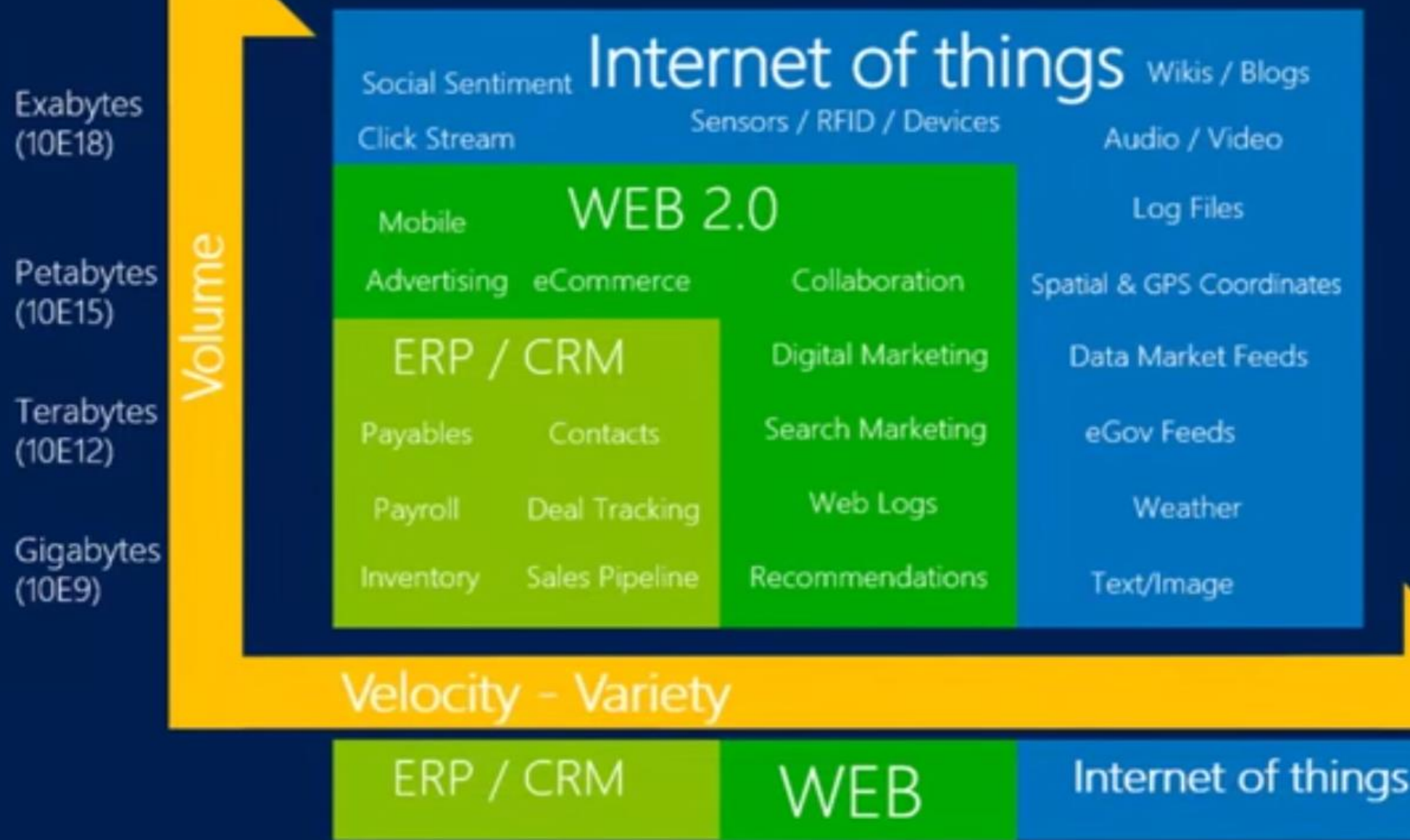


Analytics on Streaming Data with Azure Stream Analytics



Introducing Big Data

Continued



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.

Timeliness of Information

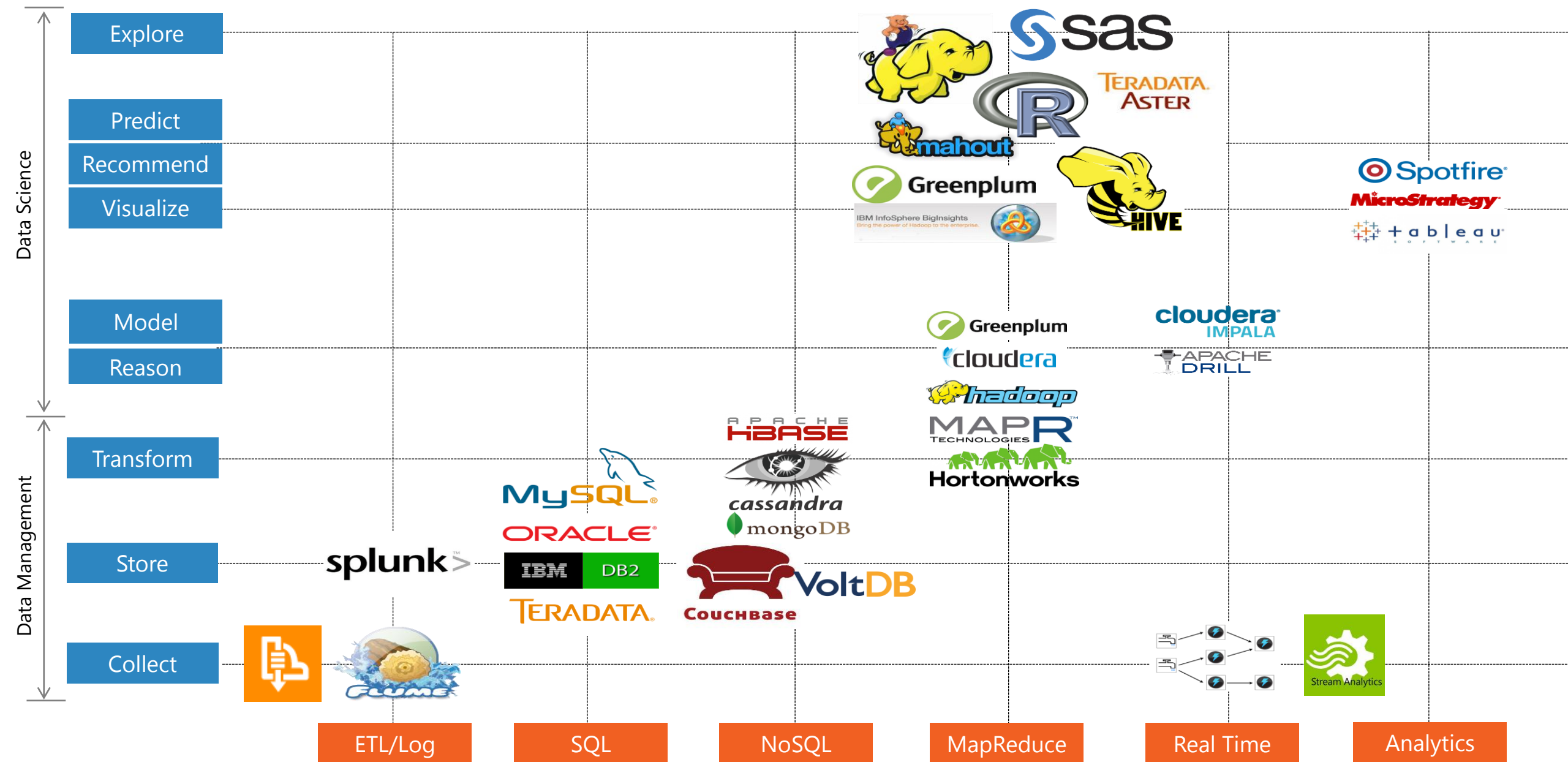


A stock is going to crash in 20 minutes.

A fire is about to start in your house.

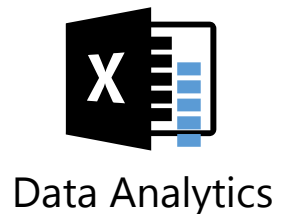
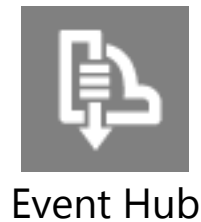
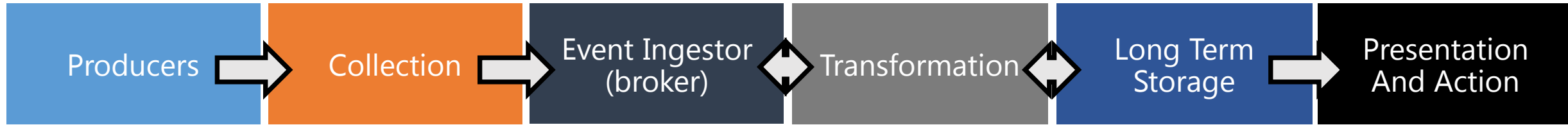


The power grid will overload in 2 minutes.

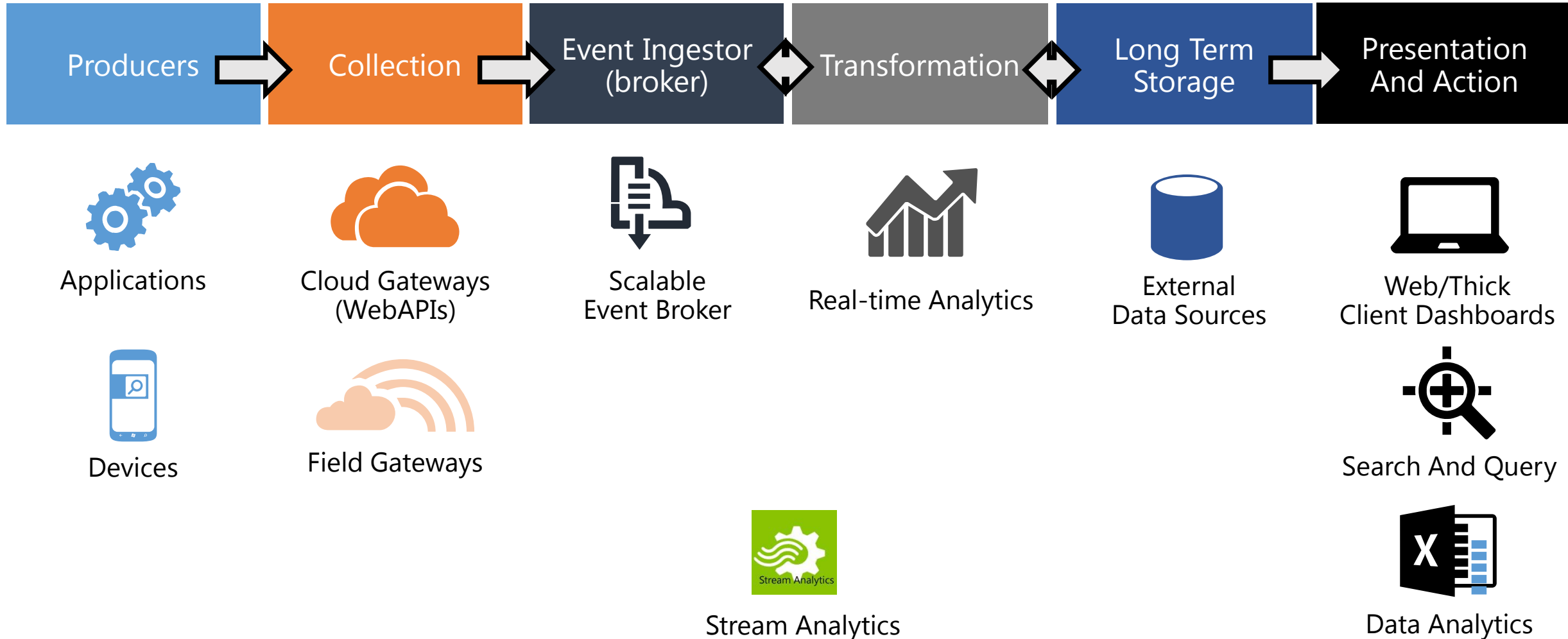


Big Data – Technology, Platforms & Products

Typical Event Processing



Typical Event Processing



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'



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...**

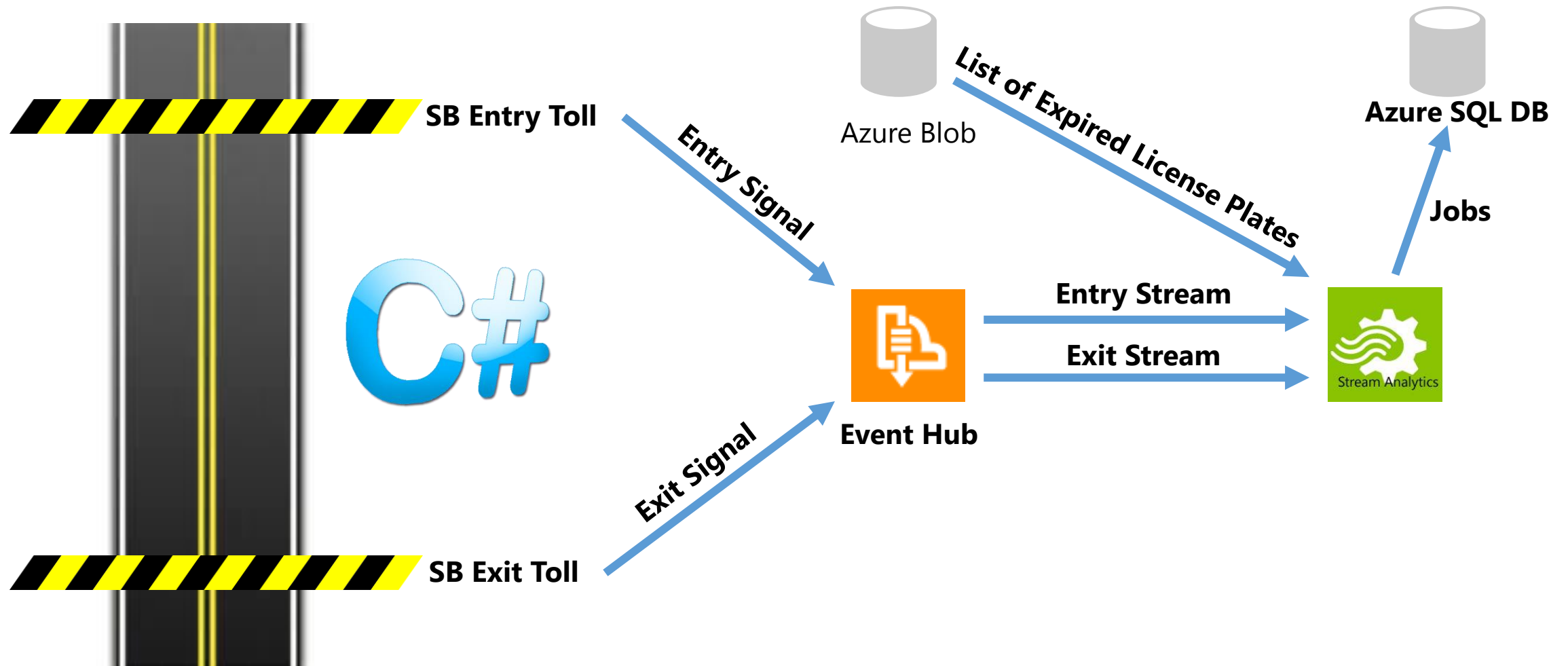


Demo

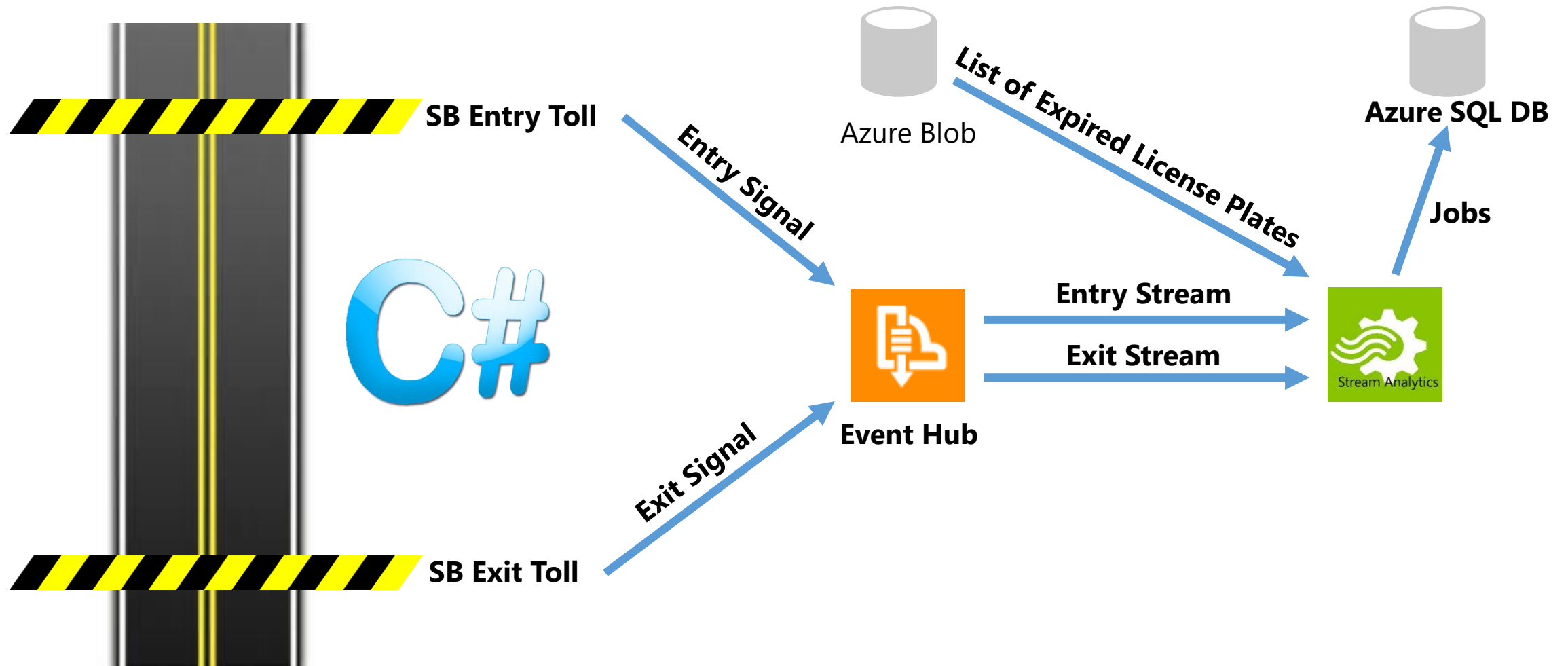
Tolls on I-405



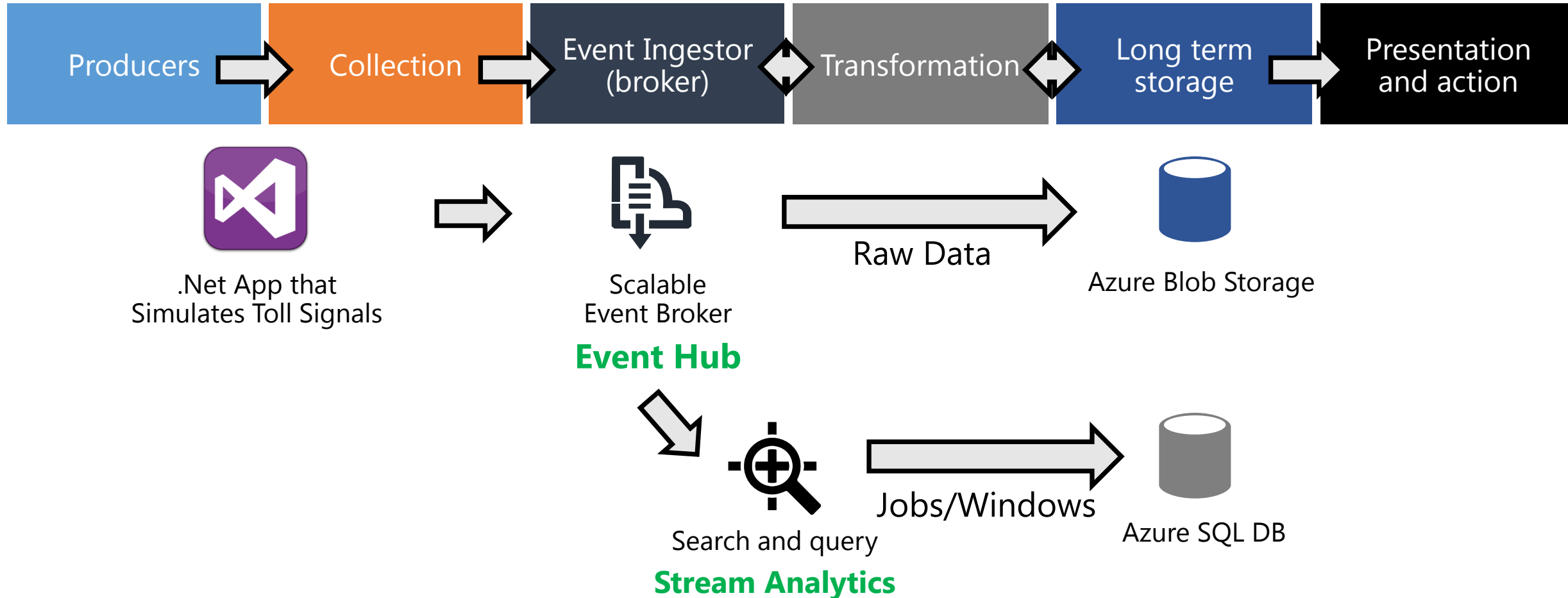
Automated Tolls



Automated Tolls



Tolls Work Process



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)
```


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

Specifications

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

Built-In Functions And Supported Types

Aggregate functions

Count, Min, Max, Avg, Sum

Scalar functions

Cast

Date and time

Datetime, Datepart, Day, Month, Year, Datediff, Dateadd

String

Len, Concat, Charindex, Substring, Patindex

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?

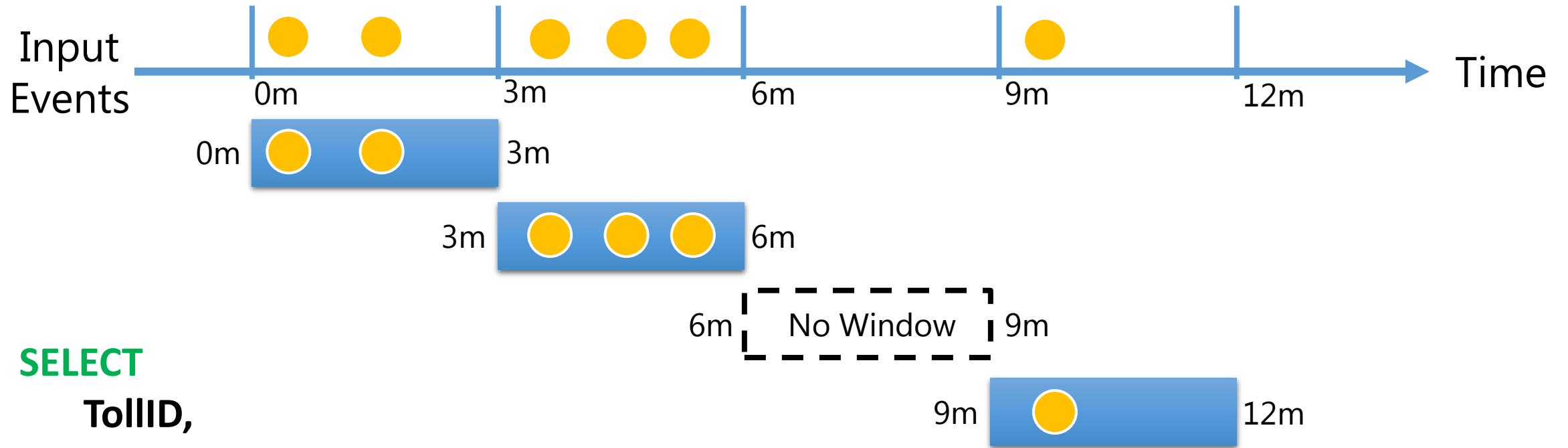
```
SELECT TollID, Count(*) AS Count
FROM EntryStream
WHERE date = 'yesterday'
GROUP BY TollID
```


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
```

Tumbling Window



SELECT

TollID,

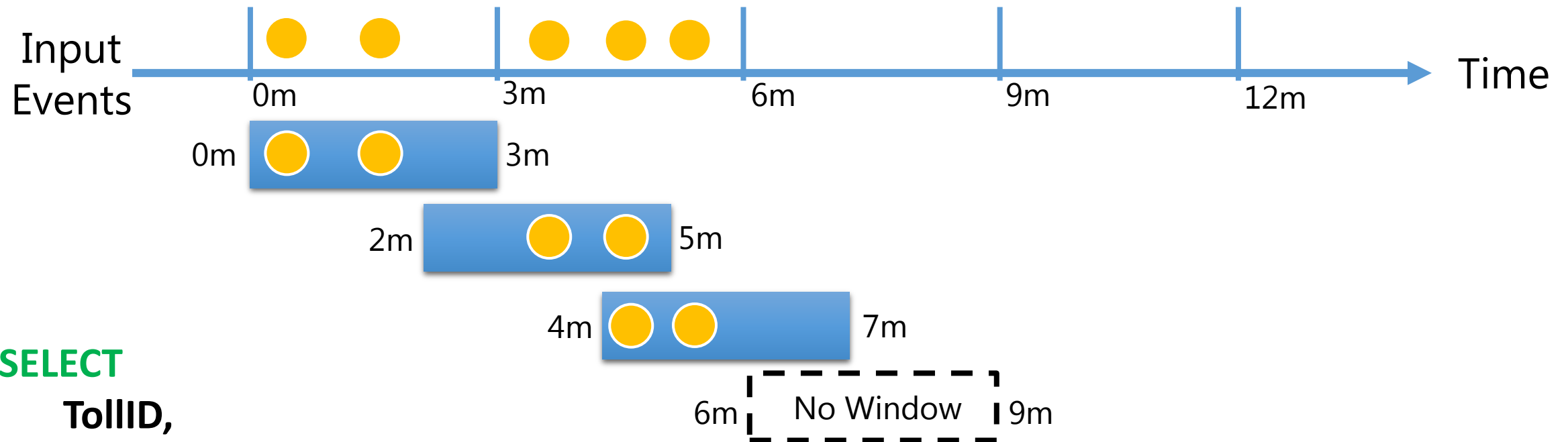
System.Timestamp AS WindowEnd,

Count(*) AS Count

FROM EntryStream TIMESTAMP BY EntryTime

GROUP BY TUMBLINGWINDOW(minute, 3), TollID

Hopping Window



SELECT

TollID,

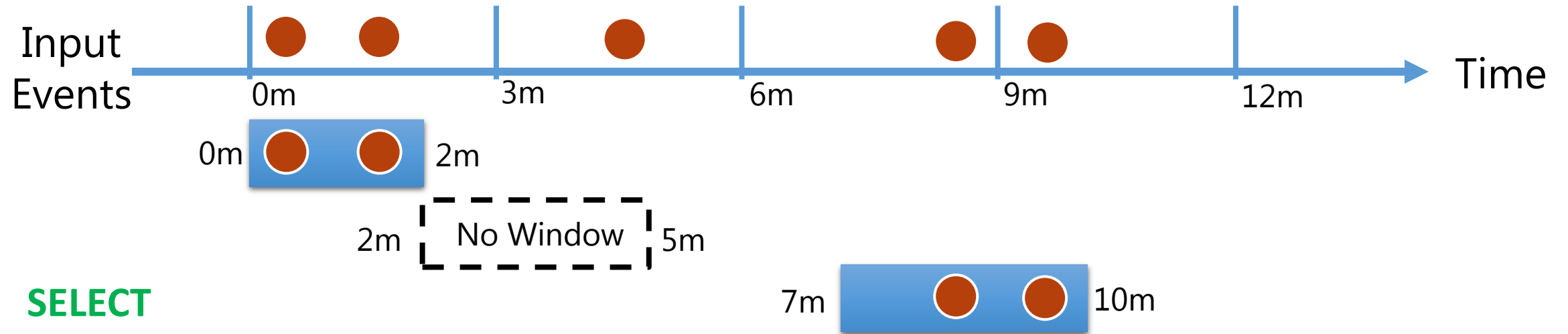
System.Timestamp AS WindowEnd,

Count(*) AS Count

FROM EntryStream TIMESTAMP BY EntryTime

GROUP BY HOPPINGWINDOW(minute, 2, 3), TollID

Sliding Window



SELECT

System.Timestamp AS WindowEnd,

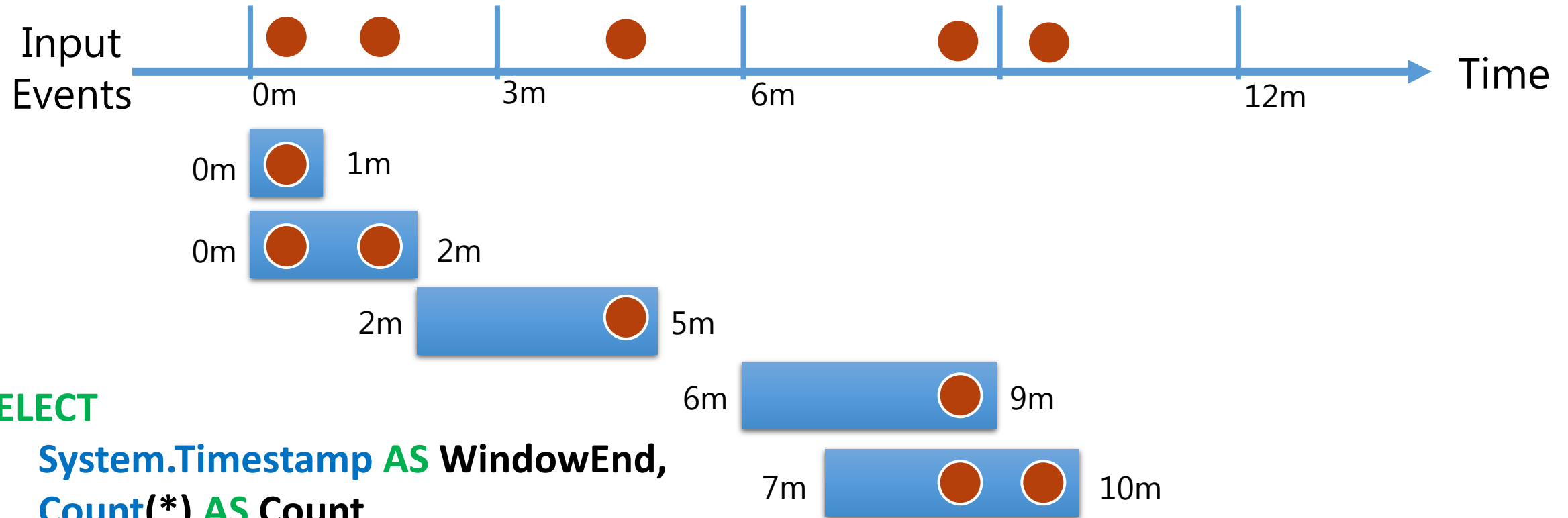
Count(*) AS Count

FROM EntryStream TIMESTAMP BY EntryTime

GROUP BY SLIDINGWINDOW(minute, 3)

HAVING CarCount > 2

Sliding Window: Without 'Having' Clause



SELECT

System.Timestamp **AS** WindowEnd,
Count(*) **AS** Count

FROM EntryStream **TIMESTAMP BY** EntryTime

GROUP BY **SLIDINGWINDOW**(minute, 3)

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
```

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
```

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

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

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

Joining Datasets

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

SELECT

```
EntryStream.EntryTime,  
EntryStream.LicensePlate,  
EntryStream.TollId,  
Registration.RegistrationId
```

(Broken at the moment)

FROM EntryStream **TIMESTAMP BY** EntryTime

JOIN Registration

ON EntryStream.LicensePlate = Registration.LicensePlate

WHERE Registration.Expired = '1'

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
```

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
```

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

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

No Caching (yet?)

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~~

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

Average of Average Approximations

