

# Analytics on Streaming Data with Azure Stream Analytics

Data Science Dojo



1 **NEW**  
DEFINITION  
IS ADDED ON  
**URBAN**  
DICTIONARY

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**

320+  
**NEW**  
**twitter**  
**ACCOUNTS**

100+  
**NEW**  
**Linked in**  
**ACCOUNTS**

1 **NEW**  
ARTICLE IS  
PUBLISHED  
associated content

6,600+  
**NEW**  
PICTURES ARE  
UPLOADED ON  
**flickr™**

50+  
**WORDPRESS**  
**DOWNLOADS**

695,000+  
**facebook**  
**STATUS**  
**UPDATES**

125+  
**PLUGIN**  
**DOWNLOADS**

79,364  
**WALL**  
**POSTS**

510,040  
**COMMENTS**

1,700+  
**Firefox**  
**DOWNLOADS**

694,445  
**SEARCH**  
**QUERIES**

168 MILLION  
**EMAILS**  
**ARE SENT**

60+  
**NEW**  
**BLOGS**

1,500+  
**BLOG**  
**POSTS**

70+  
**DOMAINS**  
**REGISTERED**

600+  
**NEW**  
**VIDEOS**



100+  
40+

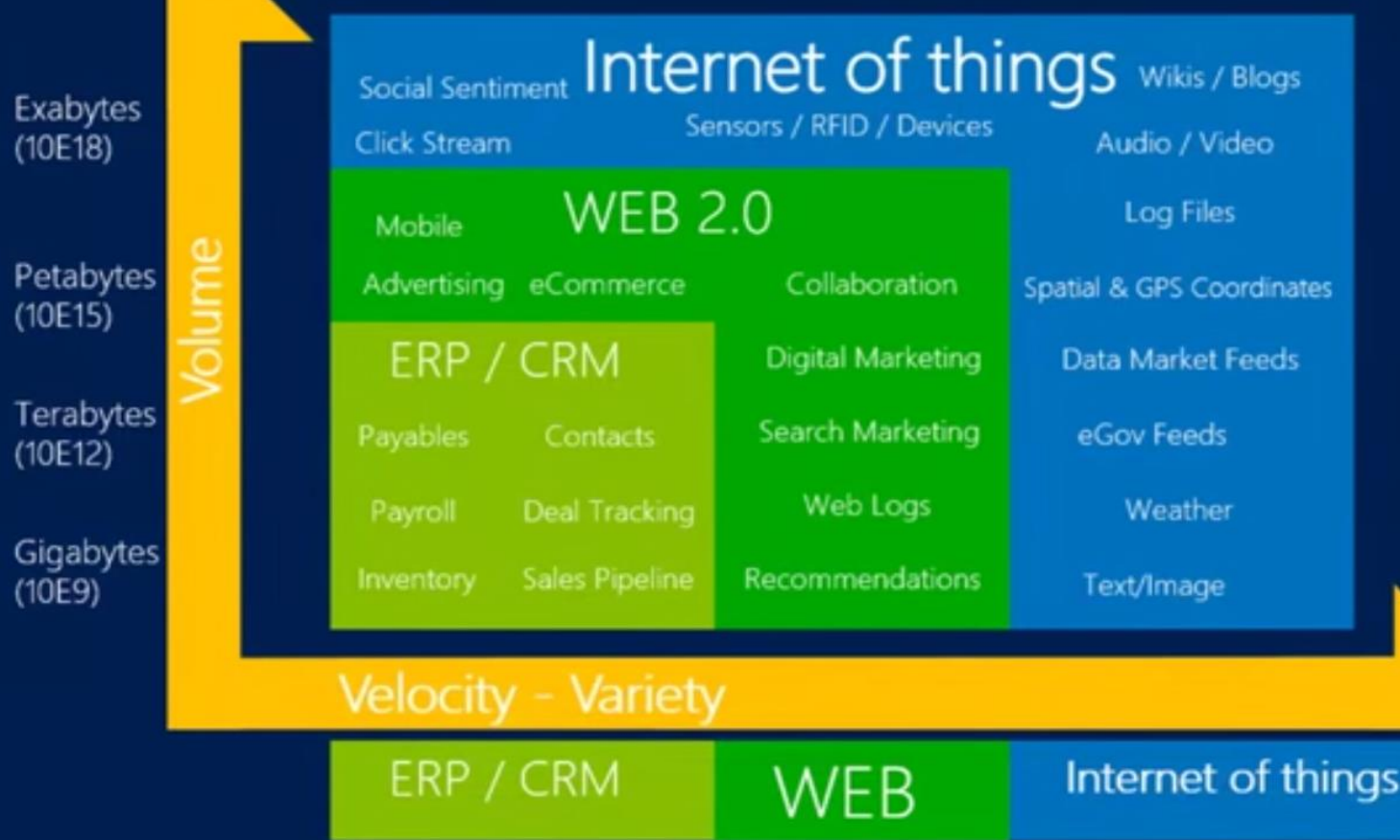
Answers.com  
YAHOO! ANSWERS

**QUESTIONS**  
ASKED ON THE  
INTERNET...

25+ HOURS  
**TOTAL**  
**DURATION**

# Introducing Big Data

## Continued



# Defining Real-time

Within seconds...

or...

Within minutes...

of an event occurring

Up to 2 hours

# Timeliness of Information



What was trending in the past 5 minutes?

Amber alert car detected!



A tornado will form in the next 30 minutes.

# Timeliness of Information



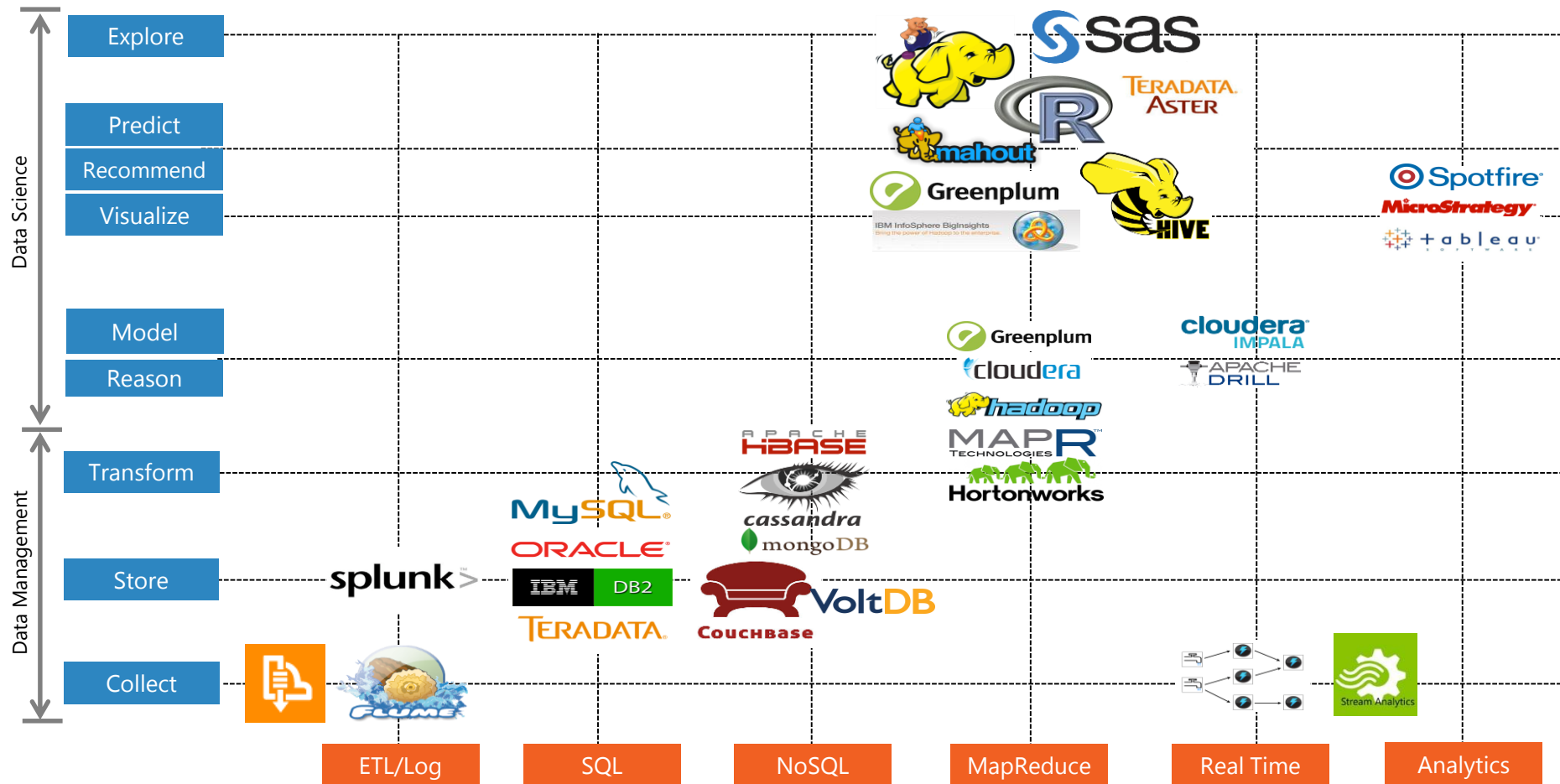
A stock is going to crash in 20 minutes.

A fire is about to start in your house.



The power grid will overload in 2 minutes.





# Typical Event Processing



**ETL Timespan**  
(Extract, Transform, Load)



# 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

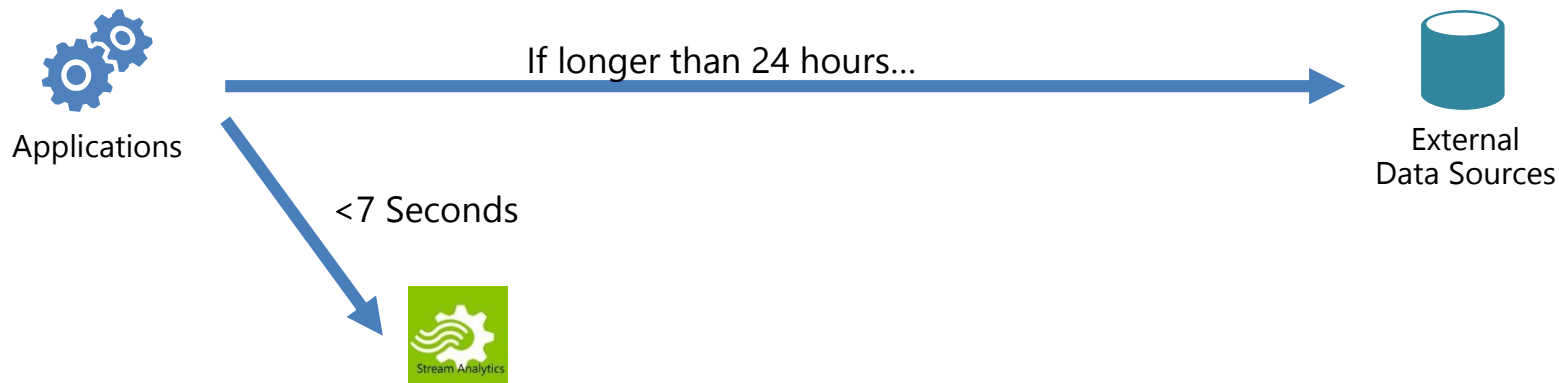


Stream Analytics



Data Analytics

# When to use Stream Processors



# ETL Should Still Happen



**Stream Processing**

**ETL**

- Stream Processing is only icing on the cake.
- It can, but should not replace a company's normal ETL cycle.

# Popular Up and Coming Event Processors



**Google DataFlow**



**Azure Stream  
Analytics**



**Amazon Kinesis**

**data**science**dojo**  
unleash the data scientist in you

# Demo

# Credit Card Transactions (swipes)



- Credit card transactions are usually done in batch as an EOTD send.
- Stream process for insights now.
- US mainland transactions



# Previously...

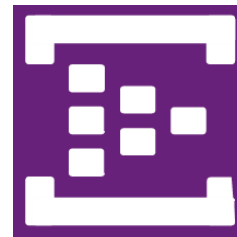


**Credit Card  
Reader  
(Synthetic)**

Swipes



**Message Broker  
(DataScienceDojo's  
Webpage)**



**Data Ingestor  
(Azure Event Hub)**




# The Streamer

- <http://demos.datasciencedojo.com/app/credit-card-streamer/>


## Credit Card Streamer

This app will simulate the kind of data streams that banks would encounter, credit card swipe data. The app will generate synthetic data from a credit card transaction (swipe) and pushes it into a given Azure Event Hub as a JSON. The application logic for this app is written entirely in JavaScript so the speed and interval of the transactions is dependent on the processing power of the user device.


### Event Hub Credentials

Event Hub Name (Need help? PDF Guide) 

Service Bus Namespace (Need help? PDF Guide) 

Shared Access Policy Name (Need help? PDF Guide) 

### Output Preview

Display Format (Data is still sent as a JSON):

```
Successfully loaded database. Ready to simulate data.
```

# The Data

```
{  
  "swipe_date":"2015-05-22T20:16:27.122Z",  
  "transaction_id":3127484,  
  "card_type":"VISA",  
  "card_number":"4913419738164560",  
  "expiration_month":"02",  
  "expiration_year":"18",  
  "cvv_code":"520",  
  "user_id":"972288",  
  "user_gender":"male",  
  "user_first_name":"Alexander",  
  "user_last_name":"Hamilton",  
  "merchant":"McDonald's",  
  "transaction_amount":13.64,  
  "balance":336.48,  
  "merchant_fee":.5,  
  "swipe_city":"New York",  
  "swipe_state":"New York",  
  "swip_city_state":"New York, NY",  
  "InstanceNo":1  
}
```

# Data vs Events

```
{  
  "swipe_date":"2015-05-22T20:16:27.122Z",  
  "transaction_id":3127484,  
  "card_type":"VISA",  
  "card_number":"4913419738164560",  
  "expiration_month":"02",  
  "expiration_year":"18",  
  "cvv_code":"520",  
  "user_id":"972288",  
  "user_gender":"male",  
  "user_first_name":"Alexander",  
  "user_last_name":"Hamilton",  
  "merchant":"McDonald's",  
  "transaction_amount":13.64,  
  "balance":336.48,
```

An event is just data  
with a timestamp



# Inside the Event Hub



**Credit Card  
Reader  
(Synthetic)**

Swipes  
→



**Message Broker  
(DataScienceDojo's  
Webpage)**

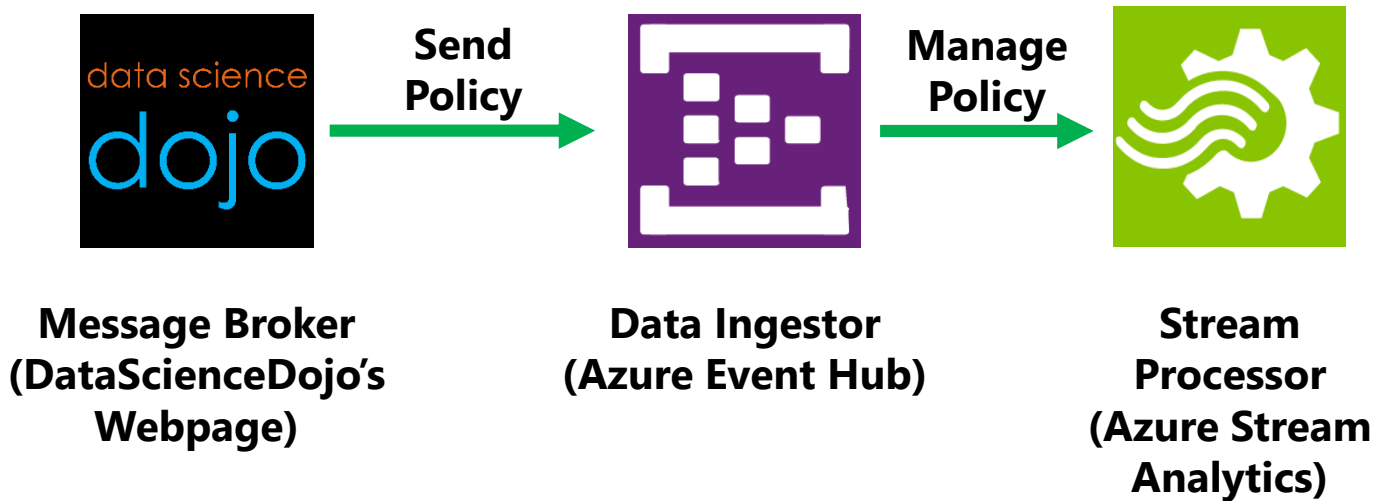
→

Service Bus Namespace:  
YourNameSpace

**Event Hub 1:**

- Credit Card Swipes

# Setting Policies



# With Stream Processor



**Credit  
Card  
Reader  
(Synthetic)**

Swipes  
→



**Message Broker  
(DataScienceDojo's  
Webpage)**



**Data Ingestor  
(Azure Event  
Hub)**



**Stream Processor 1  
(All Data)**



**Long-term Storage  
(Azure Storage)**

# More Processors

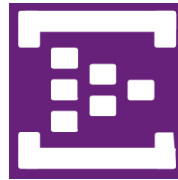


Credit Card Reader (Synthetic)

Swipes →



Message Broker (DataScienceDojo's Webpage)



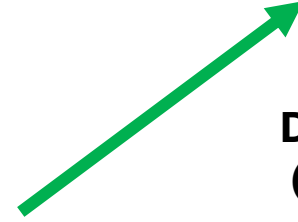
Data Ingestor (Azure Event Hub)



Stream Processor 1 (All Data)



Long-term Storage (Azure Storage)



Dashboard (PowerBI)



Stream Processor 2 (Filter/Aggregate)



Anomaly Detection Event Hub



# SQL with Data at Rest

- **Question** "Show me VISA transactions from last month."
- **Answering with a relational database**  
No problem! Here you go!
- **SELECT \***  
**FROM** credit\_db  
**WHERE** card\_type **like** VISA'

# SQL Data in Motion

- **Different Question** "Show me VISA transactions in the past 2 minutes."
- **Answering with a relational database**  
I'm not ready yet... Ask again later... Or tomorrow (after batch)...
- **Not a great solution...**



# Azure Stream Query Language

- Queries through time
- 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 as its unique identifier.
  - Temporal integrity and referential integrity
- Stream Analytics can append your events with a timestamp (bad practice if standalone)
  - The default timestamp will be when the event enters Stream Analytics
  - Can be skewed by network and hardware latency, or legacy processing
- Users can define application time stamps with the `TIMESTAMP BY` clause

# Which Timestamp?

```
{  
  "swipe_date": "2015-05-21T22:47:55.0770000Z",  
  "transaction_id": 222301082,  
  "card_type": "VISA",  
  "card_number": "40265691066025560",  
  "expiration_month": "06",  
  "expiration_year": "22",  
  "cvv_code": "3310",  
  "user_id": "690548",  
  "user_gender": "male",  
  "user_first_name": "Caden",  
  "user_last_name": "Hatton",  
  "merchant": "Macy's",  
  "transaction_amount": 4.98,  
  "balance": 7223.9,  
  "merchant_fee": 0.5,  
  "swipe_city": "New York",  
  "swipe_state": "New York",  
  "swipe_city_state": "New York, NY",  
  "InstanceNo": 1,  
  "EventProcessedUtcTime": "2015-05-21T22:47:50.0879821Z",  
  "PartitionId": 3,  
  "EventEnqueuedUtcTime": "2015-05-21T22:47:49.9850000Z"  
}
```

Time of event

Time processed by  
stream processor

Time entered broker

# Same Event...

```
{  
  "swipe_date":"2015-05-21T22:47:55.0770000Z",  
  "EventProcessedUtcTime":"2015-05-21T22:47:50.0879821Z",  
  "EventEnqueuedUtcTime":"2015-05-21T22:47:49.9850000Z"  
}
```

According to these timestamps, the event happened 5 seconds AFTER the event was processed and queued.

- How can that be?
- The event was not confined to the physical laws of space and time.

**The clock on your device matters.**

# Azure Stream Query Language

- Show me transactions as they happen.  
Write it to a blob AND powerBI.

```
SELECT *  
INTO MyBlob  
FROM SwipeStream TIMESTAMP BY swipe_date;  
SELECT *  
INTO PowerBI  
FROM SwipeStream TIMESTAMP BY swipe_date;
```



# StreamQL: Calculations

- What was our commission on each transaction?

**SELECT**

transaction\_id,  
merchant\_fee / transaction\_amount **AS** Commision

**FROM** SwipeStream

**TIMESTAMP BY** swipe\_date

# StreamQL: Filter Queries

- Show me only VISA transactions that made over \$5 revenue.

**SELECT**

swipe\_date,  
card\_type,  
merchant\_fee **AS** revenue

**FROM** SwipeStream

**TIMESTAMP BY** swipe\_date

**WHERE** card\_type **LIKE** 'VISA'

**AND** merchant\_fee < 5

SWIPE_DATE	CARD_TYPE	REVENUE
2015-05-21T2...	VISA	6.2
2015-05-21T2...	VISA	10.31
2015-05-21T2...	VISA	11.72
2015-05-21T2...	VISA	7.82
2015-05-21T2...	VISA	9.91
2015-05-21T2...	VISA	7.62
2015-05-21T2...	VISA	5.25

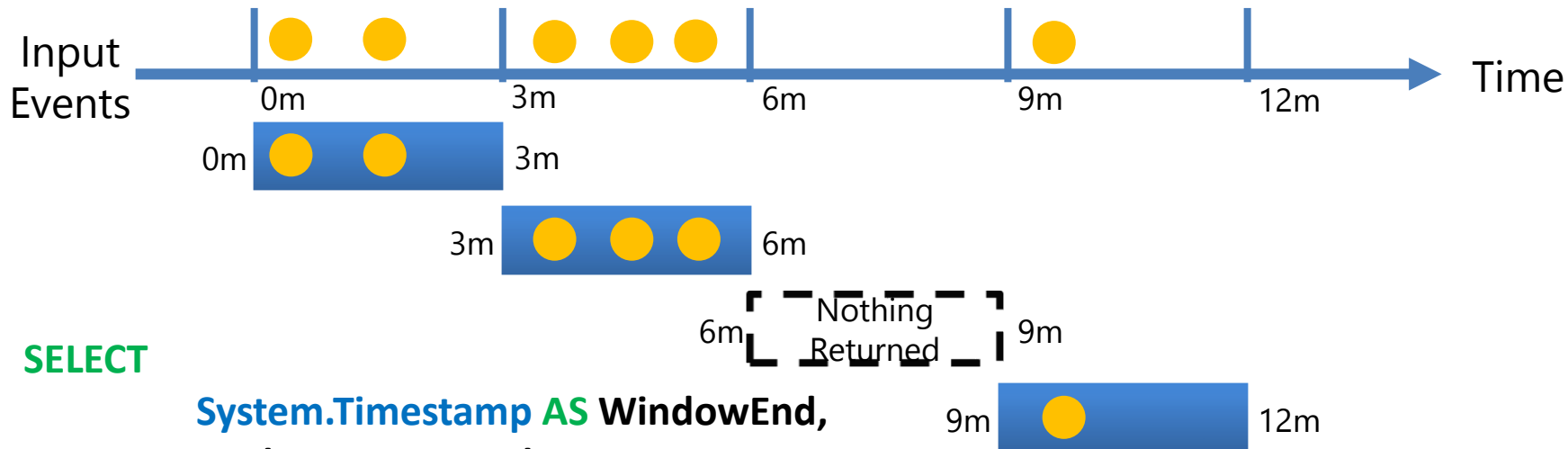
# Temporal Questions

Count the number of transactions....

- When should the counting of transactions begin?
- When should the counting of transactions end?
- How long should the transactions be counted for?
- How often do transactions need to be counted?

# Tumbling Window

How many transactions were made for each card type every 3 minute?



**SELECT**

**System.Timestamp** **AS** WindowEnd,

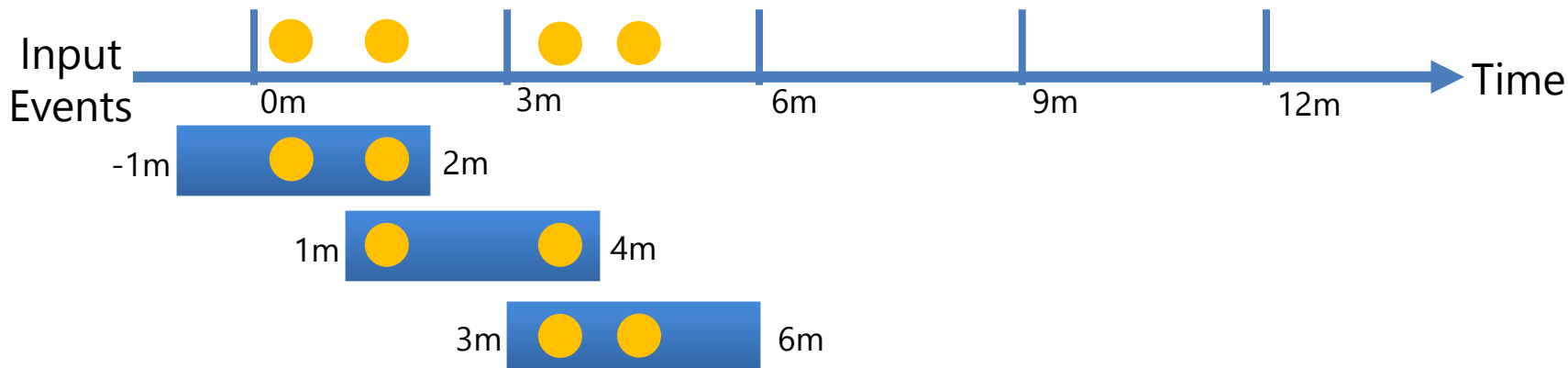
**card\_type** **AS** CardType,

**Count(\*)** **AS** Frequency

**FROM** SwipeStream **TIMESTAMP BY** swipe\_date

**GROUP BY** **TUMBLINGWINDOW**(minute, 3), card\_type

# Hopping Window



**SELECT**

**System.Timestamp** AS WindowEnd,

**card\_type** AS CardType,

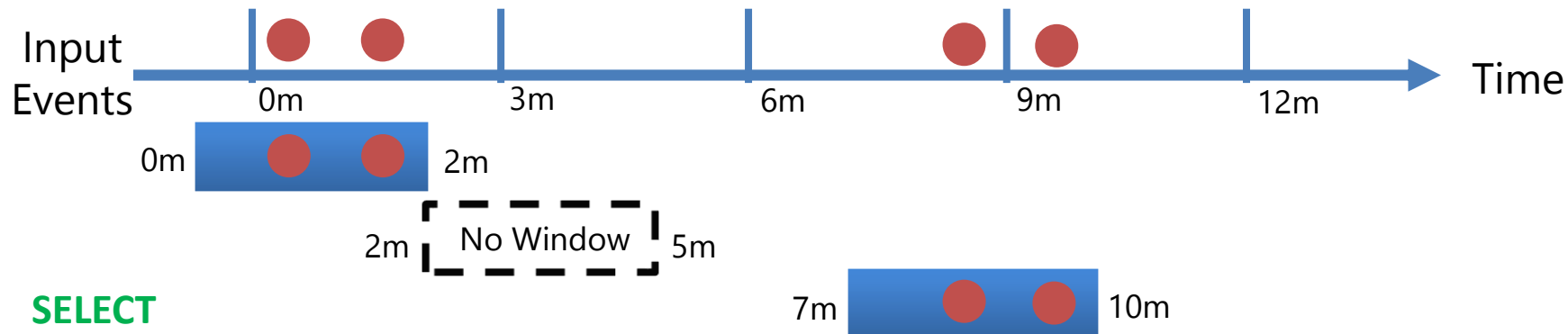
**Count(\*)** AS Frequency

**FROM** SwipeStream **TIMESTAMP BY** swipe\_date

**GROUP BY** HoppingWindow(minute, 3, 2), card\_type

5m | [ Nothing Returned ] | 8m

# Sliding Window

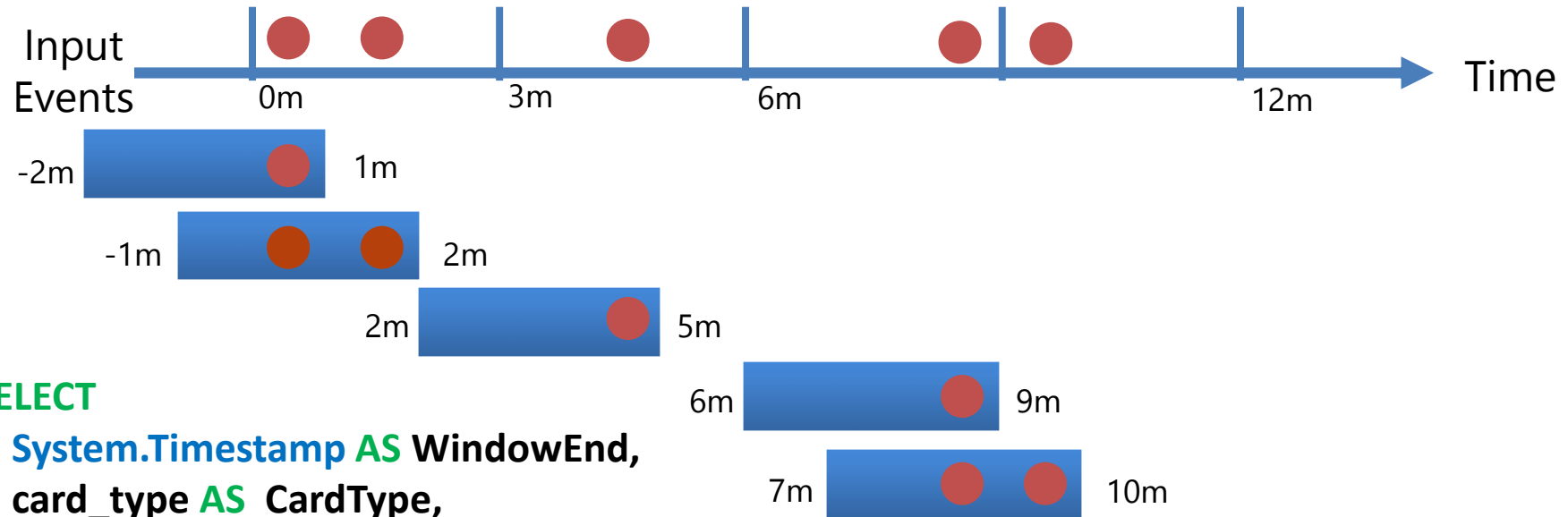


**SELECT**

```
System.Timestamp AS WindowEnd,  
card_type AS CardType,  
Count(*) AS Frequency
```

```
FROM SwipeStream TIMESTAMP BY swipe_date  
GROUP BY SlidingWindow(minute, 3), card_type  
HAVING Frequency > 2
```

# Sliding Window: Without 'Having' Clause



**SELECT**

**System.Timestamp** AS WindowEnd,

**card\_type** AS CardType,

**Count(\*)** AS Frequency

**FROM** SwipeStream **TIMESTAMP BY** swipe\_date

**GROUP BY** SlidingWindow(minute, 3), card\_type



# Sum Aggregation

- How much revenue is being accumulated from merchants every 3 minutes?

SELECT

System.Timestamp AS WindowEnd,

Sum(merchant\_fee) AS IntervalRevenue

FROM SwipeStream TIMESTAMP BY swipe\_date

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(merchant_fee) AS IntervalRevenue
```

```
FROM SwipeStream TIMESTAMP BY swipe_date
```

```
GROUP BY TUMBLINGWINDOW(minute, 3), WindowEnd
```

```
Having IntervalRevenue > 10
```

# Descriptive Statistics

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

**SELECT**

```
System.Timestamp AS WindowEnd,  
count(merchant_fee) AS CarCount,  
min(merchant_fee) AS MinRev,  
max(merchant_fee) AS MaxRev,  
avg(merchant_fee) AS AvgRev,  
stdev(merchant_fee) AS VarRev,  
sum(merchant_fee) AS TotalRev
```

```
FROM SwipeStream TIMESTAMP BY swipe_date  
GROUP BY TUMBLINGWINDOW(minute, 3)
```

# DateDiff and Time

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

**SELECT**

**System.Timestamp AS WindowEnd,**

**count(\*) AS Frequency,**

**datediff(second, min(swipe\_date), max(swipe\_date)) AS FirstLastDuration,**

**datediff(second, min(swipe\_date), System.Timestamp) AS FirstEndDuration**

**FROM SwipeStream****TIMESTAMP BY** swipe\_date

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

# Joining Stream with Reference Data

- Say we had a list of stolen credit card numbers. Let's run each transaction against this list and get the locations.

```
SELECT
    SwipeStream.swipe_date as SwipeTime,
    SwipeStream.card_number as CardNumber,
    SwipeStream.merchant as Store,
    SwipeStream.swipe_city_state as Location,
    StolenList.Stolen as Stolen
FROM SwipeStream TIMESTAMP BY swipe_date
JOIN StolenList
ON SwipeStream.card_number = StolenList.card_number
WHERE StolenList.Stolen = '1'
```

# Joining Streams, Temporally

- How long did it take for each transaction to get approval from the bank?
  - Joining on events through time
  - JOIN operator requires specifying a temporal wiggle room describing an acceptable time difference between the joined events
  - If two transactions occurred within the same join interval, then consider them the same event.

# Joining Streams

- How long did it take for each transaction to get approval from the bank?

**SELECT**

swipe.transaction\_id

swipe.swipe\_date,

bank.approval\_time,

**DATEDIFF** ( **second**, swipe.swipe\_date, bank. approval\_time) **AS** DurationInSeconds

**FROM** SwipeStream **AS** swipe **TIMESTAMP BY** swipe\_date

**JOIN** BankStream **AS** bank **TIMESTAMP BY** approval\_time

**ON** (swipe.transaction\_id = bank.transaction\_id)

**AND DATEDIFF** ( **minute**, swipe, bank ) **BETWEEN** 0 AND 15

# Joining Streams, by Window

- What was the average time that it took for transactions to get approved every 3 minutes?

**SELECT**

**System.Timestamp AS WindowEnd,**

**avg( DATEDIFF ( second, swipe.swipe\_date, bank.approval\_time )) AS ApprovalTime**

**FROM SwipeStream AS swipe TIMESTAMP BY swipe\_date**

**JOIN BankStream AS bank TIMESTAMP BY approval\_time**

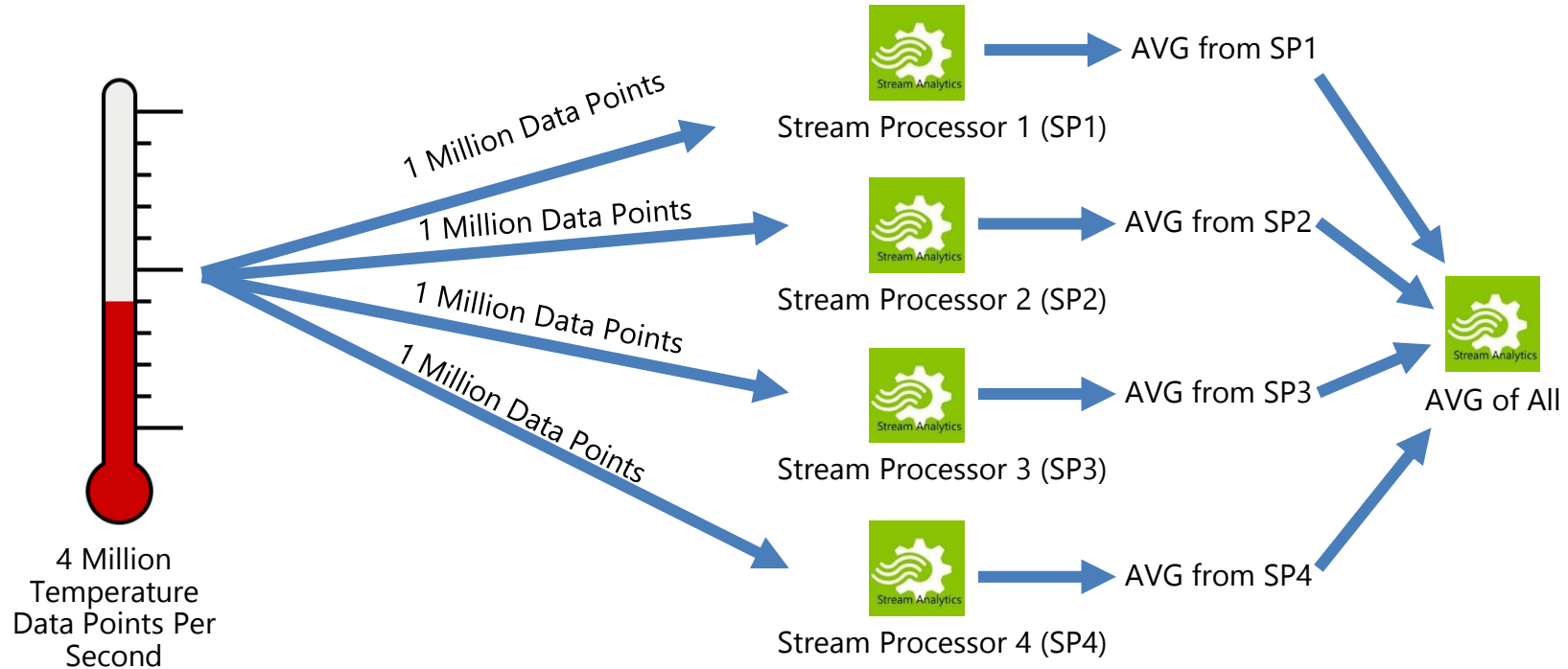
**ON (swipe.transaction\_id = bank.transaction\_id)**

**AND DATEDIFF ( minute, swipe, bank ) BETWEEN 0 AND 15**

**Group by TumblingWindow( minute, 3)**



# Average of Average Approximations



# 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**

# QUESTIONS