

Bike Sharing

Courtesy of MongoDb

Baldip Singh | Big Data Analysis | 16/09/2021

Table of Contents

[Executive summary 2](#_Toc82688610)

[Introduction 3](#_Toc82688611)

[The Data 4](#_Toc82688612)

[Data Understanding 4](#_Toc82688613)

[Data Pre-processing 6](#_Toc82688614)

[Data Model, Structure, and Implementation 7](#_Toc82688615)

[Data manipulation 9](#_Toc82688616)

[Reporting 10](#_Toc82688617)

[Queries – Baldip Singh 10](#_Toc82688618)

[Query 1 11](#_Toc82688619)

[Query 2 13](#_Toc82688620)

[Queries – Shetu Patel 16](#_Toc82688621)

[Reflections 17](#_Toc82688622)

[Encoding 17](#_Toc82688623)

[OBDC connectors 17](#_Toc82688624)

[References 18](#_Toc82688625)

[Appendices 21](#_Toc82688626)

[MongoDB 21](#_Toc82688627)

[MongoDB - Shell 21](#_Toc82688628)

[MongoDB – Charts 27](#_Toc82688629)

[RStudio 28](#_Toc82688630)

# Executive summary

This report consists detailed information regarding the transforming and loading of Bike Sharing Data. Also included are the various queries performed on the data set utilising various components of the NoSQL database platform MongoDB. When necessary, please refer to the appendices for all relevant coding.

# Introduction

On Cycle is a Bike sharing company which automate its rental and sign-up services using its automated IT system. Various systems like these exist across the world such as Next Bike in New Zealand. A bike is fitted out with components which communicate through to an application typically based on smart phones where users can rent out the bike for a designated time. These initiatives are based on principles designed to ease pressures on local infrastructure (roads and public transport systems), leading to an environmentally friendly mode of transport.

As analysts we have received metrics from these bikes for each hour of the day along with attributes such as the hour number, number of riders, day of the week etc. (explained ahead in further detail) for two full years (2011 and 2012). This flow of data is managed using a NoSQL platform – MongoDB with its excellent real time cloud-based technologies. Specifically, four components of MongoDB are being utilised in the processing, analysing and visualisation of this data.

* **MongoDB Atlas** - forming the cloud-based server to operate on via clustering and setting up data security protocols
* **MongoDB Shell** - allowing the transformation of data before it can be analysed
* **MongoDB Compass** - offering a user-friendly interface for uploading data and forming aggregation pipelines and finally
* **MongoDB Charts** - for more in-depth visualisation and dashboards

Logo, company name

Description automatically generated

A picture containing text

Description automatically generated

Graphical user interface, application

Description automatically generated

Logo, company name

Description automatically generated

# The Data

MongoDB Atlas was used to create a cluster on which to store collections and databases. While creating this cluster security protocols were set to who can access the data, this was set by whitelisting the specific IP addresses of the potential users. For the purposes of this assignment security was kept minimal and only one account was created with administrative rights which could read and write across all databases.

## Data Understanding

Supplied alongside the data set was the relevant documentation in the form of a .txt file breaking down each attribute and its meaning, also on the UCI [website](https://archive.ics.uci.edu/ml/datasets/bike+sharing+dataset) a breakdown was visible, shown below.

Text

Description automatically generated

Each row of data represents the hour of the specific day and its corresponding environmental readings, and various time attributes. Also included was the count of riders split into the types, Casual Riders and Registered Riders, a third count also existed Total Rider’s which is an aggregation of the two for that specific hour. The data is of two full years 2011 and 2012. The attributes can be broken into categories shown in the table below.

|  |  |  |
| --- | --- | --- |
| Time | Environmental | Users |
| Date | Normalised ambient Temperature (c) | Casual rider count |
| Year | Normalised felt Temperature (c) | Registered rider count |
| Month | Normalised Humidity | Total Rider count |
| Season | Normalised Windspeed |  |
| Day of the Week | Weather type |  |
| If Day is weekend or not |  |  |
| If day is a public holiday or not |  |  |

It is to note most of these time attributes have been encoded and will need to be processed for real values. Several Environmental metrics have been pre-processed as they have been normalised. These readings along with naming conventions are processed via the MongoDB – Shell.

## Data Pre-processing

The data is of ~17,000 rows which was a considerable size however if we investigate how this data was produced, we understand that this data set has been processed already to a certain degree. E.g., every time a bike was rented out the status of the user along with the hour (and other metrics mentioned) most likely would have been communicated back to the database, as there are multiple bikes across town each hour would have multiple entries for that day. However, this is not the case which leads me to believe at some stage the data has been aggregated on hour to produce the total number of riders for that specific hour on the day. Seeing as we do not have each entry but totals for specific hours, we can assume a certain level of pre-processing has occurred. However, for the purposes of this assignment this is not an issue and further processing is required in the form of renaming columns and renaming field values.

For all updating of fields and attribute names the MongoDB Shell extension available for Visual Studio Code was utilised offering an aesthetically pleasing Shell experience with its clear syntax highlighting. It was to note that multiple column names were not easily understood without documentation and for the purposes of this analysis were processed to CamelCase.

For the purposes of this assignment all key encoded attributes were replaced with actual values (e.g., Month: 1 = January) for clearer understanding, it is to note however these encoded fields can be useful for building predictive models however this was outside the scope of this assignment and therefore deemed unnecessary to keep two columns.

Weather was also numerically categorised, and generalisations were made within the categories as the categories covered various keywords for example:

1. Clear, few clouds, partly cloudy, partly cloudy Altered to “Little Cloudy”
2. Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist to “Misty/Cloudy”
3. Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds to “Light Rain/Cloudy/Thunder” and
4. Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog to “Heavy Rain/Thunder/hail”

Column Index was dropped as it was only a record index and by default MongoDB has a record index.

## Data Model, Structure, and Implementation

MongoDB works on an array-based system where each row is considered an element within the array which in this case is the table/database. Typically, elements within array’s work in pair values, each element has an index (the number at which the element occurs in the array) and the value (the information this element holds). Within MongoDB – The element is referred to as a document and the index is the autogenerated ObjectId.

Once a .csv file is uploaded using MongoDB – Compass, options such as changing the data type for columns exist which are best applied at the time of upload as changing these data types can be tedious via Shell simply due to the time it can take if there are an excessive number of attributes. Once the data types are determined and uploaded, MongoDB automatically converts each row into a document with a corresponding object Id\_.

Text

Description automatically generated with medium confidenceGraphical user interface, text, application

Description automatically generatedAs shown in the data attribute table we noticed various environmental (temps, windspeed, humidity) and time (year, date, month, weekday) based attributes. A decision was made to create nested documents within each document to provide a more efficient experience when using the MongoDB – Compass aggregation pipelines. These attributes can be accessed via dot notation when querying and when wanting to view in the documents section simply click the arrow beside the arrays creating a drop down of the relevant metrics.

Figure 2- How the Documents are displayed now

Figure 1- How the Documents were displayed Originally

Indexing can be used to help querying a database where fields have a range of names and searching of these names may be necessary. For example, if each bike was to have a name referring to the model, hypothetically could be

* “Mark 1 Supercharged”
* “Mark 1”
* “Supercharged legacy model” and
* “Legacy standard”

we can easily query for names using indexing by defining this name type to text. We can then use the find operator within MongoDB - Shell to then search all text type indexes to search for “Supercharged” which would produce the records “Mark 1 Supercharged” and “Supercharged legacy model”, essentially a search bar type functionality. This was not necessary for this data set as no real search type method was necessary and instead the $match operator was more than sufficient; therefore, no indexing was performed.

## Data manipulation

The temperature readings as mentioned were normalised, it was necessary to have the real temperature readings available as actual temperature values are much easier to relate to. To do this the normalised values were reverted to original values using the minimum and maximum values provided. Simple algebra was used to rearrange the normalisation formula.

Normalisation Formula:

Text

Description automatically generated

Formula rearranged to give true values:



It is to note that the real maximum and minimum values were only available for the temperature readings however for windspeed and humidity this was not the case and therefore obtaining the real values was not viable.

# Reporting

## Queries – Baldip Singh

As a Data Analyst – Baldip Singh was required to produce two ad-hoc reports asking two different questions.

1. When are bikes most likely to be used during the day? and
2. As it is coming to the end of year report and the boss would like the total number of riders for both registered and casual users for the first two years of operations (2011 and 2012).

Report on your findings and provide extra insight into these reports and suggestions where possible.

### Query 1

When are bikes most likely to be used during the day?

#### The Query

Below is essentially the same query done through three different components of MongoDB – Shell, Compass, and Charts. We are finding the sum or total riders, registered riders and casual riders for the specific hour and ordering in descending order.

Text

Description automatically generatedGraphical user interface, text, application

Description automatically generated

Graphical user interface, application

Description automatically generated

#### The Analysis

Both Compass and Shell provided the hours in ranking of highest total users in decreasing order based on hours. The output showed that the 17th hour of the day and 18th hour of the day had the highest total riders 336,860 and 309,772 respectively. However, running this same query through MongoDB – Charts provided an extra level of insight with the introduction of visualisations shown below.

Chart, bar chart

Description automatically generated

To answer the question, the hours with peak total users is the 17th and 18th hours however when viewing the count of casual rider’s vs registered riders throughout the day we see unique trends. Registered users are showing two peaks in total users (the 8th hour and 17th hour) whereas as casual riders are showing only one peak (around the 15th hour).

#### Suggestions

It would be best to investigate further into the demographic of the casual rider’s vs registered riders. At this stage it seems the peaks for registered riders align well with working hours i.e., there is a chance registered riders are utilising our services to get to and from work. However casual riders seem to peak through the working hours – investigation into this is necessary for a suitable marketing approach.

### Query 2

What are the total number of riders for both registered and casual users for the first two years of operations?

#### The Query

Again, below are essentially the same query done through the various components of MongoDB.

Graphical user interface, text, application

Description automatically generated

Text

Description automatically generated

Graphical user interface, application

Description automatically generated

#### The Analysis

To answer the question, the total riders for 2011 were 1,243,103 vs 2,049,576 for the year 2012 that is almost a doubling in total riders which is an awesome achievement and one the organisation should be proud of. We can also view he break down between Casual riders and Registered riders below.

#### Chart, bar chart Description automatically generated

The increase in riders is across the board for all rider types, both casual and registered riders showing a clear increase.

#### Suggestions

Invest as much as you can as there is real potential for this organisation to grow exponentially. Seasonal patterns are observed throughout the year, the Fall and Summer seasons should be considered for future marketing. Shown below is a side by side of the impact temperature and season have on riders.

The Breakdown of users for the relevant months (1 = January…) should also be considered. Casual riders peak from May through till September, pay special attention to these months when trying to convert casual riders -> registered riders.

## Queries – Shetu Patel

### Query 1

Query to show total riders ride the bike in spring season.

So, to perform the query as started with making a new connection in MongoDB and after that created a new database in it and then after created a new collection in the database in which we performed the following query. We added the new data which was updated and started analysing the data. In the aggregations tab we created a new pipeline for the query.

So, in the aggregation tab we selected our first stage for the query i.e., Project stage we need to specify which fields are to included and which fields are to exclude. So, for the query to show total riders in spring season we need Month, Season, Year and Total Riders to display the result of the query then in the next stage i.e., to match the particular season that we selected is spring and in the last stage that is to group all the selected items and display the result so in this we have to select id as Month so it will display a particular month then it will show the sum of total riders in spring season in a particular month.

As we can see below in the displayed image in the month of December in spring season there are total of 42998 riders whereas in Feb there are 151352 riders. So, in business means by performing this query at the end of the spring season company can analyse how much seasonal profit they made in a particular season in each month.

Graphical user interface, text, application

Description automatically generatedGraphical user interface, text, application

Description automatically generated

### Query 2

Query to show all the registered riders in summer season in each month.

So, to perform this query we already have our data loaded in MongoDB platform so for this we need to create a new pipeline in aggregations tab and analyze the data accordingly. Now in this query we need the result as registered riders ride the bike in summer season in each month so our first stage that is project, we will select all the required fields from the data we need Year, Month, Season and Registered Riders.

In the second stage of the query, we will match the season to summer as we need the result for summer season by selecting summer as a season it will only display the data about summer season and in the third stage that is our final stage, we will group the data to display the result. So, in group id will be month as it will show different months in which data will be displayed and the sum of registered riders to show how many riders were registered riders in each month of summer season.

As we can see in below in the displayed image in the month of April there are total of 208292 riders in summer season whereas in the month of May there are total of 256401 riders. So, in the business means this query helps the company to show all the registered riders in summer season in different months.

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

# Reflections

In terms of learning experience, this project not only allowed me to refine my reporting techniques but also allowed me to dive deeper into the NoSQL platform MongoDB. The experience was challenging however, I feel as though the skills obtained are highly translatable to other NoSQL platforms. In saying this, no project is perfect and there is always room for improvement, discussed below.

## Encoding

As mentioned, encoded attributes had the encoding replaced with the actual values and the encoded columns were not kept. This posed complications when trying to graph the monthly data in order of months as they occur. By default, MongoDB – Charts orders the x- axis either alphabetically or by ascending/descending order, this did complicate things however an awesome feature does exist where the chart data can be exported. This did require further work as the months had to be then encoded manually on MS Excel, luckily there are only 12 months in a year. Therefore, moving forward, it would be best to either keep two columns (one encoded, one with original) or in a real world setting sufficient domain knowledge would exist within the organisation to decipher that (1:January, 0:Sunday etc.).

In summary I found all encoding that relates back to cyclic events i.e., days, months, years etc. should be kept encoded for the sake of visualisations as only then trends can be observed chronologically.

## OBDC connectors

From the start I wanted to challenge myself and connect this database system to a platform purpose built for dashboarding and visualisations. Extensive time was spent trying to connect MongoDB directly to MS Tableau as I had tunnel visioned into the idea that only Tableau can create dashboards etc. I did get quite close however the final hurdle I could not overcome which was to figure out the server’s name on which our cluster was running. Later I found out this method would have been also unreliable as MongoDB does not guarantee the cluster will be operating on the same server consistently. Later I also found out MongoDB – has a visualisation component of its own MongoDB – Charts, therefore the entire process was unnecessary as there is an inbuilt component available for such tasks.

In summary my experiences proved that it is best to investigate further into the applications available within the suite you are operating in as the interconnectivity is unmatched.

# References

*MongoDB - Documentation was utilised extensively, for the sake of repetitiveness only the first page of manual is referenced*

*MongoDB. (2021). MongoDB - Documentation. Retrieved from https://docs.mongodb.com/manual/tutorial/getting-started/*

*Aggregation | MongoDB | Tutorial 10. (2017, December 22). [Video]. YouTube. https://www.youtube.com/watch?v=Kk6Er0c7srU&ab\_channel=MikeDane*

*Complete MongoDB Tutorial in 14 Minutes! (2020, February 14). [Video]. YouTube. https://www.youtube.com/watch?v=11ZJq1\_vrnE&t=771s&ab\_channel=KindsonTheTechPro*

*Create and Read operation in MongoDB. (2019, January 11). [Video]. YouTube. https://www.youtube.com/watch?v=MXIpMcMGRds&ab\_channel=HiteshChoudhary*

*Data Modeling with MongoDB. (2020a, June 9). [Video]. YouTube. https://www.youtube.com/watch?v=yuPjoC3jmPA&t=830s&ab\_channel=MongoDB*

*Data Modeling with MongoDB. (2020b, October 28). [Video]. YouTube. https://www.youtube.com/watch?v=3GHZd0zv170&t=489s&ab\_channel=MongoDB*

*HOW TO: Connect Your App To Cloud Database || MongoDB Atlas. (2020, October 7). [Video]. YouTube. https://www.youtube.com/watch?v=nwgQzuRRgec&t=1s&ab\_channel=MarinaKim*

*Intro to MongoDB Charts (Demo). (2020, July 1). [Video]. YouTube. https://www.youtube.com/watch?v=gJ85ixGCe8M&t=383s&ab\_channel=MongoDB*

*An Introduction to MongoDB Compass. (2015, November 2). [Video]. YouTube. https://www.youtube.com/watch?v=XfLeO\_LE6ic&t=29s&ab\_channel=MongoDB*

*Learn How to Update MongoDB Array Elements without Feeling Bulba-sore. (2021, March 9). [Video]. YouTube. https://www.youtube.com/watch?v=XRXjJRJ03\_A&t=371s&ab\_channel=MongoDB*

*MongoDB Charts. (2020, September 24). [Video]. YouTube. https://www.youtube.com/watch?v=ZW4FOGF5ENA&t=405s&ab\_channel=MongoDB*

*MongoDB Compass: Complete In-depth Tutorial [all features]. (2020, June 19). [Video]. YouTube. https://www.youtube.com/watch?v=ydXCcLAi5aU&t=689s&ab\_channel=CodeFirstwithHala*

*MongoDB Compass Tutorial - Aggregation. (2020, April 9). [Video]. YouTube. https://www.youtube.com/watch?v=IqBXKRrgy38&t=399s&ab\_channel=CocoZhu*

*MongoDB in 5 Minutes with Eliot Horowitz. (2018, February 13). [Video]. YouTube. https://www.youtube.com/watch?v=EE8ZTQxa0AM&t=70s&ab\_channel=MongoDB*

*MongoDB In 30 Minutes. (2016, September 4). [Video]. YouTube. https://www.youtube.com/watch?v=pWbMrx5rVBE&t=27s&ab\_channel=TraversyMedia*

*MongoDB In The Cloud With Atlas. (2018, December 3). [Video]. YouTube. https://www.youtube.com/watch?v=KKyag6t98g8&t=626s&ab\_channel=TraversyMedia*

*MongoDB Indexes - The Recipe behind Fast Query - How to Create Indexes and the B-Tree Data Structure. (2020, July 5). [Video]. YouTube. https://www.youtube.com/watch?v=IHQeDEn38BQ&t=34s&ab\_channel=CodeSports*

*MongoDB Schema Design Best Practices. (2021, April 18). [Video]. YouTube. https://www.youtube.com/watch?v=QAqK-R9HUhc&t=463s&ab\_channel=MongoDB*

*MongoDB Tutorial for Beginners - 5 - Find and Search Query. (2015, April 11). [Video]. YouTube. https://www.youtube.com/watch?v=q5EY2HRfw5c&ab\_channel=thenewboston*

*MongoDB VS Code Plugin: Tips and Tricks. (2021, April 24). [Video]. YouTube. https://www.youtube.com/watch?v=g1EjixJht6g&t=418s&ab\_channel=MongoDB*

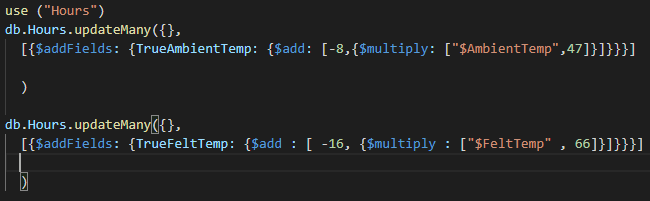
# Appendices

## MongoDB

### MongoDB - Shell

Renaming Column Names to CamelCase


3- Renaming Column Names to CamelCase



4- Adding New Calculated Fields to Dataset

A screenshot of a computer

Description automatically generated with medium confidence

5- Sample query with query output on Shell

A screenshot of a computer

Description automatically generated with medium confidence

6- Updating real Season values

Text

Description automatically generated

7- Updating real Month Values

Text

Description automatically generated

8- Updating real Year values

A screenshot of a computer

Description automatically generated with medium confidence

9- Updating real Weekday values

Text

Description automatically generated

10- Updating real Weather values

Text

Description automatically generated

11- Updating real Holiday values

Text

Description automatically generated

12- Updating real Working Day values

Text

Description automatically generated

13- Updating structure of Documents

### MongoDB – Charts

Graphical user interface, chart, line chart

Description automatically generated

14-Dashboard with Real Time Charts

## RStudio

It is to note at the final stage all processing was done through MongoDB however during the initial phase RStudio was implemented as it was the more familiar platform at the time. As time progressed and skills adapted to the MongoDB suite it was deemed the Mongo Shell was much more efficient at this thanks to its Array based Document structure.

Text

Description automatically generated

15-Replacing encoded fields with Real Values

Text

Description automatically generated

16- Dropping, Rearranging and Renaming Various Columns