EMR, Apache Zeppelin, and Jupyter: A Comprehensive Integration for Big Data Visualization.

Surya Pranav Sukumaran

What is Big Data Visualization?

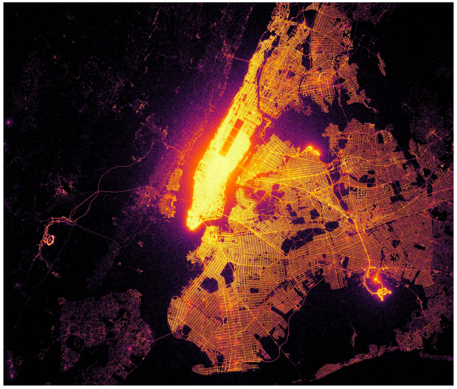
The process of representing large volumes of data in a graphical format.

Helps simplify complex datasets, revealing patterns, trends, and insights.

The dataset that is used for visualization is the NYC Taxi data that has over 5 million trips

How do we approach visualization of really large scale data?

- We will explore some methods to get started with big data visualization and look at potential upgrades when it comes to scalability



Challenges

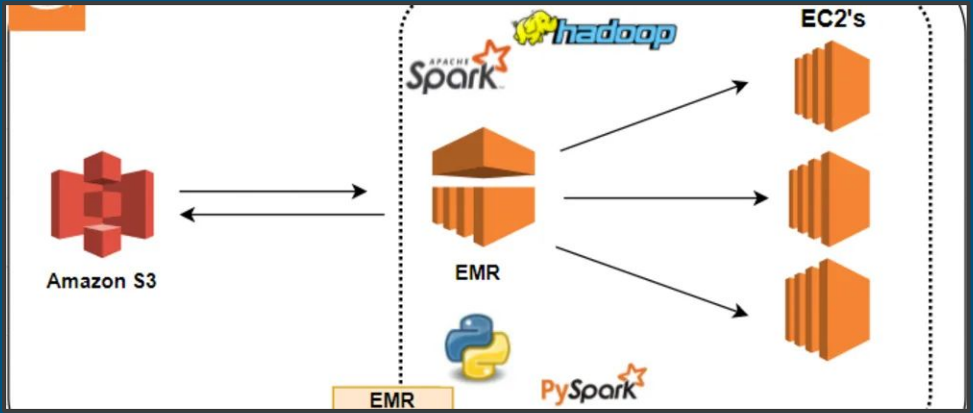
Volume: The sheer size of big data can be overwhelming; choosing what to visualize is crucial.

Variety: Diverse data types and sources require varied visualization techniques.

Velocity: The rapid rate at which big data is generated requires real-time or near-real-time visualizations.

Integrity: Visualizations can be misleading if data quality or representation isn't accurate.

Technologies Used



AWS EMR (Elastic Map Reduce)

Cloud-native big data platform that allows processing of vast amounts of data across resizable clusters of Amazon EC2 instances using popular distributed frameworks such as Hadoop and Spark

Apache Zeppelin

Jupyter Notebook

Apache Zeppelin

Open-source, web-based notebook that enables data-driven, interactive data analytics, and visualization for large datasets

- Run SQL Queries!
- Direct database connections (Ex. MongoDB, Postgres)

Pros:

Native big data integrations (EMR, Spark)

Built-in visualizations & interactivity.

Supports multiple languages in one notebook.

Role-based access & collaboration tools.

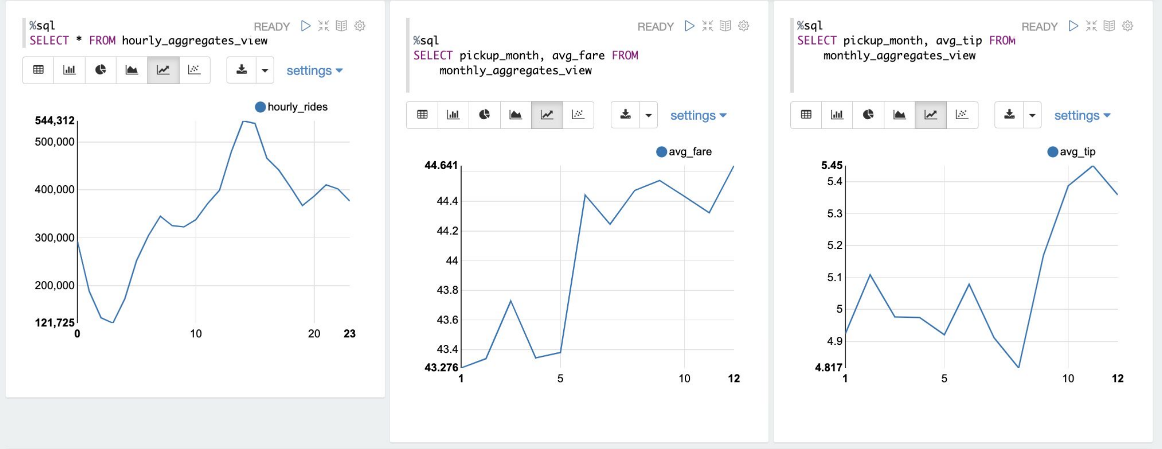
Cons:

Smaller ecosystem & community.

Limited support for some Python libraries.

Less intuitive UI for some users.

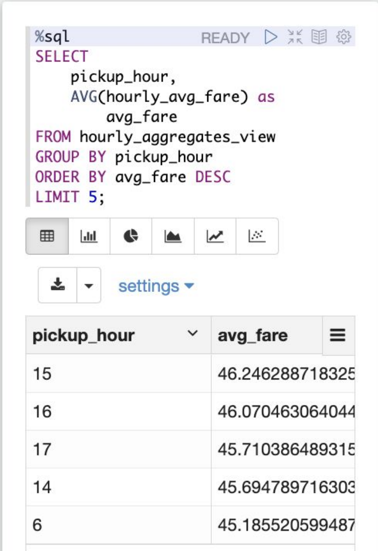
Each cell is known as a Paragraph and can be customized to present data and visualize in a seamless way



```
%sql
SELECT * FROM hourly_aggregates_view
```

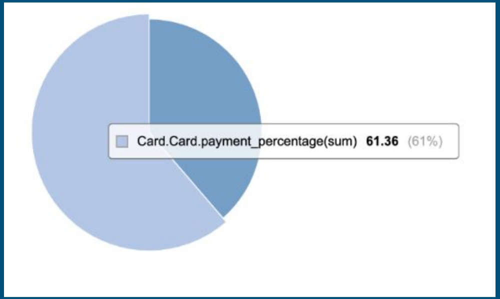
```
%sql
SELECT pickup_month, avg_fare FROM
    monthly_aggregates_view
```

```
%sql
SELECT pickup_month, avg_tip FROM
    monthly_aggregates_view
```



```
%sql
SELECT
    pickup_hour,
    AVG(hourly_avg_fare) as
        avg_fare
FROM hourly_aggregates_view
GROUP BY pickup_hour
ORDER BY avg_fare DESC
LIMIT 5;
```

pickup_hour	avg_fare
15	46.246288718325
16	46.070463064044
17	45.710386489315
14	45.694789716303
6	45.185520599487



The combination of Zeppelin's built-in visualizations, its integration with big data tools, and its interactive notebook format makes it a powerful tool for data scientists and analysts working in the big data ecosystem

Jupyter Notebook

Open-source web application that allows the creation, sharing, and execution of documents containing live code, visualizations, and narrative text.

Offers a unique blend of code, text, and visual output in a single, scrollable document.

Useful for data scientists and analysts looking to work with data

Pros:

Vast community & rich ecosystem.

Extensive library support.

Flexibility with custom widgets.

Supports various kernels & languages.

Cons:

Requires extensions for big data tools.

No built-in real-time collaboration.

Less native interactivity than Zeppelin.

Interpreter vs. Kernel:

Zeppelin: Uses interpreters, which allow for a shared context across different paragraphs (code blocks) within a notebook. This can be advantageous when loading and processing big data as data and configurations can be reused across paragraphs.

Jupyter: Utilizes kernels for each notebook, which means each notebook has its isolated environment. This isolation can sometimes mean reloading data or configurations for different notebooks.

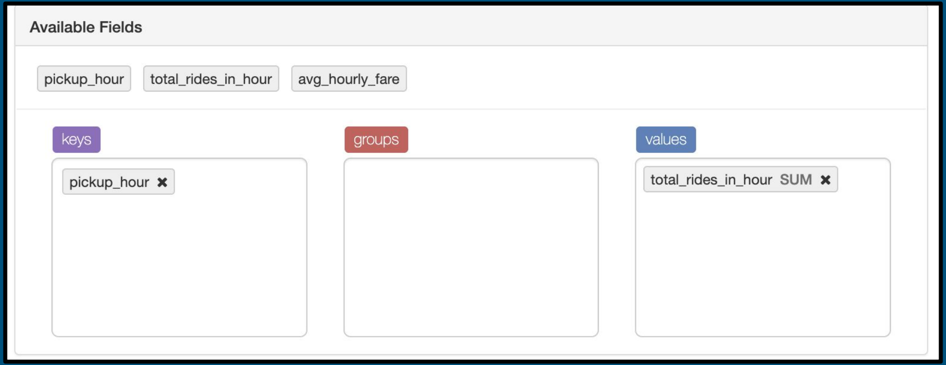
Memory Management:

Both Zeppelin and Jupyter, when integrated with Spark on EMR, leverage Spark's memory management and distributed processing capabilities. However, the way they handle Spark sessions might differ. Zeppelin's shared context can be more efficient with resource utilization, especially when working with multiple related tasks within the same notebook.

Visualization Pipeline

1. EMR Cluster Setup
2. Launch Apache Zeppelin Notebook for data ingestion
3. Preprocessing using pyspark or spark
4. Aggregate data and use SQL queries to visualize and gain initial insight
5. Save the aggregated data as parquet files
6. Load the dataset in Jupyter for more customizable visualizations using Plotly, Folium

SQL



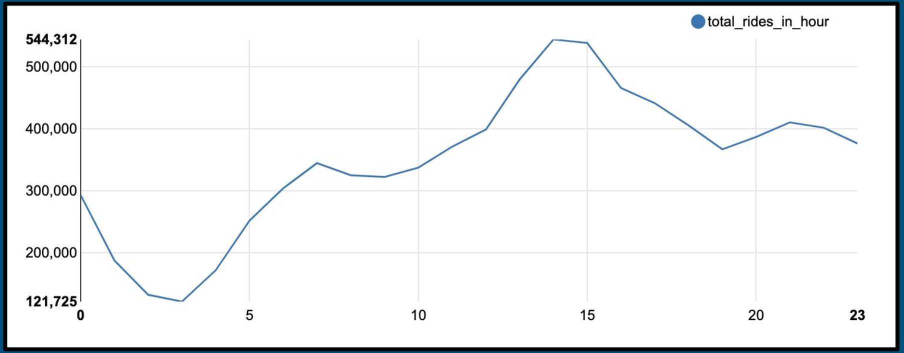
Available Fields

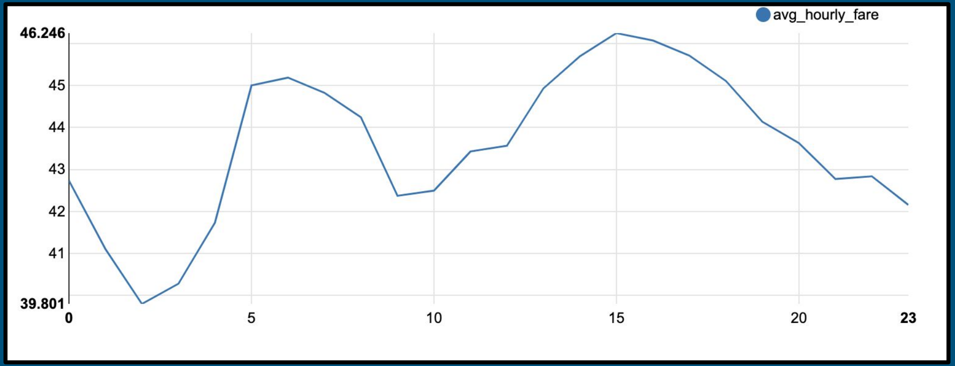
pickup_hour total_rides_in_hour avg_hourly_fare

keys: pickup_hour ✖

groups:

values: total_rides_in_hour SUM ✖

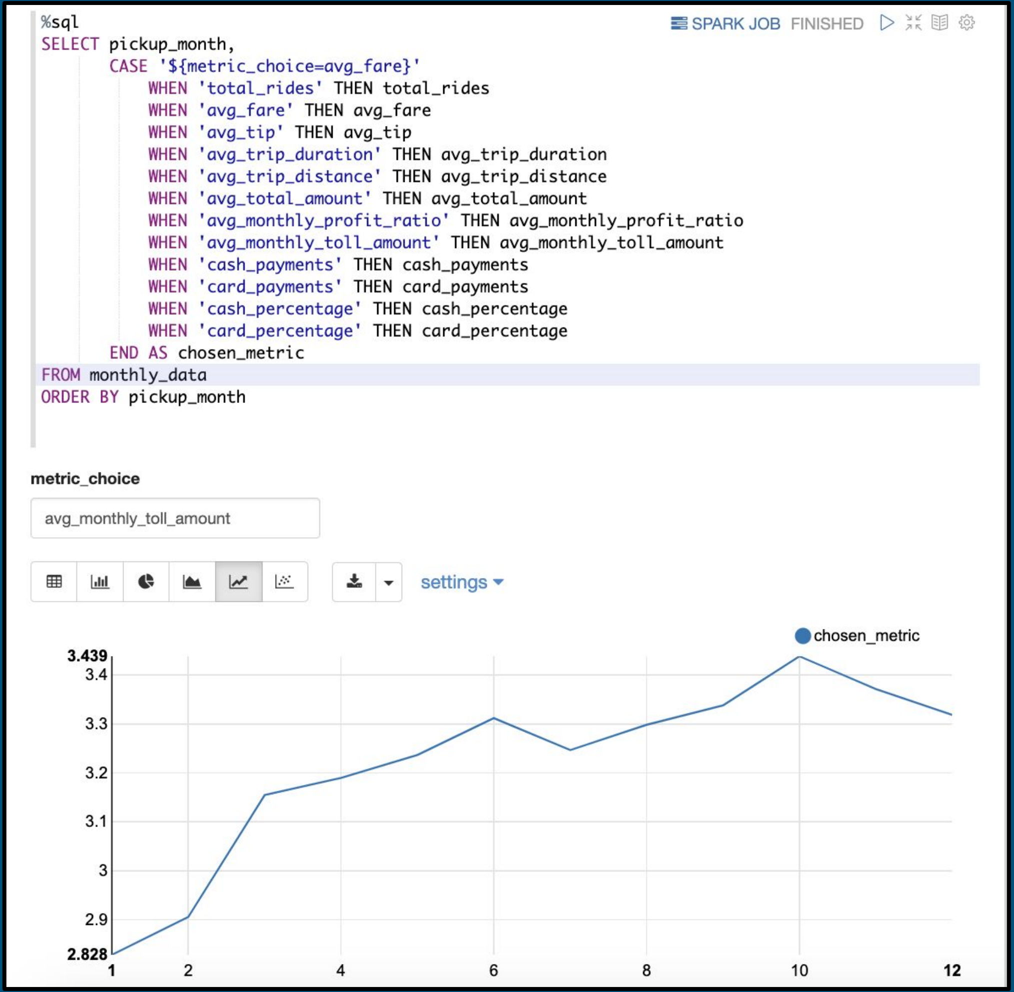




Dynamic Forms

Allow users to interactively change the parameters of their analyses without having to modify the code itself.

Instead of creating separate visualizations or tables for each metric (like average fare, total rides, etc.), we used Zeppelin's dynamic forms feature to create an input metric box.

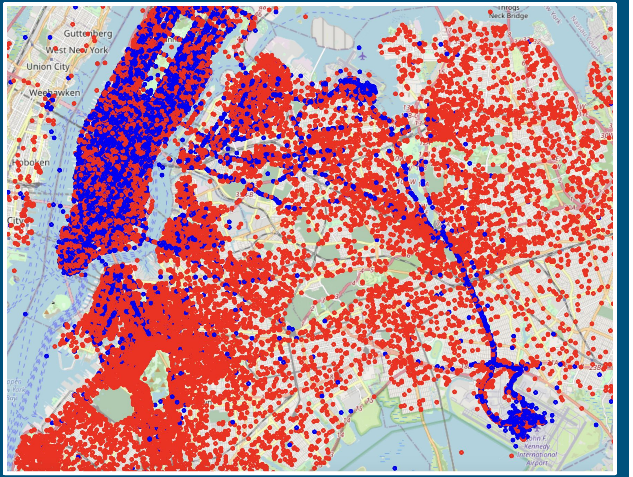


```
%sql
SELECT pickup_month,
       CASE '${metric_choice=avg_fare}'
            WHEN 'total_rides' THEN total_rides
            WHEN 'avg_fare' THEN avg_fare
            WHEN 'avg_tip' THEN avg_tip
            WHEN 'avg_trip_duration' THEN avg_trip_duration
            WHEN 'avg_trip_distance' THEN avg_trip_distance
            WHEN 'avg_total_amount' THEN avg_total_amount
            WHEN 'avg_monthly_profit_ratio' THEN avg_monthly_profit_ratio
            WHEN 'avg_monthly_toll_amount' THEN avg_monthly_toll_amount
            WHEN 'cash_payments' THEN cash_payments
            WHEN 'card_payments' THEN card_payments
            WHEN 'cash_percentage' THEN cash_percentage
            WHEN 'card_percentage' THEN card_percentage
       END AS chosen_metric
FROM monthly_data
ORDER BY pickup_month
```

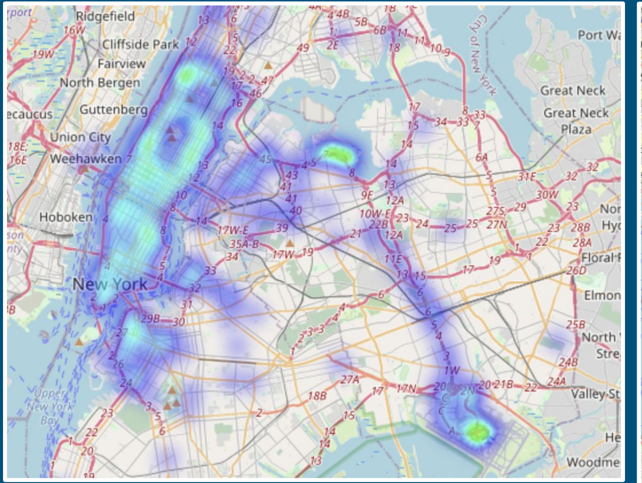
metric_choice

avg_monthly_toll_amount

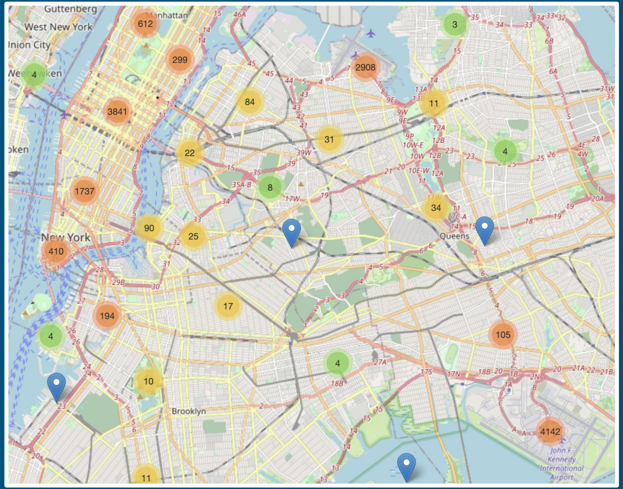
What’s Next? Geospatial Visualization using Folium



Pick Up Drop Off Points



Heat Map of Pick Up Zones



Cluster Map of Pick Up and Drop off

What about Plotly and Dash running on Jupyter Notebook?

Thank you!

I hope you have learned something new, and will be motivated to use Apache Zeppelin when working with large datasets next time!