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

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

**Summarize**
Provides a large set of functions to summarize values in a dataset. Can be used in functions with GROUP and TABLE to create Pivots.

```ecl
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);
  covFareDist := COVARIANCE(GROUP, ds.fare, ds.distance);
  correlateFareDist := CORRELATION(GROUP, ds.fare, ds.distance);
END;
crossTabDs := TABLE(ds, crossTabLayout, pickup_date);
OUTPUT(crossTabDs);
```

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

```ecl
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.pickup_date := Std.Date.FromStringToDate(LEFT.pickup_datetime[..10], '%Y-%m-%d'),
  SELF.pickup_time := Std.Date.FromStringToTime(LEFT.pickup_datetime[12..], '%H:%M:%S'),
  SELF.fare := LEFT.fare,
  SELF.distance := LEFT.distance));
OUTPUT(outputDs);
```

**Observe Subset**
Select a subset of rows in a dataset for observation.

```ecl
Input := RECORD
  STRING pickup_date;
  DECIMAL8_2 fare;
  DECIMAL8_2 distance;
END;

output := PROJECT(input, TRANSFORM(Output,
  SELF.fare := AVE(GROUP, ds.fare),
  SELF.distance := MIN(GROUP, ds.distance),
  SELF.velocity := COVARIANCE(GROUP, ds.fare, ds.distance),
  SELF.correlate := CORRELATION(GROUP, ds.fare, ds.distance);
)
OUTPUT(output);
```

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

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

OutputLayout := RECORD
  STRING pickup_datetime;
  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.pickup_datetime := Std.Date.FromStringToDate(LEFT.pickup_datetime[..10], '%Y-%m-%d'),
  'datetime := Std.Date.FromStringToTime(LEFT.pickup_datetime[12..], '%H:%M:%S'),
  SELF.fare := LEFT.fare,
  SELF.distance := LEFT.distance));
OUTPUT(outputDs);
```
Shape with Rollup
In one way, ROLLUP is used combine related records into a single aggregate record, like an aggregating SQL self join.

InputLayout := RECORD
    STRING8_10 ride_id;
    DECIMAL8_2 fare;
    DECIMAL8_2 distance;
    DECIMAL8_2 mileageDeduction := 0;
    END;

Outputs := ROLLUP(inputsDs, ride_id, aggregate); OutputLayout := RECORD
    STRING8_10 passenger_state;
    END;

InputLayout := RECORD
    STRING8_10 ride_id;
    DECIMAL8_2 fare;
    DECIMAL8_2 distance;
    DECIMAL8_2 mileageDeduction := 0;
    END;

OutputLayout := RECORD
    END;

InputLayout := RECORD
    STRING8_10 passenger_state;
    END;

OutputLayout := RECORD
    END;

Rollup records into a parent child layout.

InputLayout := RECORD
    STRING8_10 ride_id;
    DECIMAL8_2 fare;
    DECIMAL8_2 distance;
    DECIMAL8_2 mileageDeduction := 0;
    END;

OutputLayout := RECORD
    END;

InputLayout := RECORD
    STRING8_10 passenger_state;
    END;

OutputLayout := RECORD
    END;

Shape Parent Child Rollup
Rollup records into a parent child layout.

InputLayout := RECORD
    STRING8_10 ride_id;
    DECIMAL8_2 fare;
    DECIMAL8_2 distance;
    DECIMAL8_2 mileageDeduction := 0;
    END;

OutputLayout := RECORD
    END;

InputLayout := RECORD
    STRING8_10 passenger_state;
    END;

OutputLayout := RECORD
    END;

Shape with Normalize
Break contents of record into normal form.

InputLayout := RECORD
    STRING8_10 ride_id;
    DECIMAL8_2 fare;
    DECIMAL8_2 distance;
    DECIMAL8_2 mileageDeduction := 0;
    END;

OutputLayout := RECORD
    END;

InputLayout := RECORD
    STRING8_10 passenger_state;
    END;

OutputLayout := RECORD
    END;

Denormalize
Combine data from two normalized Datasets.

WeatherLayout := RECORD
    STRING8_10 weather_date;
    DECIMAL8_2 rain_quantity;
    END;

TripLayout := RECORD
    STRING8_10 pickup_date;
    DECIMAL8_2 fare;
    DECIMAL8_2 distance;
    DECIMAL8_2 mileageDeduction := 0;
    END;

Trips := DATASET([{'2019-01-01', 25.10, 5, 
    {'2019-01-01', 0.5}, 
    {'2019-01-01', 0.5}]
    Trips := DATASET([{'2019-01-01', 25.10, 5, 
    {'2019-01-01', 0.5}, 
    {'2019-01-01', 0.5}]
    OUTPUT: Contents depend on JOIN Type and Condition

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

WeatherLayout := RECORD
    STRING8_10 weather_date;
    DECIMAL8_2 rain_quantity;
    END;

TripLayout := RECORD
    STRING8_10 pickup_date;
    DECIMAL8_2 fare;
    DECIMAL8_2 distance;
    DECIMAL8_2 mileageDeduction := 0;
    END;

Trips := DATASET([{'2019-01-01', 25.10, 5, 
    {'2019-01-01', 0.5}, 
    {'2019-01-01', 0.5}]
    Output with no match in the opposite

WHERE

Try these examples at https://play.hpccsystems.com:18010/#/stub/ECL-DL/Playground

and view the results