ETL and Analysis applied to Meetup

Ideas that cities and communities across the US curious about...

Karin Brodd
Chandler McCann
Natarajan Shankar
Dan Watson

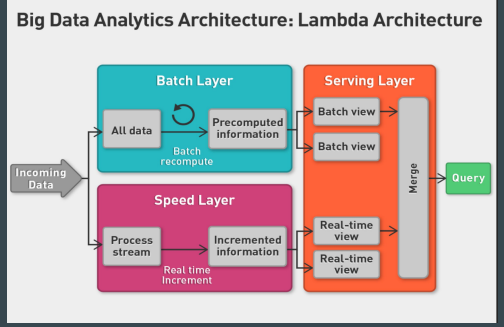
Project Scope

- Apply the ETL concepts learned in this class
  - Pick a provider of a live data stream
  - Establish an architecture for the end to data flow and implement
  - Refine the chosen architecture through issues encountered
  - With a working implementation:
    - Extract information about the curiosities and interests in geographical regions and subgroups (of practical scope)
      - Which communities are relatively more active throughout the US?
      - Are similar groups more active in one geographic region than in another?
      - What possibly accounts for these differences?
- Project the architecture and implementation towards a future form!

Ingestion of live data stream - Data Source : Meetup.com

- Why meetup.com?
  - Real data source, streamed
  - Meetup topics reflect public inclination and sentiment
  - Measure of frequency of similar meetings across the nation will augment and outperform national polls
  - Meetup.com offers diverse API
    - Data diversity via events and get_events API
  - Meets velocity requirements, many hundred JSON records per query
- Data Ingestion Layer
  - Preprocessing
    - Segment the incoming stream into individual JSON records
    - Built-in filtering: Write only the clean records to Database
      - Dirty records are dropped prior to database write

Initial proposed architecture... based on Classic Lambda



API Call 1 (Events)

API Call 2 (groups)

API Call n (tbd)

All data

Batch Layer

All data (Amazon EBS, Hadoop)

Apply schema on write

NoSQL Database (Amazon EBS)

Serving Layer (Apache Spark)

Table update

Table update

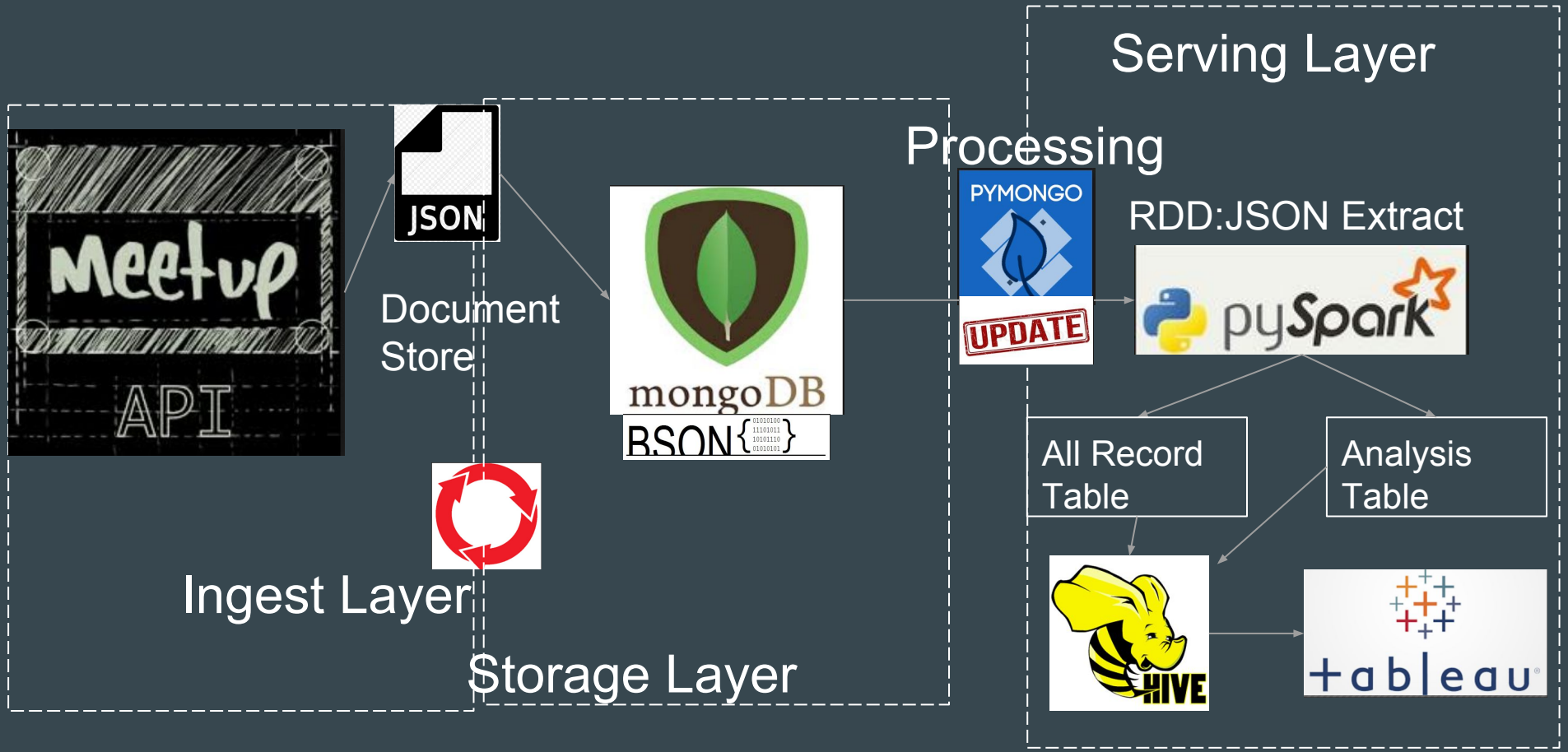
Table update

Merge

Data Viz (Tableau)

Query & Data Viz

Refined Architecture and Data Flow



Tradeoffs, Design Choices, Rationale

- **Limit the number of spout types**
  - Design for multiple types of data spouts but focus on meetup.com get_event()
- **Use in stream schema application and data cleanup**
  - Chosen data spout provides well formatted data
  - Schema on Write is not necessary, JSON is already well formed
  - Drop bad JSON in stream
- **Choose MongoDB over S3**
  - Need a self managed DIY app rather than a managed service
  - Couple the DB very closely with the API process
  - Document store in Mongo well suited for JSON
- **Write persistent Hive tables from Pyspark**
  - Data manipulation in Pyspark with data frames, has long term machine learning abilities
  - Native hive context in Spark allows easy save of Hive tables
  - Easy access to Hive from Tableau
    - Trouble getting the spark SQL thrift server to work

The long road from Mongo to Spark : Advanced Analytics, Complex Computation

Versatile

Cumbersome lynch pin

PYMONGO_SPARK
on top of
Mongo-Hadoop

Convenient


```
import pymongo_spark
pymongo_spark.activate()
```

- MongoDB Conn
- mongoRDD(conn)
- JSON Extract
- DF manipulation
- Scale

Easily connect to a mongoDB with a mongoRDD

```
conn = "mongodb://meetup_user:%40B%24%40rp1Dy%25C3@ec2-54-144-14-181.c
rdd = sc.mongoRDD(conn)
2/04 02:43:47 WARN SampleSplitter: Not enough documents for more than
new_rdd = rdd.map(lambda x: dict([(i, x[i]) for i in x if i != '_id'])
                ).map(lambda x: json.dumps(x, ensure_ascii=False).encc
                ).map(lambda x: "".join(x.split("\\n"))
)
df = sqlContext.jsonRDD(new_rdd)
```

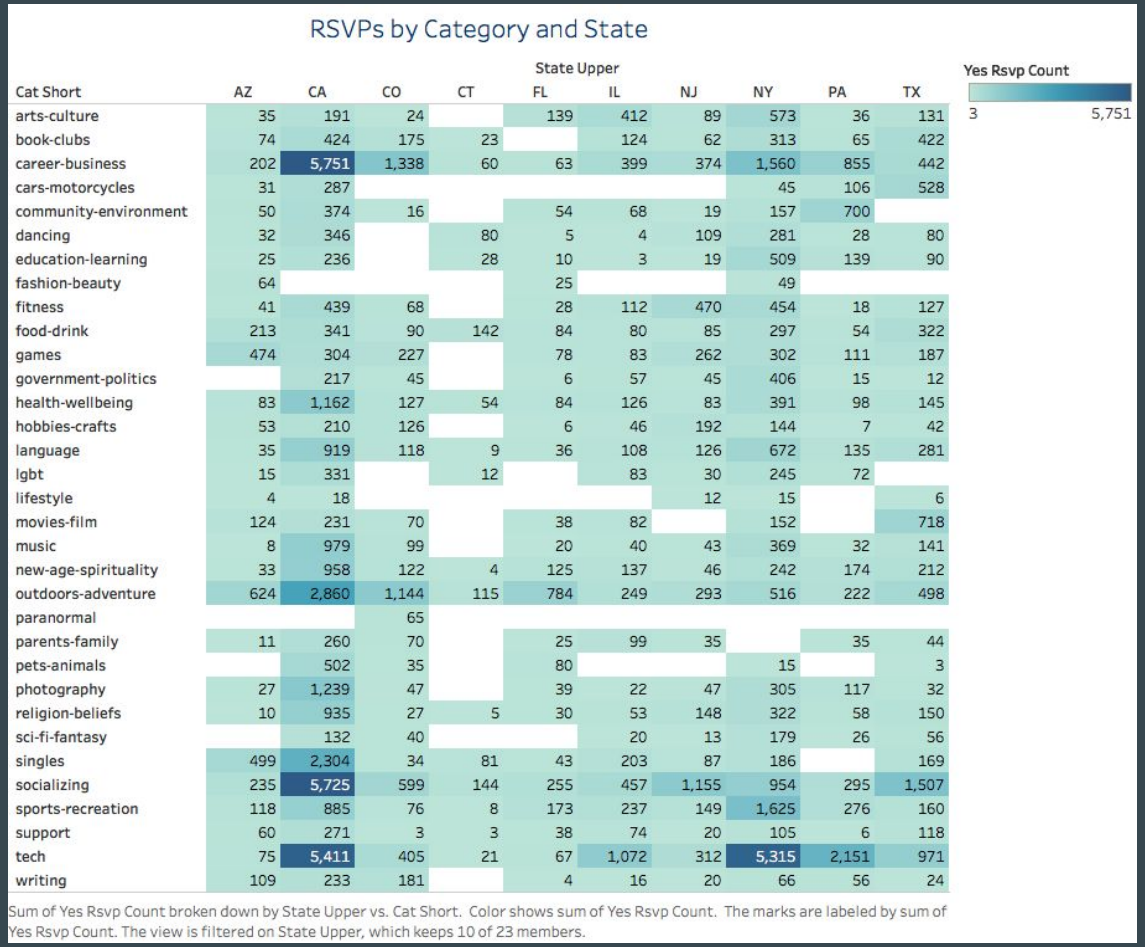
Schema automatically inferred from JSON

```
>>> df.printSchema()
root
 |-- created: long (nullable = true)
 |-- description: string (nullable = true
 |-- distance: double (nullable = true)
 |-- duration: long (nullable = true)
 |-- event_url: string (nullable = true)
 |-- fee: struct (nullable = true)
 |    |-- accepts: string (nullable = tru
 |    |-- amount: double (nullable = true
 |    |-- currency: string (nullable = tr
 |    |-- description: string (nullable =
 |    |-- label: string (nullable = true)
 |    |-- required: string (nullable = tr
 |-- group: struct (nullable = true)
```

Create columns from nested JSON fields easily

```
sqlContext.sql("CREATE TABLE groupCity AS SELECT group.name, venue.city FROM events_temp")
```

Sample Data Table - Visualization via Tableau



RSVPs by Category and State

State Upper

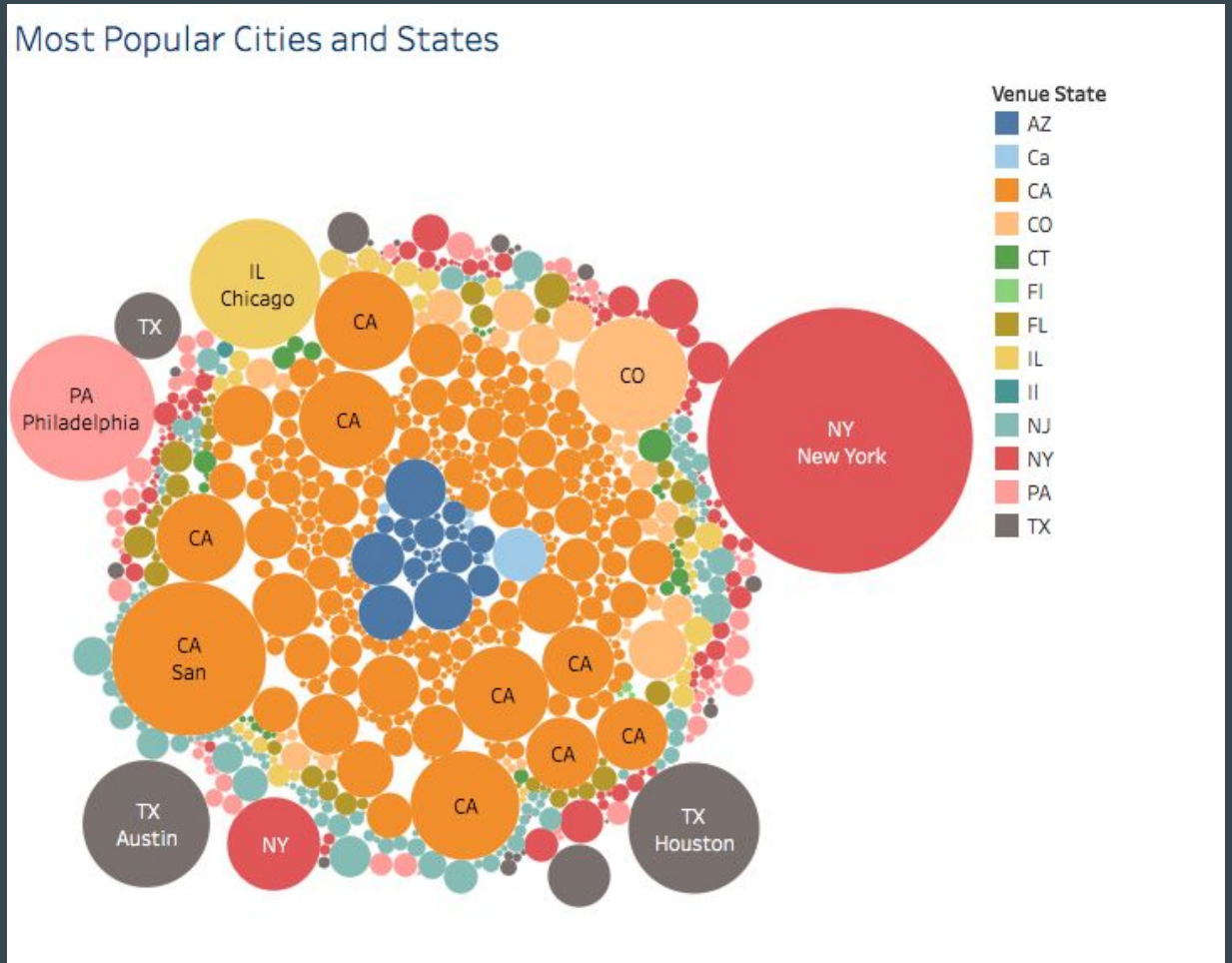
Cat Short	AZ	CA	CO	CT	FL	IL	NJ	NY	PA	TX
arts-culture	35	191	24		139	412	89	573	36	131
book-clubs	74	424	175	23		124	62	313	65	422
career-business	202	5,751	1,338	60	63	399	374	1,560	855	442
cars-motorcycles	31	287						45	106	528
community-environment	50	374	16		54	68	19	157	700	
dancing	32	346		80	5	4	109	281	28	80
education-learning	25	236		28	10	3	19	509	139	90
fashion-beauty	64				25			49		
fitness	41	439	68		28	112	470	454	18	127
food-drink	213	341	90	142	84	80	85	297	54	322
games	474	304	227		78	83	262	302	111	187
government-politics		217	45		6	57	45	406	15	12
health-wellbeing	83	1,162	127	54	84	126	83	391	98	145
hobbies-crafts	53	210	126		6	46	192	144	7	42
language	35	919	118	9	36	108	126	672	135	281
lgbt	15	331		12		83	30	245	72	
lifestyle	4	18					12	15		6
movies-film	124	231	70		38	82		152		718
music	8	979	99		20	40	43	369	32	141
new-age-spirituality	33	958	122	4	125	137	46	242	174	212
outdoors-adventure	624	2,860	1,144	115	784	249	293	516	222	498
paranormal			65							
parents-family	11	260	70		25	99	35		35	44
pets-animals		502	35		80			15		3
photography	27	1,239	47		39	22	47	305	117	32
religion-beliefs	10	935	27	5	30	53	148	322	58	150
sci-fi-fantasy		132	40			20	13	179	26	56
singles	499	2,304	34	81	43	203	87	186		169
socializing	235	5,725	599	144	255	457	1,155	954	295	1,507
sports-recreation	118	885	76	8	173	237	149	1,625	276	160
support	60	271	3	3	38	74	20	105	6	118
tech	75	5,411	405	21	67	1,072	312	5,315	2,151	971
writing	109	233	181		4	16	20	66	56	24

Yes Rsvp Count

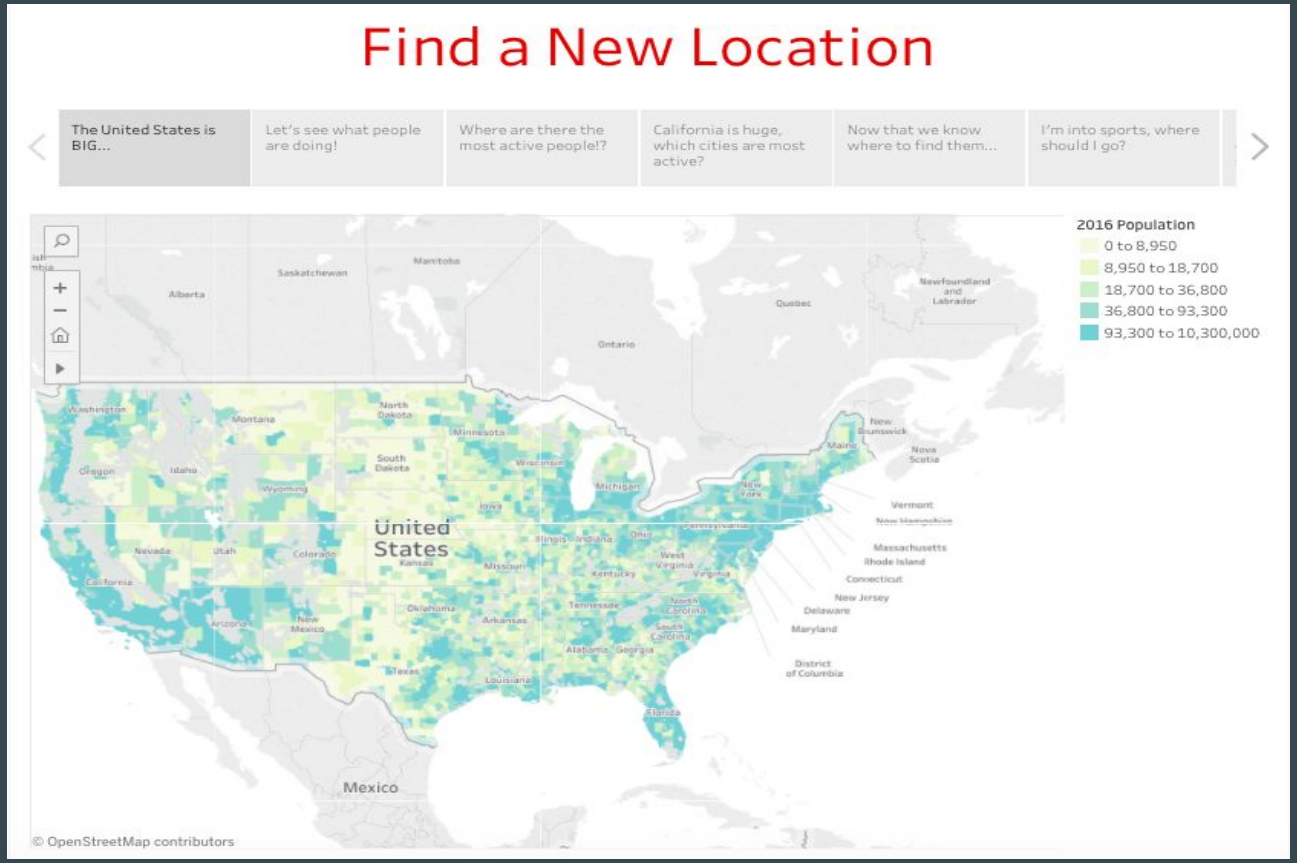
3 5,751

Sum of Yes Rsvp Count broken down by State Upper vs. Cat Short. Color shows sum of Yes Rsvp Count. The marks are labeled by sum of Yes Rsvp Count. The view is filtered on State Upper, which keeps 10 of 23 members.

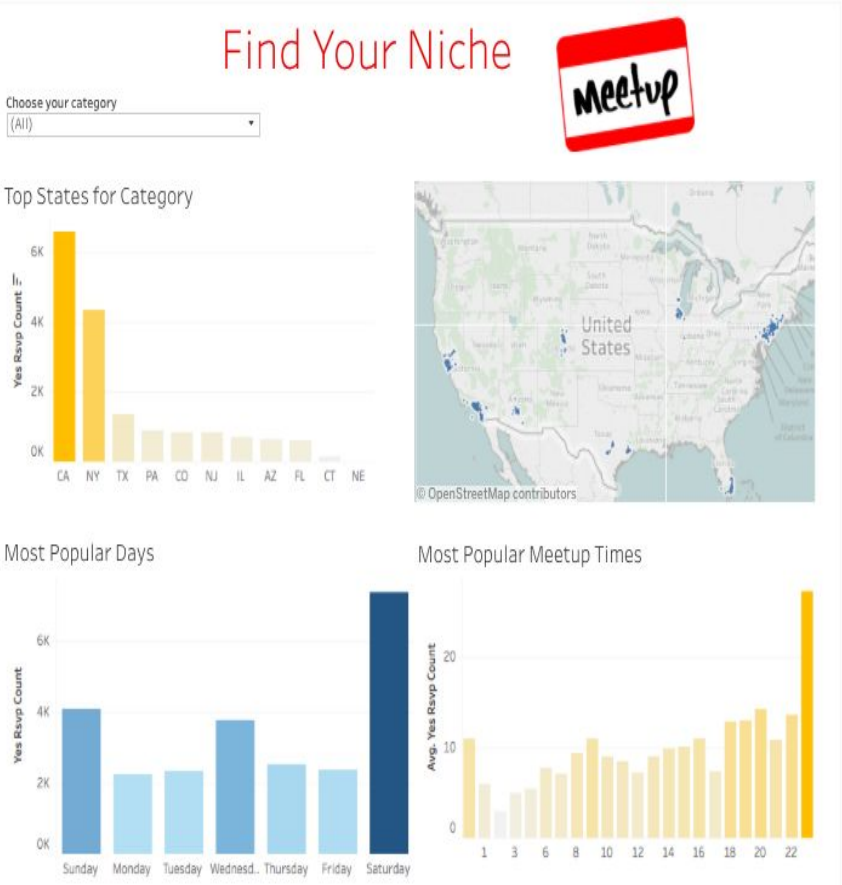
Sample Data Viz - Tableau

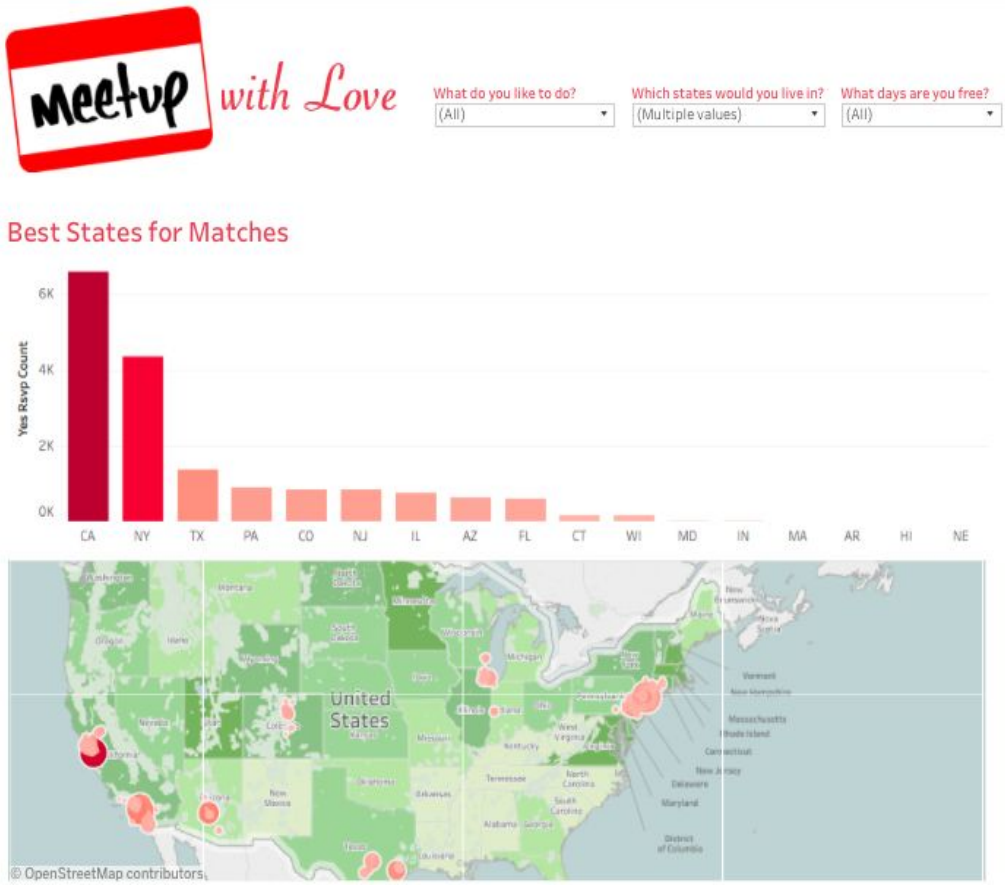


Output: Stories



Output: Dashboards





Complexity and Storage Needs

- Complexity
  - Project complexity is moderated by design
  - Spouts are well behaved, present well formed data
  - Data velocity is throttled by API throttling
    - System is designed to handle higher velocity
  - Database kept co-located with Ingest, provides managed latency
    - Overcomes inability to extract data over relatively slow internet
  - Hive and Map Reduce well understood but not pushed to limit
    - Hive latency issues due to the map reduce framework not magnified
- Storage
  - Tested at order of 86Mb/hour, can be scaled by 100x
  - To meet higher scale, Ingest layer may need distributed Storage architecture
    - Distributed storage provides capability to handling diverse data sets
    - Can push magnification of I/O rate by 100x

Challenges and Remedies

- API
  - Meetup api is limited in scope. For instance, querying by epoch is not supported!
  - Some required fields needs special parameter tags and are expensive to extract
  - API requests are limited to 200 per hour, need to throttle rate
- Processing
  - Multiple points of interface: PyMongo/ MongoHadoop/ PyMongoSpark/ Spark interface, many points of data exchange
- Front End
  - Despite the processing, still have minor issues with redundancy
    - Solution - opt for stricter constraints
  - Timezones
    - Known problem with all programming
    - ST solution- convert time from epoch to date-time then adjust by state
    - LT Solution- create tables for offsets based upon zipcodes and recognition of

Future Product Improvements

- Extend filtering and grouping algorithm for Precise Calculations
- Support Deeper Insights via:
  - Group analysis
  - Keyword Search
  - Domain knowledge
- Seek out and group Demographic Information
  - Focus on individuals, not groups
  - Pull information from profiles and connect to attended events
- Refine Ease of Use
  - Even with more robust information, we have a business tool
  - Need to make user friendly with controlled/verified output
- Extend application capability
  - Extract public sentiment

Scaling : Now and Future

- Support multiple data spouts, each with own schema
- Extend I/O Scaling: Obtaining and saving data
  - Scale the Database architecture, possibly distributed
  - Storage buffer in Ingest layer
- API
  - Extend API calls for more subfield level queries
- Analysis
  - Extend to predictive and prescriptive conclusions
- Extend towards capability for custom output
    - GUI output for casual applications
    - CDN for advanced users

Did Project meet set goals?

- ETL infrastructure for Meetup data
- Summarization of data across 12 large metropolitan areas
- Summarization of recent data
    - What activities are communities throughout the United States excited about?
    - Most prominent meetings in the US in December 2016
    - Does how people connect through activity differ by region?