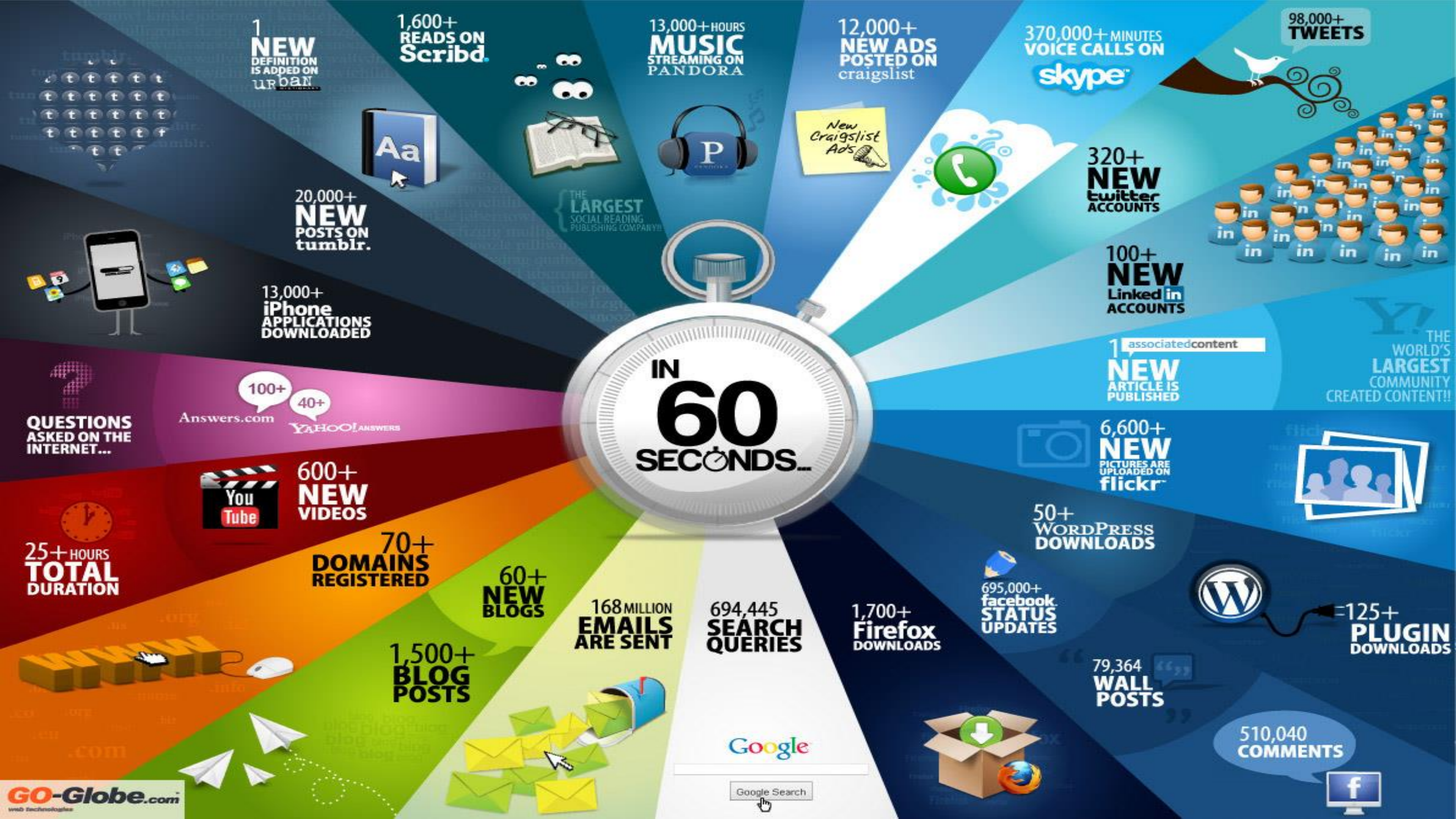
Streaming Processing with Azure Stream Analytics

Data Science Dojo

datasciencedojo

data science for everyone



IN 60 SECONDS...

1 NEW DEFINITION IS ADDED ON urban DICTIONARY

1,600+ READS ON Scribd.

THE LARGEST SOCIAL READING PUBLISHING COMPANY!!

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 LinkedIn ACCOUNTS

1 NEW associatedcontent ARTICLE IS PUBLISHED

THE WORLD'S LARGEST COMMUNITY CREATED CONTENT!!

6,600+ NEW PICTURES ARE UPLOADED ON flickr

50+ WORDPRESS DOWNLOADS

=125+ PLUGIN DOWNLOADS

695,000+ facebook STATUS UPDATES

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

25+ HOURS TOTAL DURATION

600+ NEW VIDEOS

QUESTIONS ASKED ON THE INTERNET...

100+ Answers.com

40+ YAHOO! ANSWERS

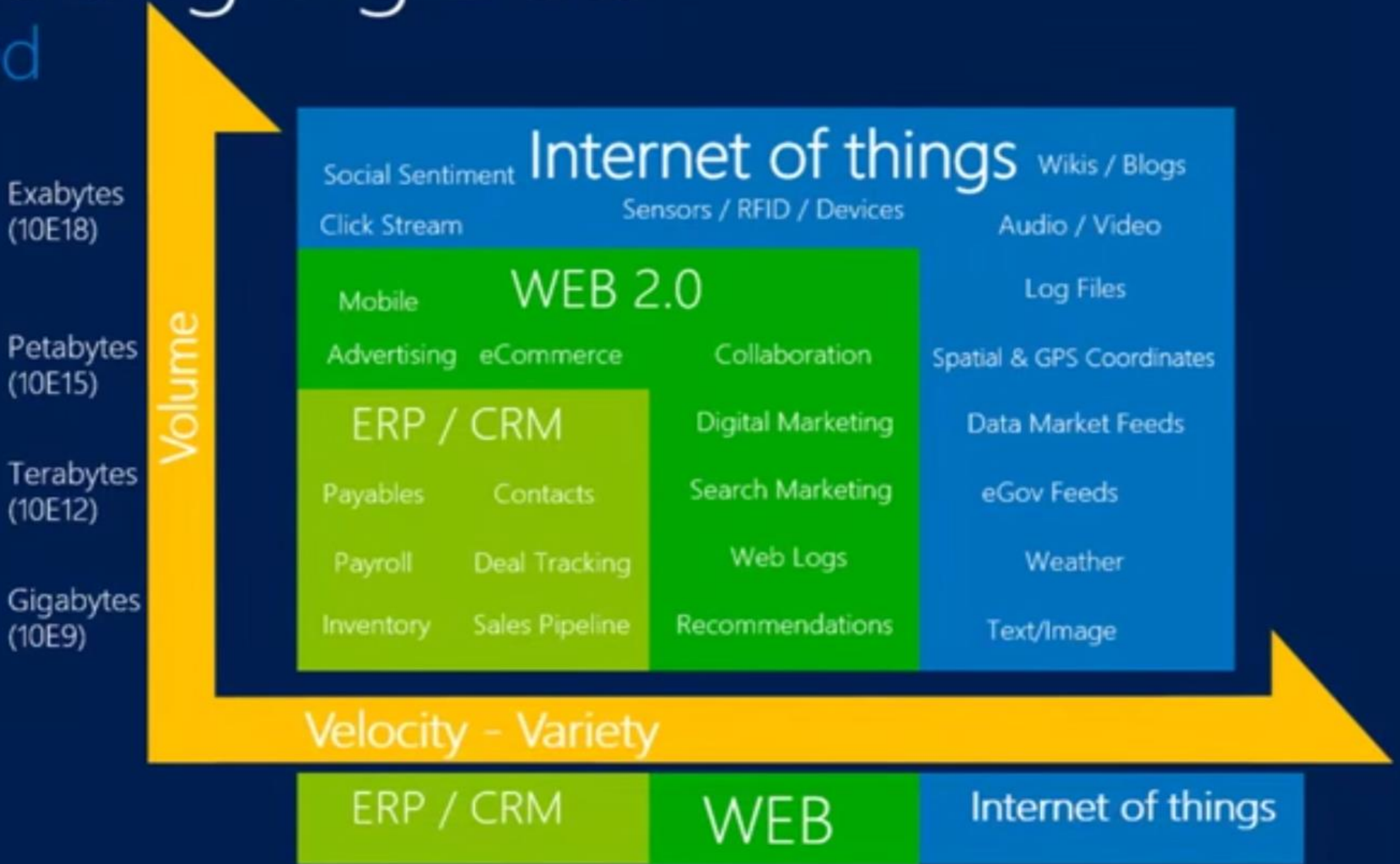
13,000+ iPhone APPLICATIONS DOWNLOADED

20,000+ NEW POSTS ON tumblr.

GO-Globe.com web technologies

Introducing Big Data

Continued



Exabytes (10E18)

Petabytes (10E15)

Terabytes (10E12)

Gigabytes (10E9)

Volume

Internet of things

Social Sentiment

Wikis / Blogs

Click Stream

Sensors / RFID / Devices

Audio / Video

Log Files

Spatial & GPS Coordinates

Data Market Feeds

eGov Feeds

Weather

Text/Image

WEB 2.0

Mobile

Advertising

eCommerce

Collaboration

Digital Marketing

Search Marketing

Web Logs

Recommendations

ERP / CRM

Payables

Contacts

Payroll

Deal Tracking

Inventory

Sales Pipeline

Velocity - Variety

ERP / CRM

WEB

Internet of things

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!

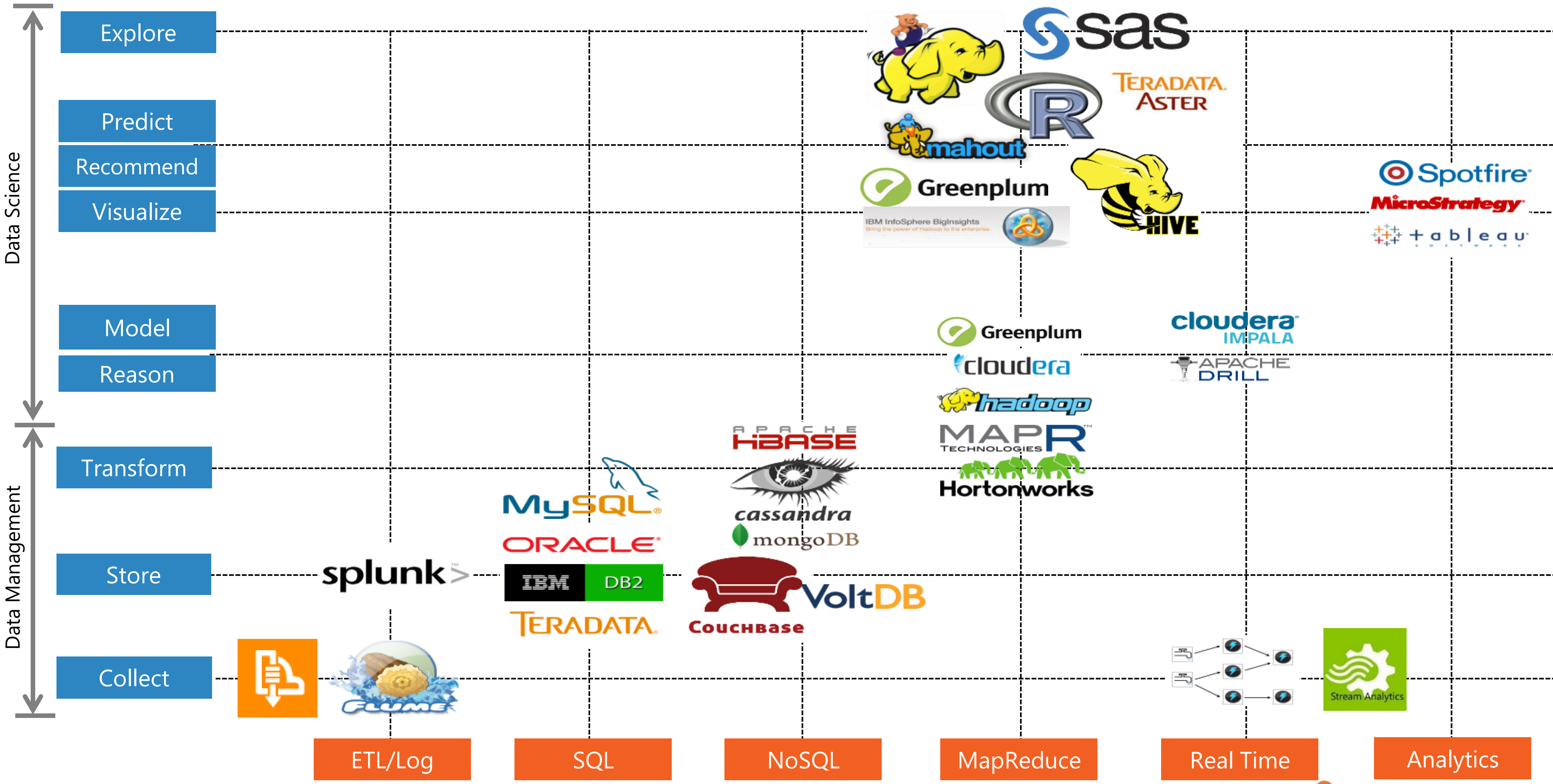
A tornado will form in the next 30 minutes.

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.



Data Science

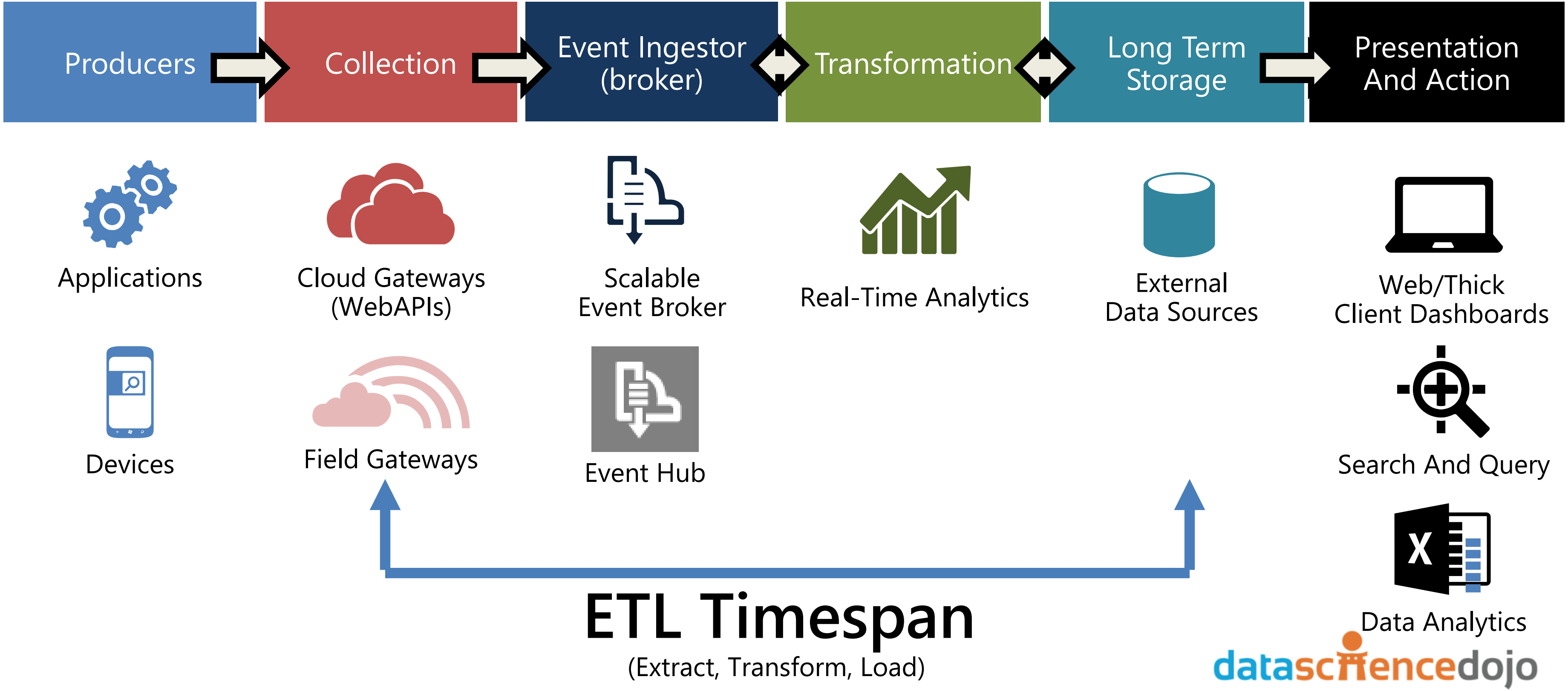
- Explore
- Predict
- Recommend
- Visualize
- Model
- Reason

Data Management

- Transform
- Store
- Collect

ETL/Log | SQL | NoSQL | MapReduce | Real Time | Analytics

Typical Event Processing



Producers → Collection → Event Ingestor (broker) ↔ Transformation ↔ Long Term Storage → Presentation And Action

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

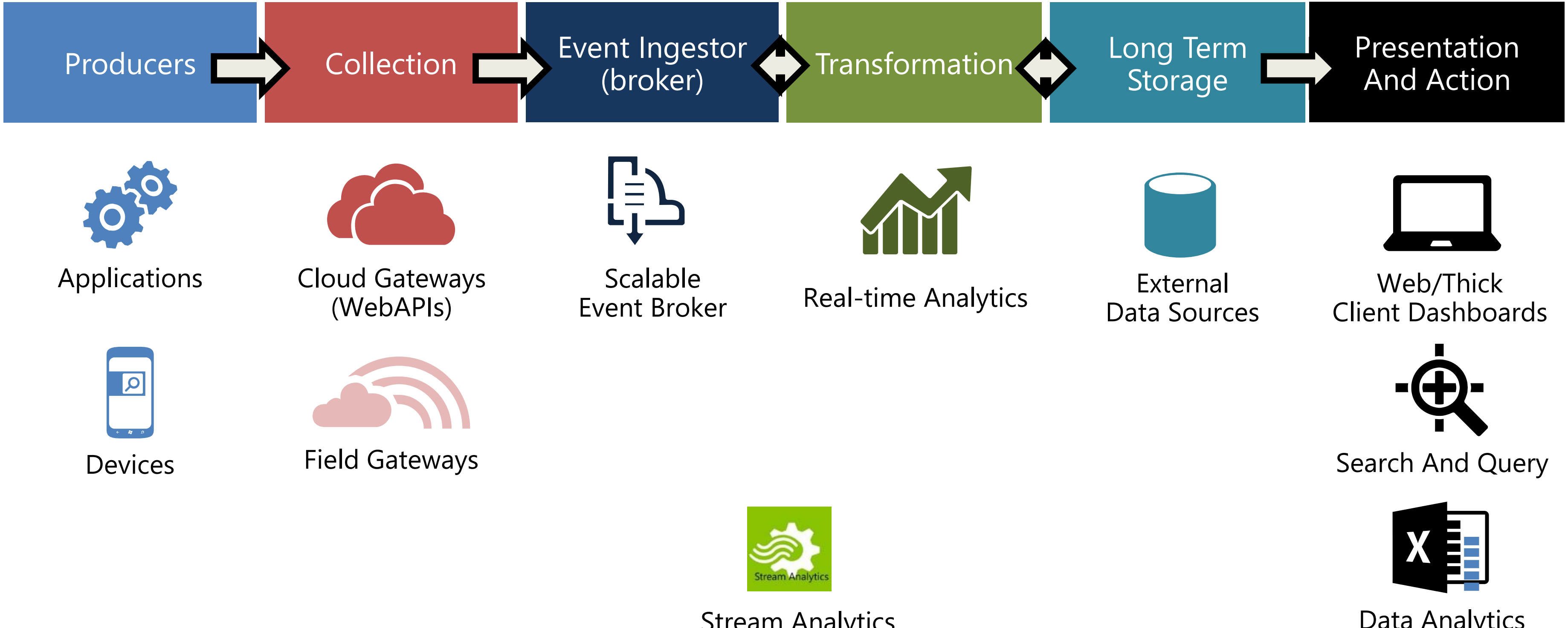
ETL Timespan

(Extract, Transform, Load)

datasciencedojo

data science for everyone

Typical Event Processing



Producers → Collection → Event Ingestor (broker) ↔ Transformation ↔ Long Term Storage → Presentation And Action

Applications

Devices

Cloud Gateways (WebAPIs)

Field Gateways

Scalable Event Broker

Real-time Analytics

Stream Analytics

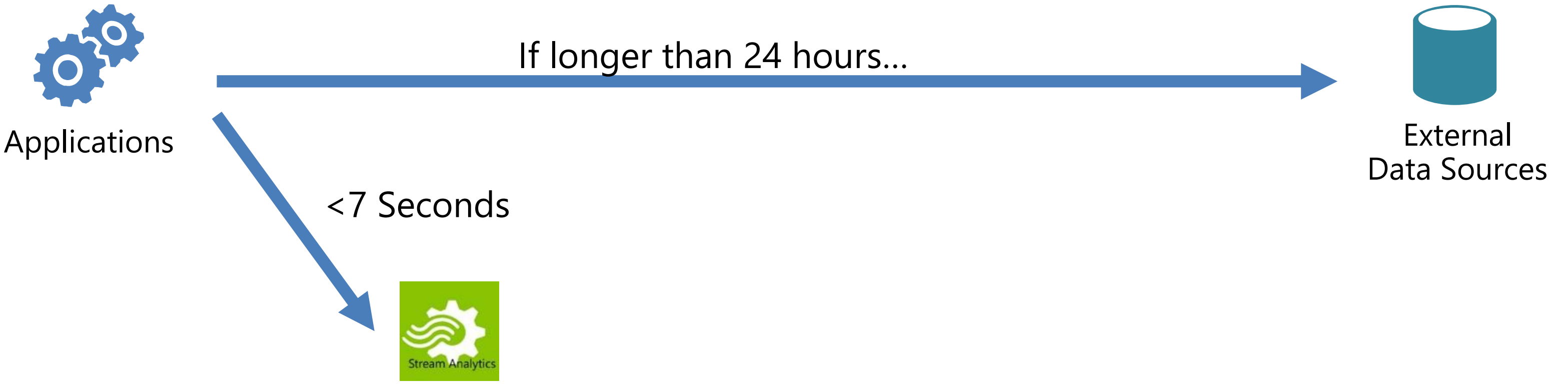
External Data Sources

Web/Thick Client Dashboards

Search And Query

Data Analytics

When to use Stream Processors



Applications

If longer than 24 hours…

External Data Sources

<7 Seconds

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

APACHE STORM™

Distributed • Resilient • Real-time

Cloud Dataflow Beta

Google DataFlow

Spark Streaming

Azure Stream Analytics

Amazon Kinesis

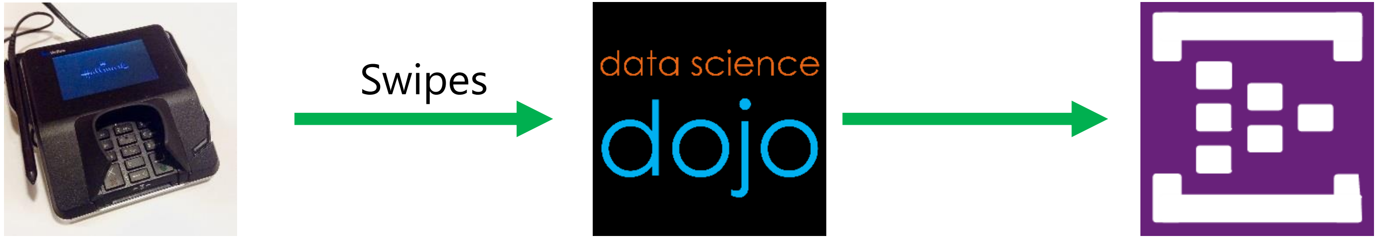
Demo

Credit Card Transactions (swipes)

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

datasciencedojo
data science for everyone

Previously…



Swipes

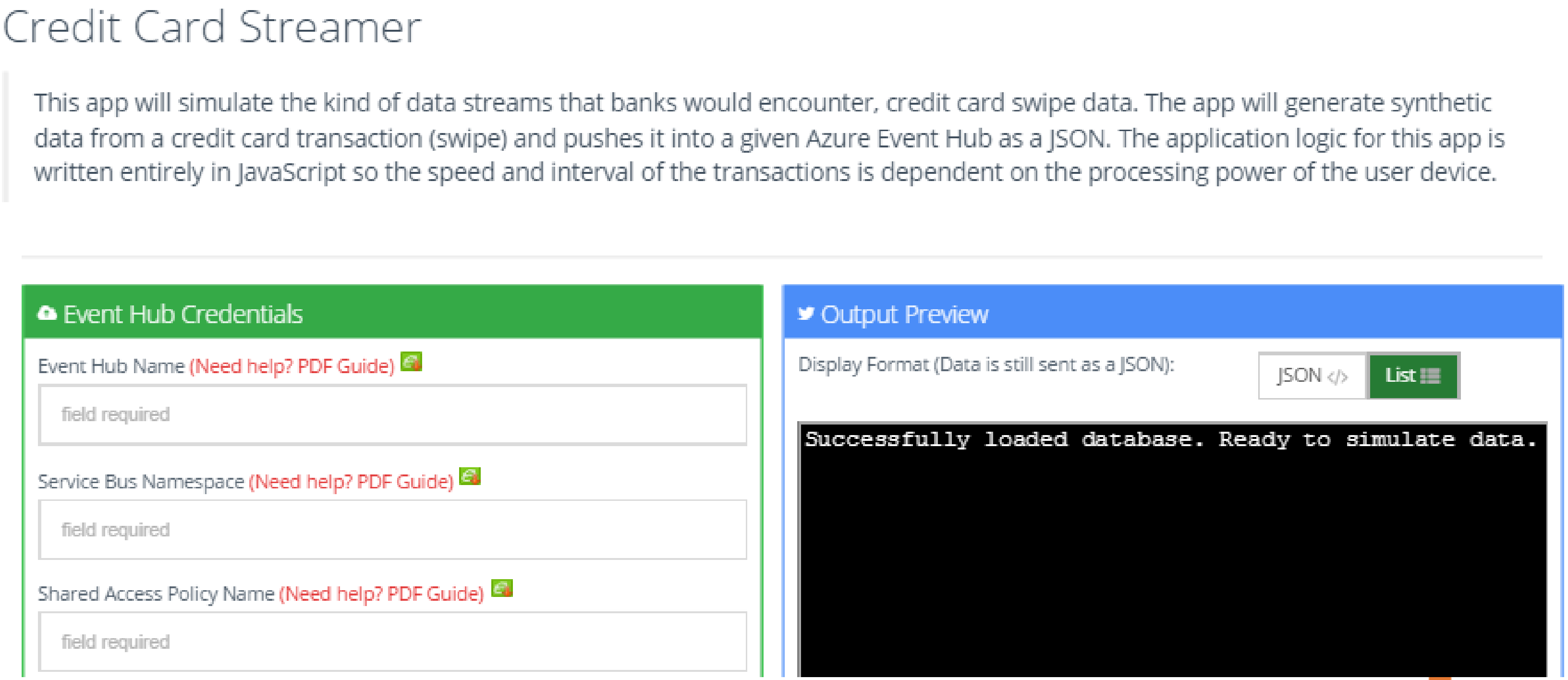
Credit Card Reader (Synthetic)

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)

field required

Service Bus Namespace (Need help? PDF Guide)

field required

Shared Access Policy Name (Need help? PDF Guide)

field required

Output Preview

Display Format (Data is still sent as a JSON): JSON </> List

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

datasciencedojo
data science for everyone

The Data

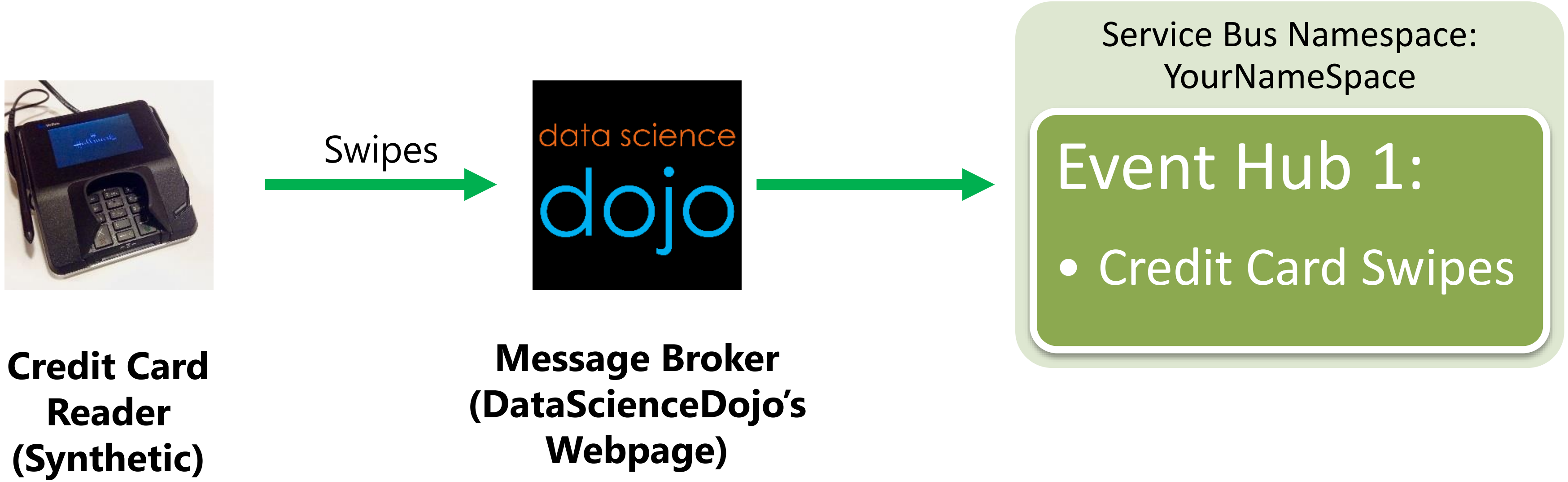
```
{
    "swipe_date":"2015-05-22T20:16:27.122Z",
    "transaction_id":3127484,
    "card_type":"VISA",
    "card_number":"4913419738164560",
    "expiration_month":"02",
    "expiration_year":"18",
    "cvv_code":"520",
    "user_id":"972288",
    "user_gender":"male",
    "user_first_name":“Alexander",
    "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



Swipes

Credit Card Reader (Synthetic)

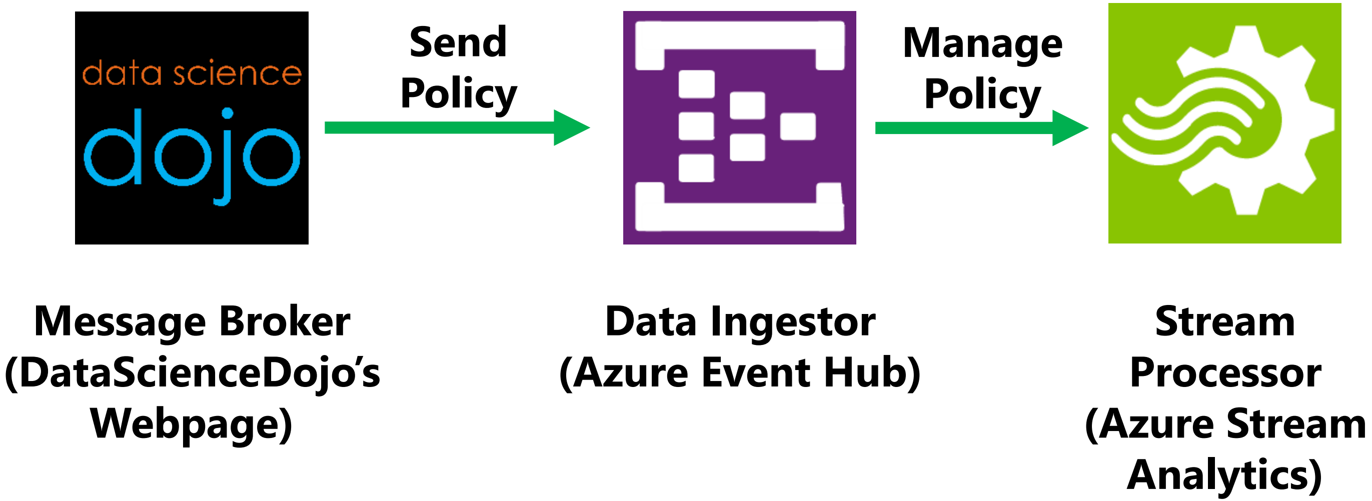
Message Broker (DataScienceDojo's Webpage)

Service Bus Namespace: YourNameSpace

Event Hub 1:

- Credit Card Swipes

Setting Policies



Send Policy

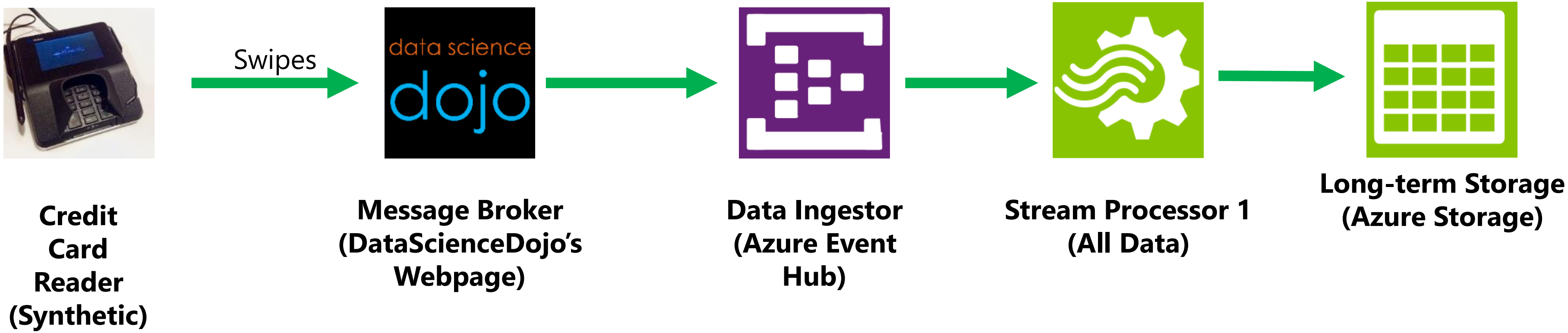
Manage Policy

Message Broker (DataScienceDojo’s Webpage)

Data Ingestor (Azure Event Hub)

Stream Processor (Azure Stream Analytics)

With Stream Processor



Swipes

Credit Card Reader (Synthetic)

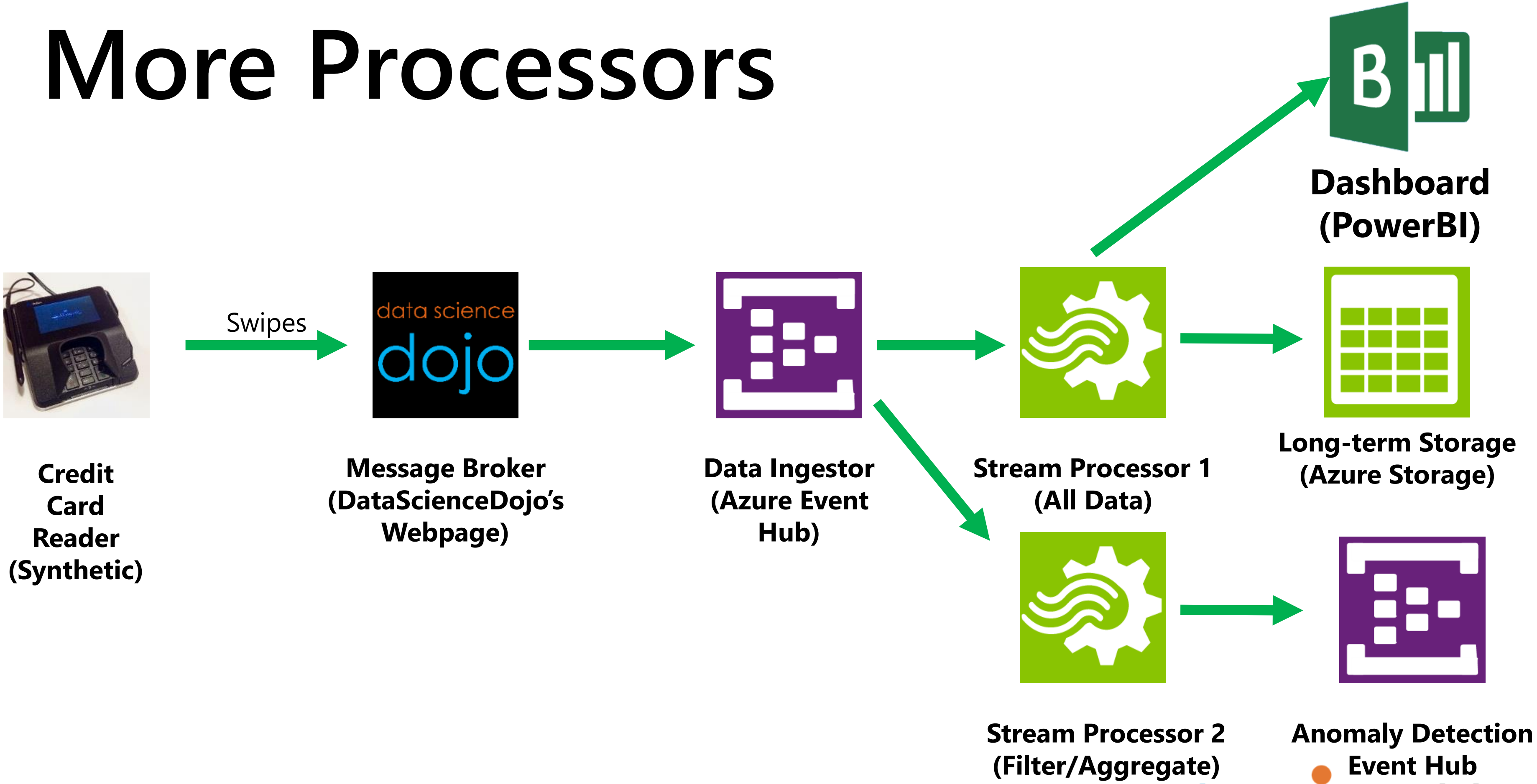
Message Broker (DataScienceDojo's Webpage)

Data Ingestor (Azure Event Hub)

Stream Processor 1 (All Data)

Long-term Storage (Azure Storage)

More Processors



Swipes

Credit Card Reader (Synthetic)

Message Broker (DataScienceDojo's Webpage)

Data Ingestor (Azure Event Hub)

Stream Processor 1 (All Data)

Dashboard (PowerBI)

Long-term Storage (Azure Storage)

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'

datasciencedojo
data science for everyone

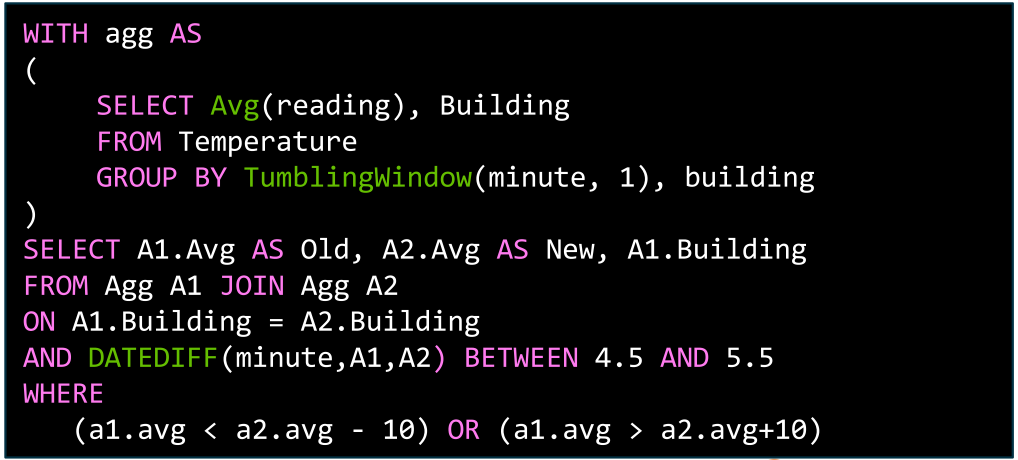
SQL Data in Motion

- **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)
```

datasciencedojo

data science for everyone

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",
   "swip_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

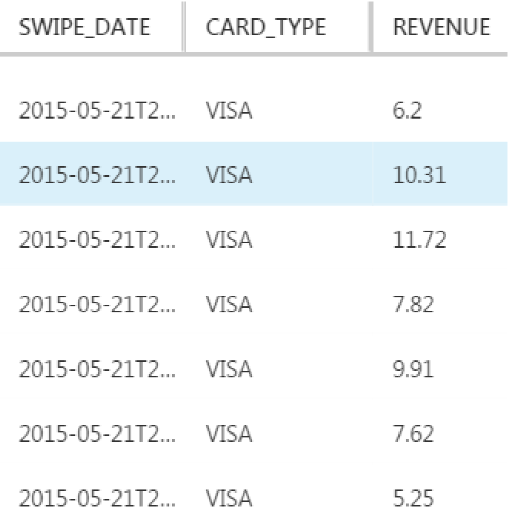
- What was our commission on each transaction?

```
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

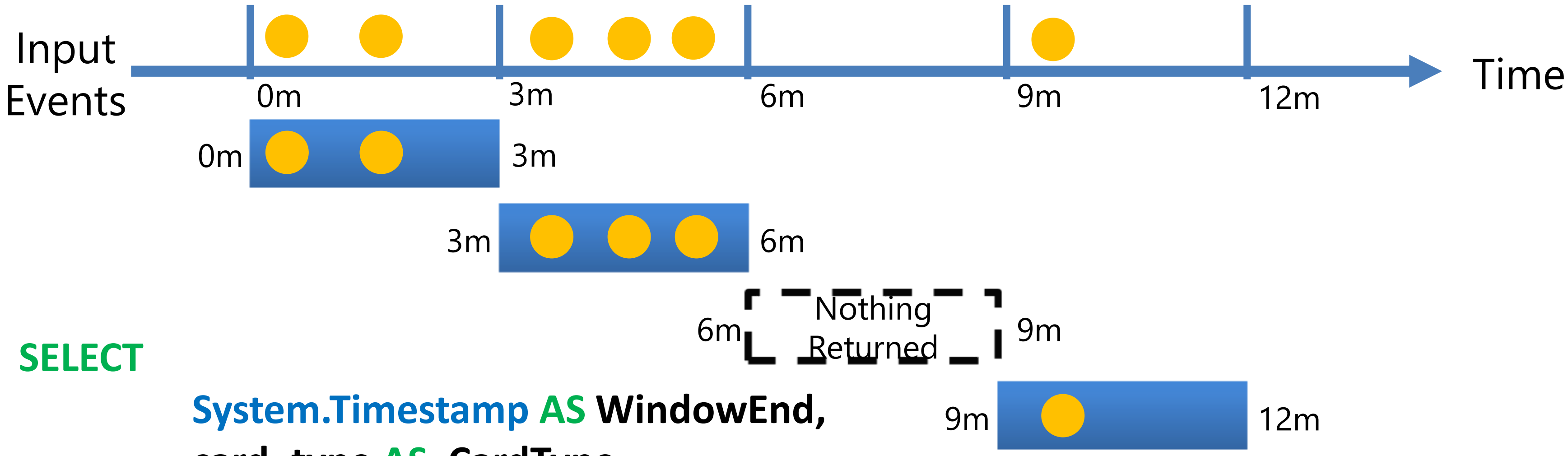
Temporal Questions

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

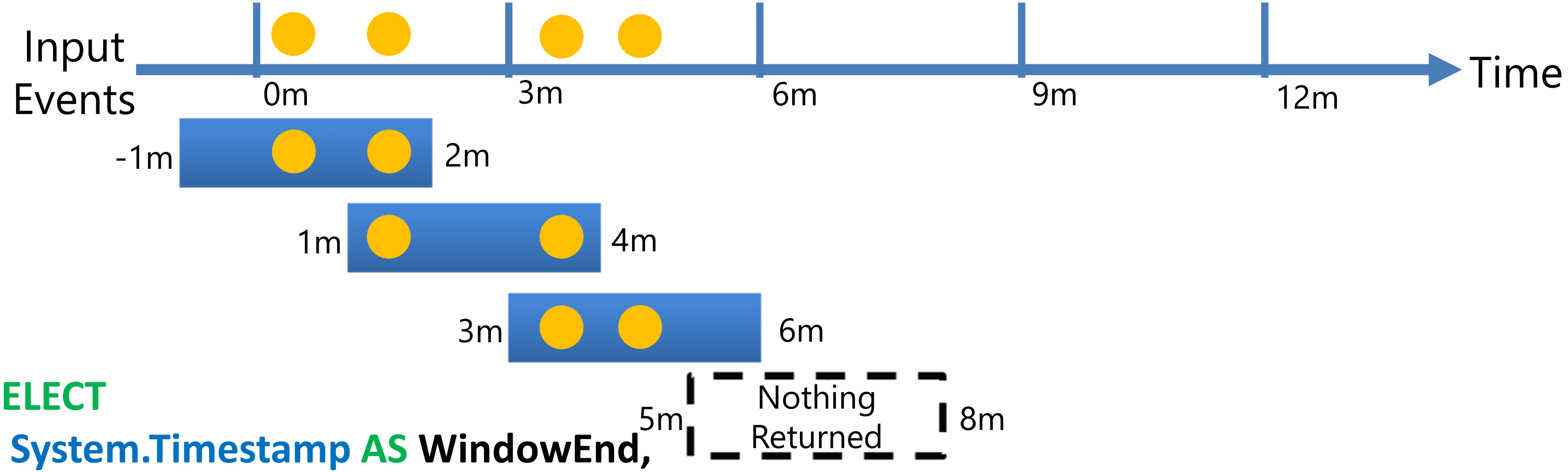
How many transactions were made for each card type every 3 minute?



```
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
```

datasciencedojo
data science for everyone

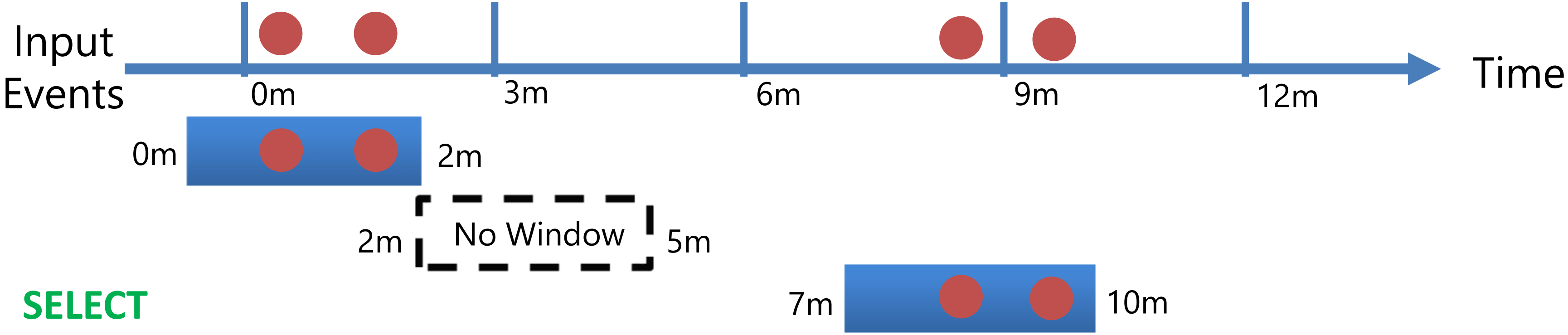
Hopping Window



```
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
```

datasciencedojo
data science for everyone

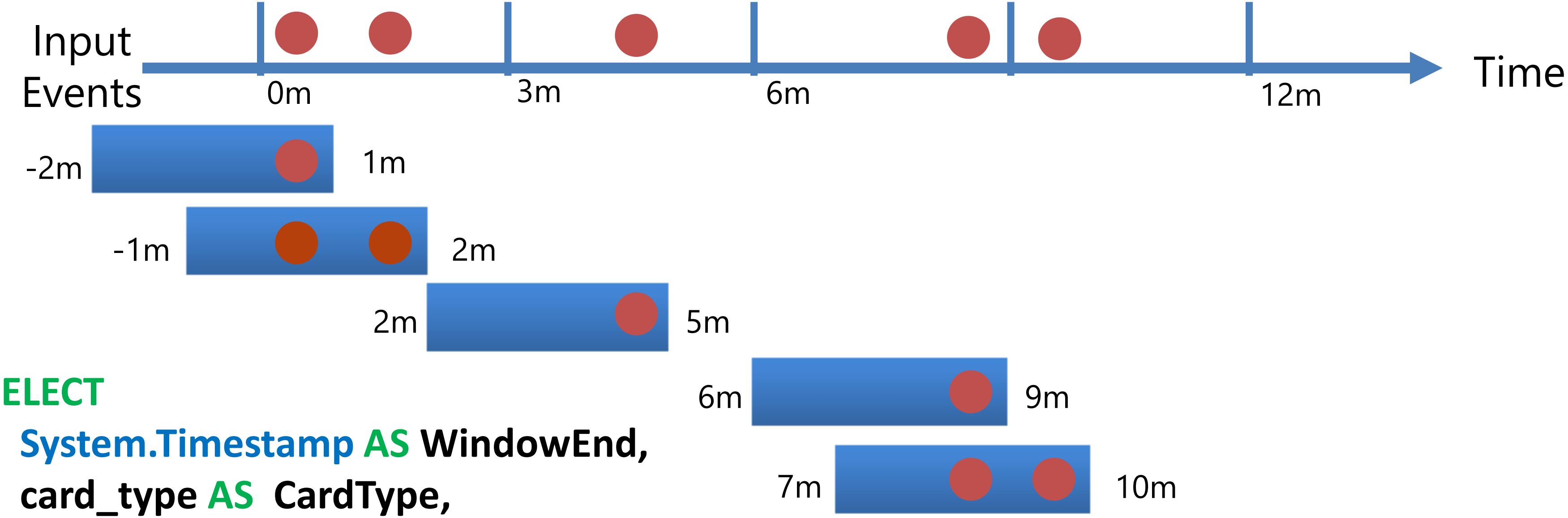
Sliding Window



```
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
```

datasciencedojo
data science for everyone

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
```

datasciencedojo
data science for everyone

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 SwipeStreamTIMESTAMP 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,
       StoleList.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?

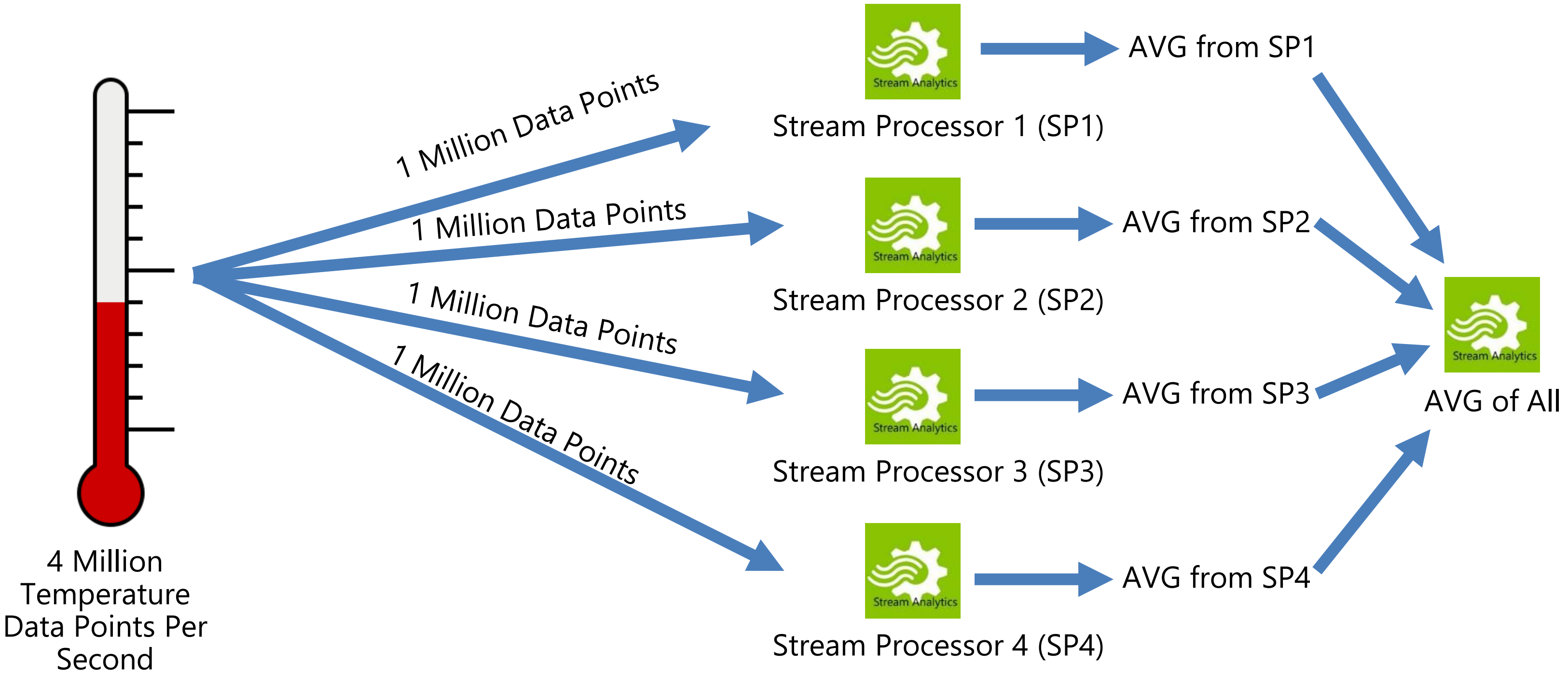
```
SELECT
    swipe.transaction_id
    swipe.swipe_date,
    bank.approval_time,
    DATEDIFF ( second, swipe.swipe_date, bank. approval_time) AS DurationInSeconds
FROM SwipeStream AS swipe TIMESTAMP BY swipe_date
JOIN BankStream AS bank TIMESTAMP BY approval_time
ON (swipe.transaction_id = bank.transaction_id)
    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



4 Million Temperature Data Points Per Second

1 Million Data Points

1 Million Data Points

1 Million Data Points

1 Million Data Points

Stream Processor 1 (SP1)

Stream Processor 2 (SP2)

Stream Processor 3 (SP3)

Stream Processor 4 (SP4)

AVG from SP1

AVG from SP2

AVG from SP3

AVG from SP4

AVG of All

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

QUESTIONS

datasciencedojo
data science for everyone