LexisNexis®
RISK SOLUTIONS

Introduction to HPCC Systems

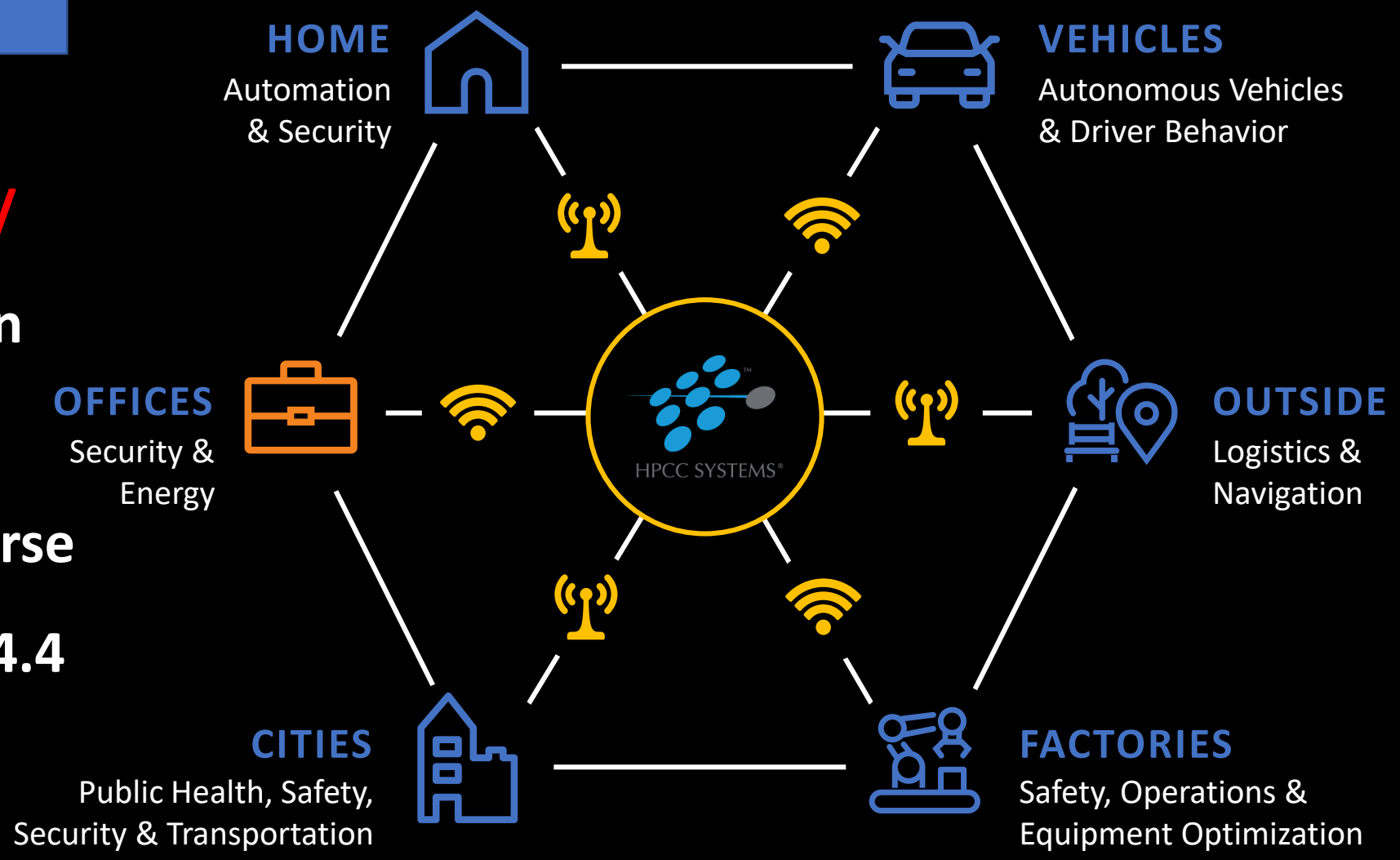
Dan S. Camper, Sr. Architect, HPCC Solutions Lab
Lili Xu, Software Engineer III, HPCC Solutions Lab

February 9, 2019

HPCC SYSTEMS®

By Year 2020

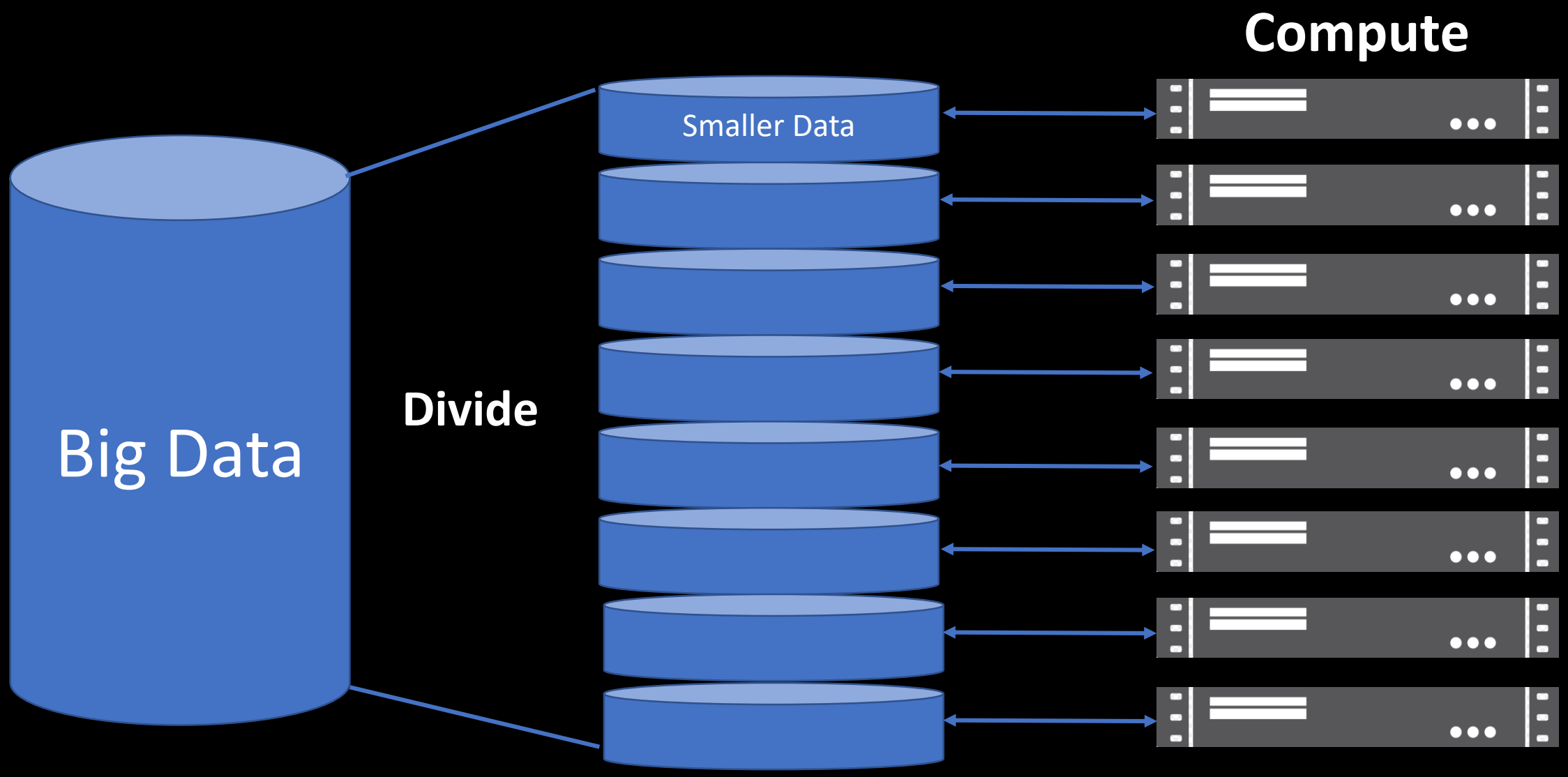
- **1.7 Megabytes of data / second for every human being**
- **Our accumulated universe of data will grow from 4.4 Zettabytes today to 44 Zettabytes by 2020**



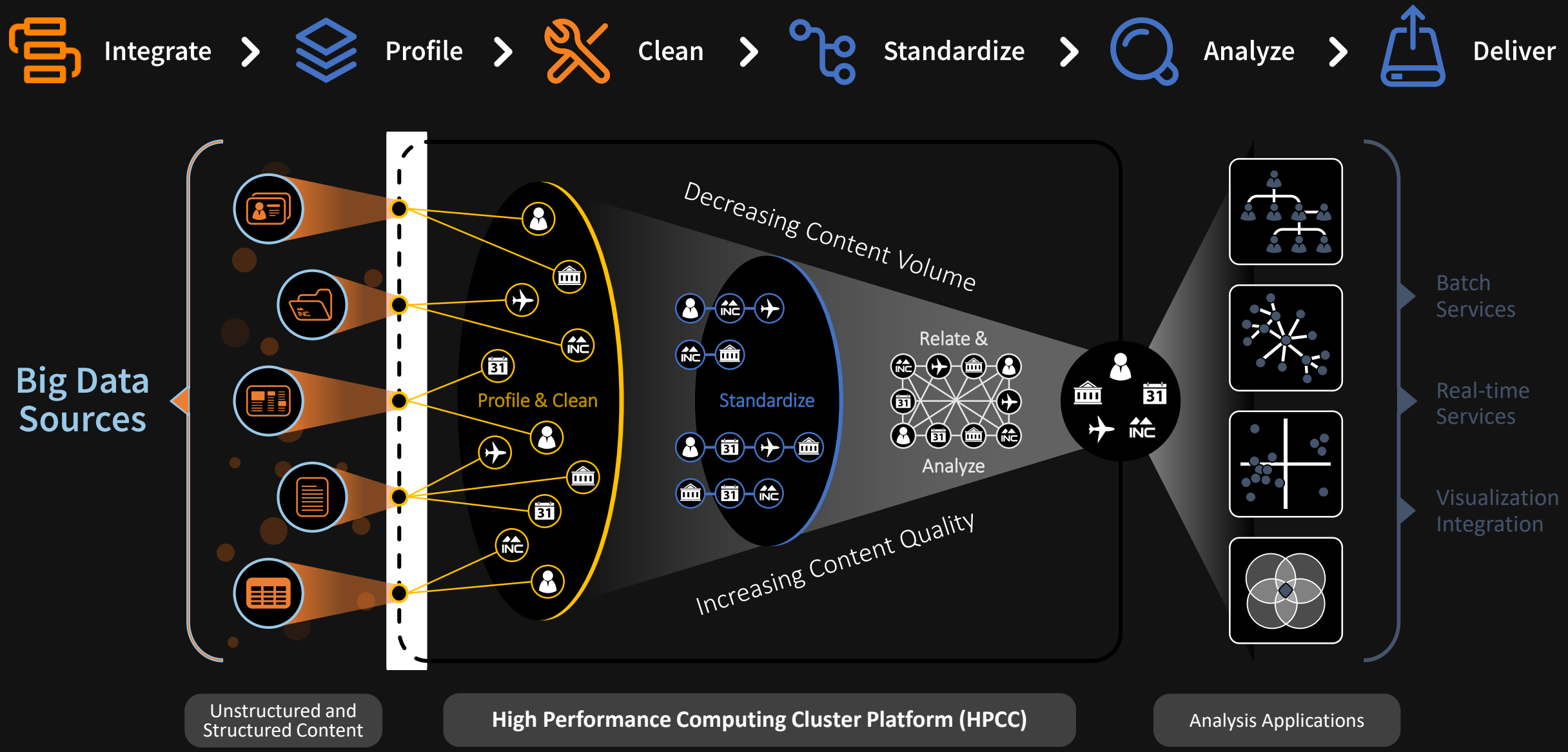
Data Scientists need in 2018

- **500,000 Jobs available**
- **Only 200,000 available Data Scientists to fill these positions**

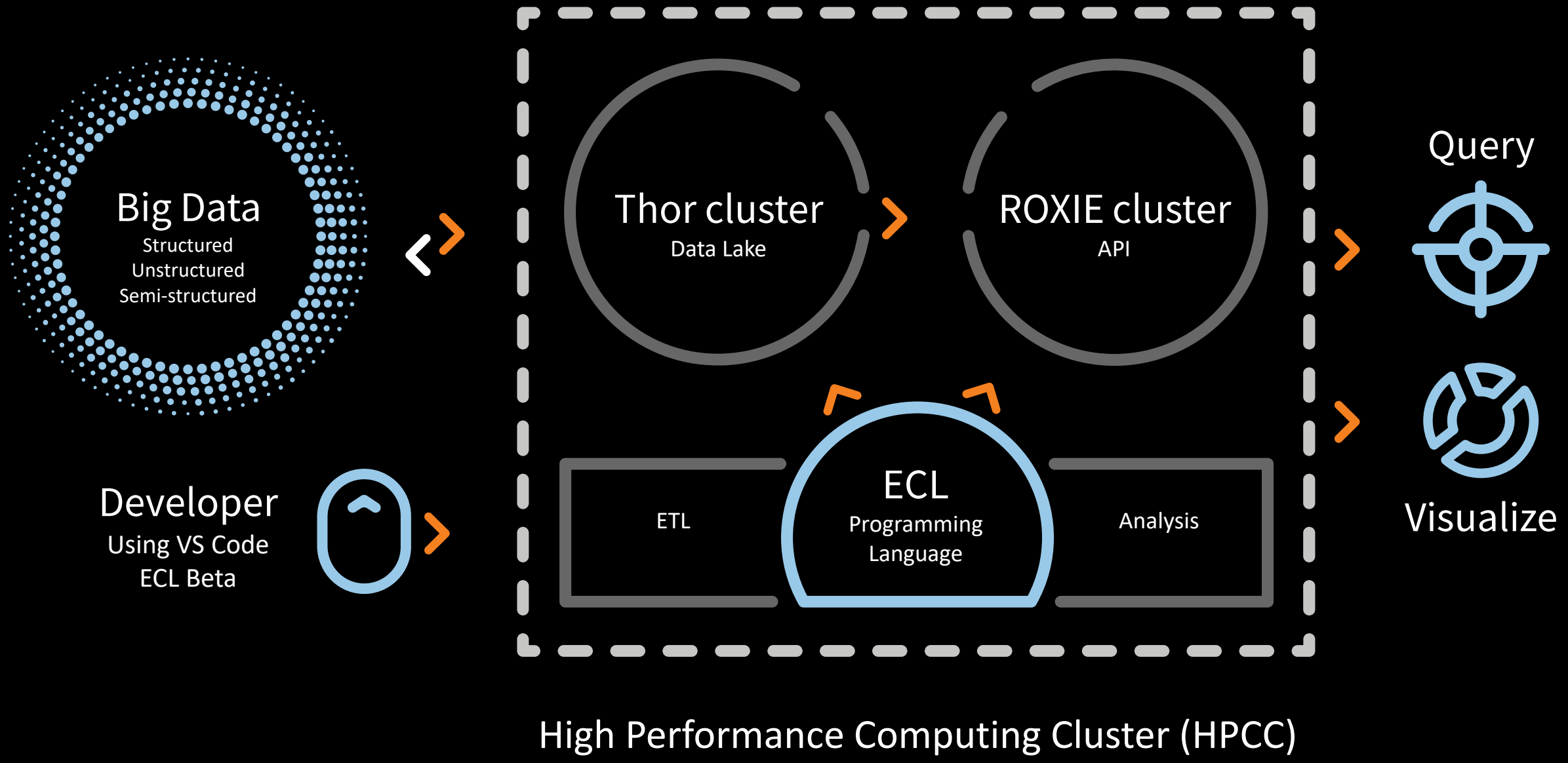
Anatomy of a Big Data processing system



Big Data Workflow



HPCC Systems Intro



ECL: What is it?

- Declarative Programming Language
  - "… a programming paradigm … that expresses the logic of a computation without describing its control flow." – Wikipedia
- Designed For Big Data Scenarios
- Any Cluster Size
- Source-to-source compiler
  - ECL code translated to C++ that is compiled to shared libraries and executed within a custom framework
- Useful Documentation (these links are in the #general Slack channel)
  - Language Reference
    - https://hpccsystems.com/training/documentation/ecl-language-reference/html
  - Standard Library
    - https://hpccsystems.com/training/documentation/standard-library-reference/html

ECL: Basic, But Important, Stuff

- Two Statement Types
  - Definition
    - Assign an expression to an attribute
  - Action
    - Actually do something that affects the outside world
- Plot Twist
  - You can define an attribute as an action

ECL: Common Data Types

- Character
  - STRING[n]
  - UTF8
  - UNICODE[_locale][n]
- Numeric
  - INTEGER[n]
  - UNSIGNED[n]
  - REAL[n]
  - DECIMAL<n>[_y]
  - UDECIMAL<n>[_y]
- Other
  - BOOLEAN
  - SET OF <type>
  - RECORD
  - DATASET

ECL: Important Minutia

- Case-insensitive
- Whitespace insensitive (within reason)
- String literals are quoted with apostrophes
- Semicolon terminator
- C++/Java style commenting
- Definition (assignment) operator is `:=`
- Equality test operator is =
- Attributes can be defined only once
- Single-pass code parser
  - Only previously-defined attributes can be referenced
- Only those definitions that contribute to a result are actually compiled and used
- There are no loops

ECL and SQL

```
new_data := PROJECT
{
    input_data,
    TRANSFORM
        {
            RECORDOF(LEFT),
            SELF.age := LEFT.age + 1,
            SELF.current_year := 2019,
            SELF := LEFT
        }
};
```

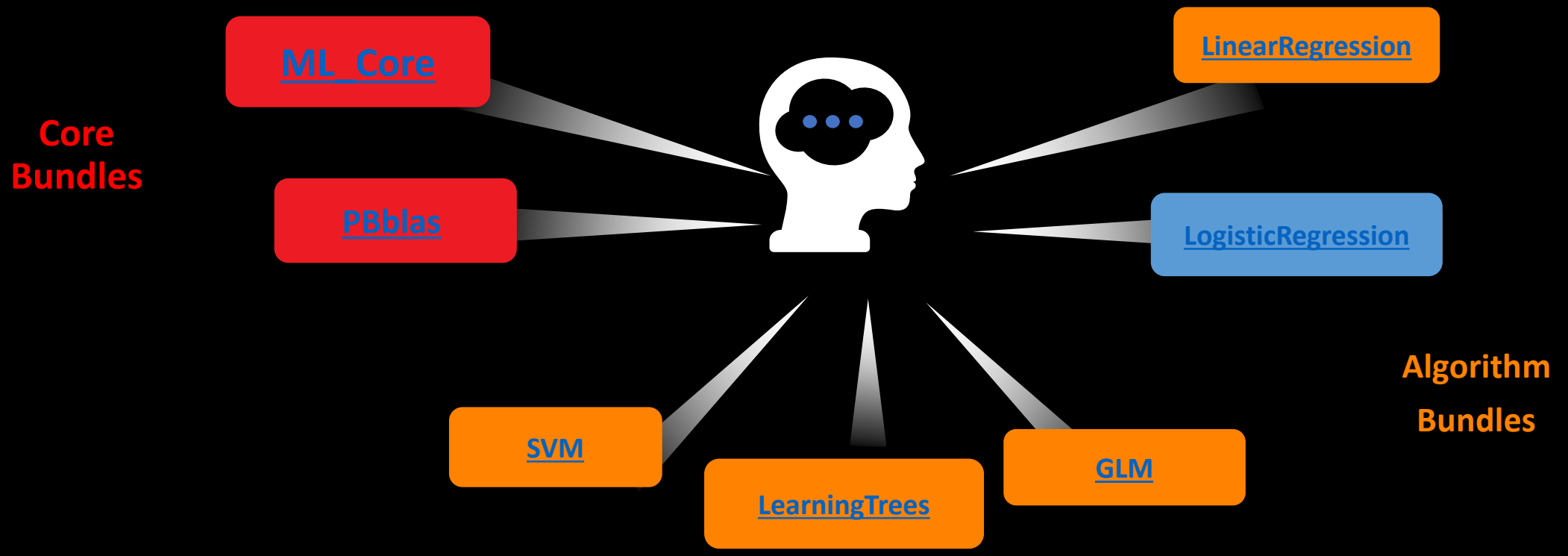
```
CREATE TABLE
    new_data
SELECT
    fname,
    mname,
    lname,
    age + 1 AS age,
    2019 AS current_year
FROM
    input_data;
```

ECL and SQL

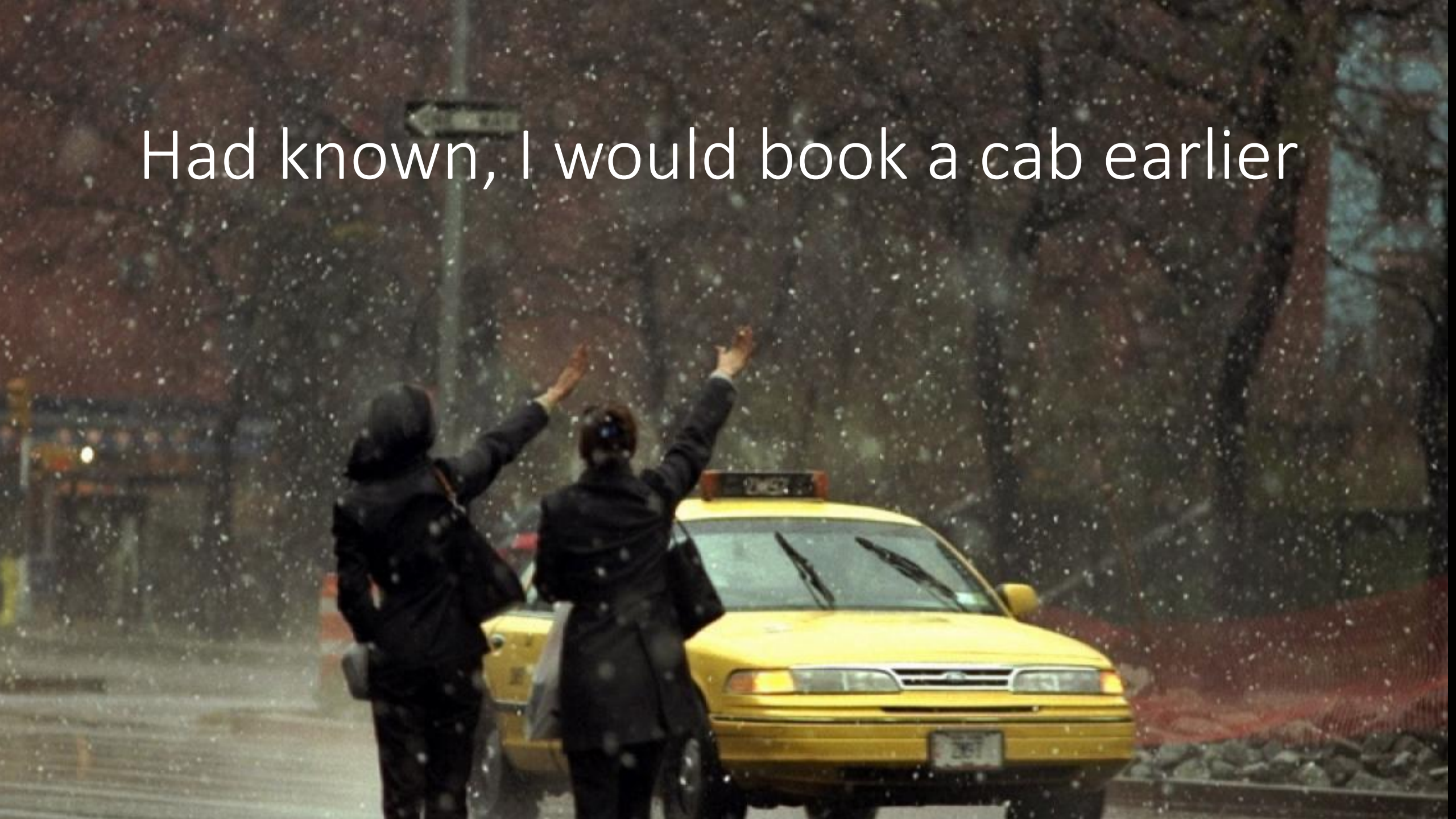
```
TABLE
{
    input_data,
    {
        make,
        model,
        DECIMAL8_2 ave_price := AVE(GROUP, sale_price)
    },
    make, model
};
```

```
SELECT
    make, model, AVE(sale_price) AS ave_price
FROM
    input_data
GROUP BY
    make, model
ORDER BY
    make, model;
```

Machine Learning on HPCC Systems Platform



New York City Taxi Trips


Had known, I would book a cab earlier

Prediction

NYC Taxi Data

48 GB
241M RECORDS
JAN 2015 – JUN 2016
16 MONTH
W/ WEATHER INFO

HPCC Systems Machine Learning

Step 1: Setup the model

- Model := LogisticRegression(100 , 0.001);

Step 2: Train the model

- Training := Model.Fit(Training Data)

Step 3: Test the model

- Testing := Model.Predict(Testing Data)

Feature Engineer

wi	Classifier	Class	Precision	Recall	FPR
1	1	1	0.428571429	0.486486486	0.533333333
1	1	0	0.525	0.466666667	0.513513514

Take Action Early!

wi	Classifier	Class	Precision	Recall	FPR
1	1	0	0.333	0.031	0.040
1	1	1	0.608	0.960	0.969

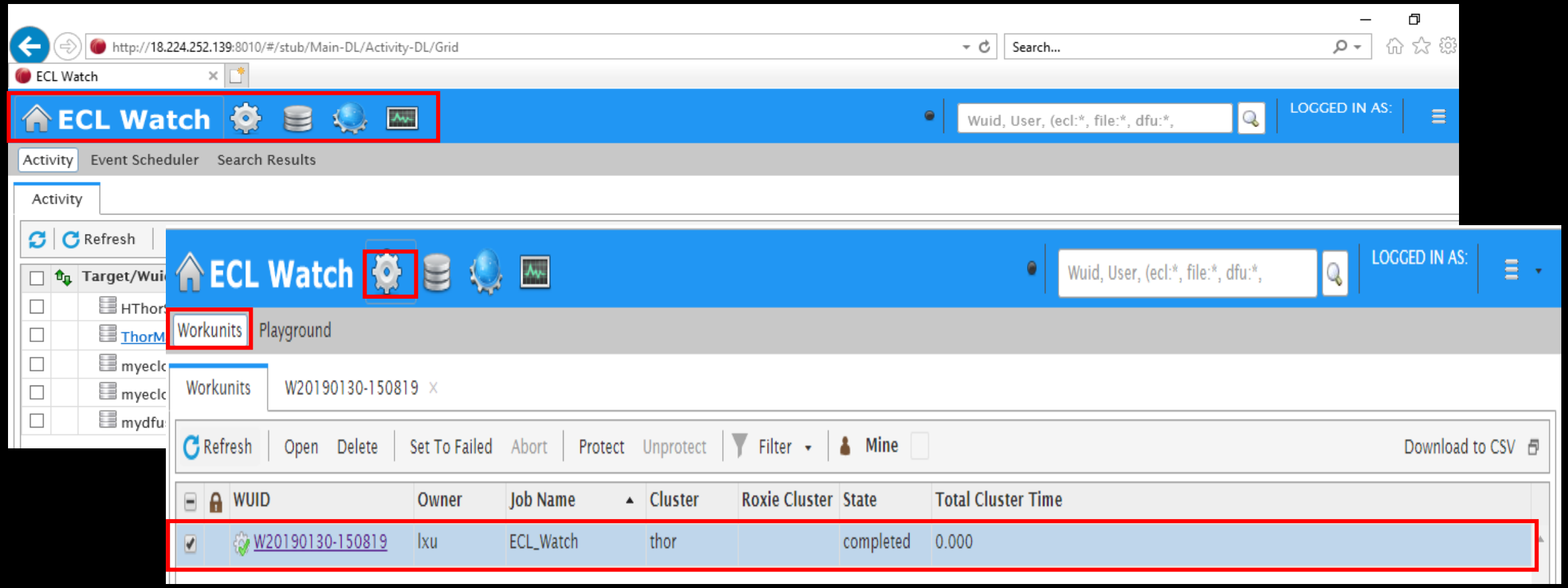
HPCC Systems Workshop Cluster

http://18.224.252.139:8010

Workshop Codebase

https://github.com/lilyclemson/Train

ECL-Watch -- A Big Data Application Monitor



ECL Playground -- Online IDE

ECL Watch

Workunits | Playground

ECL Playground

Sample: Simple Filter

Source code Scratch Pad

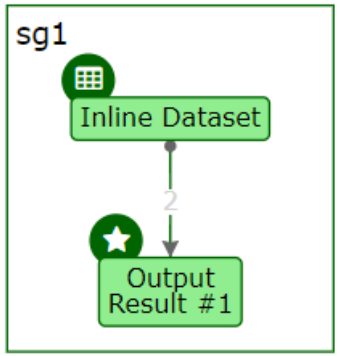
```
/*
    Example code - use without restriction.
*/
Layout_Person := RECORD
  UNSIGNED1 PersonID;
  STRING15  FirstName;
  STRING25  LastName;
END;

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

somePeople := allPeople(LastName = 'Smith');

//  Outputs  ---
somePeople;
```

Execution Graph



Submit | Target: thor | completed

Execution Results

Refresh | Download: Zip GZip XLS CSV | Filter

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

1 - 2 of 2 results

Result 1

Easy Test on HPCC Systems Platform

ECL Playground Cont.,

ECL Watch

Wuid, User, (ecl:*, file:*, dfu:*,

LOGGED IN AS:

Workunits | Playground

ECL Playground

Sample: 1_Sample_DATASET.ecl

- 1_Sample_DATASET.ecl
- 2_Sample_PROJECT.ecl
- 3_Sample_JOIN.ecl
- 4_Sample_Validate.ecl
- 5_Sample_Profiling.ecl
- 6_Sample_TABLE.ecl

```
//READ the Sample Taxi Data
//Define dataframe
Layout:= RECORD
    STRING  VendorID;                   // Franchisee ID (?)
    STRING  tpep_pickup_datetime;
    STRING  tpep_dropoff_datetime;
    STRING  passenger_count;
    STRING  trip_distance;              // Scale:  miles
    STRING  pickup_longitude;
    STRING  pickup_latitude;
    STRING  rate_code_id;
    STRING  store_and_fwd_flag;         // Y/N
    STRING  dropoff_longitude;
    STRING  dropoff_latitude;
```

Submit | Target: thor | completed

Refresh | Download: Zip GZip XLS CSV | Filter

##	vendorid	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance	p

1 - 50 of 100 results

« ‹ 1 2 › » 50

Sample_Dataset

NYC Taxi Data

vendo	tpep_pickup_datetime	tpep_dropoff_datetime	passe	trip_	pickup_longitude	pickup_latitude	rate_	store	dropoff_longitude	dropoff_latitude
2	2016-06-09 21:06:36	2016-06-09 21:13:08	2	.79	-73.983360290527344	40.760936737060547	1	N	-73.977462768554688	40.753978729248047
2	2016-06-09 21:06:36	2016-06-09 21:35:11	1	5.22	-73.981719970703125	40.736667633056641	1	N	-73.981636047363281	40.670242309570313
2	2016-06-09 21:06:36	2016-06-09 21:13:10	1	1.26	-73.994316101074219	40.751071929931641	1	N	-74.004234313964844	40.742168426513672
2	2016-06-09 21:06:36	2016-06-09 21:36:10	1	7.39	-73.98236083984375	40.773891448974609	1	N	-73.929466247558594	40.851539611816406
2	2016-06-09 21:06:36	2016-06-09 21:23:23	1	3.10	-73.987106323242187	40.733173370361328	1	N	-73.985908508300781	40.766445159912109
2	2016-06-09 21:06:36	2016-06-09 21:								
2	2016-06-09 21:06:36	2016-06-09 21:								
1	2016-06-09 21:06:37	2016-06-09 21:								
1	2016-06-09 21:06:37	2016-06-09 21:								
1	2016-06-09 21:06:37	2016-06-09 21:								
1	2016-06-09 21:06:37	2016-06-09 21:								
2	2016-06-09 21:06:37	2016-06-09 21:								
2	2016-06-09 21:06:37	2016-06-09 21:								
2	2016-06-09 21:06:37	2016-06-09 21:								

NYC Taxi Trip Data

date	minutes_after_midnight	summary	temperature	precipintensity	preciptype	windspeed	visibility	cloudcover
20150103	720	Clear	37.38	0.001	rain	8.39	3.03	0
20150103	780	Overca…	36.25	0.008	rain	7.4	3.25	1
20150103	840	Light	0	34.69	0.028	0	7.04	1.59
20150103	900	Light	0	34.86	0.036	0	5.75	1.61
20150103	960	Light	0	35.49	0.044	0	7.44	1.58
20150103	1020	Light	0	36.38	0.038	0	7.46	2.61
20150103	1080	Light	0	37.75	0.043	0	7.32	2.66
20150103	1140	Rain	39.04	0.069	rain	7.31	3	1
20150103	1200	Rain	39.97	0.065	rain	8.47	2.31	0
20150103	1260	Light	0	40.79	0.031	0	8.31	2.43
20150103	1320	Light	0	40.78	0.03	0	8.24	2.16
20150103	1380	Light	0	40.91	0.028	0	8.36	1.98

NYC Weather Data

Sample 1: READ Data

Step 1: Define Dataframe

```
Layout := RECORD
   STRING  VendorID;                    // Franchisee ID (?)
   STRING  tpep_pickup_datetime;
   STRING  tpep_dropoff_datetime;
   STRING  trip_distance;               // Scale:  miles
   STRING  store_and_fwd_flag;          // Y/N
   STRING  payment_type;                // 1 = credit; 2 = cash; there are others
  .......
END;
```

Step2: Import Data

```
//Define data directory
Path := '~yellow_tripdata_2016-06.csv';
Read_data := DATASET( PATH, Layout, CSV(HEADING(1) ));
```

Sample 2: Transform Data

Step 1: Define Dataframe

```
Layout := RECORD
   Date.Date_t    date;
   Date.Time_t    time;
   UNSIGNED2     minutes_after_midnight;
END;
```

Step2: Transform Data

```
Reform_data := PROJECT( data, TRANSFORM( Layout,

        SELF.date :=Std.Date.FromStringToDate(LEFT.tpep_pickup_datetime[..10], '%Y-%m-%d'),
        SELF.time :=Std.Date.FromStringToTime(LEFT.tpep_pickup_datetime[12..], '%H:%M:%S'),
        SELF.minutes_after_midnight:= Std.Date.Hour(SELF.pickup_time)* 60 ....)
 );
```

Sample 3: Combine Data

```
A_join_B :=  JOIN(  A,  B,  LEFT.date = RIGHT.date,
 TRANSFORM
  (
     Layout_of_A_JOIN_B,
     SELF.date := LEFT.date,
     SELF.minutes_after_midnight := RIGHT.minutes_after_midnight,
     SELF.summary := RIGHT.summary,
     SELF.temperature := RIGHT.temperature,
     SELF.windSpeed := RIGHT.windSpeed,
     SELF.visibility := RIGHT.visibility,
     SELF.cloudCover := RIGHT.cloudCover,
     SELF := LEFT
  ),
  LEFT OUTER
);
```

Sample 4: Validate Data

ECL Watch

Wuid, User, (ecl:*, file:*, dfu:*,

LOGGED IN AS:

Workunits | Playground

ECL Playground

Sample: 4_Sample_Validate.ecl

```
IMPORT ML_Core.Types AS Types;
IMPORT STD;

//Data Validation
//Read integrated data
//Define dataframe
lsample_weather_raw := RECORD
  Types.t_RecordID id := 0;
  Std.Date.Date_t         date;
  Std.Date.Seconds_t      minutes_after_midnight;
  STRING                  summary;                // 1 Breezy; 2    Clear; 3 Possible; 4 Snow; 5 Windy; 6 ''; 7 Mostly;
                                                  // 8 Heavy; 9 Humid; 10 Overcast;11   Foggy; 12 Light; 13 Partly; 14 Rain
  DECIMAL6_3              temperature;            // Fahrenheit
  UDECIMAL6_3             precipIntensity;        // [0.00, 92.03]
  STRING                  precipType;             // '', 'SNOW' , 'RAIN'
  UDECIMAL4_2             windSpeed;              // MPH [0.00 - 87.75]
  UDECIMAL4_2             visibility;             // Miles [0.00 - 20.23]
  UDECIMAL4_2             cloudCover;             // [0.00 - 1.00]
```

Sample 5: Profile Data

```
IMPORT DataPatterns;
//Profile data
Data_Profiling:= DataPatterns.Profile(Data);
```

attribute	given_attribute_type	best_attribute_type	rec_count	fill_count	fill_rate	cardinality	cardinality_breakdo…		modes		min_length	max_length
							value	rec_count	value	rec_count		
temperature	decimal6_3	decimal6_3	2843404	34954	1.229301	3	79.03	18205	79.03	18205	5	5

popular_patterns			rare_patterns			is_numeric	numeric_min	numeric_max	numeric_mean	numeric_std_dev	numeric_lower_quartile	numeric_median
data_pattern	rec_count	example	data_pattern	rec_count	example							
99.99	34954	66.87				true	66.87	79.03	73.7058	5.6317	66.87	79.03

numeric_lower_quartile	numeric_median	numeric_upper_quartile	numeric_correlations	
			attribute	corr
66.87	79.03	79.03	windspeed	0.9680…
			id	0.1215…
			visibility	0.0547…
			date	0.0263…
			minutes_after_midnight	-0.149…
			precipintensity	-0.542…
			cloudcover	-0.542…

Sample 6: Analyze Data

```
//Aggregate the total taxi trips per day
trips_per_day := TABLE(taxi_data, {date, INTEGER trips := COUNT(GROUP)}, date);
```

##	date	trips
1	20160601	107307
2	20160602	87887
3	20160603	61900
4	20160604	156190
5	20160605	81823
6	20160606	187326
7	20160607	108800
8	20160608	104989
9	20160609	8973
10	20160610	32548
11	20160611	263528
12	20160612	4778

Useful Links

- Open Source HPCC Systems Platform: Home Page
- Internship Program
- Online Training
- Download Page
- Our GitHub portal
- Community Forums
- Getting Started with ECL
- Advanced ECL
- Latest Release and Documentation
- Supported plugins, connectors, third party modules and bundles
- Machine Learning on HPCC Systems
- VS Code and HPCC Systems Installation Cheat Sheet

Join our Community

Help us make HPCC Systems better. Register on our community portal.

LexisNexis®
RISK SOLUTIONS