Analyzing Ferrari's last years in F1

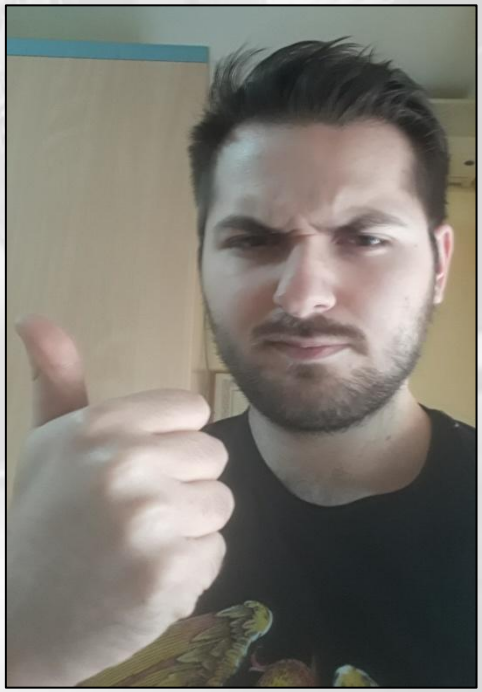
- Neo4j portion -

Luca Sannino 1542194

- Hi, my name is Luca… and I've recently become the new team principal at Ferrari!

2014-2019

2019-2020



2020-????

- As a team principal, it is my duty to keep improving the team so that we can compete with the other constructors and, hopefully, win the season.

- However, before I can improve something, I first need to understand WHAT it needs to be improved!

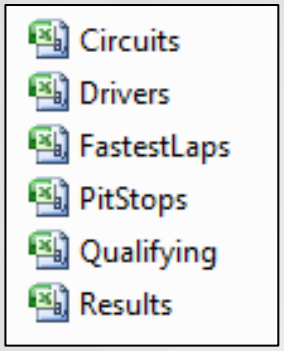
• So I've selected three areas where I feel the team is not meeting my expectations.

POTENTIAL ISSUES

1 – Our pit stops are subpar
2 – Vettel is underperforming
3 – The car is slower than Mercedes'

• The objective of this project is to change that "*where I feel*" to "*where I know*". How?
• By using a graph database to query data obtained after an integration process.
• This portion will focus on the graph database!

- My instruments for this portion will be:



A series of .csv files containing F1 stats obtained after integrating various sources with Talend

Neo4j: a software that lets the user create a graph database and query the data inside it

A brief overview on Neo4j

- Property-graph database
- Written in Java
- Available in two versions: server and desktop
- Desktop version uses JVM, server version can be accessed through REST operations
- Useful for querying thanks to its Cypher language
- Sharding not supported: the entire graph must be saved inside a machine
- However it supports primary-secondaries architectures

POTENTIAL ISSUE 1

Are our pit stops subpar?

• Last year Red Bull registered the quickest pit stop in the entire history of Formula 1.

WHAT I WANT TO KNOW

Was it just luck or a symptom of a bigger problem: that we are just slower?

- The objective of this exercise will be to load in the graph database a .csv file containing all pit stops since 2011, and then querying the data in order to find our pit stops in the last season, calculate some stats, do the same for Red Bull, and then compare the results and see how we stack up.

- Let’s begin by loading the data in the graph database!

SOURCE FILE

- Pitstops(race, circuit, race_date, year, surname, name, stop, lap, duration, constructor)

LOADING INTO THE GRAPH

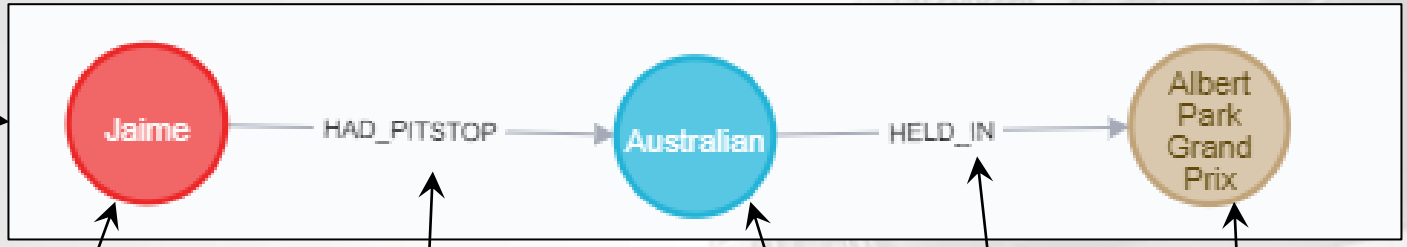
```
LOAD CSV WITH HEADERS FROM 'file:///Pitstops.csv' AS line
FIELDTERMINATOR ';'
MERGE (x: Race {name: line.race, date: line.
toInteger(line.year)})
MERGE (z: Circuit {name: line.circuit})
MERGE (y:Driver {surname: line.surname, name: line.name})
CREATE (y)-[:HAD_PITSTOP {stop: toInteger(line.stop), lap:
toInteger(line.lap), duration_in_ms: toInteger(line.duration_in_ms),
constructor: line.constructor}]->(x)
MERGE (x)-[:HELD_IN]->(z)
```

Must specify if a .csv file contains headers or how the data is divided.

Merge: if a particular node already exists, it doesn't create a new one.

RESULT

race	circuit	race_date	year	surname	name	stop	lap	duration_	constructor
Australian Grand Prix	Albert Park Grand Prix Circuit	27/03/2011	2011	Alguersuari	Jaime	1	1	26898	Toro Rosso



:DRIVER {surname, name}

:HAD_PITSTOP {stop, lap, duration, constructor}

:RACE {name, date, year}

:HELD_IN

:CIRCUIT {name}

- LABELS
- RELATIONSHIPS
- PROPERTIES

QUERY 1

Average pit stop time per Ferrari and RedBull drivers last season

```
MATCH (d)-[dp:HAD_PITSTOP]->(r)
WHERE r.year = 2019 and (dp.constructor = "Ferrari" or
dp.constructor = "Red Bull")
RETURN d.surname as surname, dp.constructor as constructor,
avg(dp.duration_in_ms) as average_in_ms
```

QUERY 2

Average pit stop time per Ferrari and RedBull last season

```
MATCH (d)-[dp:HAD_PITSTOP]->(r)
WHERE r.year = 2019 and (dp.constructor = "Ferrari" or
dp.constructor = "Red Bull")
RETURN dp.constructor as constructor, avg(dp.duration_in_ms) as
average_in_ms
```

QUERY 3

Return quickest Ferrari and Red Bull pit stops for 2019

```
MATCH (d)-[dp:HAD_PITSTOP]->(r)
WHERE r.year = 2019 and (dp.constructor = "Ferrari" or
dp.constructor = "Red Bull")
RETURN dp.constructor as constructor, min(dp.duration_in_ms) as
quickest_in_ms
```

QUERY 4

Return slowest Ferrari and Red Bull pit stops for 2019

```
MATCH (d)-[dp:HAD_PITSTOP]->(r)
WHERE r.year = 2019 and (dp.constructor = "Ferrari" or
dp.constructor = "Red Bull")
RETURN dp.constructor as constructor, max(dp.duration_in_ms) as
slowest_in_ms
```

CONCLUSIONS

surname	constructor	average_in_ms
"Gasly"	"Red Bull"	23657.105263157893
"Verstappen"	"Red Bull"	23682.968749999996
"Leclerc"	"Ferrari"	24281.25
"Vettel"	"Ferrari"	24942.513513513513
"Albon"	"Red Bull"	25059.714285714283

CONCLUSIONS

surname	constructor	average_in_ms
"Gasly"	"Red Bull"	23657.105263157893
"Verstappen"	"Red Bull"	23682.968749999996
		24281.25
		24942.513513513513
		25059.714285714283

Looking at the results, and since Albon raced in too few races to make his average relevant, it's fair to say that our pit stops are slower. We need either to improve or account for a ~1 second delay when making strategies that involve pitting.

POTENTIAL ISSUE 2

Is Vettel underperforming?

• Last year many claimed Vettel was feeling the pressure from his new teammate, Leclerc, which led him to some major mistakes.

WHAT I WANT TO KNOW

Is Vettel in a difficult situation, or are people just blinded by the hype for our younger driver?

• The objective of this exercise will be to load in the graph database a .csv file containing all the results since F1 started, and then querying the data in order to find out how many times Vettel has placed in a better place than his teammate per season, so that we can see if there is a downgrade in his performance across the years.

• Let’s begin by loading the data in the graph database!

SOURCE FILE

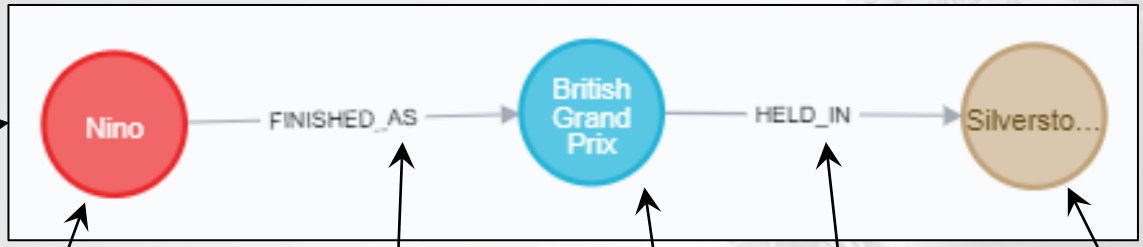
- Results(race_name, date, year, circuit, surname, name, constructor, grid, position, status)

LOADING INTO THE GRAPH

```
LOAD CSV WITH HEADERS FROM 'file:///Results.csv' AS line
FIELDTERMINATOR ';'
MERGE (x:Race {name: line.race_name, date: line.date, year:
toInteger(line.year)})
MERGE (z: Circuit {name: line.circuit})
MERGE (y:Driver {surname: line.surname, name: line.name})
CREATE (y)-[:FINISHED_AS {grid: toInteger(line.grid), position:
line.position, status: line.status, constructor: line.constructor}]-
>(x)
MERGE (x)-[:HELD_IN]->(z)
```

RESULT

race_name	date	year	circuit	surname	name	constructo	grid	position	status
British Grand Prix	########	1950	Silverston	Farina	Nino	Alfa Rome	1	1	Finished



:DRIVER {surname, name}

:FINISHED_AS {constructor, grid, position, status}

:RACE {name, date, year}

:HELD_IN

:CIRCUIT {name}

- LABELS
- RELATIONSHIPS
- PROPERTIES

QUERY 1

List of races where Vettel won against his teammate

```
MATCH (v)-[vr:FINISHED_AS]->(r)<-[tr:FINISHED_AS]-(t)
WHERE v.surname="Vettel" and ID(v) <> ID(t) and vr.constructor =
tr.constructor and (toInteger(vr.position) < toInteger(tr.position) or
tr.position = "\\N" and vr.position <> "\\N")
RETURN r.name as race, r.date as date, r.year as year, vr.position as
vettel_position, t.surname as teammate, tr.position as
teammate_position, vr.constructor as constructor
ORDER BY date
```

QUERY 2

Number of times Vettel has beaten his teammate per season

```
MATCH (v)-[vr:FINISHED_AS]->(r)<-[tr:FINISHED_AS]-(t)
WHERE v.surname="Vettel" and ID(v) <> ID(t) and vr.constructor =
tr.constructor and (toInteger(vr.position) < toInteger(tr.position) or
tr.position = "\\N" and vr.position <> "\\N")
RETURN r.year as year, t.surname as teammate, count(*) as
vettel_wins
ORDER BY year
```

QUERY 3

List of races where Vettel lost against his teammate

```
MATCH (v)-[vr:FINISHED_AS]->(r)<-[tr:FINISHED_AS]-(t)
WHERE v.surname="Vettel" and ID(v) <> ID(t) and vr.constructor =
tr.constructor and (toInteger(vr.position) > toInteger(tr.position) or
tr.position <> "\\N" and vr.position = "\\N")
RETURN r.name as race, r.date as date, r.year as year, vr.position as
vettel_position, t.surname as teammate, tr.position as
teammate_position, vr.constructor as constructor
ORDER BY date
```

QUERY 4

Number of times Vettel has lost against his teammate per season

```
MATCH (v)-[vr:FINISHED_AS]->(r)<-[tr:FINISHED_AS]-(t)
WHERE v.surname="Vettel" and ID(v) <> ID(t) and vr.constructor =
tr.constructor and (toInteger(vr.position) > toInteger(tr.position) or
tr.position <> "\\N" and vr.position = "\\N")
RETURN r.year as year, t.surname as teammate, count(*) as
vettel_lost
ORDER BY year
```

CONCLUSIONS

2008	"Bourdais"	12
2009	"Webber"	8
2010	"Webber"	11
2011	"Webber"	16
2012	"Webber"	13
2013	"Webber"	18
2014	"Ricciardo"	5
2015	"R♦ikk♦nen"	14
2016	"R♦ikk♦nen"	14
2017	"R♦ikk♦nen"	16
2018	"R♦ikk♦nen"	12
2019	"Leclerc"	12

- Even though Vettel had a hard time against Leclerc, it was surprising to learn that it wasn't any better last year or in 2014, where Ricciardo clearly gave a stronger performance!

POTENTIAL ISSUE 3

Are we slower than Mercedes?

- It’s not a secret that Mercedes is dominating the recent era of Formula 1.

2014	Lewis Hamilton [30]	29	Mercedes	Mercedes
2015	Lewis Hamilton [30]	30	Mercedes	Mercedes
2016	Nico Rosberg [33]	31	Mercedes	Mercedes
2017	Lewis Hamilton [30]	32	Mercedes	Mercedes
2018	Lewis Hamilton [30]	33	Mercedes	Mercedes
2019	Lewis Hamilton [30]	34	Mercedes	Mercedes

WHAT I WANT TO KNOW

Is this the result of unlucky circumstances or is our car just slower?

- The objective of this exercise will be to load in the graph database a .csv file containing all fastest laps per pilot and race since 2004, and then query the data in order to collect all fastest laps from races since 2014 in which each driver from both Ferrari and Mercedes had the opportunity to register a valid fast lap, and then sum the fast lap times per season and per pilot, and ultimately compare the results and see who has a quicker time.

- Let’s begin by loading the data in the graph database!

SOURCE FILE

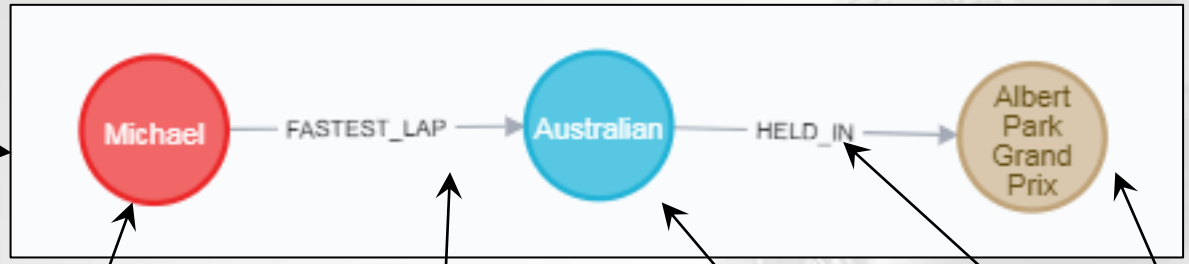
- FastestLaps(race_name, date, year, circuit, surname, name, stop, constructor, fastestLap, fastestLapTime)

LOADING INTO THE GRAPH

```
LOAD CSV WITH HEADERS FROM 'file:///FastestLaps.csv' AS line
FIELDTERMINATOR ';'
MERGE (x:Race {name: line.race_name, date: line.date, year:
toInteger(line.year)})
MERGE (z: Circuit {name: line.circuit})
MERGE (y:Driver {surname: line.surname, name: line.name})
CREATE (y)-[:FASTEST_LAP {fastestLap: toInteger(line.fastestLap),
fastestLapTime: line.fastestLapTime, constructor:
line.constructor}]->(x)
MERGE (x)-[:HELD_IN]->(z)
```

RESULT

race_name	date	year	circuit	surname	name	constructo	fastestLap	fastestLapTime
Australian Gra	07/03/2004	2004	Albert Par	Schumacher	Michael	Ferrari	29	1:24.125



:DRIVER {surname, name}

:FASTEST_LAP {constructor, fastestLap, fastestLapTime}

:RACE {name, date, year}

:HELD_IN

:CIRCUIT {name}

- LABELS
- RELATIONSHIPS
- PROPERTIES

QUERY 1

Return all races from >= 2014 where all Mercedes and Ferrari pilots registered a fast lap

```
MATCH (r: Race)<-[fl: FASTEST_LAP]-(d: Driver)
WHERE r.year >= 2014 and fl.fastestLap >= 10 and (fl.constructor =
"Ferrari" or fl.constructor = "Mercedes")
WITH r, COUNT(*) as c
WHERE c = 4
RETURN r.name as name, r.date as date, r.year as year
ORDER BY year
```

WITH lets you carry the results of a query to the next one.

QUERY 2

Return all fastest laps from all >= 2014 races where all Mercedes and Ferrari pilots registered a fast lap

```
MATCH (r: Race)<-[fl: FASTEST_LAP]-()
WHERE r.year >= 2014 and fl.fastestLap >= 10 and (fl.constructor =
"Ferrari" or fl.constructor = "Mercedes")
WITH r, COUNT(*) as c
WHERE c = 4
MATCH (r)<-[fl: FASTEST_LAP]-(d: Driver)
WHERE fl.constructor = "Ferrari" or fl.constructor = "Mercedes"
RETURN r.name as name, r.date as date, r.year as year, d.surname
as surname, fl.fastestLapTime as fastestLapTime
ORDER BY year
```

QUERY 3

Return the sum in ms of all fastest laps from all >= 2014 races where all Mercedes and Ferrari pilots registered a fast lap

```
MATCH (r: Race)<-[fl: FASTEST_LAP]-()
WHERE r.year >= 2014 and fl.fastestLap >= 10 and (fl.constructor =
"Ferrari" or fl.constructor = "Mercedes")
WITH r, COUNT(*) as c
WHERE c = 4
MATCH (r)<-[fl: FASTEST_LAP]-(d: Driver)
WHERE fl.constructor = "Ferrari" or fl.constructor = "Mercedes"
WITH r, d, [item in split(fl.fastestLapTime, ":") | toInteger(item)] AS
timeComponents1, [item in split(fl.fastestLapTime, ".") |
toInteger(item)] AS timeComponents2
RETURN r.year as year, d.surname as surname,
sum(timeComponents1[0]*60000+timeComponents1[1]*1000+timeCo
mponents2[1]) AS total_time
ORDER BY year
```

We can't sum dates, so we need to split and convert the format 'mm.ss.SSS' to ms

QUERY 4

Return the sum in mm:ss.SSS of all fastest laps from all >= 2014 races where all Mercedes and Ferrari pilots registered a fast lap

```
MATCH (r: Race)<-[fl: FASTEST_LAP]-()
WHERE r.year >= 2014 and fl.fastestLap >= 10 and (fl.constructor =
"Ferrari" or fl.constructor = "Mercedes")
WITH r, COUNT(*) as c
WHERE c = 4
MATCH (r)<-[fl: FASTEST_LAP]-(d: Driver)
WHERE fl.constructor = "Ferrari" or fl.constructor = "Mercedes"
WITH r, d, [item in split(fl.fastestLapTime, ":") | toInteger(item)] AS
timeComponents1, [item in split(fl.fastestLapTime, ".") |
toInteger(item)] AS timeComponents2
RETURN r.year as year, d.surname as surname,
apoc.date.format(sum(timeComponents1[0]*60000+timeComponent
s1[1]*1000+timeComponents2[1]), "ms", "mm:ss.SSS") AS total_time
ORDER BY year
```

NOTE: in order to use apoc we need to install the plugin!

CONCLUSIONS

2017	"Hamilton"	"21:46.959"
2017	"Bottas"	"21:50.936"
2017	"Vettel"	"21:47.379"
2017	"R?ikk?nen"	"21:52.122"
2018	"Bottas"	"27:29.129"
2018	"Hamilton"	"27:25.946"
2018	"Vettel"	"27:26.812"
2018	"R?ikk?nen"	"27:33.768"
2019	"Vettel"	"27:58.008"
2019	"Hamilton"	"27:48.313"
2019	"Bottas"	"27:48.603"
2019	"Leclerc"	"28:07.056"

- This data shows that we clearly need to keep improving the performance of our car because Mercedes is still ahead!

FINAL REMARKS

- By integrating and querying data, I could take a authentic look at how various aspects of the team have performed and where - and if - they fell short.

- The answers I gained from this analysis will help me make decisions in the future that will be based on facts and not subjective feelings, which was exactly my initial objective!

BONUS

Let’s add and query more data!

UPDATING THE GRAPH 1

Add more informations to the circuits

```
LOAD CSV WITH HEADERS FROM 'file:///Circuits.csv' AS line
FIELDTERMINATOR ';'
MERGE (c: Circuit {name: line.name})
SET c.location = line.location
SET c.country = line.country
SET c.lat = line.lat
SET c.lon = line.lon
```

UPDATING THE GRAPH 2

Add more informations to the drivers

```
LOAD CSV WITH HEADERS FROM 'file:///Drivers.csv' AS line
FIELDTERMINATOR ';'
MERGE (d: Driver {surname: line.surname, name: line.forename})
SET d.number = line.number
SET d.code = line.code
SET d.dob = line.dob
SET d.nationality = line.nationality
```

UPDATING THE GRAPH 3

Add information about qualifying

```
LOAD CSV WITH HEADERS FROM 'file:///Qualifying.csv' AS line
FIELDTERMINATOR ';'
MATCH (d: Driver {surname: line.surname, name: line.name})
MATCH (r: Race {name: line.race, date: line.race_date, year:
toInteger(line.year)})
MERGE (d)-[:QUALIFIED_AS {position: line.position, q1: line.q1, q2:
line.q2, q3: line.q3, constructor: line.constructor}]->(r)
```

QUERY 1

All wins by Ferrari drivers since the beginning of F1

```
MATCH (d)-[vr:FINISHED_AS]->(r)
WHERE vr.constructor = "Ferrari" and vr.position = "1"
RETURN r.date as date, d.surname as surname, d.name as name,
r.name as circuit
ORDER BY date
```

QUERY 2

All wins by Michael Schumacher when he was in Ferrari

```
MATCH (d)-[vr:FINISHED_AS]->(r)
WHERE vr.constructor = "Ferrari" and vr.position = "1" and d.name =
"Michael" and d.surname = "Schumacher"
RETURN r.date as date, r.name as circuit
ORDER BY date
```

QUERY 3

All DNF from Ferrari

```
MATCH (d)-[vr:FINISHED_AS]->(r)
WHERE vr.constructor = "Ferrari" and vr.position = "\\N"
RETURN r.date as date, r.name as circuit, d.surname as surname,
d.name as name, vr.status as status
ORDER BY date
```

QUERY 4

All Italian pilots who drove for Ferrari

```
MATCH (d: Driver)-[r]->()
WHERE r.constructor = "Ferrari" and d.nationality = "Italian"
RETURN DISTINCT d.surname as surname, d.name as name
```

QUERY 5

All races by Michael Schumacher where he both took pole position and won, when he was in Ferrari

```
MATCH (d)-[vr:FINISHED_AS]->(r)<-[qa:QUALIFIED_AS]-(d)
WHERE vr.constructor = "Ferrari" and vr.position = "1" and d.name =
"Michael" and d.surname = "Schumacher" and qa.position = "1"
RETURN r.date as date, r.name as circuit
ORDER BY date
```

QUERY 6

Fastest qualifying lap at Monza by a Ferrari driver

```
MATCH (d)-[qa:QUALIFIED_AS]->(r)-[:HELD_IN]->(c)
WHERE qa.constructor = "Ferrari" and c.name = "Autodromo
Nazionale di Monza"
WITH min(qa.q3) as fastest_qualify
MATCH (d)-[qa:QUALIFIED_AS]->(r)
WHERE qa.q3 = fastest_qualify
RETURN d.surname as surname, fastest_qualify, r.date as date
```

INDEX CREATION

```
CREATE INDEX FOR (d: Driver) ON (d.surname)
```

- Time to answer the “All Ferrari wins by Michael Schumacher” query before the index:

```
Started streaming 72 records after 2 ms and completed after 60 ms.
```

- Same query, but after the index:

```
Started streaming 72 records after 1 ms and completed after 9 ms.
```

REFERENCES

- Kaggle F1 Dataset: https://www.kaggle.com/rohanrao/formula-1-world-championship-1950-2020

- Talend Website: https://www.talend.com/

- Neo4j Website: https://neo4j.com/