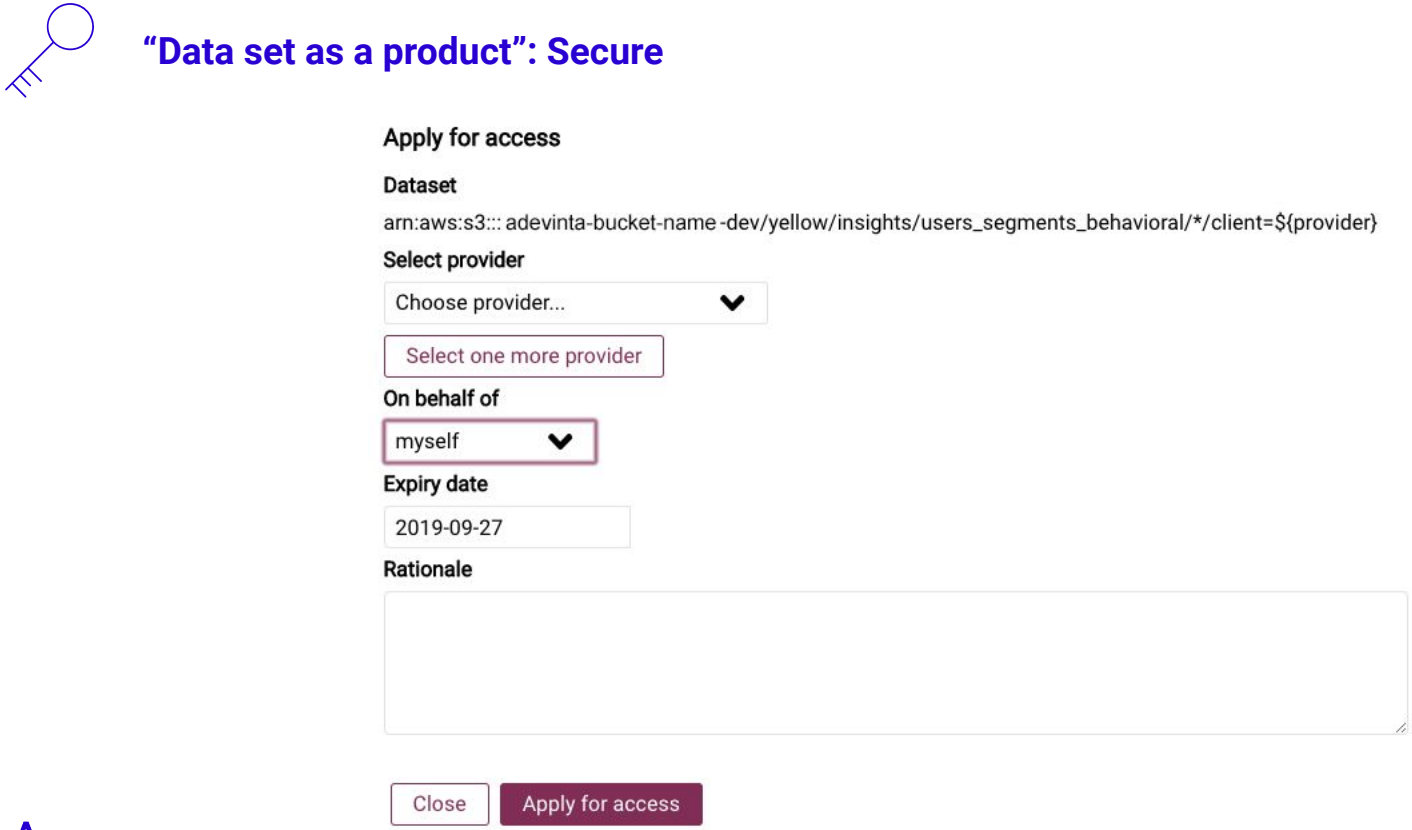
“Data set as a product”: Inter operable

- Defining a common nomenclature is a must in all layers of the platform
- Usage of schema.org to identify the same object across different domains

```
"adType": {
      "description": "Type of the ad",
        "enum": [
            "buy",
            "sell",
            "rent",
             "let",
             "lease",
             "swap",
             "give",
             "jobOffer"
    ]
}
```

Governance of data sets

“Data set as a product”: Secure



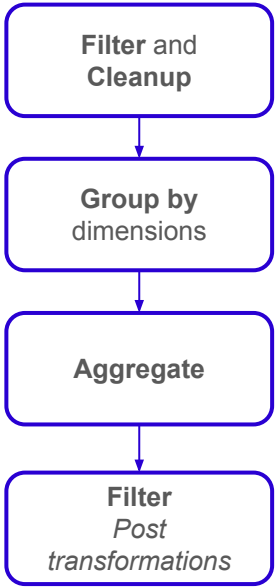
03 Building common infrastructure as a platform

Building common infrastructure as a platform

Use-case: metrics calculation

Patterns for business metrics calculation:

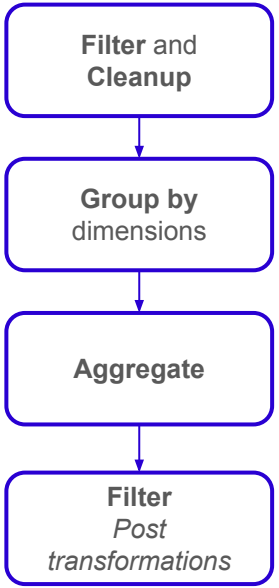
- Metrics need to use specific events (filter)
- Some transformations applied before aggregating
- Group by several dimensions
- Aggregation function (count, count distinct, sum,...)
- Some transformations applied after aggregating
- Different periods of calculation day, week, month, 7d, 28d



Building common infrastructure as a platform

Use-case: metrics calculation

```
val simpleMetric: Metric = withSimpleMetric(
   metricId = AdsWithLeads,
   cleanupTransformations = Seq(
      filterEventTypes(List(isLeadEvent(EventType, ObjectType)))
   ),
   dimensions = Seq(DeviceType, ProductType, TrackerType),
   aggregate = countDistinct(AdId),
   postTransformations = Seq(
      withConstantColumn(Period, period)(_),
      withConstantColumn(ClientId, client)(_))
)
```

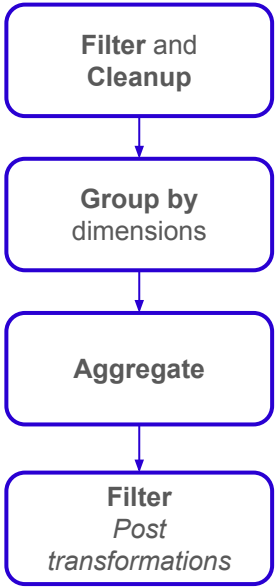


Building common infrastructure as a platform

Use-case: metrics calculation

This configuration is then passed to the `cube()` function in Spark. The `cube()` function “calculates subtotals and a grand total for every permutation of the columns specified”.

```
val simpleMetricWithSubtotals: Metric =
   simpleMetric.withSubtotals(Seq(DeviceType, ProductType, TrackerType)
)
```



Building common infrastructure as a platform

Use-case: metrics calculation

```
private val metricDefinitions: Seq[MetricDefinition] = List(
  MetricDefinition(
    metricIdentifiers(Sessions),
    countDistinct(SessionId)
  ),
  MetricDefinition(
    metricIdentifiers(LoggedInSessions),
    countDistinct(SessionId),
    filterEventTypes(List(col(EventIsLogged) === 1)) _
  ),
  MetricDefinition(
    metricIdentifiers(AdsWithLeads),
    countDistinct(AdId),
    filterEventTypes(List(isLeadEvent(EventType, ObjectType))) _
  )
)
```

Building common infrastructure as a platform

Use-case: Recency-Frequency-Monetization (RFM) user segmentation

```
val df = spark.read.parquet(path)
              .groupBy("user_id")
              .agg(
                  count(col("event_id")).as("total_events"),
                  countDistinct()(col("session_id")).as("total_sessions")
          )


val dfWithSegments = df.transform(withSegment("segment_chain",
  Seq(
    SegmentDimension(col("total_events"), "events_percentile", 0.5, 0.8),
    SegmentDimension(col("total_sessions"), "sessions_percentile", 0.5, 0.8)
  )
))
```

The **withSegment** method requires a name to store the output of the segmentation and a list of all dimensions that will be used. You can tune the thresholds for each segment dimension.

Building common infrastructure as a platform

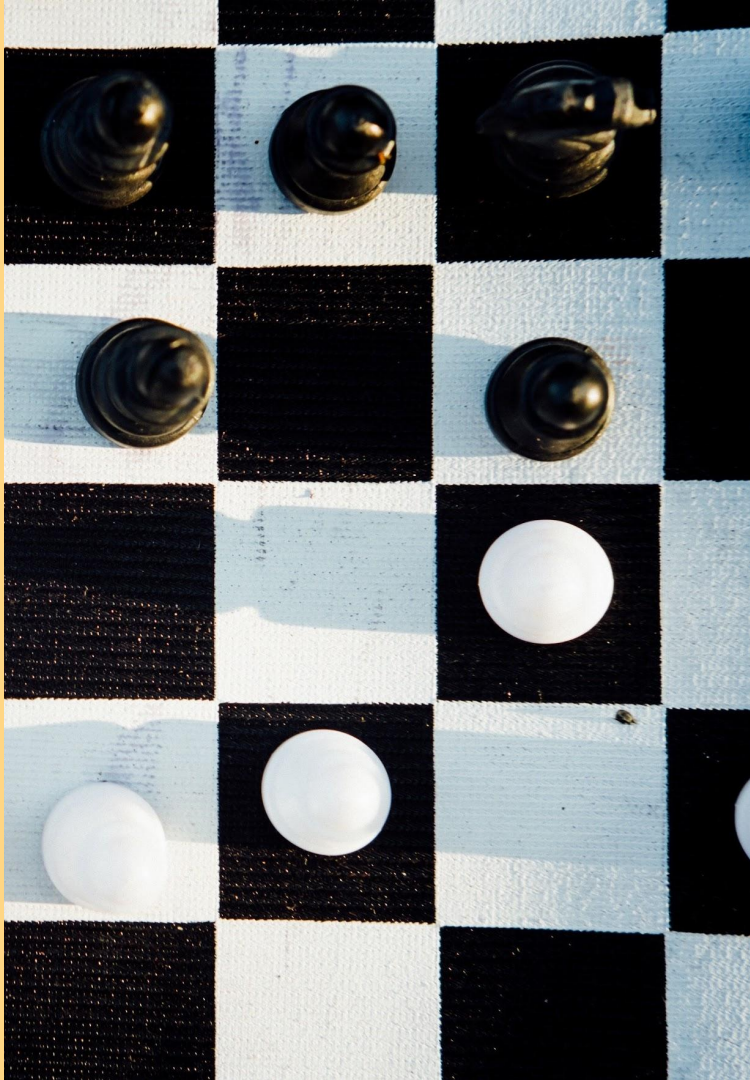
Use-case: Recency-Frequency-Monetization (RFM) user segmentation

```
val myMap = Map[String, String](
    "LL" -> "That's a low active user" ,
    "LM" -> "Users that do few events in different sessions" ,
    "LH" -> "Users that do almost nothing but somehow generate many sessions"  ,
    "ML" -> "Meh... in little sessions" ,
    "MM" -> "Meh... in medium sessions" ,
    "MH" -> "Meh... in multiple sessions" ,
    "HL" -> "Users that do a lot of things in a row" ,
    "HM" -> "Users that do a lot of things along the day" ,
    "HH" -> "Da best users"
)

dfWithSegments.transform(withSegmentMapping( "segment_name", col("segment_chain"), myMap))
```

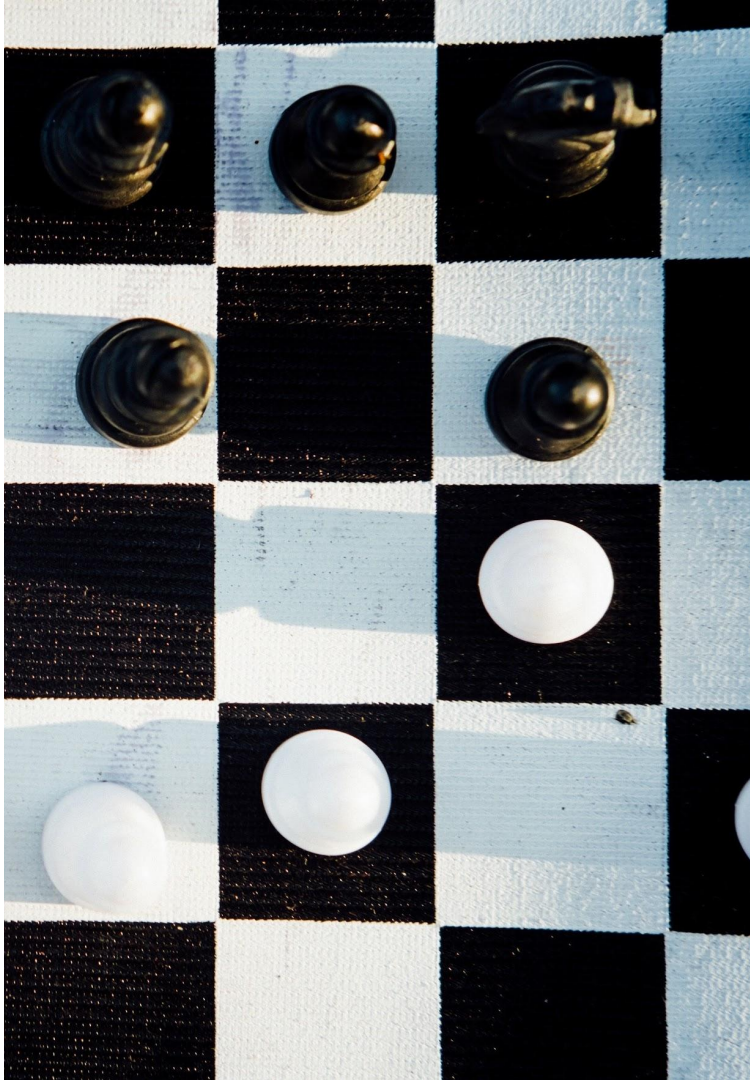
The **withSegmentMapping** method applies a map to the result of the segmentation to add meaningful names to the user segments.

What have we learned?



What have we learned?

- Federation gives autonomy
- Non-invasive governance is key
- Balance the delivery of business value vs tooling



Adevinta

Thank you!

Xavier Gumara Rigol
@xgumara