HPCC SYSTEMS

AN INTRODUCTION

Big Data

Handling massive volume of both structured and unstructured.

Structured Data

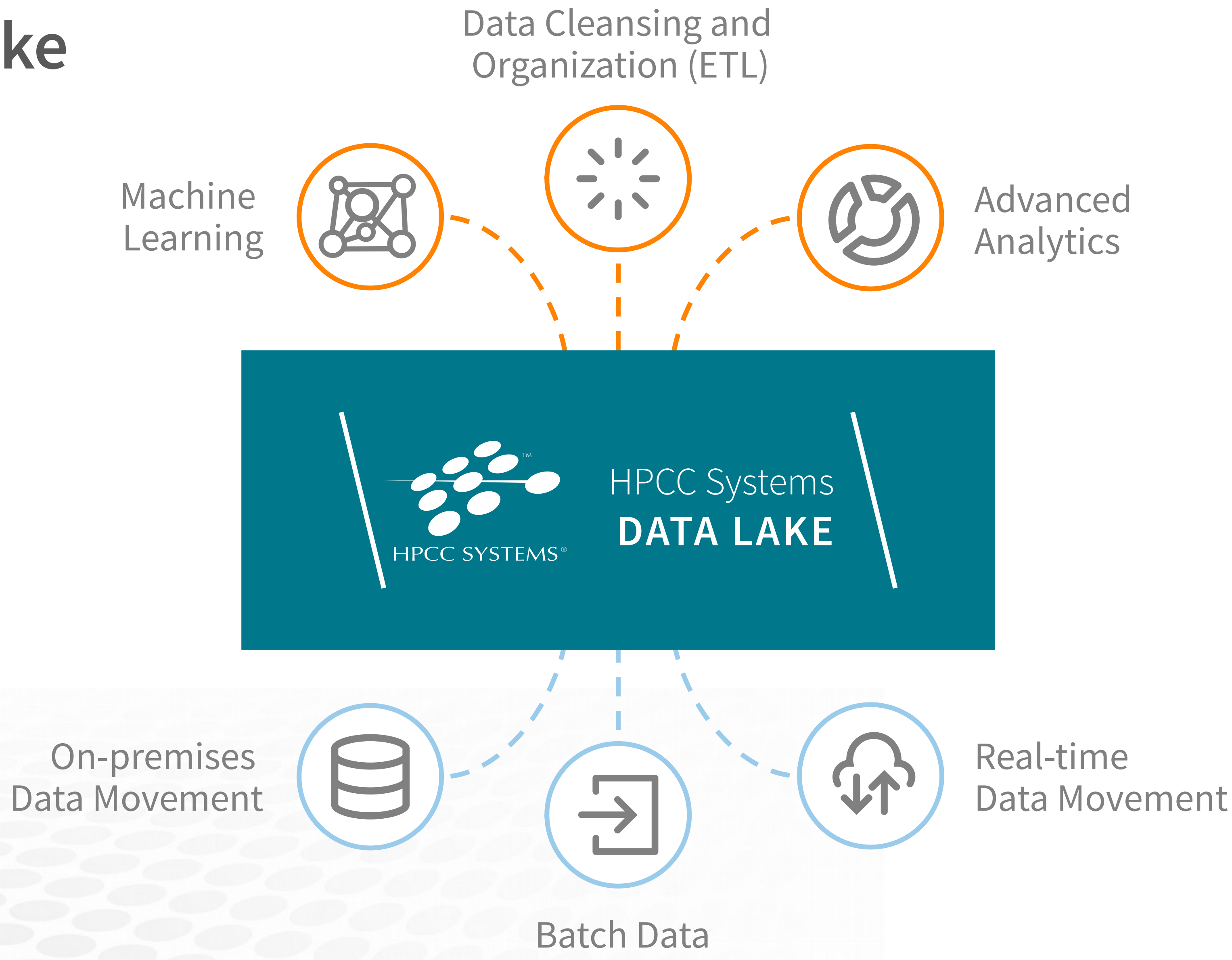
Data that can be processed, stored, and retrieved in a fixed format.
Ex: dates, salary table

Unstructured Data

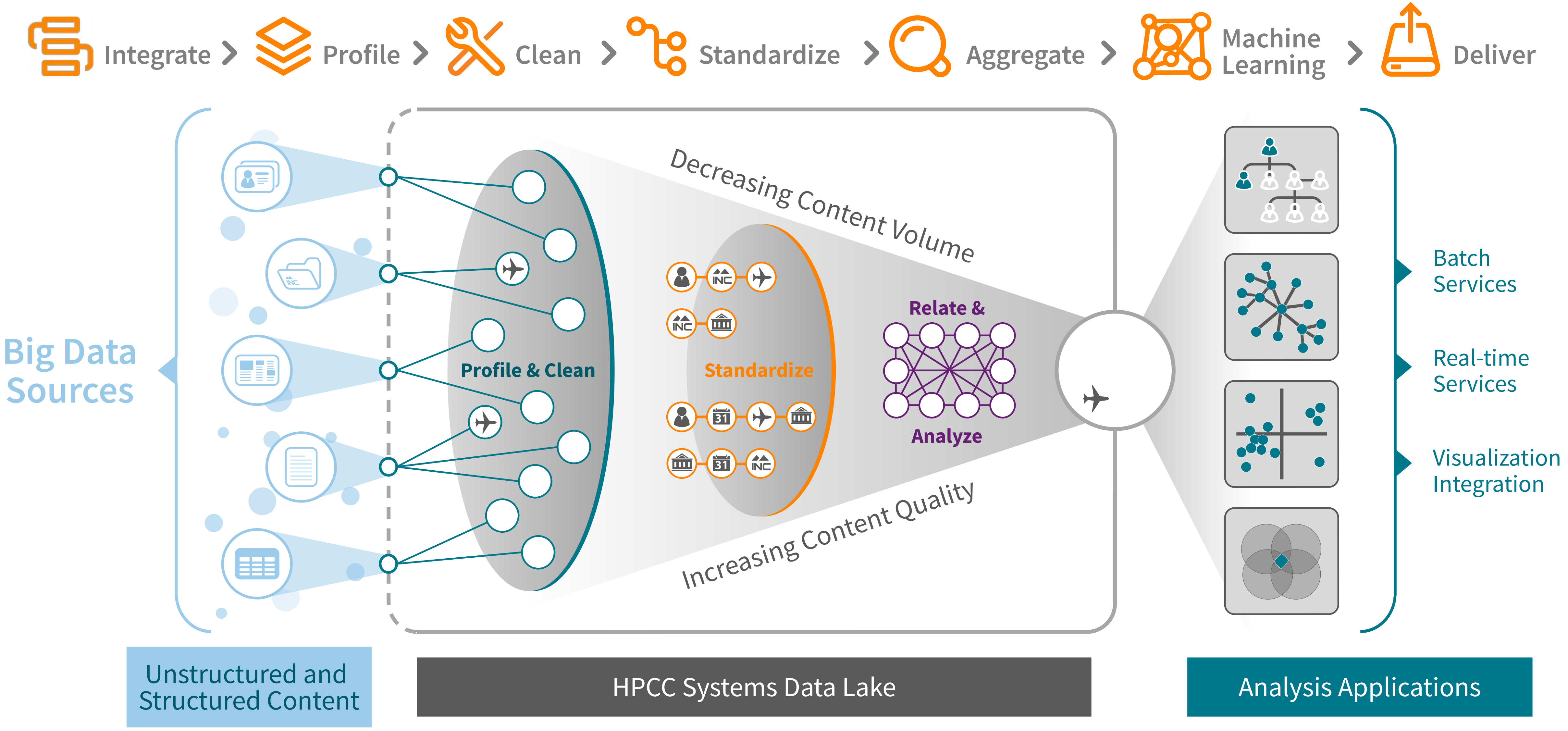
Data that lacks any specific form or structure whatsoever. This makes it very difficult and time-consuming to process and analyze unstructured data.
Ex: emails, videos.

Dipping into a Data Lake

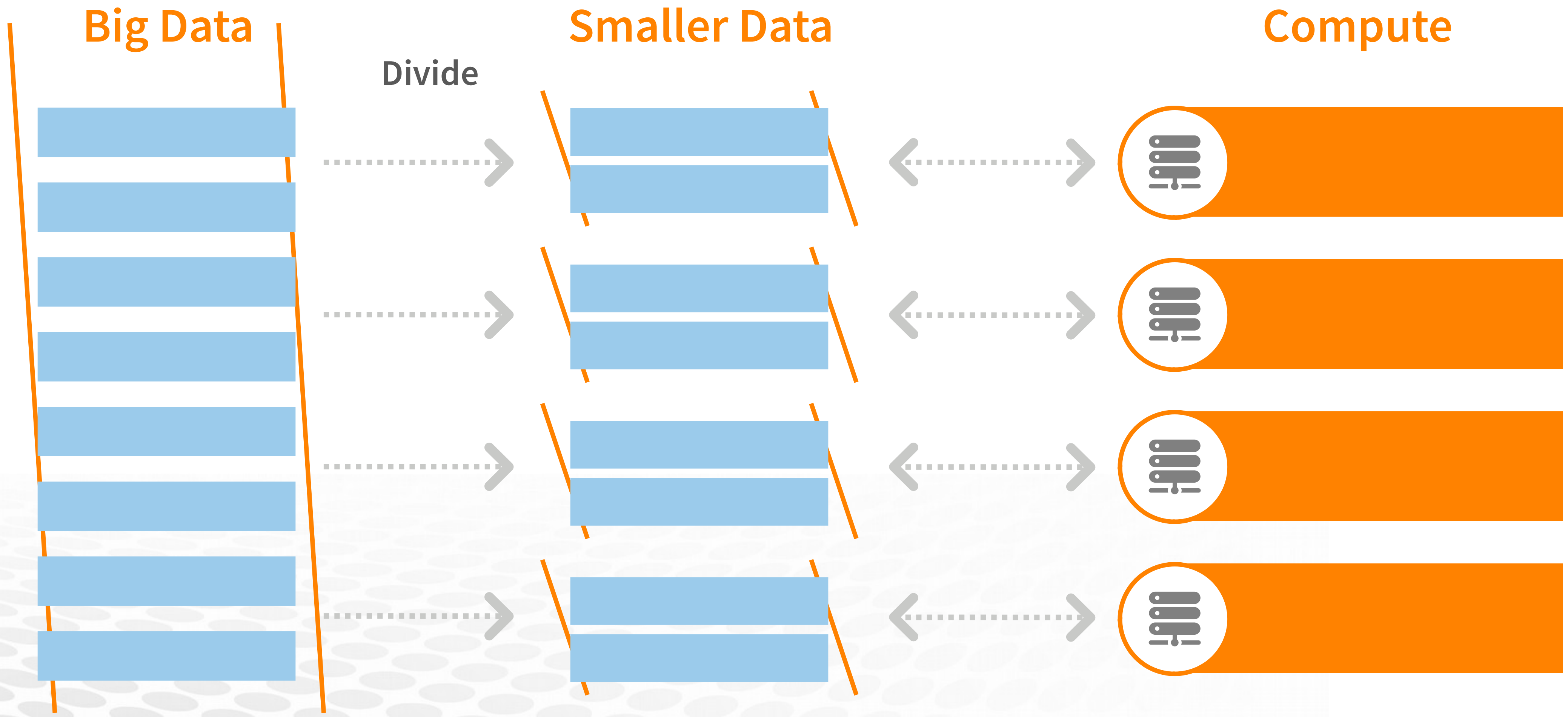
- A central (logical) repository of data
- Unlimited storage
- Schema on read
- Low cost of storage and compute
- File metadata tracking
- High performance processing



HPCC Systems (Small to Big Data) ETL



Anatomy of a Big Data Processing System



The Challenge

Use Case: Smart Hat

4,000 workers die and millions are injured annually while working on the industrial floor

Very high cost for maintaining safety in industrial businesses

The Solution + Result

Use Case: Smart Hat

Sensor-Equipped Wi-Fi Hardhats

Pushing floor information to a monitoring station, which uses a prediction engine to forecast emergency situations

Produced an industrial wearable that uses
IoT and wireless communications systems
to **protect and empower industrial workers**

Mobile Strategy Games

The mobile games industry is worth billions of dollars, with companies spending vast amounts of money on the development and marketing of these games to an equally large market. This dataset includes fields such as Name, Release Date, Description, avg rating, etc.

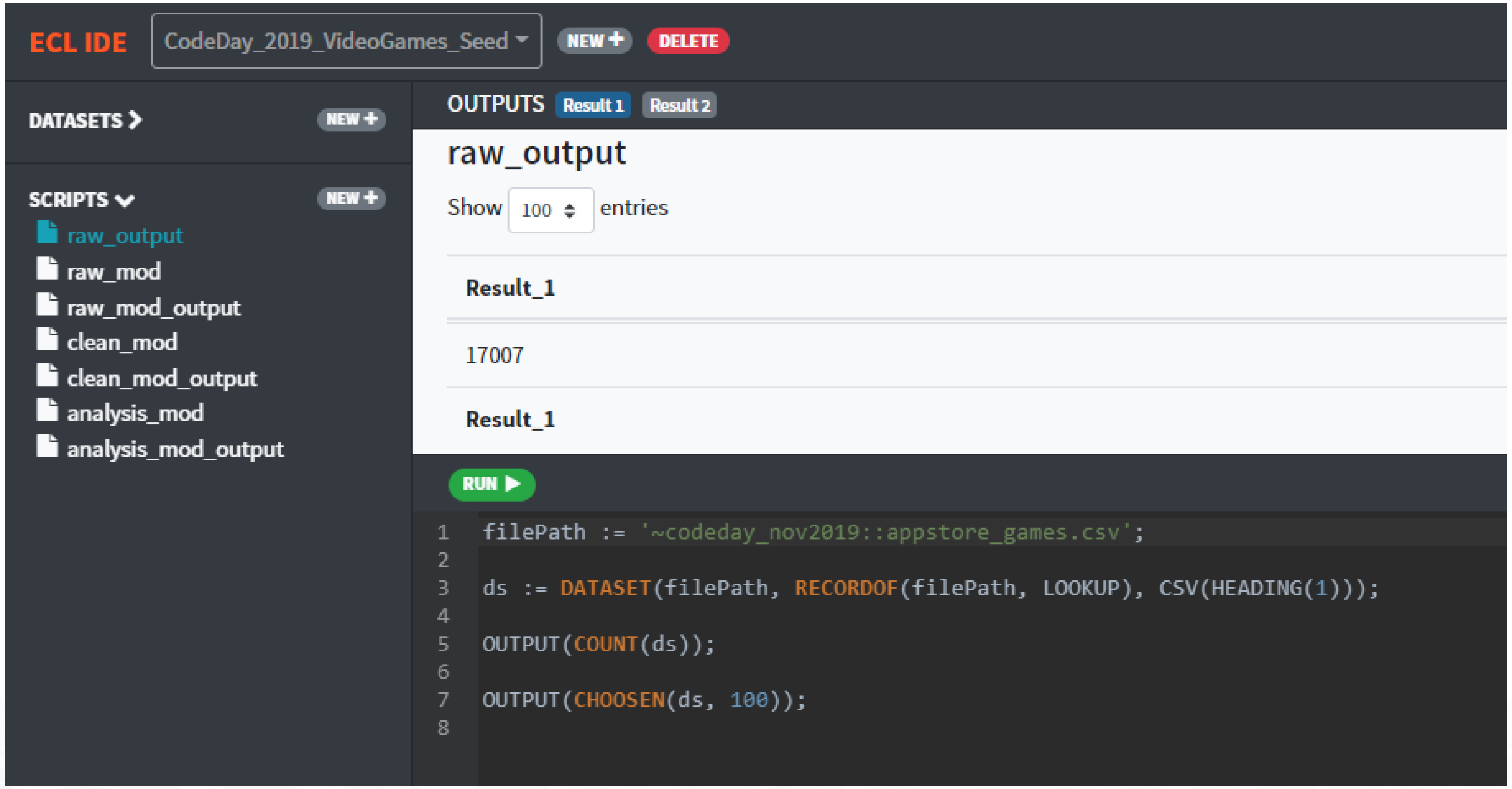
Spotify Playlists Dataset

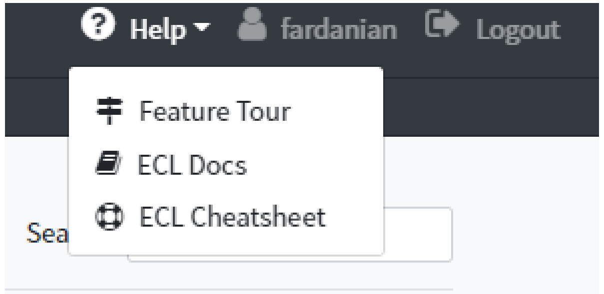
This dataset is based on the subset of users in the #nowplaying dataset who publish their #nowplaying tweets via Spotify. In principle, the dataset holds users, their playlists and the tracks contained in these playlists.

Cloud IDE

1. Register ECL Cloud IDE on Campus https://ide.hpccsystems.com/auth/login

Cloud IDE





Need help? Visit https://srnd.to/eclide

ECL

ECL is language design to query/manipulate huge data and is used for ETL (Extract, Transform, and Load) and data visualization.

- ECL is not case-sensitive.
- White space is ignored, allowing formatting for readability as needed.
- Single-line comments must begin with //.
- Block comments must be delimited with /* and */.
- ECL uses the standard object.property syntax to qualify Definition scope and disambiguate field references within tables:
  - ModuleName.Definition //reference a definition from another module/folder
  - Dataset.Field //reference a field in a dataset or recordset

Record Structure

Defines the layout o fields in the dataset, order of the fields should be the same as the dataset.

```
Layout := RECORD
   STRING pickup;
   INTEGER fare;
END;
```

File Dataset

A physical data file on disk. It can be defined directly, or can be brought in.

```
memDs := DATASET([{'2015-01-01 01:08:56', 25.10},
             {'2015-01-01 02:10:22', 40.15}], Layout);

fileDs := DATASET(
       '~Sample::file::csv',  Layout, CSV);
```

OUTPUT

```
Layout_Person := RECORD
  UNSIGNED1 PersonID;
  STRING15  FirstName;
  STRING25  LastName;
END;

allPeople := DATASET([ {1,'Fred','Smith'},
                  {2,'Joe','Blow'},
                  {3,'Jane','Smith'},
                  {4, 'Blue', 'Saturn'},
                  {5, 'Silver', 'Moon'}]
                 ,Layout_Person);

OUTPUT(allPeople, NAMED('allPeople'));      // Same as allPeople;
```

##	personid	firstname	lastname
1	1	Fred	Smith
2	2	Joe	Blow
3	3	Jane	Smith
4	4	Blue	Saturn
5	5	Silver	Moon

```
OUTPUT((CHOOSEN(allPeople, 2)), NAMED('FirstTwoRecs'));
```

##	personid	firstname	lastname
1	1	Fred	Smith
2	2	Joe	Blow

Filter

```
Layout_Person := RECORD
  UNSIGNED1 PersonID;
  STRING15  FirstName;
  STRING25  LastName;
END;

allPeople := DATASET([ {1,'Fred','Smith'},
                 {2,'Joe','Blow'},
                 {3,'Jane','Smith'}],Layout_Person);

allPeople(LastName = 'Smith');
```

##	personid	firstname	lastname
1	1	Fred	Smith
2	3	Jane	Smith

Sort

```
SortedPerson := SORT(Person, LastName, FirstName)
```

##	personid	firstname	lastname
1	2	Joe	Blow
2	1	Fred	Smith
3	3	Jane	Smith

Aggregation

```
rec := RECORD
  INTEGER Num1;
  INTEGER Num2;
  INTEGER Num3;
END;

DS := DATASET([{20,45,34}, {909,56,45}, {30,-1,90}], rec);

COUNT(DS);   //Returns 3
MAX(DS, Num1);  //Returns  909
MIN(DS, Num2);   //Returns -1
AVE(DS, Num1)    //Returns 319.6666666666667
TRUNCATE(AVE(DS, Num1));     //Returns 319
SUM(DS, Num1 + Num3);   //Returns 1128
```

Module

The MODULE structure is a container that allows you to group related definitions.
The *parameters* passed to the MODULE are shared by all the related *members* definitions.

Variable Scope:

- Local definitions are visible only through the next EXPORT or SHARED definition (including *members* of the nested MODULE structure, if the next EXPORT or SHARED definition is a MODULE).
- SHARED definitions are visible to all subsequent definitions in the structure (including *members* of any nested MODULE structures) but not outside of it.
- EXPORT definitions are visible within the MODULE structure (including *members* of any subsequent nested MODULE structures) and outside of it .

```
MyMod := MODULE
  SHARED x := 88; // local
  SHARED y := 42; // local

  EXPORT See := 'This is how a module works.’; // public
  EXPORT res := Y * 2; // public
END;

OUTPUT(MyMod.See);
```

##	Result_1
1	This is how a module works.

```
OUTPUT(MyMod.Res, Named('ViewResult'));
```

##	ViewResult
1	84

TRANSFORM

Takes an input, and modifies it into an output.

PROJECT

Applies a TRANSFORM to

- (LEFT: refers to dataset getting passed in.)

```
NameRec := RECORD
    STRING FirstName;
    STRING LastName;
END;

NameDS := DATASET([{'Sun','Shine'}, {'Blue','Moon'}, {'Silver','Rose'}], NameRec);

NameOutRec := RECORD
    STRING FirstName;
    STRING LastName;
    STRING CatValues;
     INTEGER RecCount
END;

NameOutRec CatThem(NameRec L, INTEGER C) := TRANSFORM
    SELF.CatValues := L.FirstName + ' ' + L.LastName;
    SELF.RecCount := C;
    SELF := L;
END;
CatRecs := PROJECT(NameDS, CatThem(LEFT,COUNTER));
```

##	firstname	lastname	catvalues	reccount
1	Sun	Shine	Sun Shine	1
2	Blue	Moon	Blue Moon	2
3	Silver	Rose	Silver Rose	3

TABLE

Creates a temporary dataset in memory, GROUP option can be used.

```
Layout := RECORD
   STRING10 pickup_date;
   DECIMAL8_2 fare;
   DECIMAL8_2 distance;
END;
Ds := DATASET([{'2015-01-01', 25.10, 5}, {'2015-01-01', 40.15, 8}, {'2015-01-02', 30.10, 6}, {'2015-01-02', 25.15, 4}], Layout);

crossTabLayout := RECORD
  ds.pickup_date;
  avgFare := AVE(GROUP, ds.fare);
  totalFare := SUM(GROUP, ds.fare);
END;
crossTabDs := TABLE(ds, crossTabLayout, pickup_date);
```

pickup_date	avgfare	totalfare
2015-01-01	32.625	65.25
2015-01-02	27.625	55.25

JOIN

The JOIN function produces a result set based on the intersection of two or more datasets or indexes.

INNER: Only those records that exist in both datasets.

LEFT OUTER: At least one record for every record in the left.

RIGHT OUTER: At least one record for every record in the right.

LEFT ONLY: One record for each left record with no match in the left.

RIGHT ONLY: One record for each left record with no match in the right.

FULL ONLY: One record for each left and right record with no match in the opposite.

```
MyRec := RECORD
	STRING1 Value1;
	STRING1 Value2;
END;

LeftFile := DATASET([{'C','A'},  {'X','B'},   {'A','C'}],  MyRec);

RightFile := DATASET([{'C','X'},  {'B','Y'},  {'A','Z'}], MyRec);

MyOutRec := RECORD
	STRING1 Value1;
	STRING1 LeftValue2;
	STRING1 RightValue2;
END;

MyOutRec JoinThem(MyRec L, MyRec R) := TRANSFORM
	SELF.Value1 := IF(L.Value1<>'', L.Value1, R.Value1);
	SELF.LeftValue2 := L.Value2;
	SELF.RightValue2 := R.Value2;
END;
```

Left Dataset

##	value1	value2
1	C	A
2	X	B
3	A	C

Right Dataset

##	value1	value2
1	C	X
2	B	Y
3	A	Z

```
InnerJoinedRecs := JOIN(LeftFile,RightFile,
                   LEFT.Value1 = RIGHT.Value1,
                   JoinThem(LEFT,RIGHT));
```

##	value1	leftvalue2	rightvalue2
1	C	A	X
2	A	C	Z

```
LOutJoinedRecs := JOIN(LeftFile,RightFile,
                   LEFT.Value1 = RIGHT.Value1,
                   JoinThem(LEFT,RIGHT),
                   LEFT OUTER);
```

##	value1	leftvalue2	rightvalue2
1	C	A	X
2	X	B	
3	A	C	Z

```
LOnlyJoinedRecs := JOIN(LeftFile,RightFile,
                   LEFT.Value1 = RIGHT.Value1,
                   JoinThem(LEFT,RIGHT),
                   LEFT ONLY);
```

##	value1	leftvalue2	rightvalue2
1	X	B	

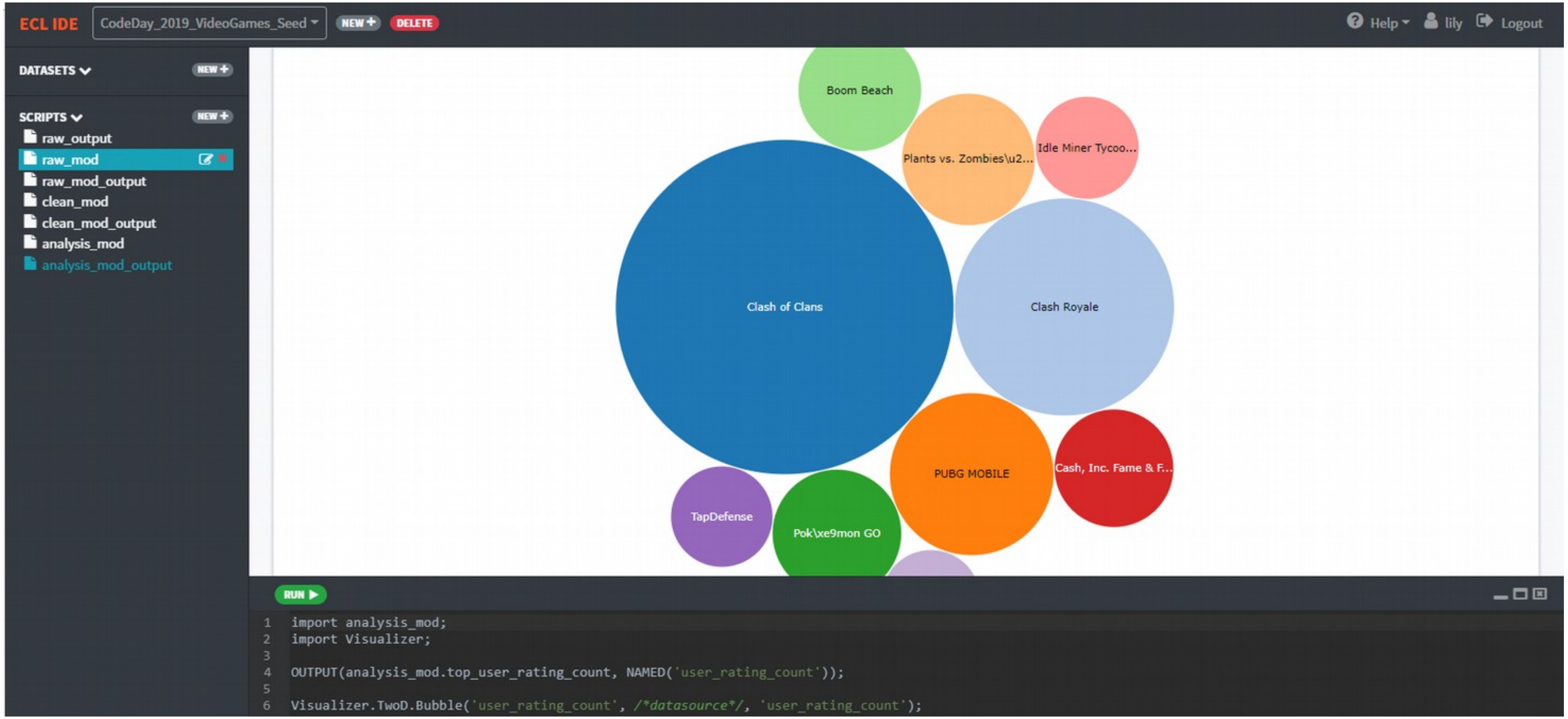
Visualization

Methods include

- Two-Dimensional
- Multi-Dimensional Methods
- Geospatial
- General

A basic visualization typically requires the following steps:

1. Creation of a suitable dataset.
2. Output the dataset with a suitable name, so that visualization can locate the data.
3. Create (and output) the visualization, referencing the named output from step




```
top_user_rating_count := TOPN( TABLE(clean_mod.games_ds, {name, user_rating_count}) , 10, -user_rating_count);
OUTPUT(analysis_mod.top_user_rating_count, NAMED('user_rating_count'));
Visualizer.TwoD.Bubble('user_rating_count', /*datasource*/, 'user_rating_count');
```

THANK YOU!

LEARN MORE AT HPCCSYSTEMS.COM