

# CHAPTER 14

## EPL Reference: Data Windows

Giulian Guillaume - Ramzy Elwan - **Lucas Beauvais**

# SUMMARY

1. Introduction to the data windows
2. Illustrative example in which data windows can be useful
3. Explanation of this example
4. Data Windows and Special Derived-Value Windows
5. Bibliography

# Introduction

Here, we are talking about data windows in Event Processing Language, this type of data helps us to understand how the data works depending on specific temporal windows. It makes the analysis of the data in a timeline easier

Data windows retain incoming events until an expiry policy indicates to release events (tumbling window) but can also work with events one after the other without any restart (sliding window).

Thus data windows are a means of indicating what subset of events to analyze. Two or more data windows can be combined. This allows a set of events retained by one data window to be placed into a union or an intersection with the set of events retained by one or more other data windows

# Illustrative example

We are responsible of a digital streaming platform and we want our subscribers to have the best experience possible by giving them the most accurate recommendations depending on what they usually watch.

Therefore we are going to use an EPL Windows data to know how to give good recommendations thanks to the videos subscribers are watching in a specific period of time

We can compare this way of thinking with how YouTube works:

Clicks: YouTube assumes that if you click on a video, you find it satisfying.

Watch time: To refine its system, the platform also relies on this criterion, which allows it to more effectively filter its recommendations among a series of videos on the same topic (highlights vs tennis match analysis, for example).

Survey responses: "Valued watch time", or the time spent watching a video that truly interests you, is measured based on a star rating (from 1 to 5) from surveys conducted with users.

Shares and likes: A shared or liked video constitutes a strong signal from a user about their appreciation of the video they just watched.

# Syntaxe

## EPL Module Text

Enter EPL Here:

```
SELECT video_id, SUM(views) AS total_views
FROM VideoViews
WINDOW TumblingWindow 30 minutes EVERY 5 minutes
GROUP BY video_id;
```

## Beginning Of Time

Provide a timestamp to start at:

## Advance Time and Send Events

Enter sequence of time and events:

```
INSERT INTO VideoViews (view_id, video_id, view_timestamp, user_id) VALUES
(1, 101, '2022-03-01 08:00:00', 1001),
(2, 102, '2022-03-01 08:05:00', 1002),
(3, 101, '2022-03-01 08:10:00', 1003),
(4, 103, '2022-03-01 08:15:00', 1004),
(5, 102, '2022-03-01 08:20:00', 1005),
(6, 101, '2022-03-01 08:25:00', 1006),
(7, 104, '2022-03-01 08:30:00', 1007),
```

- VideoViews contains the views of each video,
- video\_id identify one of the videos watched
- views gives the number of views.
- The request WINDOW TumblingWindow 30 minutes EVERY 5 minutes identify a window of 30 minutes with an update every 5 minutes.
- GROUP BY video\_id regroups datas by video\_id, helping us to get the total number of views for each video in the las 30 minutes.
- SUM(views) AS total\_views calculates the sum of views for each group of video\_id, it gives us the total number of views for each video watched in our specific window.
- This request allows us to follow in a real timelinel the viewing trends of vies on the platform, by updating the statistics every 5 minutes for the last 30 minutes given.

## 14.4. Special Derived-Value Windows

This part explains the use of "Special Derived-Value Windows" to aggregate or derive information from an event stream.

- These windows can be used alone or combined with data windows.
- They act in a similar way to aggregation functions, but can post several derived fields at once, including properties of the last event received.
- They do not retain events.

These windows are useful for aggregating and analyzing data in real time, calculating statistics, regressions, correlations and weighted averages from event streams.

## 14.4.1. Size Derived-Value Window (size) or std:size)

Example: you work for a trading company and you need to monitor the volume of transactions for different financial instruments, such as shares, futures or currencies. You have a database containing data on trades made at different times of the day.

You can use the **size** function to count the number of transactions made for each financial instrument over a given period of time. This would allow you to monitor trading activity and identify moments of peak activity or emerging trends in the market.

For example, suppose you wanted to count the number of equity trades made in the last hour. You could run an SQL query like this

### EPL Module Text

Enter EPL Here:

```
SELECT
    size() AS transaction_count
FROM TransactionData
WHERE instrument = 'stock' AND timestamp >= DATE_SUB(NOW(), INTERVAL 1 HOUR)
```

In this example, TransactionData is the table containing transaction data, and instrument is a column indicating the type of financial instrument. The size() function counts the number of share transactions over the last hour. The results of this query can be used to monitor trading activity and make operational decisions accordingly.

## 14.4.2. Univariate Statistics Derived-Value Window (uni or stat:uni)

Example: you work for a brokerage firm and you need to monitor share price movements for a certain portfolio of shares. You have a database containing data on share prices at different times of the day.

Using the uni function, you can calculate statistics such as the mean and standard deviation of share prices over a specific period of time, for example a trading day. This can help you understand market volatility and make informed decisions for your clients.

For example, you could run an SQL query like this:

### EPL Statements

#### EPL Module Text

Enter EPL Here:

```
SELECT
  uni(price) AS average_price,
  SQRT(uni(POWER(price - uni(price), 2))) AS price_stddev
FROM StockTickEvent
WHERE timestamp >= '2024-03-28 09:00:00' AND timestamp < '2024-03-29 09:00:00';
```

In this example, the query calculates the mean and standard deviation of share prices for a given trading day. These statistics can then be used by financial analysts to assess the risk and performance of the equity portfolio, and to make informed investment decisions.

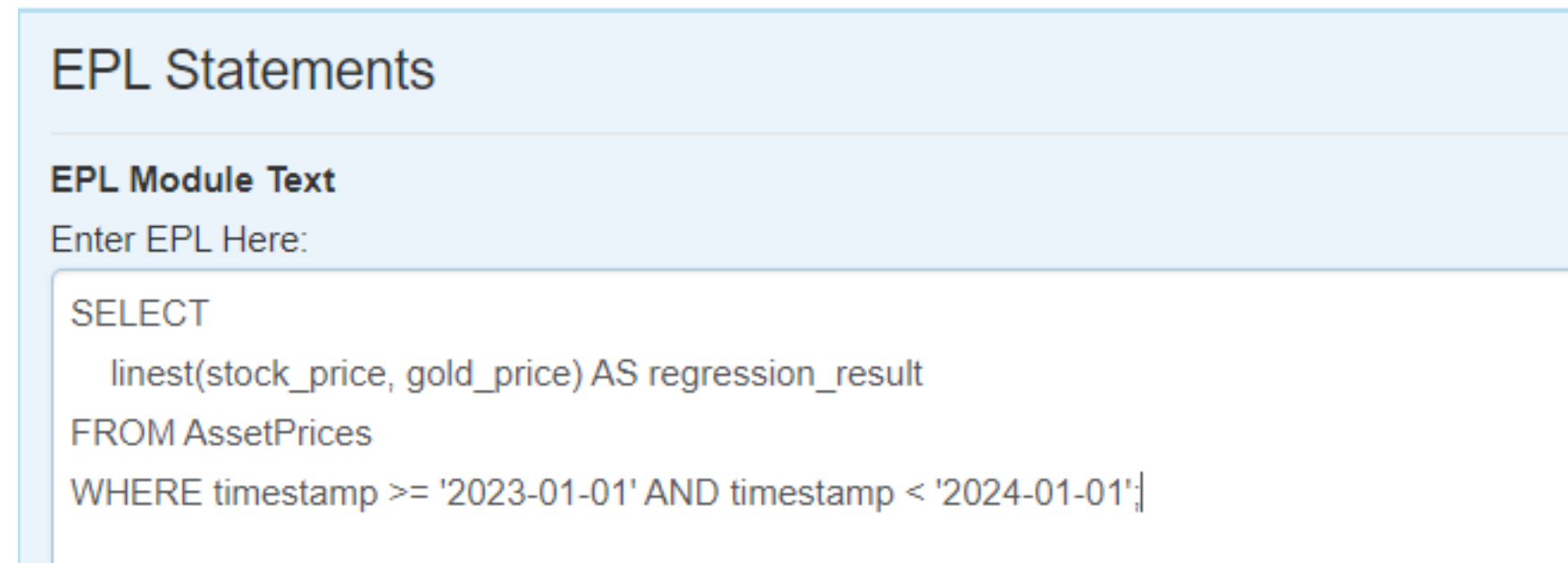


## 14.4.3. Regression Derived-Value Window (linest or stat:linest)

Example: you work for an asset management company and you want to evaluate the relationship between two financial assets, for example the price of a share and the price of a commodity, over a given period of time. You have a database containing data on the prices of these assets at different times.

You can use the linest function to calculate the linear regression between the prices of these two assets. This would allow you to determine whether there is a linear relationship between the two and, if so, calculate parameters such as the slope of the regression line and the intercept with the y-axis.

For example, suppose you wanted to evaluate the relationship between the price of a share and the price of gold over a period of one year. You could run an SQL query like this:



The screenshot shows a web-based interface for entering EPL (Embedded PL/SQL) statements. It has a light blue header with the text 'EPL Statements'. Below this is a section titled 'EPL Module Text' with a prompt 'Enter EPL Here:'. A text area contains the following SQL query:

```
SELECT
    linest(stock_price, gold_price) AS regression_result
FROM AssetPrices
WHERE timestamp >= '2023-01-01' AND timestamp < '2024-01-01';
```

In this example, stock\_price represents the share price and gold\_price represents the gold price. The linest function will calculate the linear regression between these two price series over a period of one year. The results of the regression can then be used to analyse the correlation between the two assets and make informed investment decisions.

## 14.4.4. Correlation Derived-Value Window (correl or stat:correl)

Example: you work for a financial analysis company and you need to measure the correlation between two financial assets, for example a company's share price and the return on a stock market index, over a given period of time. You have a database containing data on share prices and stock index returns at different points in time.

You can use the correl function to calculate the correlation between these two assets. This would allow you to determine whether they are moving together in the same direction, in opposite directions or whether they are independent.

For example, suppose you wanted to assess the correlation between the share price of company XYZ and the performance of the S&P 500 index over a six-month period. You could run an SQL query like this:

### EPL Module Text

Enter EPL Here:

```
SELECT
    correl(stock_price, sp500_return) AS correlation_result
FROM AssetData
WHERE timestamp >= '2023-07-01' AND timestamp < '2024-01-01';
```

In this example, stock\_price represents the share price of company XYZ and sp500\_return represents the return on the S&P 500 index. The correl function will calculate the correlation between these two sets of data over a six-month period. The correlation results can then be used to assess the degree of relationship between the two assets and make investment decisions accordingly.

## 14.4.5. Weighted Average Derived-Value Window (weighted\_avg or stat:weighted\_avg)

Suppose you work for a portfolio management company and you want to calculate the weighted average of stock returns in a stock market index, taking into account the market capitalisation of each company. You have a database containing data on share prices and the market capitalisation of the companies included in the index.

You can use the weighted\_avg function to calculate the weighted average of stock returns, where the weights are given by the market capitalisation of each company. This would give you a representative aggregate measure of the index's performance.

For example, suppose you wanted to calculate the weighted average of stock returns in the S&P 500 index for last month. You could run an SQL query like this:

### EPL Module Text

Enter EPL Here:

```
SELECT
    weighted_avg(return, market_cap) AS weighted_avg_return
FROM StockData
WHERE timestamp >= '2024-02-01' AND timestamp < '2024-03-01';
```

In this example, return represents each company's share return and market\_cap represents each company's market capitalisation. The weighted\_avg function will calculate the weighted average of the share returns for last month, using the market capitalisation as a weight. The results of this weighted average can then be used to evaluate the performance of the S&P 500 index over the given period.

## BIBLIOGRAPHY

<https://www.youtube.com/watch?v=PNgH0n0qk7w>

<https://www.blogdumoderateur.com/youtube-systeme-recommandation-videos/>

<https://blog.youtube/inside-youtube/on-youtubes-recommendation-system/>

<https://youtube.com/watch?v=zAmJPdZu8Rg>

[https://esper.espertech.com/release-8.9.0/reference-esper/html\\_single/index.html#stat-views](https://esper.espertech.com/release-8.9.0/reference-esper/html_single/index.html#stat-views)