Analytics on Streaming Data with Azure Stream Analytics

datasciencedojo
unleash the data scientist in you

1 NEW DEFINITION IS ADDED ON urban
1,600+ READS ON Scribd.
13,000+ HOURS MUSIC STREAMING ON PANDORA
12,000+ NEW ADS POSTED ON craigslist
370,000+ MINUTES VOICE CALLS ON skype
98,000+ TWEETS
20,000+ NEW POSTS ON tumblr.
THE LARGEST SOCIAL READING PUBLISHING COMPANY!!
New Craigslist Ads
320+ NEW twitter ACCOUNTS
100+ NEW Linked in ACCOUNTS
13,000+ iPhone APPLICATIONS DOWNLOADED
IN 60 SECONDS...
1 associatedcontent NEW ARTICLE IS PUBLISHED
THE WORLD'S LARGEST COMMUNITY CREATED CONTENT!!
QUESTIONS ASKED ON THE INTERNET...
100+ Answers.com
40+ YAHOO! ANSWERS
6,600+ NEW PICTURES ARE UPLOADED ON flickr
600+ NEW VIDEOS
50+ WordPress DOWNLOADS
25+ HOURS TOTAL DURATION
70+ DOMAINS REGISTERED
695,000+ facebook STATUS UPDATES
125+ PLUGIN DOWNLOADS
60+ NEW BLOGS
168 MILLION EMAILS ARE SENT
694,445 SEARCH QUERIES
1,700+ Firefox DOWNLOADS
1,500+ BLOG POSTS
79,364 WALL POSTS
510,040 COMMENTS
Google
Google Search
GO-Globe.com

Introducing Big Data

Continued

Real Time Analytics

Within seconds…

or…

Within minutes…

of an event occurring.

Up to 2 hours.

datasciencedojo
unleash the data scientist in you

Timeliness of Information

What was trending in the past 5 minutes?

Your high school friend is also in Vegas RIGHT NOW.

A tornado will form in the next 30 minutes.

datasciencedojo
unleash the data scientist in you

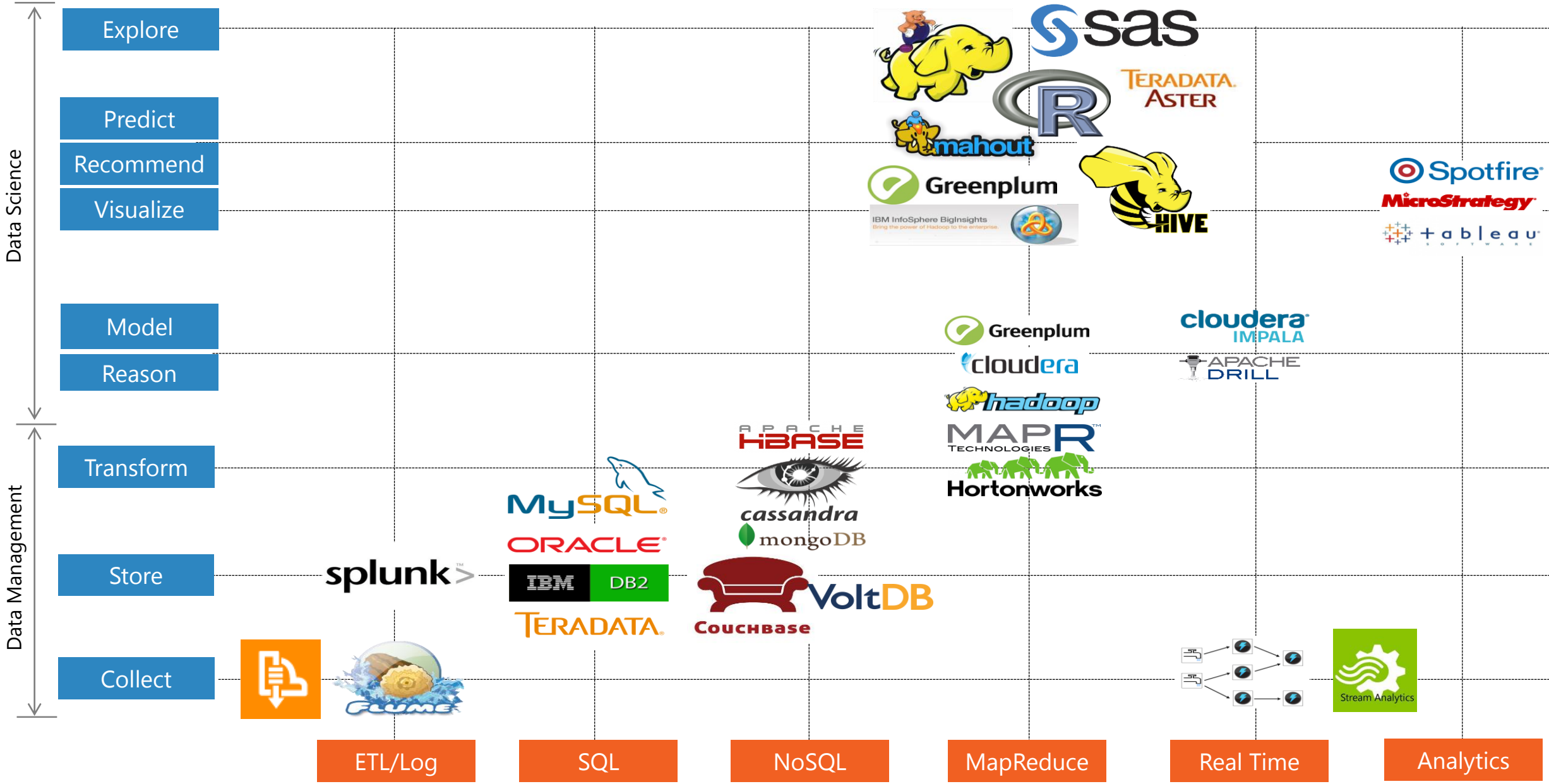
Timeliness of Information

A stock is going to crash in 20 minutes.

A fire is about to start in your house.

The power grid will overload in 2 minutes.

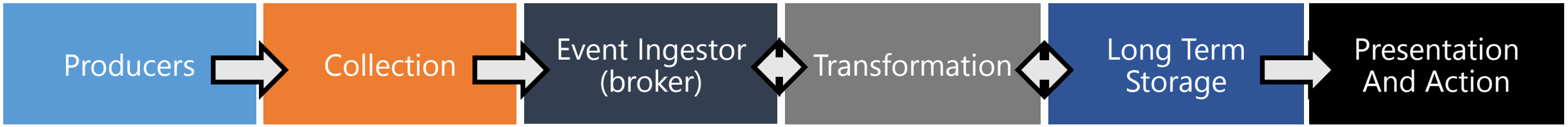
datasciencedojo
unleash the data scientist in you



Big Data – Technology, Platforms & Products

datasciencedojo
unleash the data scientist in you

Typical Event Processing



Applications

Devices

Cloud Gateways (WebAPIs)

Field Gateways

Scalable Event Broker

Event Hub

Real-Time Analytics

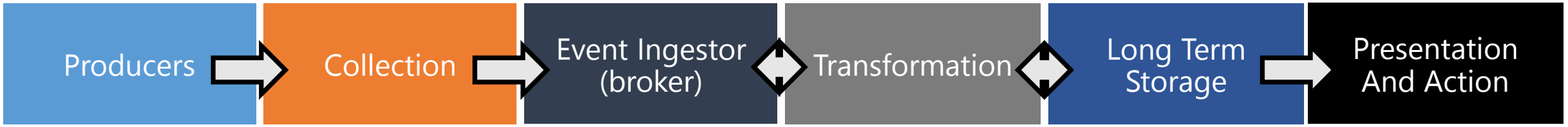
External Data Sources

Web/Thick Client Dashboards

Search And Query

Data Analytics

Typical Event Processing



Applications

Cloud Gateways (WebAPIs)

Scalable Event Broker

Real-time Analytics

External Data Sources

Web/Thick Client Dashboards

Devices

Field Gateways

Search And Query

Stream Analytics

Stream Analytics

Data Analytics

Data at Rest

- **Question** "How many red cars are in the parking lot?"
- **Answering with a relational database**
  Walk out to the parking lot
  Count vehicles that are: Red, Car
- **SELECT COUNT**(*) **FROM** ParkingLot
  WHERE type **=** 'Auto'
  AND color **=** 'Red'

datasciencedojo
unleash the data scientist in you

Data in Motion

- **Different Question** "How many red cars have passed exit 18A on A-10 in the last hour?"
- **Answering with a relational database**
  Pull over, park all vehicles in a lot, keep them there for an hour
  Count vehicles in the lot
- **Not a great solution…**

datasciencedojo
unleash the data scientist in you

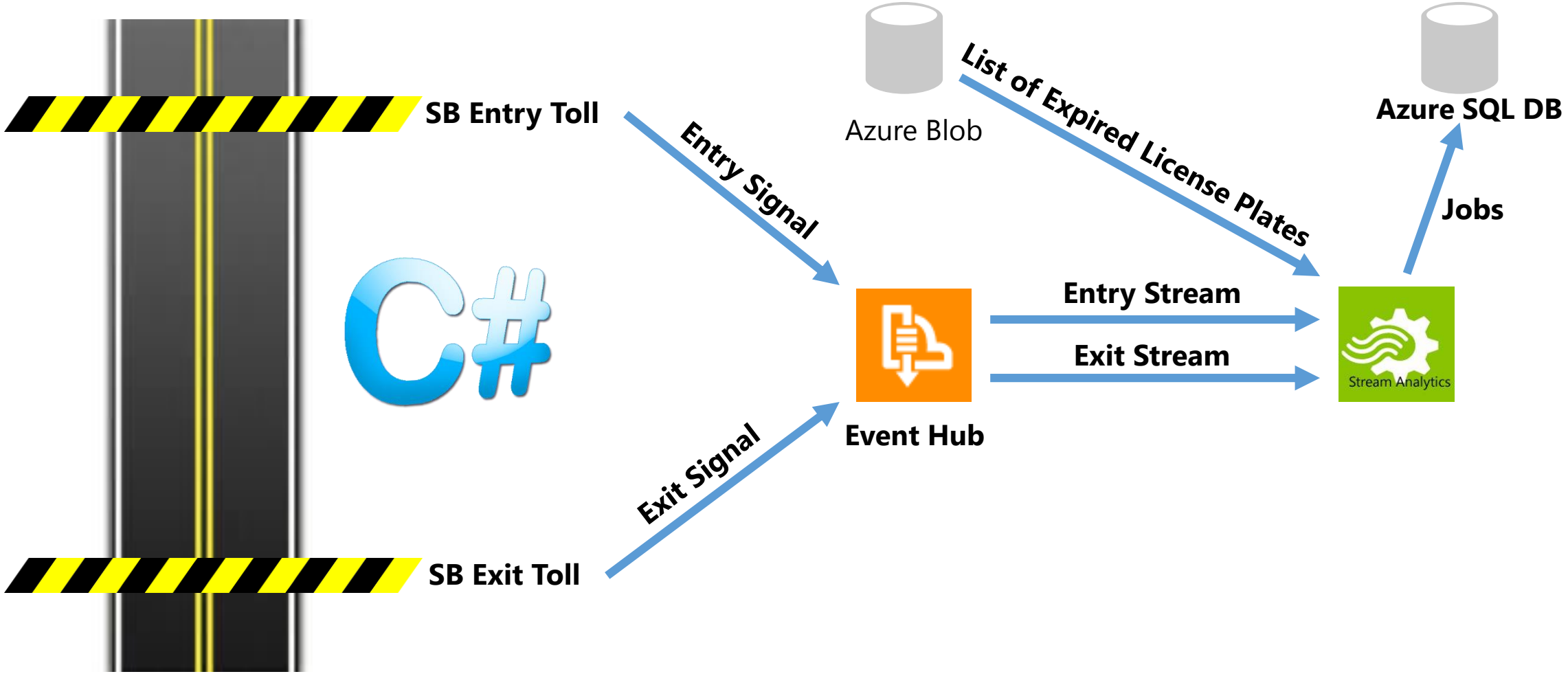
Demo

datasciencedojo
unleash the data scientist in you

Tolls on I-405

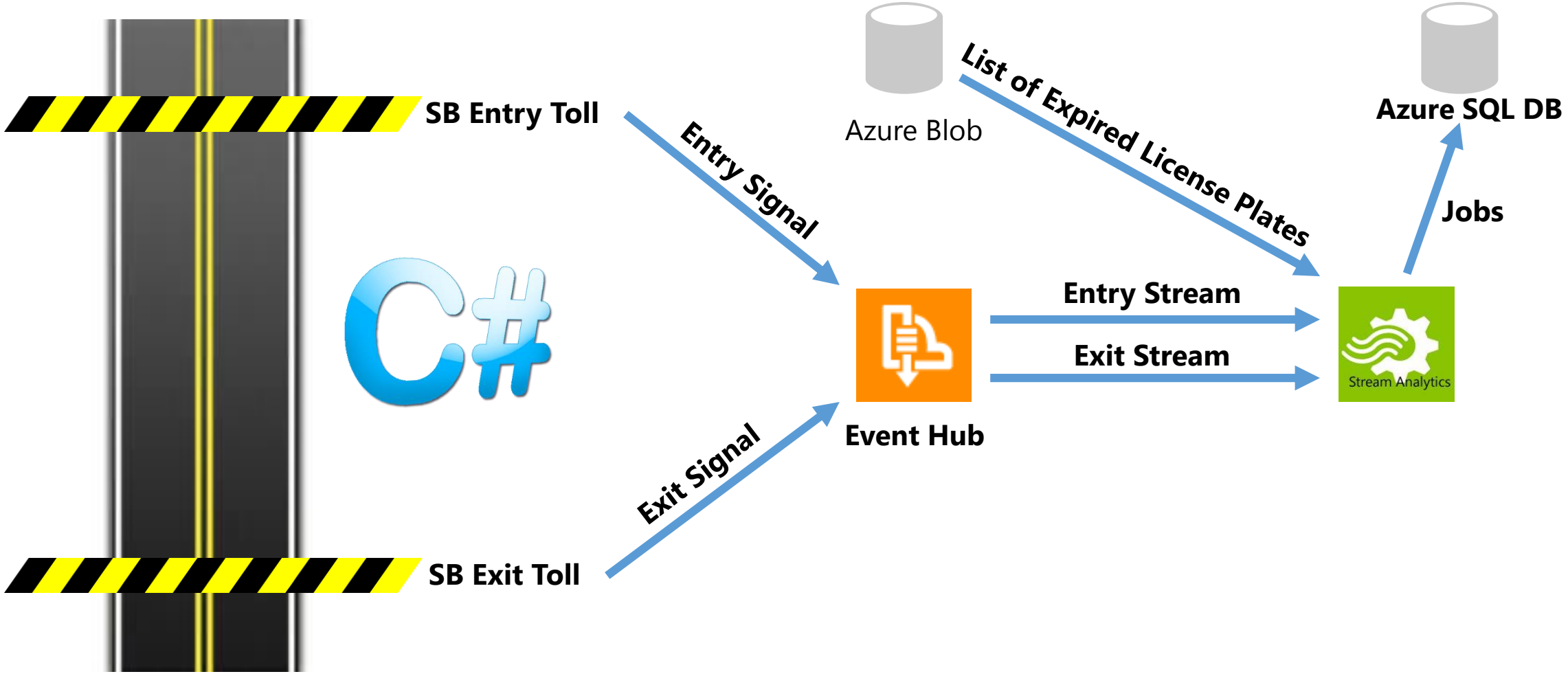
datasciencedojo
unleash the data scientist in you

Automated Tolls



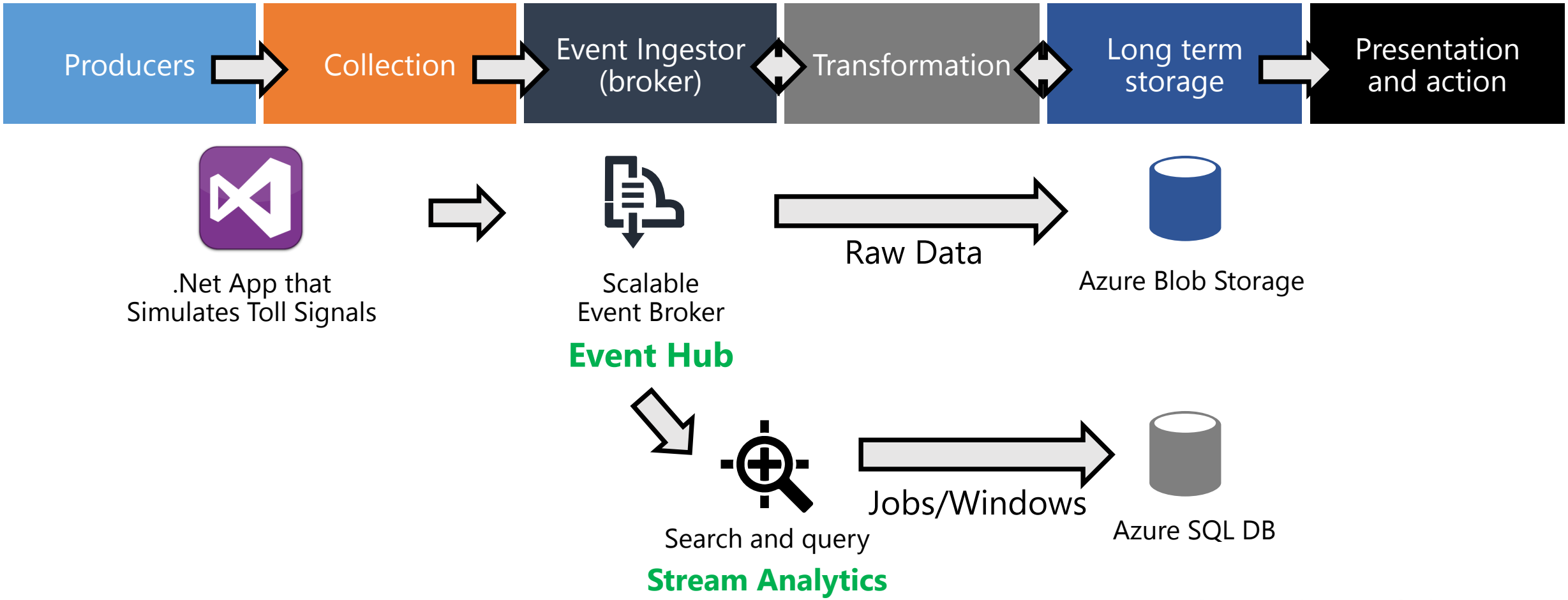
datasciencedojo
unleash the data scientist in you

Automated Tolls



datasciencedojo
unleash the data scientist in you

Tolls Work Process



datasciencedojo
unleash the data scientist in you

Azure Stream Query Language

- Simple SQL dialect
  - Familiar – learning curve reduction
  - High-Level – expression of intent, not implementation
  - Maintainable – focus on the essentials of the problem
- Extended in natural ways to express temporal concepts
  - WINDOW – multiple kinds
    - Tumbling, hopping, sliding
  - TIMESTAMP BY, BETWEEN
  - DATEDIFF in joins
  - PARTITION BY for scale-out

```
WITH agg AS
(
    SELECT Avg(reading), Building
    FROM Temperature
    GROUP BY TumblingWindow(minute, 1), building
)
SELECT A1.Avg AS Old, A2.Avg AS New, A1.Building
FROM Agg A1 JOIN Agg A2
ON A1.Building = A2.Building
AND DATEDIFF(minute,A1,A2) BETWEEN 4.5 AND 5.5
WHERE
   (a1.avg < a2.avg - 10) OR (a1.avg > a2.avg+10)
```

datasciencedojo
unleash the data scientist in you

Temporal System

- Every event is a point in time, and thus must come with a timestamp
  - Remember how relational DBs need a PK? Temporal systems need a timestamp.
- Stream Analytics can append your events with a timestamp (bad practice if standalone)
  - Can be skewed by network and hardware latency
- Users can define application time stamps with the TIMESTAMP BY clause
- Aggregations have timestamps at the end of the window

datasciencedojo
unleash the data scientist in you

Specifications

- Analyze millions of events per SECOND
- Fault tolerant
- SQL spoken here
- Fully managed service by Azure

datasciencedojo
unleash the data scientist in you

Built-In Functions And Supported Types

Aggregate functions

Count, Min, Max, Avg, Sum

Scalar functions

Cast

Date and time

Datename, Datepart, Day, Month, Year, Datediff, Dateadd

String

Len, Concat, Charindex, Substring, Patindex

datasciencedojo
unleash the data scientist in you

Traditional SQL

How many vehicles passed through each toll booth yesterday?

- Why can't we ask how many cars have gone through so far today?

```
SELECT TollID, Count(*) AS Count
FROM EntryStream
WHERE date = 'yesterday'
GROUP BY TollID
```

datasciencedojo
unleash the data scientist in you

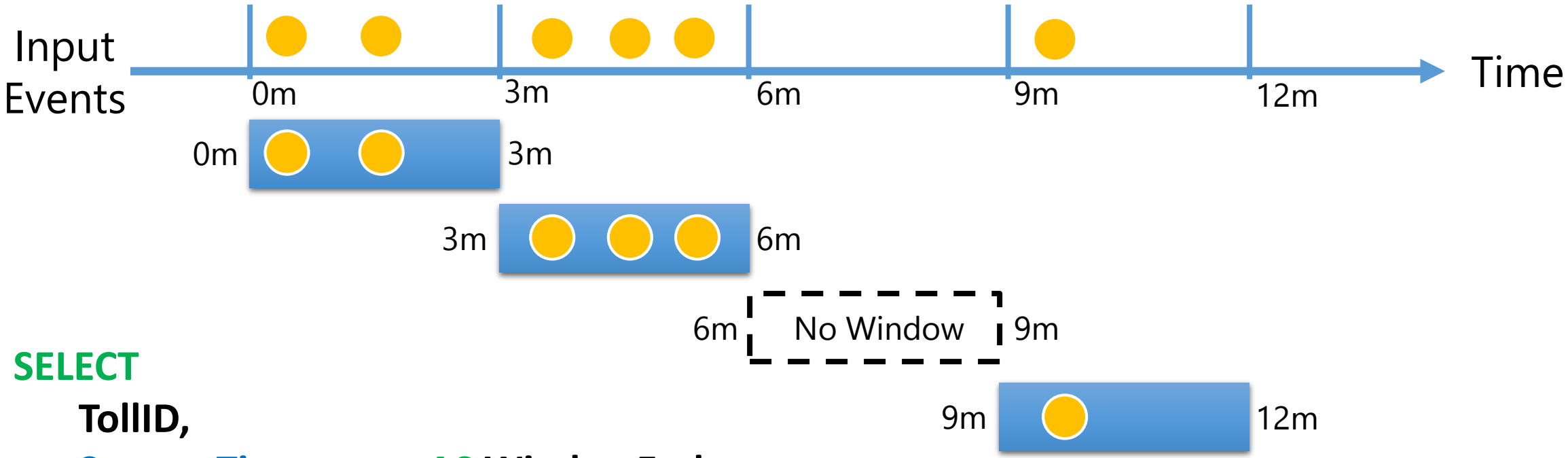
Azure Stream Query Language

How many vehicles pass through each toll booth every 3 minutes?

```
SELECT TollID, System.Timestamp AS WindowEnd, Count(*) AS Count
FROM EntryStream TIMESTAMP BY EntryTime
GROUP BY TUMBLINGWINDOW(minute, 3), TollID
```

datasciencedojo
unleash the data scientist in you

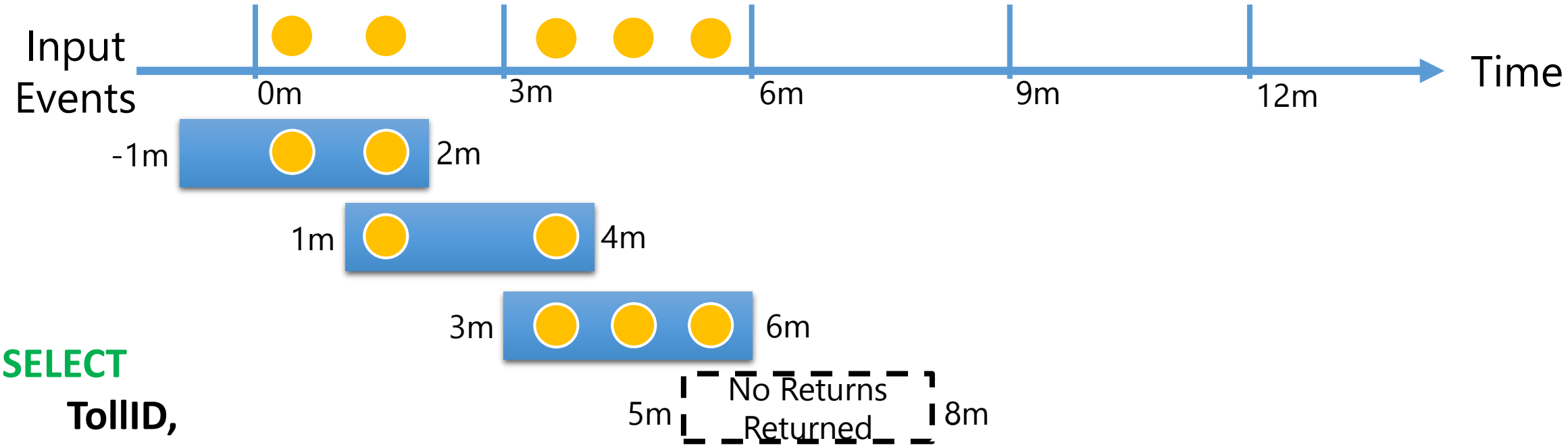
Tumbling Window




```
SELECT
    TollID,
    System.Timestamp AS WindowEnd,
    Count(*) AS Count
FROM EntryStream TIMESTAMP BY EntryTime
GROUP BY TUMBLINGWINDOW(minute, 3), TollID
```

datasciencedojo
unleash the data scientist in you

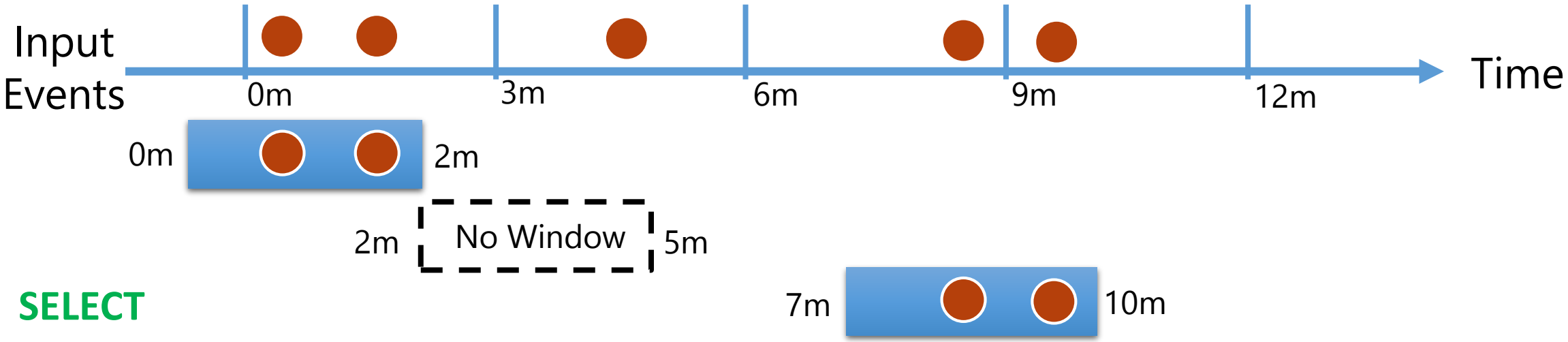
Hopping Window




```
SELECT
    TollID,
    System.Timestamp AS WindowEnd,
    Count(*) AS Count
FROM EntryStream TIMESTAMP BY EntryTime
GROUP BY HOPPINGWINDOW(minute, 3, 2), TollID
```

datasciencedojo
unleash the data scientist in you

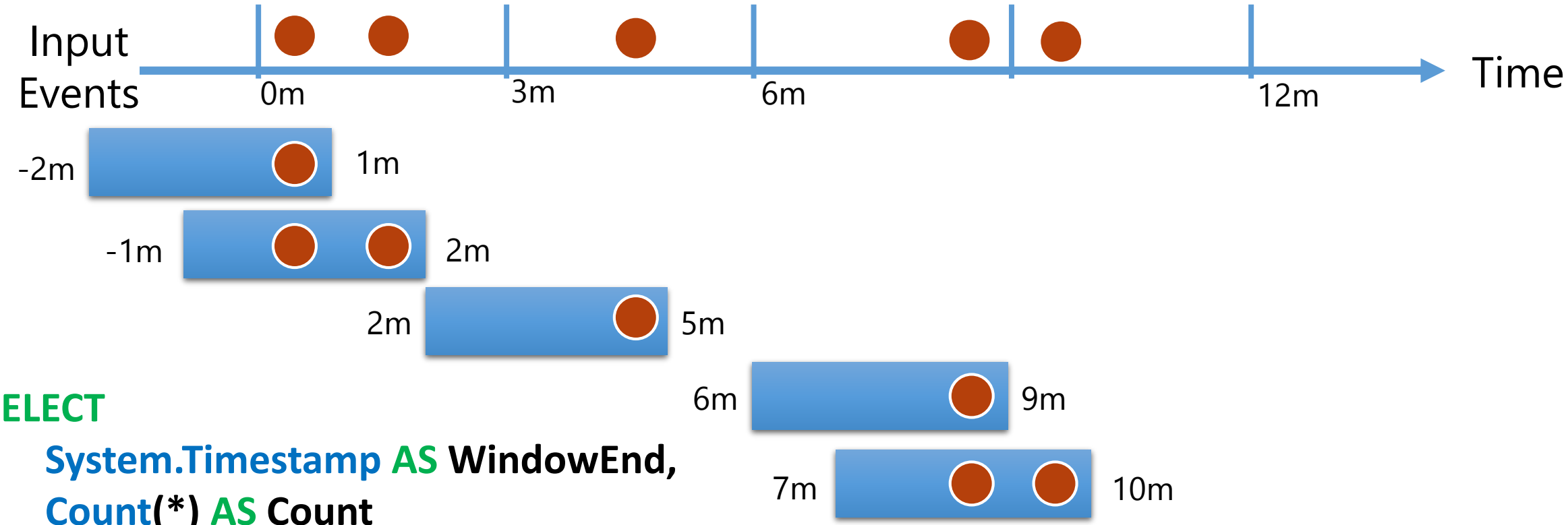
Sliding Window




```
SELECT
    System.Timestamp AS WindowEnd,
    Count(*) AS Count
FROM EntryStream TIMESTAMP BY EntryTime
GROUP BY SLIDINGWINDOW(minute, 3)
HAVING CarCount > 2
```

datasciencedojo
unleash the data scientist in you

Sliding Window: Without 'Having' Clause




```
SELECT
    System.Timestamp AS WindowEnd,
    Count(*) AS Count
FROM EntryStream TIMESTAMP BY EntryTime
GROUP BY SLIDINGWINDOW(minute, 3)
```

datasciencedojo
unleash the data scientist in you

Sum Aggregation

How much toll revenue is being accumulated every 3 minutes?

```
SELECT
    System.Timestamp AS WindowEnd,
    Sum(TollAmount) AS IntervalRevenue
FROM EntryStream TIMESTAMP BY EntryTime
GROUP BY TUMBLINGWINDOW(minute, 3), WindowEnd
```

datasciencedojo
unleash the data scientist in you

Sum Aggregation: With Filtering

Which 3-minute time interval made more than $10?

```
SELECT
    System.Timestamp AS WindowEnd,
    Sum(TollAmount) AS IntervalRevenue
FROM EntryStream TIMESTAMP BY EntryTime
GROUP BY TUMBLINGWINDOW(minute, 3), WindowEnd
Having IntervalRevenue > 10
```

datasciencedojo
unleash the data scientist in you

Descriptive Statistics

Generate descriptive statistics for toll booth 2 every 3 minutes (car count, min, max, average, standard deviation, and total revenue).

```
SELECT
    System.Timestamp AS WindowEnd,
    count(TollAmount) AS CarCount,
    min(TollAmount) AS MinRev,
    max(TollAmount) AS MaxRev,
    avg(TollAmount) AS AvgRev,
    stdev(TollAmount) AS VarRev,
    sum(TollAmount) AS TotalRev
FROM EntryStream TIMESTAMP BY EntryTime
GROUP BY TUMBLINGWINDOW(minute, 3), WindowEnd
```

datasciencedojo
unleash the data scientist in you

DateDiff and Time

What is the duration between the first car in the window and the last car in the window? What was the duration between the first car in the window and the end of the window?

```
SELECT
    System.Timestamp AS WindowEnd,
    count(*) AS CarCount,
    datediff(second, min(EntryTime), max(EntryTime)) AS FirstLastDuration,
    datediff(second, min(EntryTime), System.Timestamp) AS FirstEndDuration
FROM EntryStream TIMESTAMP BY EntryTime
WHERE TollId = 2
GROUP BY TUMBLINGWINDOW(minute, 3), WindowEnd
HAVING count(*) >= 2
```

datasciencedojo
unleash the data scientist in you

Join

How long did it take for each car to pass through the toll zone?

- JOIN operator requires specifying a temporal wiggle room describing an acceptable time difference between the joined events
- Use DATEDIFF function to specify that events should be no more than 15 minutes from each other

datasciencedojo
unleash the data scientist in you

Joining Stream with Reference Data

Who has expired license plates? Let's issue them a citation.

```
SELECT
   EntryStream.EntryTime,
   EntryStream.LicensePlate,
   EntryStream.TollId,
   Registration.RegistrationId
FROM EntryStream TIMESTAMP BY EntryTime
JOIN Registration
ON EntryStream.LicensePlate = Registration.LicensePlate
WHERE Registration.Expired = '1'
```

datasciencedojo
unleash the data scientist in you

Joining Streams

How long did it take for each car to pass through the toll zone? (in seconds)

```
SELECT
    en.TollId,
    en.LicensePlate,
    en.EntryTime, ex.ExitTime,
    DATEDIFF ( second, en.EntryTime, ex.ExitTime ) AS DurationInMinutes
FROM EntryStream AS en TIMESTAMP BY EntryTime
JOIN ExitStream AS ex TIMESTAMP BY ExitTime
ON (en.LicensePlate = ex.LicensePlate)
     AND DATEDIFF ( minute, en, ex ) BETWEEN 0 AND 15
```

datasciencedojo
unleash the data scientist in you

Joining Streams, by Window

What was the average time that it took for cars to go through the toll zone, every 3 minutes? (in seconds)

```
SELECT
    en.TollId,
    en.LicensePlate,
    avg( DATEDIFF ( second, en.EntryTime, ex.ExitTime )) AS DurationInMinutes
FROM EntryStream AS en TIMESTAMP BY EntryTime
JOIN ExitStream AS ex TIMESTAMP BY ExitTime
ON (en.LicensePlate = ex.LicensePlate)
     AND DATEDIFF ( minute, en, ex ) BETWEEN 0 AND 15
Group by TumblingWindow( minute, 3), en.TollId, en.LicensePlate
```

datasciencedojo
unleash the data scientist in you

DATEDIFF, integer only

How long (in HOURS) does it take for each car to pass through the toll zone?

- Known bug right now: Decimal floats cut off, returns only 0

(Broken at the moment)

```
SELECT
   en.TollId, en.LicensePlate, en.EntryTime, ex.ExitTime,
   DATEDIFF ( hour, en.EntryTime, ex.ExitTime ) AS DurationHours
FROM EntryStream AS en TIMESTAMP BY EntryTime
JOIN ExitStream AS ex TIMESTAMP BY ExitTime
ON (en.LicensePlate = ex.LicensePlate)
AND DATEDIFF ( hour, en, ex ) BETWEEN 0 AND 1
```

datasciencedojo
unleash the data scientist in you

Calculations

How fast (mph) was each car traveling through the toll zone?

Assume the toll zone was 1.5 miles long.

(Broken at the moment)

```
SELECT
    en.TollId, en.LicensePlate, en.EntryTime, ex.ExitTime,
    1.5 / DATEDIFF ( hour, en.EntryTime, ex.ExitTime ) AS MPH
FROM EntryStream AS en TIMESTAMP BY EntryTime
JOIN ExitStream AS ex TIMESTAMP BY ExitTime
ON (en.LicensePlate = ex.LicensePlate)
AND DATEDIFF ( hour, en, ex ) BETWEEN 0 AND 1
```

datasciencedojo
unleash the data scientist in you

Caching On Having Only

Who was speeding through the toll zone?

- Simple question… but the query below will break.

```
SELECT
    en.TollId, en.LicensePlate, en.EntryTime, ex.ExitTime,
    1.5 / DATEDIFF ( hour, en.EntryTime, ex.ExitTime ) AS MPH
FROM EntryStream AS en TIMESTAMP BY EntryTime
JOIN ExitStream AS ex TIMESTAMP BY ExitTime
ON (en.LicensePlate = ex.LicensePlate)
AND DATEDIFF ( hour, en, ex ) BETWEEN 0 AND 1
~~WHERE MPH > 62~~
```

datasciencedojo
unleash the data scientist in you

StreamQL Quirks

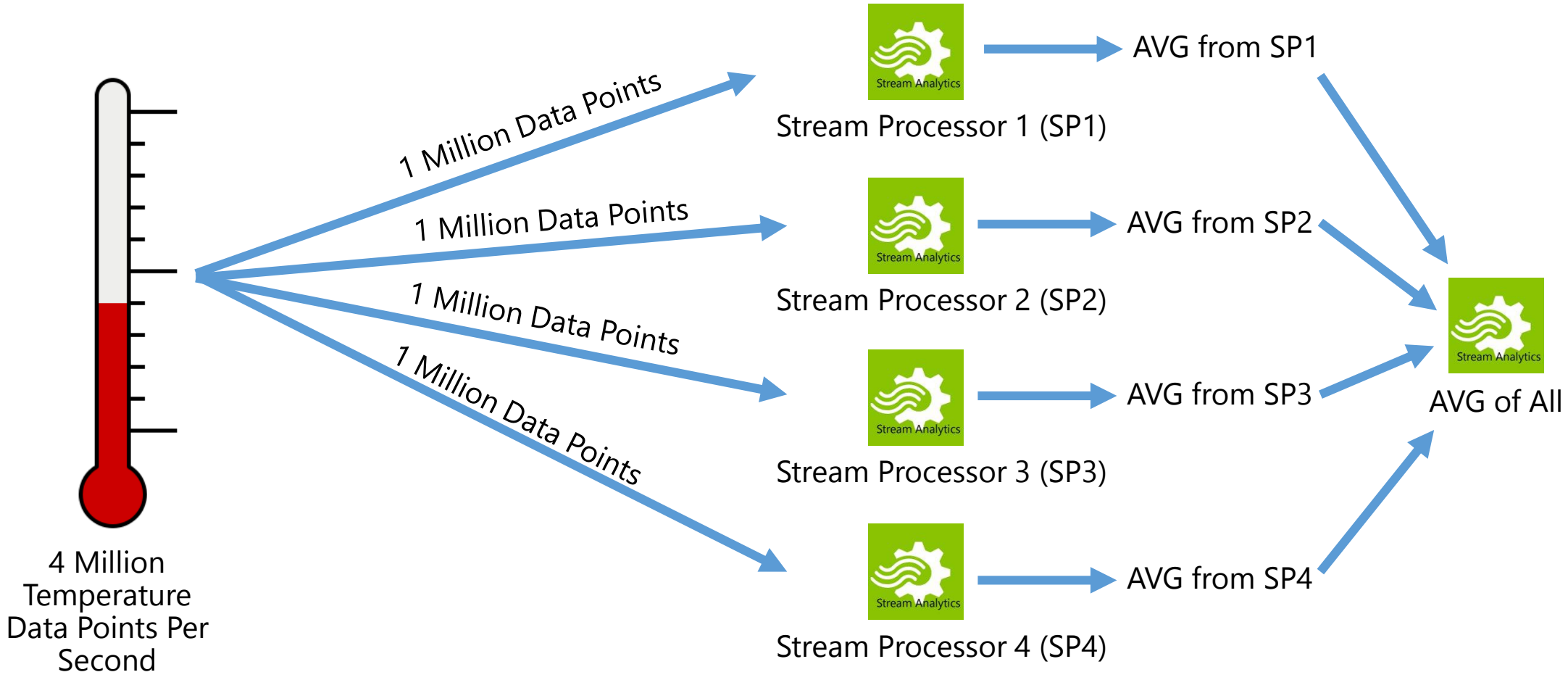
Who was speeding through the toll zone?

No caching -- must rewrite calculations…

```
SELECT
   en.TollId, en.LicensePlate, en.EntryTime, ex.ExitTime,
   1.5 / DATEDIFF ( hour, en.EntryTime, ex.ExitTime ) AS MPH
FROM EntryStream AS en TIMESTAMP BY EntryTime
JOIN ExitStream AS ex TIMESTAMP BY ExitTime
ON (en.LicensePlate = ex.LicensePlate)
AND DATEDIFF ( hour, en, ex ) BETWEEN 0 AND 1
WHERE 1.5 / DATEDIFF ( hour, en.EntryTime, ex.ExitTime ) > 62
```

datasciencedojo
unleash the data scientist in you

Average of Average Approximations



datasciencedojo
unleash the data scientist in you