

# ECL Cheat Sheet

A simple introduction to ECL — so you can master it with ease.

[https://github.com/hpcc-systems/HPCC-ECL-Training/blob/master/CheatSheet/ECL\\_Cheat\\_Sheet.pdf](https://github.com/hpcc-systems/HPCC-ECL-Training/blob/master/CheatSheet/ECL_Cheat_Sheet.pdf)



## Dataset

A representation of data on disk or created in memory. Most ECL functions return a DATASET.

```
Layout := RECORD
  STRING pickup_dt;
  DECIMAL8_2 fare;
END;

//Reading embedded data
memDs := DATASET([{'2019-01-01 01:08:56', 25.10},
                 {'2019-01-01 02:10:22', 40.15}],
                Layout);

OUTPUT(memDs);

//Reading CSV file data
fileDs := DATASET(
  '~\tutorials::cheatsheet::in::sample_trip_1.csv',
  Layout, CSV);

OUTPUT(fileDs);
```

INPUT	
pickup_dt	Fare
2019-01-01 01:08:56	25.10
2019-01-01 02:10:22	40.15



OUTPUT	
typ	val
sum	65.25
ave	32.63
min	25.1
max	40.15
count	2

```
Layout := RECORD
  STRING pickup_dt;
  DECIMAL8_2 fare;
END;

ds := DATASET([{'2019-01-01 01:08:56', 25.10},
               {'2019-01-01 02:10:22', 40.15}],
              Layout);

sumVal := SUM(ds, ds.fare);
avgVal := AVE(ds, ds.fare);
minVal := MIN(ds, ds.fare);
maxVal := MAX(ds, ds.fare);
countVal := COUNT(ds);

OUTPUT(DATASET([{'sum', sumVal},
                {'avg', avgVal},
                {'min', minVal},
                {'max', maxVal},
                {'count', countVal}],
              {String typ, DECIMAL8_2 val}));
```

## Observe Subset

Select a subset of rows in a dataset for observation.

```
Layout := RECORD
  STRING10 pickup_date;
  DECIMAL8_2 fare;
  DECIMAL8_2 distance;
END;

ds := DATASET([{'2019-01-01', 25.10, 5},
               {'2019-01-01', 40.15, 8},
               {'2019-01-02', 30.10, 6},
               {'2019-01-02', 25.15, 4}], Layout);

//Filter records by fields
filterDs := ds(pickup_date='2019-01-01');

//Remove duplicate records
dedupDs := DEDUP(SORT(ds, pickup_date),
                pickup_date);

//Returns top N records
chosenDs := CHOSEN(ds, 2);//Return top 2 records

//Return top N records after sorting
topDs := TOPN(ds, 2, pickup_date);

//Return sample part of set
sampleDs := SAMPLE(ds, 2, 1);//return every 2nd record

//Return sample set of records
enthDs := ENTH(ds, 1, 2, 1);//1 out of every 2

OUTPUT(filterDs);
OUTPUT(dedupDs);
OUTPUT(topDs);
OUTPUT(sampleDs);
OUTPUT(enthDs);
```

INPUT		



OUTPUT		

## Shape with Project

Used to transform datasets with the same number of records but transformed columns.

```
IMPORT Std;

InputLayout := RECORD
  STRING pickup_datetime;
  DECIMAL8_2 fare;
  DECIMAL8_2 distance;
END;

OutputLayout := RECORD
  Std.Date.Date_t pickup_date;
  Std.Date.Time_t pickup_time;
  DECIMAL8_2 fare;
  DECIMAL8_2 distance;
END;

inputDs := DATASET([{'2019-01-01 10:00:00', 25.10, 5},
                   {'2019-01-01 11:00:00', 40.15, 8},
                   {'2019-01-02 10:00:00', 30.10, 6},
                   {'2019-01-02 11:00:00', 25.15, 4}],
                  InputLayout);

outputDs := PROJECT(inputDs, TRANSFORM(OutputLayout,
  SELF.p Pickup_datetime :=
  Std.Date.FromStringToDate(LEFT.pickup_datetime[..<10],
  '%Y-%m-%d'),
  SELF.p Pickup_time :=
  Std.Date.FromStringToTime(LEFT.pickup_datetime[12..],
  '%H:%M:%S'),
  SELF.f Fare := LEFT.fare,
  SELF.d Distance := LEFT.distance));

OUTPUT(outputDs);
```

INPUT		
pickup_datetime	fare	dist
2019-01-01 10:00:00	25.10	5
2019-01-01 11:00:00	40.15	8
2019-01-02 10:00:00	30.10	6
2019-01-02 10:00:00	25.15	4



OUTPUT		
20190101	100000	25.10 5
20190101	110000	40.15 8
20190102	100000	30.10 6
20190102	110000	25.15 4

## Group

Easily work with cross tab functionality by using GROUP and TABLE functions.

```
Layout := RECORD
  STRING10 pickup_date;
  DECIMAL8_2 fare;
  DECIMAL8_2 distance;
END;

ds := DATASET([{'2019-01-01', 25.10, 5},
               {'2019-01-01', 40.15, 8},
               {'2019-01-02', 30.10, 6},
               {'2019-01-02', 25.15, 4}], Layout);

crossTabLayout := RECORD
  ds.pickup_date;
  avgFare := AVE(GROUP, ds.fare);
  totalFare := SUM(GROUP, ds.fare);
  varianceFare := VARIANCE(GROUP, ds.fare);
  coVarianceFareDist := COVARIANCE(GROUP,
  ds.fare, ds.distance);
  correlateFareDist := CORRELATION(GROUP,
  ds.fare, ds.distance);
END;

crossTabDs := TABLE(ds, crossTabLayout, pickup_date);
OUTPUT(crossTabDs);
```

INPUT		
pickup_date	fare	distance
2019-01-01	25.10	5
2019-01-01	40.15	8
2019-01-02	30.10	6
2019-01-02	25.15	4



OUTPUT					
pickup_date	avgfare	totalfare	variance fare	covariancefareDist	correlatefareDist
2019-01-01	32.625	62.25	56.62	11.28	1
2019-01-02	27.625	55.25	6.125	2.47	1

## Shape with Rollup

In one way, ROLLUP is used combine related records into a single aggregate record, like an aggregating SQL self join.

```
Layout := RECORD
  STRING10 pickup_date;
  DECIMAL8_2 fare;
  DECIMAL8_2 distance;
  DECIMAL8_2 mileageDeduction := 0;
END;

inputDs := DATASET([{'2019-01-01', 25.10, 5},
  {'2019-01-01', 40.15, 8},
  {'2019-01-02', 30.10, 6},
  {'2019-01-02', 25.15, 4}], Layout);

// Rollup (aggregate) data daily
outputDs := ROLLUP(SORT(inputDs, pickup_date),
  LEFT.pickup_date=RIGHT.pickup_date,
  TRANSFORM(Layout,
    SELF.pickup_date :=
    LEFT.pickup_date,
    SELF.fare := LEFT.fare +
    RIGHT.fare,
    SELF.distance := LEFT.distance
    + RIGHT.distance,
    SELF.mileageDeduction :=
    self.distance * 0.545));

OUTPUT(outputDs);
```

## Shape with Normalize

Break contents of record into normal form.

```
IMPORT Std;

InputLayout := RECORD
  UNSIGNED ride_id;
  STRING passenger_state;
END;

inputDs := DATASET([{'1, 'group cool talkative'},
  {'2, 'calm quiet'},
  {'3, 'temper nasty'},
  {'4, 'drunk smell'}], InputLayout);

OutputLayout := RECORD
  UNSIGNED ride_id;
  STRING100 word;
END;

wordDs := NORMALIZE(inputDs,

STD.Str.WordCount(LEFT.passenger_state),
  TRANSFORM(OutputLayout,
    SELF.ride_id :=
    LEFT.ride_id,
    SELF.word :=
    STD.Str.ToUpperCase(
  STD.Str.GetNthWord(LEFT.passenger_state,
  COUNTER))));

OUTPUT(wordDs);
```

INPUT		OUTPUT	
ride_id	passenger_state	ride_id	word
1	group cool talkative	1	GROUP
2	calm quiet	1	COOL
3	temper nasty	1	TALKATIVE
4	drunk smell	2	CALM
		2	QUIET
		3	TEMPER
		3	NASTY
		4	DRUNK
		4	SMELL

OUTPUT: SHAPING WITH ROLLUP

pickup_date	fare	distance	mileagededuction
2019-01-01	65.25	13	7.09
2019-01-02	55.25	20	5.45

OUTPUT: SHAPING WITH PARENT CHILD ROLLUP

pickup_date	trips		
	pickup_date	fare	distance
2019-01-01	2019-01-01	25.1	5
	2019-01-01	40.15	8
2019-01-02	2019-01-02	30.1	6
	2019-01-02	25.15	4

## Shape Parent Child Rollup

Rollup records into a parent child layout.

```
InputLayout := RECORD
  STRING10 pickup_date;
  DECIMAL8_2 fare;
  DECIMAL8_2 distance;
END;

OutputLayout := RECORD
  STRING10 pickup_date;
  DATASET(InputLayout) trips;
END;

inputDs := DATASET([{'2019-01-01', 25.10, 5},
  {'2019-01-01', 40.15, 8},
  {'2019-01-02', 30.10, 6},
  {'2019-01-02', 25.15, 4}],
  InputLayout);

groupDs := GROUP(SORT(inputDs, pickup_date),
  pickup_date);

tempDs := ROLLUP(groupDs, GROUP,
  TRANSFORM(OutputLayout,
    SELF.pickup_date := LEFT.pickup_date,
    SELF.trips := ROWS(LEFT)));

OUTPUT(tempDs);
```

INPUT

pickup_datetime	fare	distance
2019-01-01 10:00:00	25.10	5
2019-01-01 11:00:00	40.15	8
2019-01-02 10:00:00	30.10	6
2019-01-02 10:00:00	25.15	4



## Denormalize

Combine data from two normalized Datasets.

```
WeatherLayout := RECORD
  STRING10 weather_date;
  UNSIGNED hour;
  DECIMAL8_2 rain_quantity;
END;

TripLayout := RECORD
  STRING10 pickup_date;
  DECIMAL8_2 fare;
  DECIMAL8_2 distance;
  DATASET(WeatherLayout) weatherDs ;
END;

tripDs := DATASET(
  [{'2019-01-01', 25.10, 5, []},
  {'2019-01-01', 40.15, 8, []},
  {'2019-01-02', 30.10, 6, []},
  {'2019-01-02', 25.15, 4, []}], TripLayout);

weatherDs := DATASET(
  [{'2019-01-01', 1, 0.5},
  {'2019-01-01', 2, 1},
  {'2019-01-02', 1, 0},
  {'2019-01-02', 2, 0}], WeatherLayout);

outputDs := DENORMALIZE(
  tripDs, weatherDs,
  LEFT.pickup_date=RIGHT.weather_date,
  GROUP,
  TRANSFORM(TripLayout,
    SELF.pickup_date := LEFT.pickup_date,
    SELF.fare := LEFT.fare,
    SELF.distance := LEFT.distance,
    SELF.weatherDs := ROWS(RIGHT)));

OUTPUT(outputDs);
```

INPUT

pickup_date	fare	distance	weather_date	hour	rain_quantity
2019-01-01	25.10	5	2019-01-01	1	0.5
2019-01-01	40.15	8	2019-01-01	2	1
2019-01-02	30.10	6	2019-01-02	1	0
2019-01-02	25.15	4	2019-01-02	2	0

OUTPUT

pickup_date	fare	distance	weatherds	weather_date	hour	rain_quantity
2019-01-01	21.5	5	2019-01-01	1	0.5	
				2	1	
2019-01-01	40.15	8	2019-01-01	1	0.5	
				2	1	
2019-01-02	30.1	6	2019-01-02	1	0	
				2	0	
2019-01-02	25.15	4	2019-01-02	1	0	
				2	0	

## Combine

Used to transform datasets with the same number of records but transformed columns.

```
TripLayout := RECORD
  STRING10 pickup_date;
  DECIMAL8_2 distance;
END;

WeatherLayout := RECORD
  STRING10 weather_date;
  DECIMAL8_2 rain_quantity;
END;

tripDs := DATASET(
  [{'2019-01-01', 11000},
  {'2019-01-02', 12500},
  {'2019-01-03', 11800},
  {'2019-01-04', 13000}], TripLayout);

weatherDs := DATASET(
  [{'2019-01-01', 0.5},
  {'2019-01-02', 1},
  {'2019-01-05', 0},
  {'2019-01-06', 0}], WeatherLayout);
```

INPUT

pickup_date	distance	weather_date	rain_quantity
2019-01-01	11000	2019-01-01	0.5
2019-01-01	12500	2019-01-02	1
2019-01-03	11800	2019-01-05	0
2019-01-04	13000	2019-01-06	0

Try the code at  
<https://play.hpcsystems.com:18010/#/stub/ECL-DL/Playground>  
 and view the results

```
JOIN(tripDs, weatherDs, LEFT.pickup_date=RIGHT.weather_date); //Only those records that exist in both
JOIN(tripDs, weatherDs, LEFT.pickup_date=RIGHT.weather_date, LEFT OUTER); //At Least one record for every record in the left
JOIN(tripDs, weatherDs, LEFT.pickup_date=RIGHT.weather_date, RIGHT OUTER); //At Least one record for every record in the right
JOIN(tripDs, weatherDs, LEFT.pickup_date=RIGHT.weather_date, FULL OUTER); //At Least one record for every record in the left and right
JOIN(tripDs, weatherDs, LEFT.pickup_date=RIGHT.weather_date, LEFT ONLY); //One record for each left record with no match in the right
JOIN(tripDs, weatherDs, LEFT.pickup_date=RIGHT.weather_date, RIGHT ONLY); //One record for each right record with no match in the left
JOIN(tripDs, weatherDs, LEFT.pickup_date=RIGHT.weather_date, FULL ONLY); //One record for each left and right record with no match in the opposite
```