

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 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

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

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
2019-01-01	25.10	5
2019-01-01	40.15	8
2019-01-02	30.10	6
2019-01-02	25.15	4

+

weather_date	hour	rain_quantity
2019-01-01	1	50.5
2019-01-01	2	1
2019-01-02	1	0
2019-01-02	2	0

↓

OUTPUT						
pickup_date	fare	distance	weathersds	weather_date	hour	rain_quantity
2019-01-01	21.5	5	2019-01-01	1	0.5	
			2019-01-01	2	1	
2019-01-01	40.15	8	2019-01-01	1	0.5	
			2019-01-01	2	1	
2019-01-02	30.1	6	2019-01-02	1	0	
			2019-01-02	2	0	
2019-01-02	25.15	4	2019-01-02	1	0	
			2019-01-02	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		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.hpccsystems.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
```