

# Business Intelligence per i Big Data

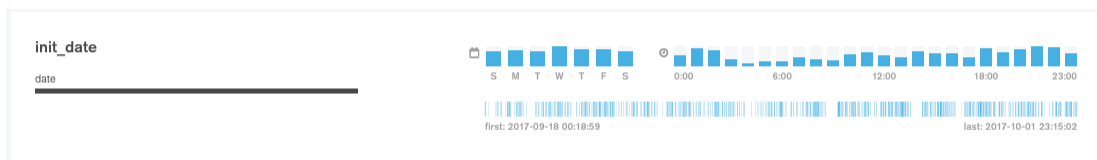
## Esercitazione n. 6

### SOLUZIONI

Nota: alcune visualizzazioni presenti nel documento potrebbero non corrispondere completamente a quelle nella versione utilizzata di MongoDB Compass.

#### 1. Analizzare la base dati con lo *schema analyzer* (Parkings)

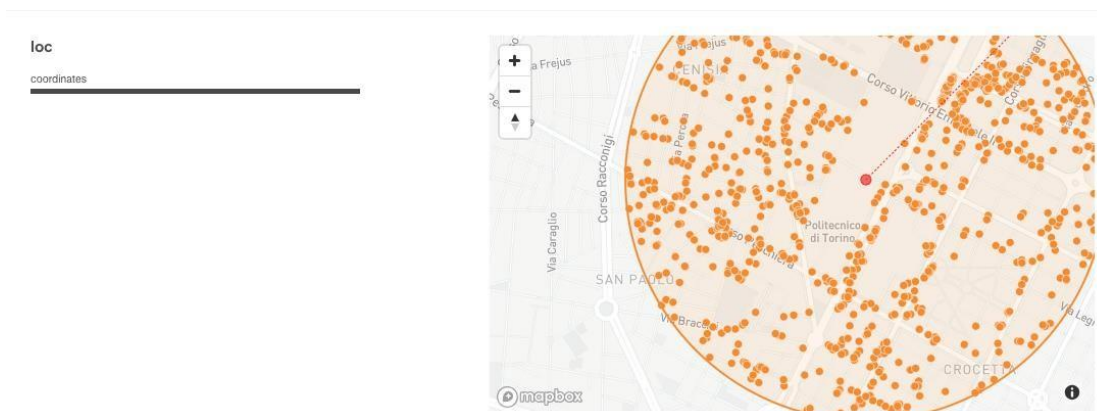
1. Identificare l'intervallo/gli intervalli orari con maggiore **richiesta di parcheggio (inizio stazionamento)** di veicoli.



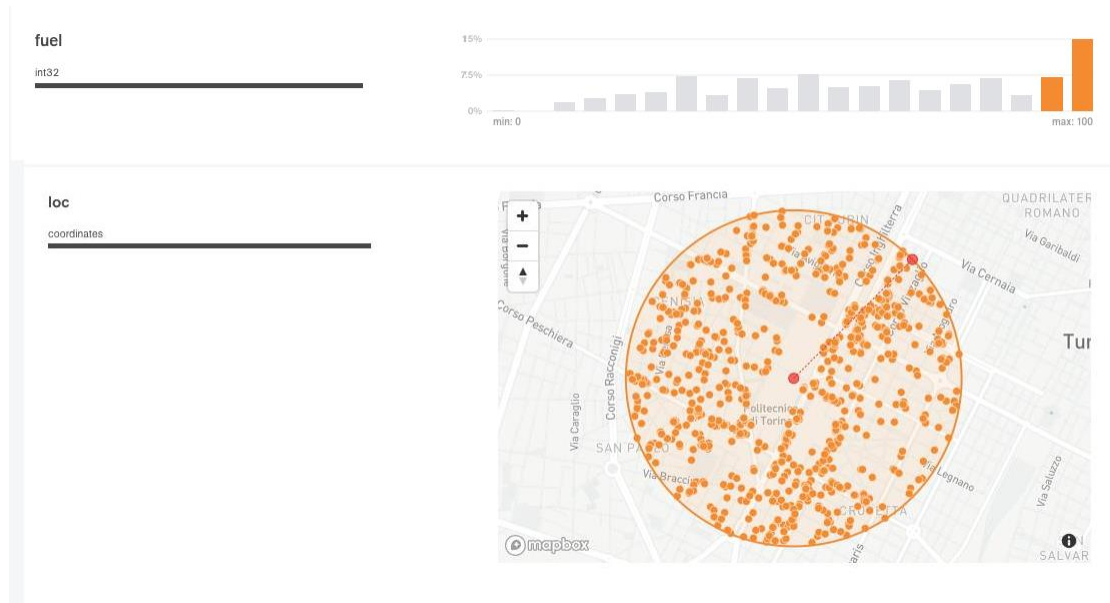
2. Identificare l'intervallo/gli intervalli orari nei quali i veicoli **vengono noleggiati (fine stazionamento)** più di frequente.



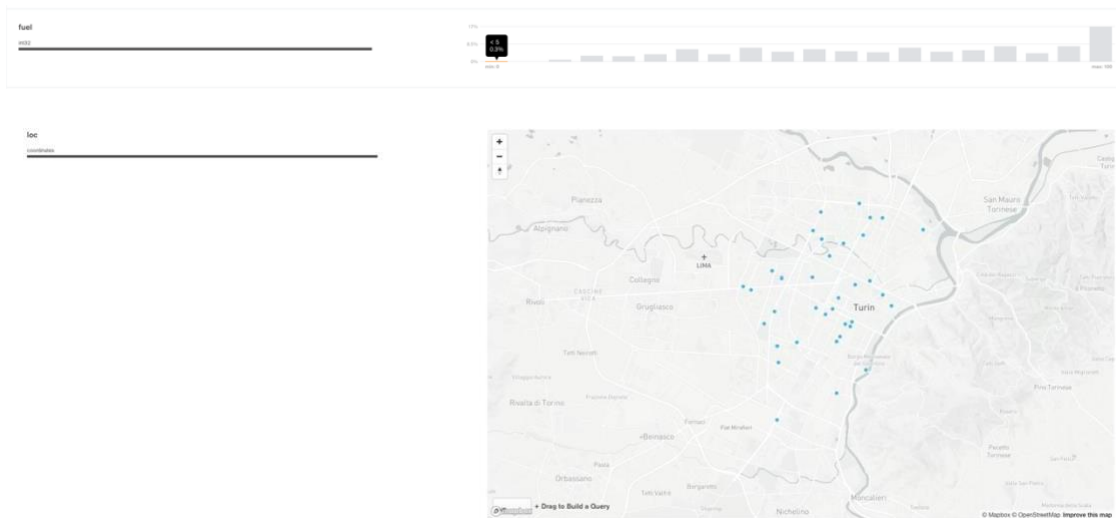
3. Filtrare sulla mappa una zona di interesse e analizzare l'intervallo/gli intervalli orari di **inizio noleggio (fine stazionamento)** più frequenti.



- Per i veicoli filtrati al passo precedente, visualizzare solo quelli che hanno un livello di carburante residuo maggiore del 90%.



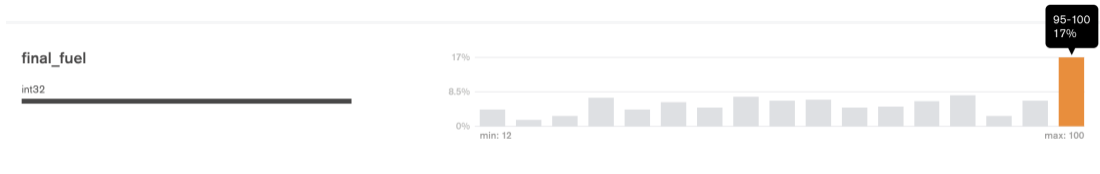
- Visualizzare su mappa i veicoli che hanno un livello di carburante residuo inferiore al 5%.



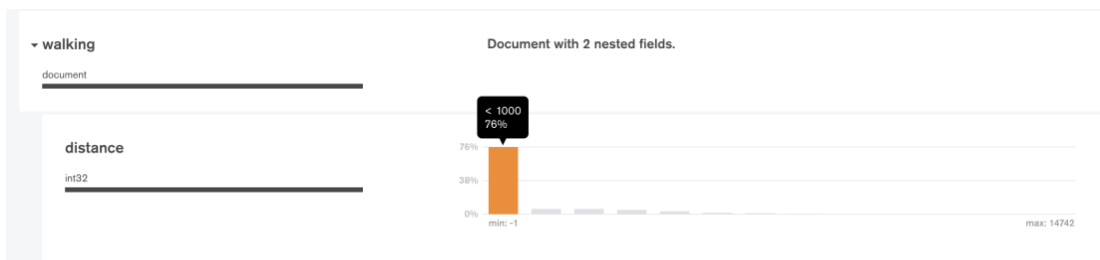
- (Bookings) Identificare la/le percentuali più frequenti di livello di carburante all'inizio del noleggio



- (Bookings) Identificare la/le percentuali più frequenti di livello di carburante a fine noleggio



- (Bookings) Identificare il range di distanza percorsa a piedi più frequente per raggiungere il veicolo.



## 2. Interrogare la base dati

- Trovare le targhe e gli indirizzi di parcheggio dei veicoli che hanno iniziato il noleggio (finito stazionamento) dopo le 6.00 del 30-09-2017.  
(Hint: usare la funzione Date("<YYYY-mm-ddTHH:MM:ss>"))

FILTER {final\_date: {\$gte : Date("2017-09-30T06:00:00Z")}}

PROJECT {address: 1, plate: 1, \_id: 0}

SORT

COLLATION

SKIP 0 LIMIT 0

VIEW LIST TABLE

Displaying documents 1 - 20 of 10062

plate: "429/FK630LX"	address: "Via Amalia Guglielminetti, 24, 10136 Torino TO"
plate: "500/FK903LX"	address: "Via Candiolo, 54, 10127 Torino TO"
plate: "321/FF296SJ"	address: "Via Zino Zini, 139, 10134 Torino TO"
plate: "340/FF144VY"	address: "Piazzetta Operaie della Manifattura Tabacchi, 10154 Torino TO"

2. Trovare gli indirizzi e il livello di carburante residuo per le auto che hanno avuto durante lo stazionamento almeno il 70% di carburante residuo e ordinare i risultati in base al loro livello di carburante decrescente.

ⓘ FILTER {fuel: {\$gte : 70}}

ⓘ PROJECT {address: 1, fuel: 1, \_id: 0}

ⓘ SORT {fuel: -1}

ⓘ COLLATION

ⓘ SKIP 0 ⓘ LIMIT 0

fuel:100  
address: "Via Don Bartolomeo Grazioli, 26, 10137 Torino TO"

fuel:100  
address: "Via Antonio Rosmini, 6, 10126 Torino TO"

fuel:100  
address: "Via Bistagno, 5, 10136 Torino TO"

fuel:100  
address: "Piazza Gian Lorenzo Bernini, 20, 10143 Torino TO"

3. Trovare la targa, tipo di motore, e livello di carburante dei veicoli di 'car2go' che hanno buone condizioni interne ed esterne.

ⓘ FILTER {vendor : "car2go", exterior: "GOOD", interior: "GOOD"}

ⓘ PROJECT {plate: 1, engineType: 1, fuel:1, \_id: 0}

ⓘ SORT

ⓘ COLLATION

ⓘ SKIP 0 ⓘ LIMIT 0

1	"262/FF153SJ"	62	"CE"
2	"244/FF119SJ"	50	"CE"
3	"256/FF144SJ"	100	"CE"
4	"516/FK993MC"	84	"CE"
5	"155/FF180NT"	75	"CE"
6	"009/FF757KW"	21	"CE"

- (Bookings) Per i noleggi che hanno richiesto una camminata maggiore di 15 km per arrivare al veicolo, visualizzare l'orario di inizio noleggio e il livello di carburante a inizio noleggio. Visualizzare i risultati ordinati in base al livello di carburante iniziale decrescente.

```

FILTER {"walking.distance" : {$gte : 15000} }
PROJECT {init_date: 1, init_fuel: 1}
SORT {init_fuel: -1}
COLLATION

```

- (Bookings) Raggruppare i documenti in base al loro livello di carburante a **fine** noleggio. Per ogni gruppo visualizzare il livello di carburante medio a **inizio** noleggio.

```

$group = {
  _id: "$final_fuel",
  avg_init_fuel: {
    $avg: "$init_fuel"
  }
}

```

Output after \$group stage (Sample of 20 documents)

_id	avg_init_fuel
39	39.88235294117647
59	60.04111405835544

- (Bookings) Visualizzare la distanza media percorsa nei noleggi per ciascun fornitore del servizio. In media con quale fornitore del servizio gli utenti percorrono una distanza maggiore?

```

$group = {
  _id: "$vendor",
  avg_init_fuel: {
    $avg: "$distance"
  }
}

```

Output after \$group stage (Sample of 2 documents)

_id	avg_distance
car2go	1915.44424954083
enjoy	2430.8068234979614

7. (Bookings) Quali sono i primi tre veicoli che hanno percorso una maggiore distanza considerando tutti i noleggi? Visualizzarne la targa e il conteggio in ordine decrescente per conteggio.

The screenshot shows a MongoDB Atlas pipeline editor with three stages:

- \$group stage:** The code defines a group by `_id` and aggregates `sum_distance` for each group. The output shows three documents with `sum_distance` values of 15746, 17743, and 161/FL467CG.
- \$sort stage:** The code sorts the documents by `sum_distance` in descending order. The output shows the top three documents with `sum_distance` values of 428935, 388515, and 428935.
- \$limit stage:** The code limits the results to 3 documents. The output shows the top three documents with `sum_distance` values of 428935, 388515, and 428935.

8. (Bookings) Quanti sono i veicoli che, considerando tutti i noleggi, hanno percorso almeno 350000m?

The screenshot shows a MongoDB Atlas pipeline editor with three stages:

- \$group stage:** The code defines a group by `_id` and aggregates `sum_distance` for each group. The output shows two documents with `sum_distance` values of 15746 and 17743.
- \$match stage:** The code filters documents where `sum_distance` is greater than or equal to 350000. The output shows two documents with `sum_distance` values of 362942 and 357386.
- \$count stage:** The code counts the number of documents. The output shows a single document with `conteggio` value of 21.

9. (Bookings) Ripetere la query precedente considerando solamente i veicoli di proprietà 'car2go'.

The screenshot shows a data pipeline with four stages:

- Stage 1: \$match**
  - Query: 

```
1- /**
2- * query: The query in MQL.
3- */
4- {
5-   vendor: 'car2go'
6- }
```

  - Output: Three documents with fields like `_id`, `init_fuel`, `city`, `walking`, `init_address`, `vendor`, and `final_time`.
- Stage 2: \$group**
  - Query: 

```
1- /**
2- * _id: The id of the group.
3- * field: The first field name.
4- */
5- {
6-   _id: '$plate',
7-   sum_distance: {
8-     $sum: '$distance'
9-   }
10- }
```

  - Output: Three documents with `_id` and `sum_distance`.
- Stage 3: \$match**
  - Query: 

```
1- /**
2- * query: The query in MQL.
3- */
4- {
5-   sum_distance: { $gte: 350000 }
6- }
```

  - Output: Three documents with `_id` and `sum_distance`.
- Stage 4: \$count**
  - Query: 

```
1- /**
2- * Provide the field name for the count.
3- */
4- 'conteggio'
```

  - Output: A single document with `conteggio: 6`.

10. (Parkings) Trovare le macchine parcheggiate a meno di 1km da Piazza San Carlo (coordinate 7.683016, 45.067764).

Hint: usare l'operatore [\\$geoWithin](#) insieme a [\\$centerSphere](#).

```
{ loc : { $geoWithin : { $centerSphere : [ [ 7.683016, 45.067764 ] , 1 / 6378.1 ] } } }
```

The screenshot shows a data visualization interface with a map. At the top, there is a filter bar with the query: `{ loc : { $geoWithin : { $centerSphere : [ [ 7.683016, 45.067764 ] , 1 / 6378.1 ] } } }`. Below the filter bar, there is a section titled "loc" with a "coordinates" field. The main part of the interface is a map showing a dense cluster of blue dots representing parking locations around Piazza San Carlo. The map includes a scale bar, a compass, and a "Leaflet | © 1987-2019 HERE | Terms of Use" footer.

11. (Parkings) Ripetere l'interrogazione al passo precedente con un punto di interesse personale nell'area metropolitana di Torino (e.g. indirizzo di casa) usando Open Street Maps per trovare le coordinate esatte ([www.openstreetmap.org](http://www.openstreetmap.org), invertire l'ordine delle coordinate).

Come la query precedente cambiando le coordinate con quelle d'interesse.