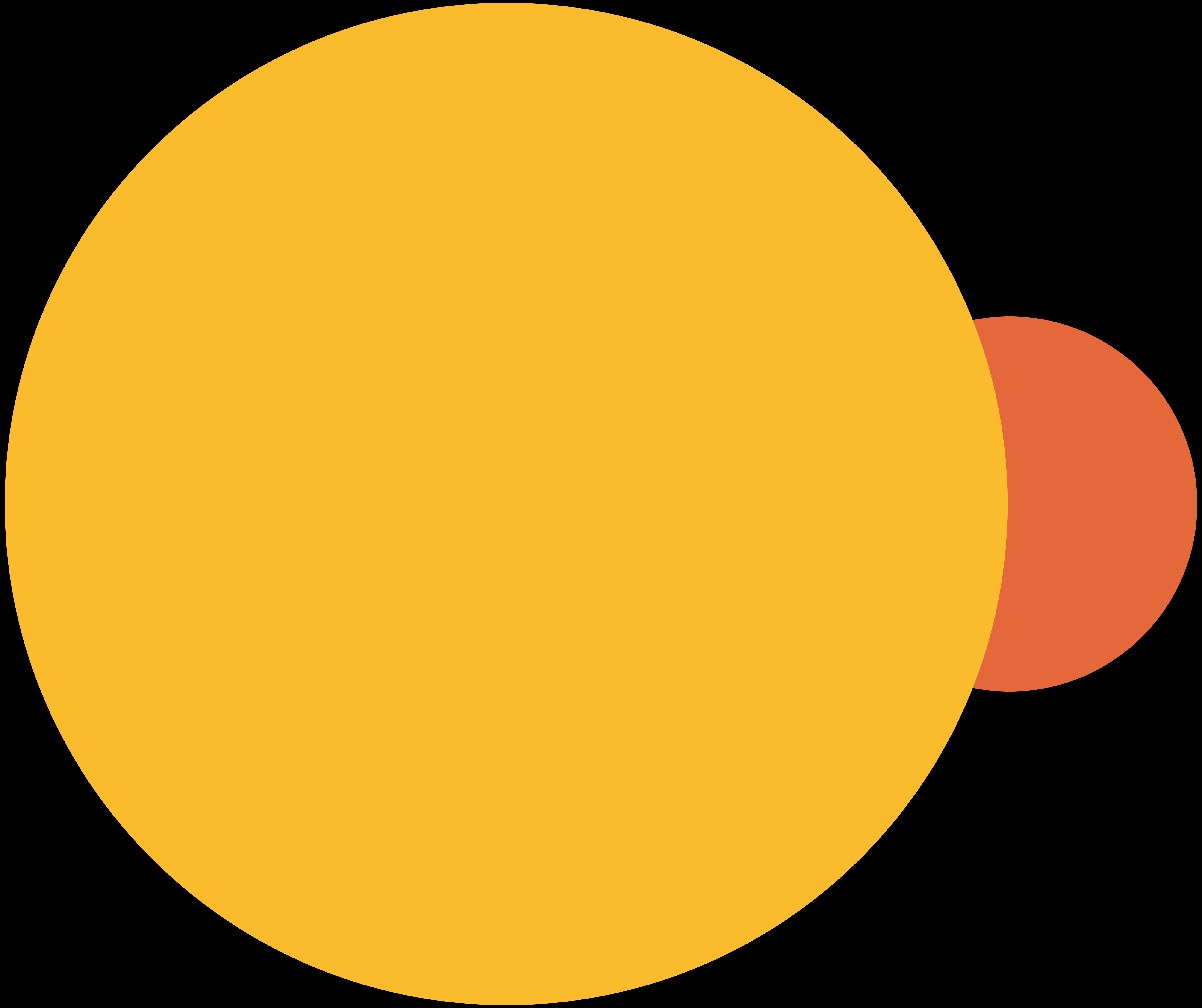
ZOOMDATA®

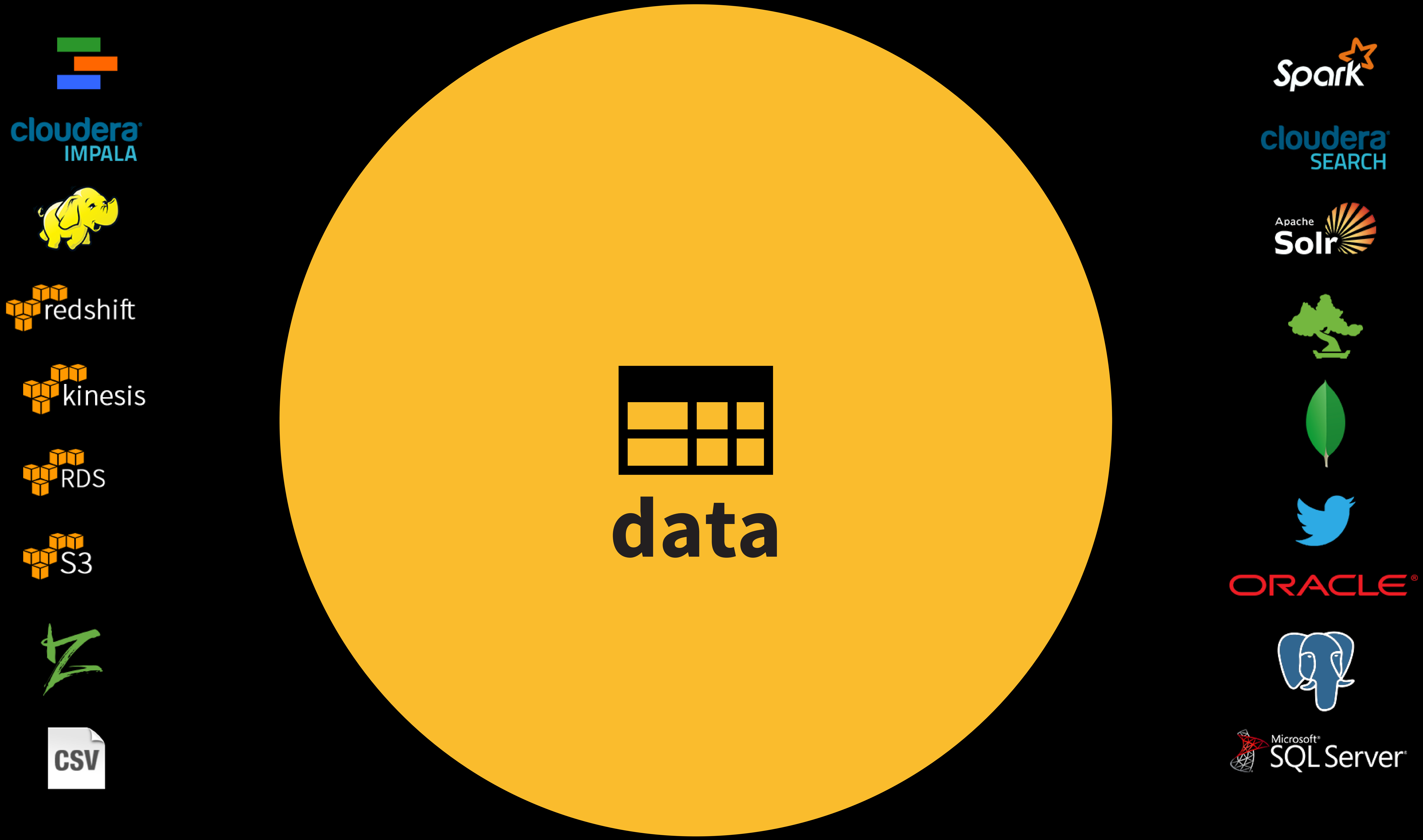
BI Style Analytics with Spark

Justin Langseth / Farzad Aref

Spark Summit 2014, San Francisco, CA

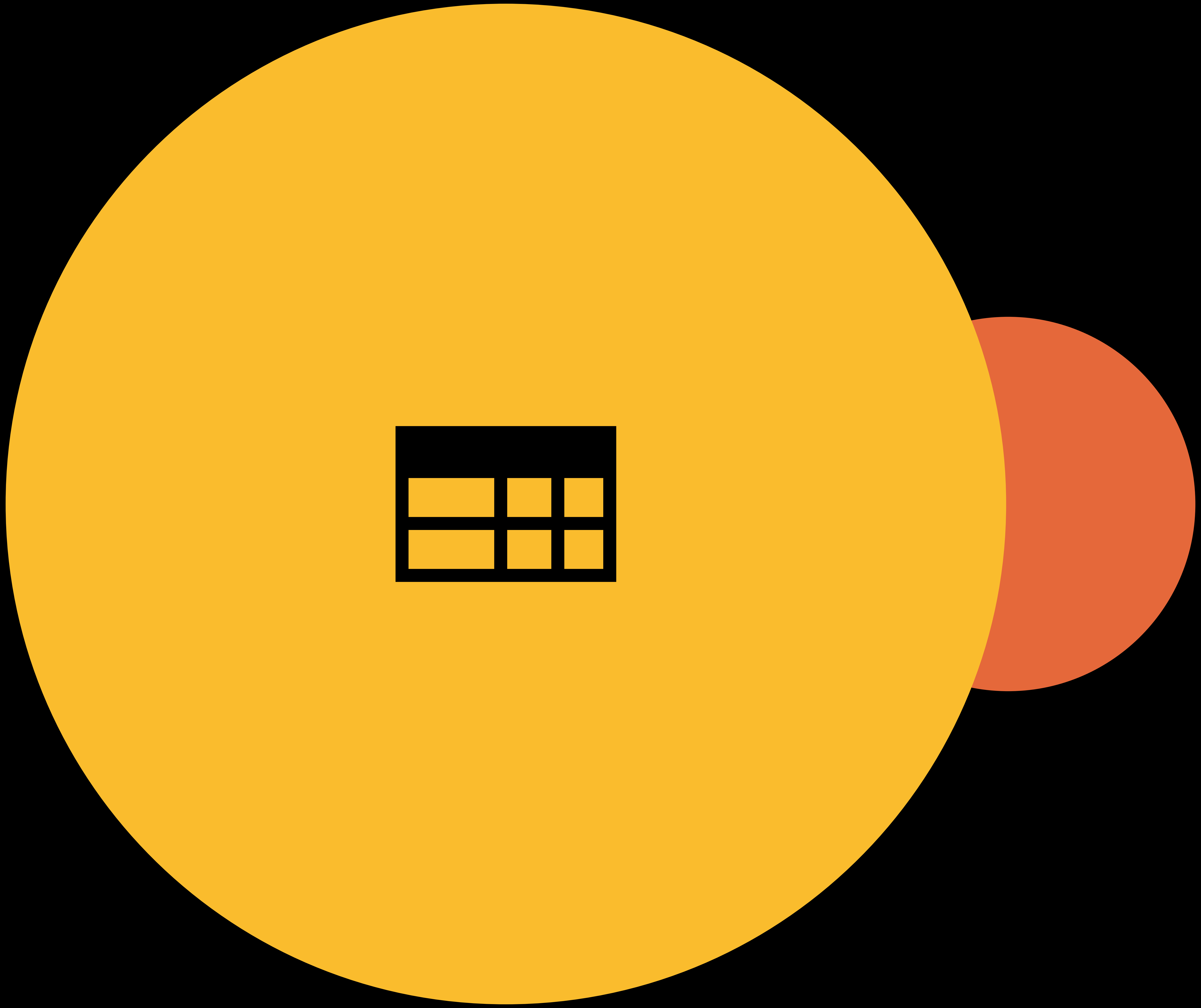


data

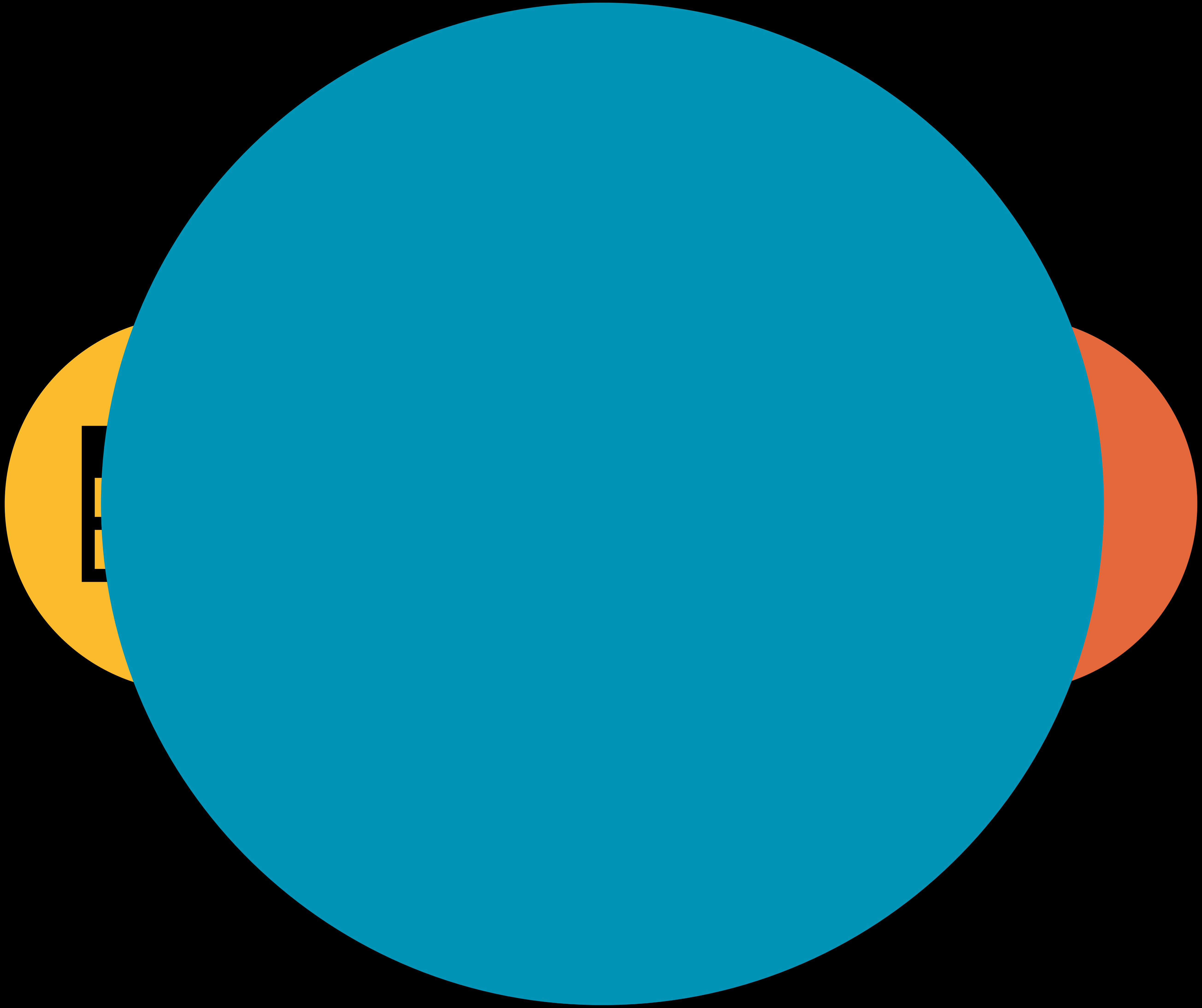


data

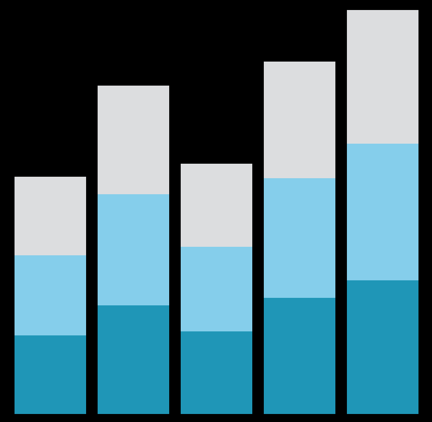
cloudera
IMPALA
redshift
kinesis
RDS
S3
CSV
Spark
cloudera
SEARCH
Apache
Solr
ORACLE
Microsoft
SQL Server



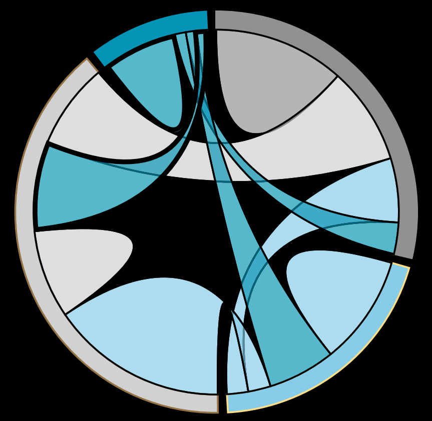
data

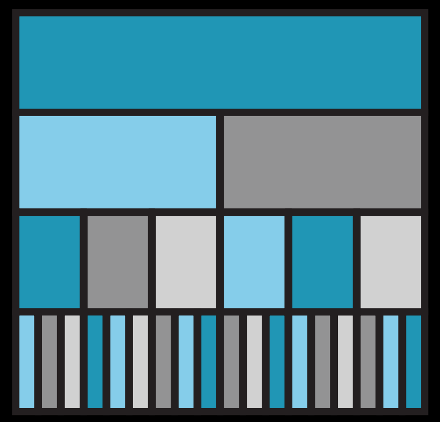


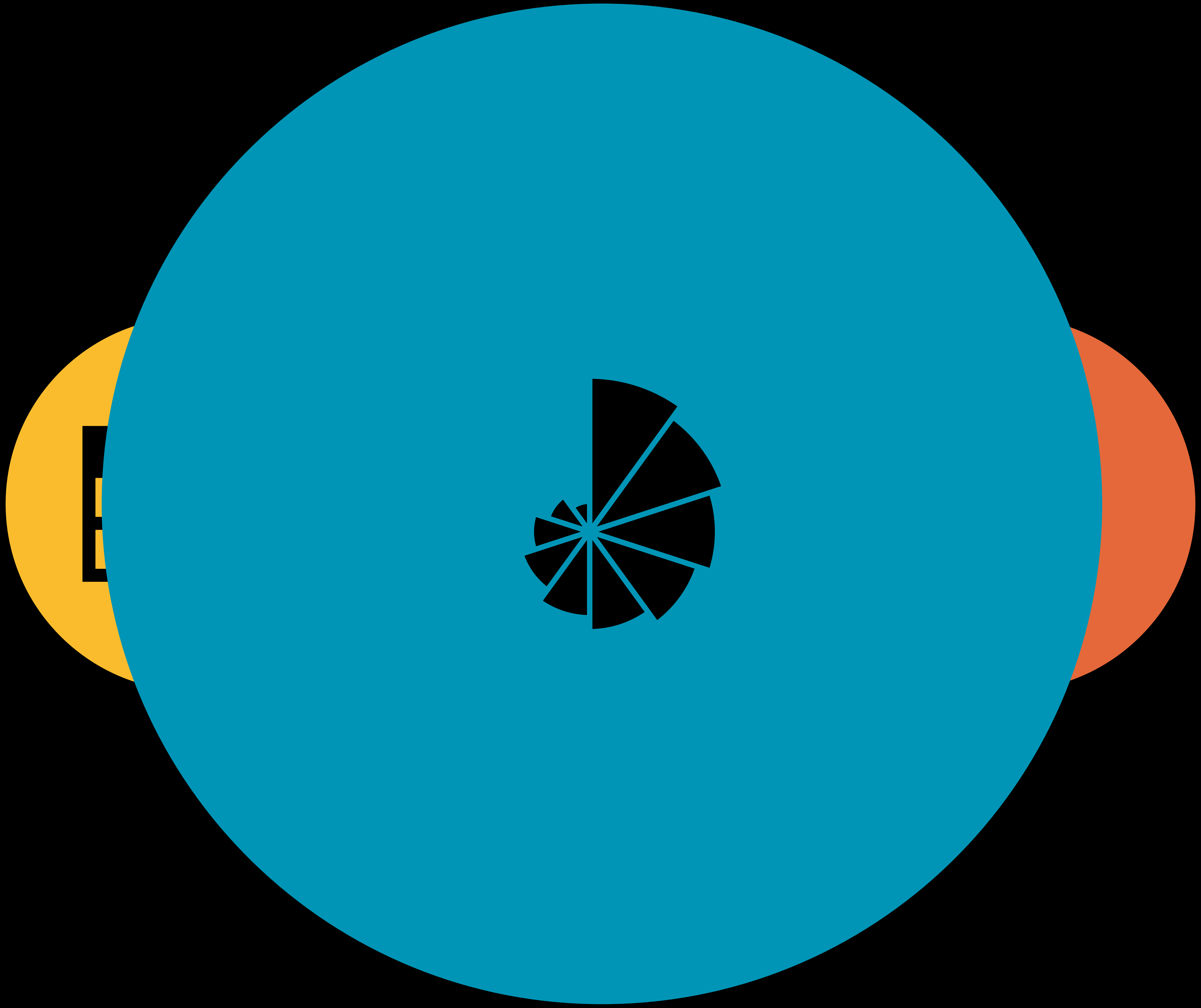
visualizations

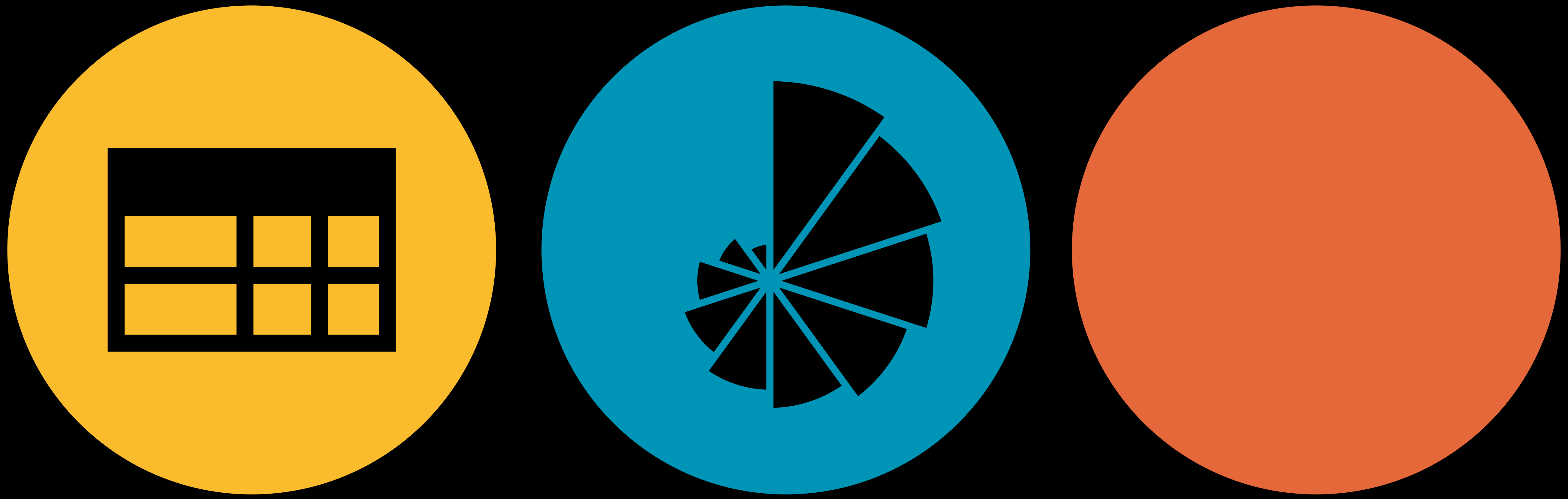


visualizations



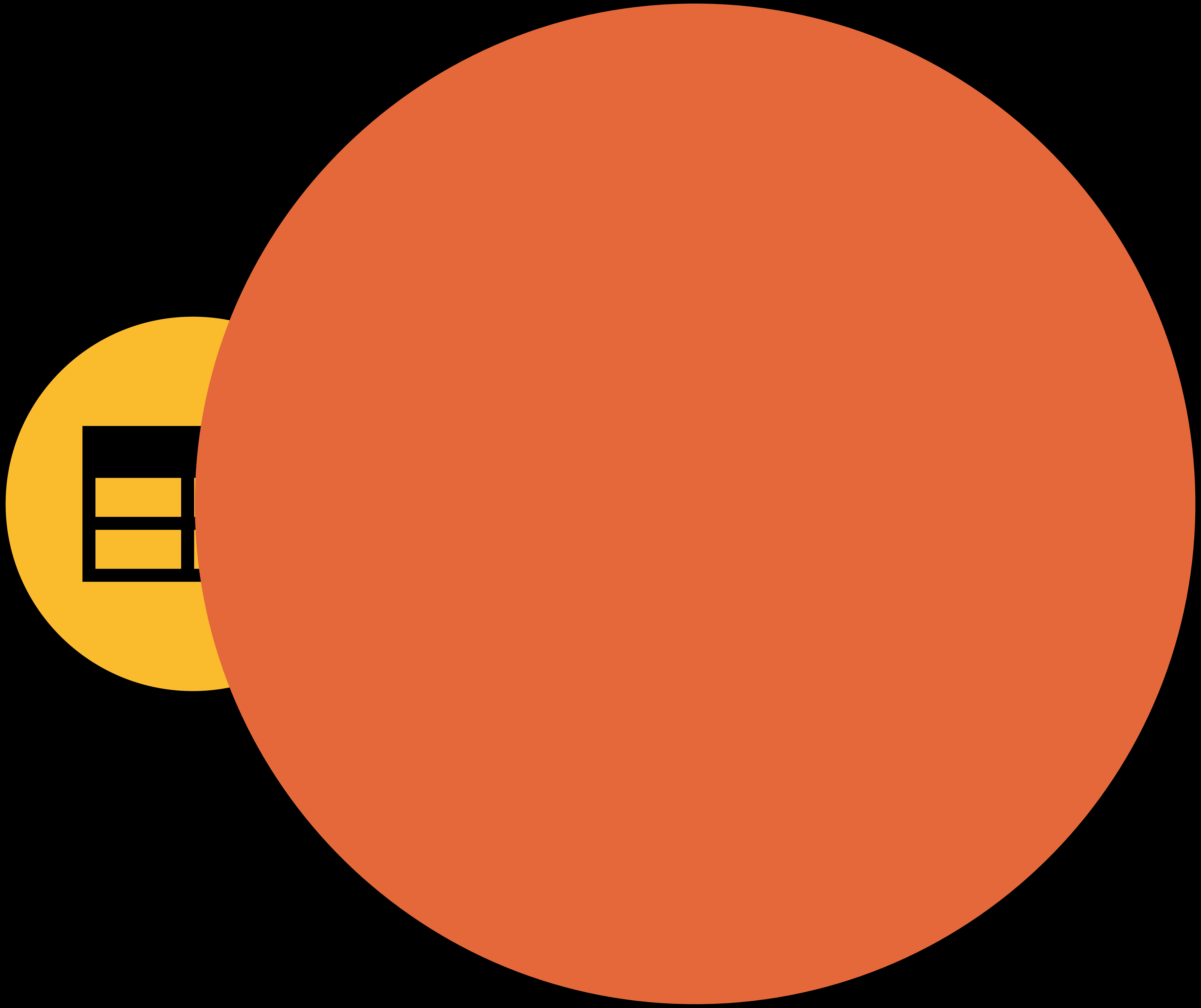






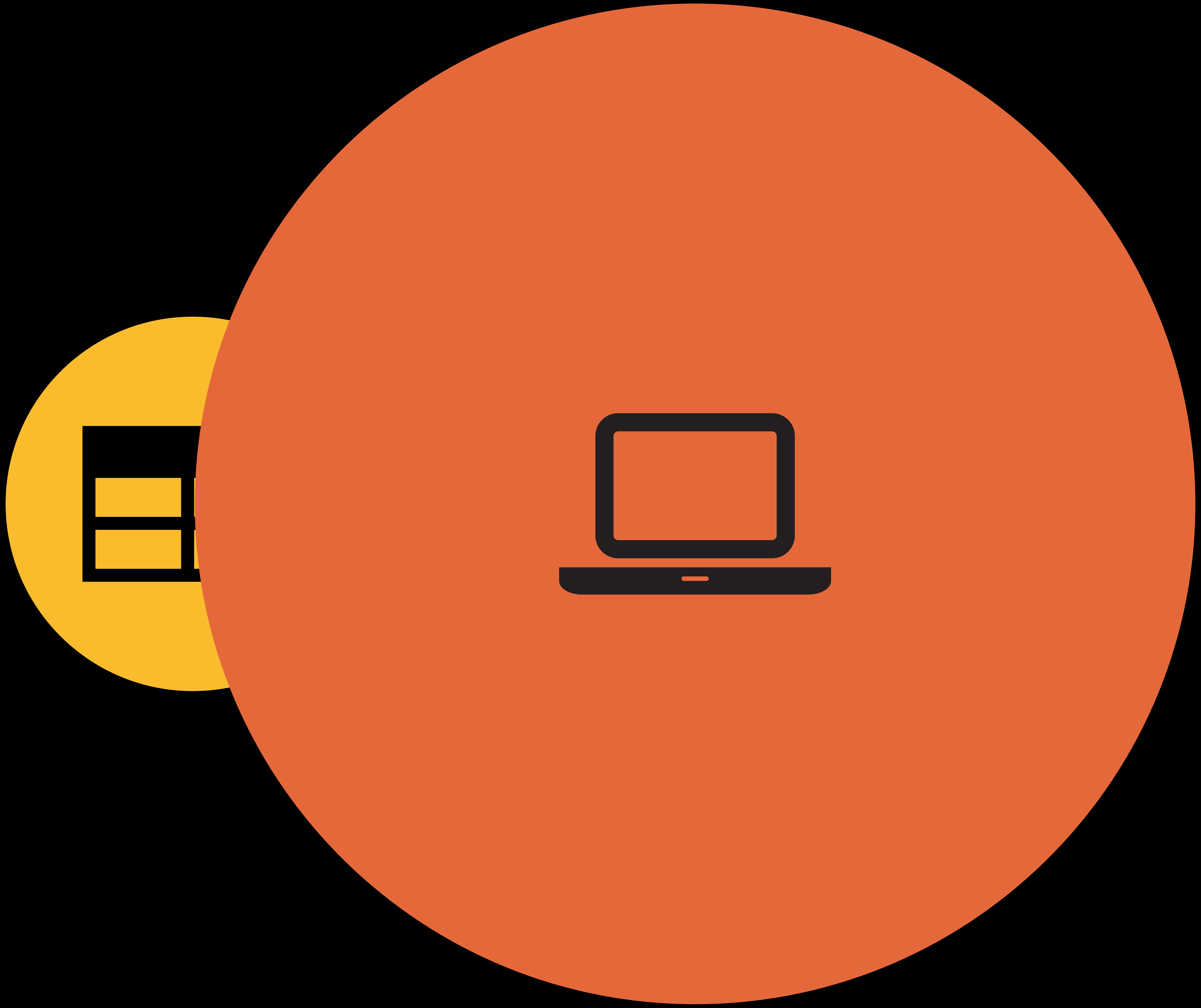
data

visualizations



devices

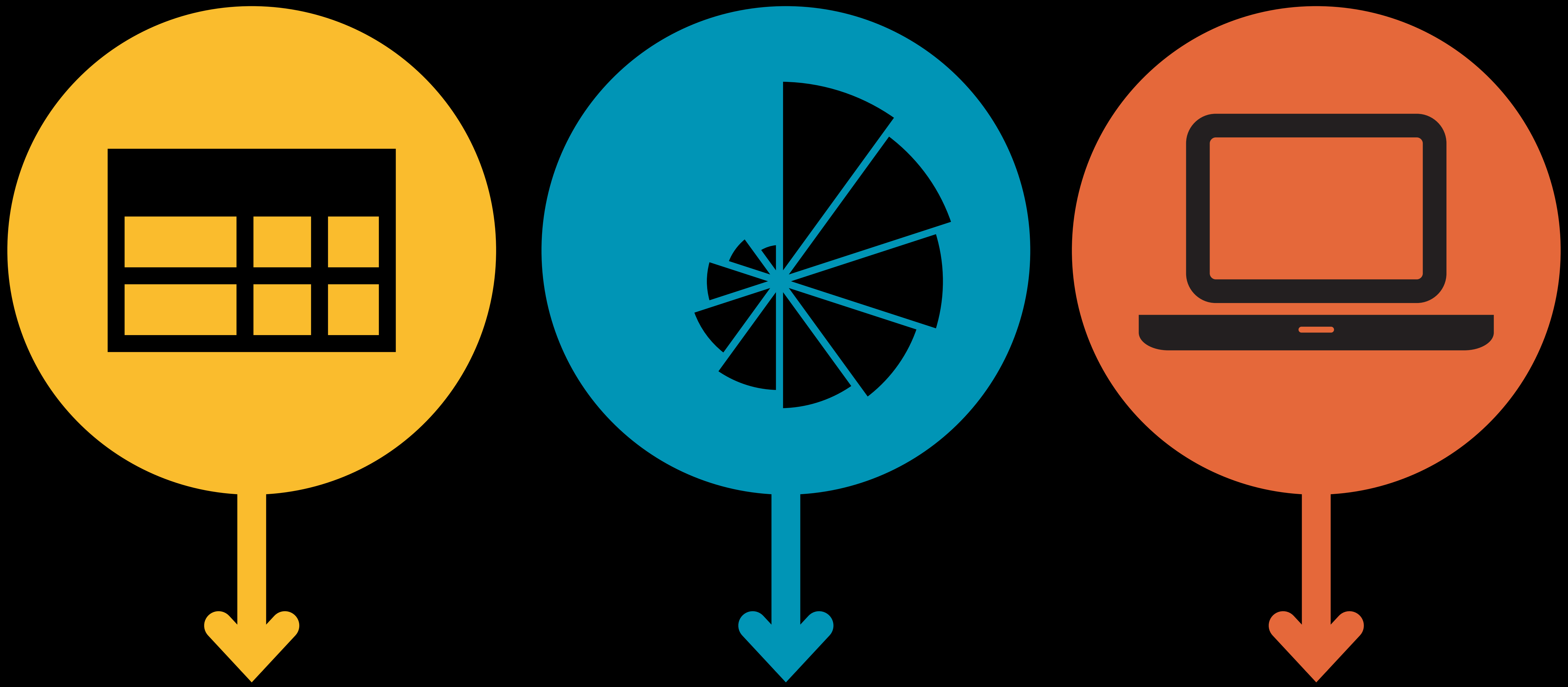
devices

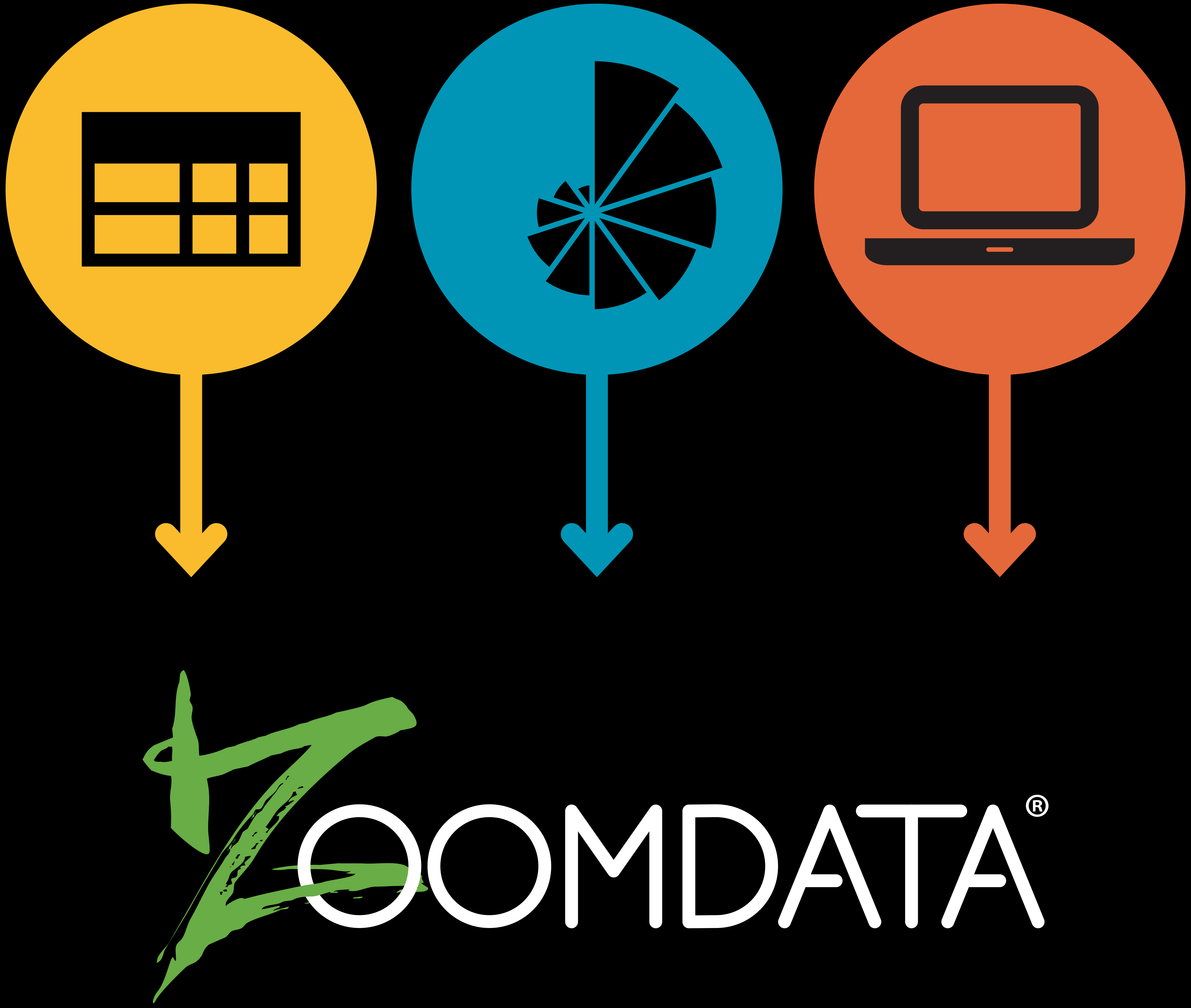


data

visualizations

devices




ZOOMDATA®

architecture

Data Connector
Hadoop
NextGen SQL
No SQL
RawCSV
Real-Time Stream Connector
Kinesis
Twitter
Cloud Connector
JSON API
Search Connectors
SOLR/Lucene
Elasticsearch
Spark It (Optional)
DataBricks
Local
Embedded
Connector Library
Connector Studio
Stream Math Library
Math Studio
Visualization Library
Visualization Studio
Data Connector Engine
Stream Processing Engine
Visualization Engine
Data Fusion Fabric
Data Cache
Data Profiler
Data Transformer
Data Controller
Browser
Widget
Mobile
Gestural
Zoomdata User
JavaScript Embed
iFrame Embed
Custom Application

architecture

Data Connector
Hadoop
NextGen SQL
No SQL
RawCSV
Real-Time Stream Connector
Kinesis
Twitter
Cloud Connector
JSON API
Search Connectors
SOLR/Lucene
Elasticsearch
Spark It (Optional)
DataBricks
Local
Embedded
Connector Library
Connector Studio
Stream Math Library
Math Studio
Visualization Library
Visualization Studio
Data Connector Engine
Stream Processing Engine
Visualization Engine
Data Fusion Fabric
Data Cache
Data Profiler
Data Transformer
Data Controller
Browser
Widget
Mobile
Gestural
Zoomdata User
JavaScript Embed
iFrame Embed
Custom Application

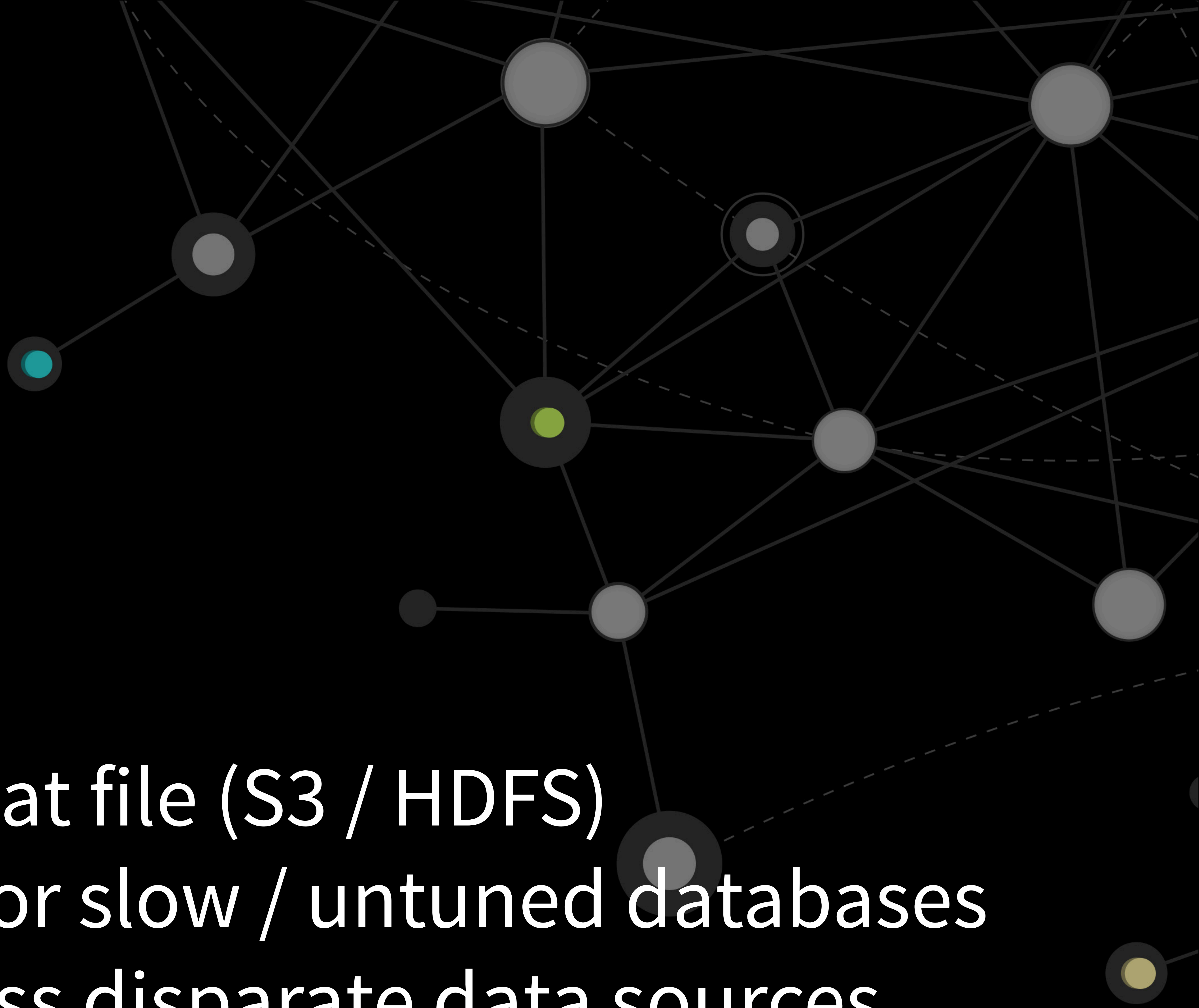
Why we’re excited about Spark

- Distributed and fast! (in memory)
- Flexible (Java / Scala / SQL / Python)
- Rich math library (MLlib,GraphX, Bagel)

We use Spark for

- Holding small datasets
- Holding aggregated datasets
- Data fusion across disparate sources
- Complex math

Benefits of Spark for us



- We can point it to any flat file (S3 / HDFS)
- Level the playing field for slow / untuned databases
- Fuse data and join across disparate data sources (SQL / noSQL / Hadoop / Search / Cloud)

Benefits of DataBricks for us

- One-step cluster setup
- Rich Math Studio to allow for complex calculations across different sources
- Direct access to RDDs

Some of our innovations

- Progressive loading into Spark (RDS/SQL sources)
- Spark analytics without SQL (w/Java, not Shark)
- Data sharpening via microqueries (non Spark’d sources)
- Sample to full (Spark’d sources)

Current challenges and next steps

- Evaluate Spark 1.0
- Sharing Spark contexts
- Sharing RDDs across contexts

Initial SparkSQL / Schema RDD findings

- Offset is not implemented
- Partitioned parquet files are not supported
- SparkSQL doesn’t allow for fetching field names and types for parquet files. We had to use com.twitter.parquet-tools to do this

Initial SparkSQL / Schema RDD findings

- Can’t find escape symbol for SQL reserved words
- “INT96” parquet type is not implemented in Spark, but Impala stores timestamps using this type
- Looks like persisting of parquet files in memory is not implemented in Spark. This can be a performance issue

killer demo