Team Gold

By Fahmida Bilqis, Leyla Tabarrok, Mandana Atunrase & Tricia Tan

We have approached this challenge in the context of gathering information as Lewis Hamilton's constructor team, ready to strategise for the next Grand Prix!

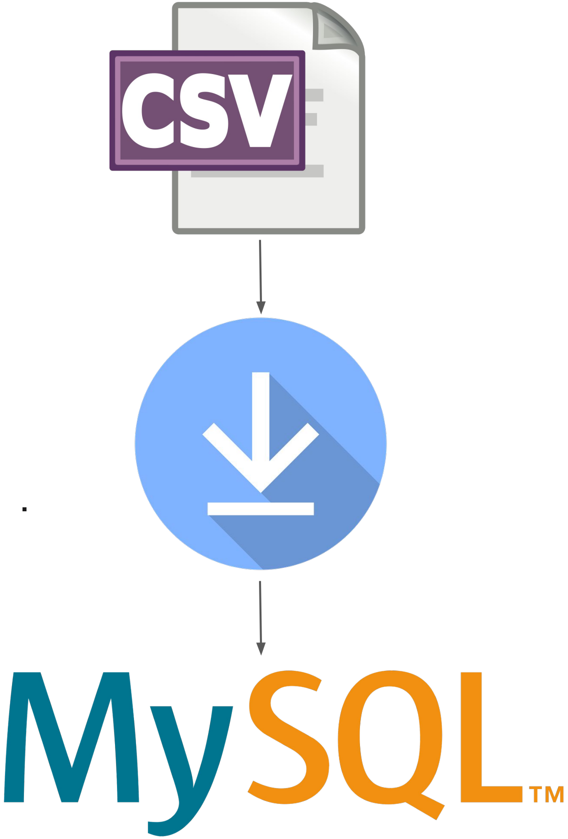
Practice Rounds: On your marks!

How did we prepare ourselves for the challenge?

- Prior to our initial meeting, Team Gold agreed to download the csv files, look at each table, find out how F1 works and read through the task guide.
- During out team meeting we:
  - Introduced and gave a little background to ourselves.
  - Discovered that between us, our coding experience and personal aims for the MOOC challenge were wide ranging.
  - Agreed that we wanted to give our challenge a context to help us focus because there was so much data to potentially work with.
  - Were aware that there was limited time to explore the data and that we would be working independently but at the same time wanted to support each other in the team
  - Looked at the initial EER diagram with disconnected schemas of imported csv files and agreed on our PKs and FKs.
- With this in mind we set ourselves the following objectives:
  - Position the challenge into the context of gathering information as Lewis Hamilton's constructor team, ready to strategise for the next Grand Prix
  - Within this context explore coding at our own level of understanding to pull out information within three categories: Drivers, Constructors and Races.
  - Keep track of coding by posting our codes into dedicated threads in Slack
  - Use Slack to get support from fellow team mates during the process
  - Throughout the week explore the data and paste any coding you have into the applicable thread with a statement about what information it is showing or question that it is answering
  - At the end of the week there was another meeting to review what we've done and discuss the slide presentation
- Tricia also produced a detailed document explaining the meaning of each column in the schemas so that we had a clear guide to refer to.

Qualifying : Get Set!

This project was focused on analysing data from the Formula 1 dataset.



The initial phase consisted of downloading the CSV files for the Formula 1 data which is sourced from kaggle. The database software we used is called MySQL which allows you to create a database and tables to allow the storage of data.

The DDL statement ‘CREATE’ was used to make a database called Formula1 along with the ‘USE’ statement and the data was then imported into this database to allow us to start querying. Although this had taken a while all data was successfully imported.

Along with this, an EER (Enhanced -Entity-Relationship) diagram which is a data model diagram was produced where primary keys and foreigns keys were assigned to certain columns.

Referring to the EER diagram was very useful when exploring the coding as it gave an overview of the schemas and helped us to understand the information contained within the database.

Our approach to writing the code differed as we are on different levels when it came to our experience with SQL. For some of us the knowledge from MOOC sprint sessions equipped us with the ability to write different queries that ranged from simple to complex code as well as the numerous resources online from informative websites like W3schools and coding tutorials on youtube. For another team member they found it helped to create a simple code, run it, then keep buildings up by inserting additional steps and running between steps. This helped them to pinpoint any errors and understand each step.

Pit Stop: *Challenges we faced and how we would approach the challenge next time*

- Be mindful of **order**: need to set PKs before FKs
- Difficult to use aggregate functions on time when it was a text data type
  - ✓ Resolved this by converting into milliseconds and changing data type to BIGINT
- A more **open exercise:** from being given questions during the MOOC to having to having come up with our own.
- Hard to decide between categories — an overlap with information about racers, drivers and constructors.
- Working **remotely** in a team with **new people**
  - ✓ Scheduled online meetings
  - ✓ Maintained contact and shared code on slack.
- We **supported** different knowledge and experience levels
  - ✓ Turned to and encouraged each other when stuck
  - ✓ Filled in the gaps and built on each other's code

Next steps

- Our coding was intentionally exploratory and so the information we extracted, although organised into the 3 categories, was disjointed.
- We could also have planned what we were going to ask and assigned queries to team members to ensure coherent code.
- It would be useful to collate all the findings in a summary to see where we had gaps in our information and carry out the coding to address these omissions, making the information we had gathered fully coherent.

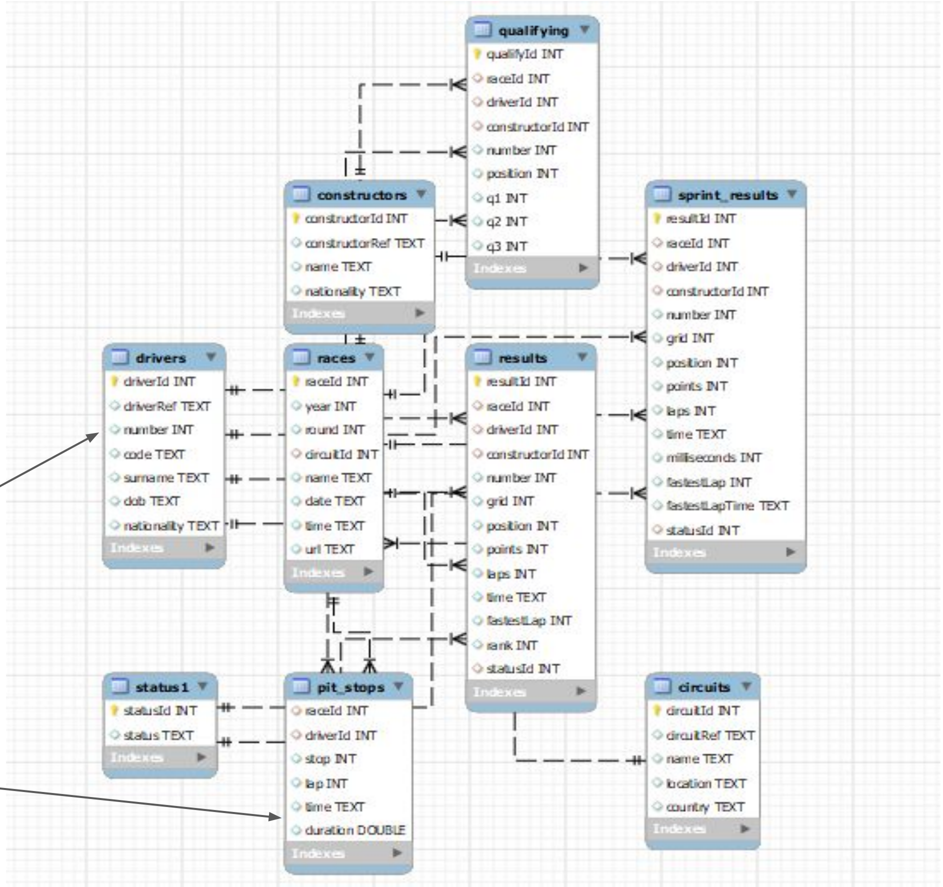
EER DIAGRAM

The EER diagram has lines connecting each entity which is known as the tables within our database.

The entities also have different attributes which are the columns as shown in the diagram. The lines are a visual representation of the relationships between the tables.

For example for the Drivers table , **driverId** is a primary key in this but its a foreign key in the results table. It's also a foreign key in the circuits, pit stops and qualifying table.
This represents a 1 to many relationship.

However not all tables have a primary key e.g. the pit stops table has foreign keys but not a primary key.



qualifying
qualifyId INT
raceId INT
driverId INT
constructorId INT
number INT
position INT
q1 INT
q2 INT
q3 INT
Indexes

constructors
constructorId INT
constructorRef TEXT
name TEXT
nationality TEXT
Indexes

sprint_results
resultId INT
raceId INT
driverId INT
constructorId INT
number INT
grid INT
position INT
points INT
laps INT
time TEXT
milliseconds INT
fastestLap INT
fastestLapTime TEXT
statusId INT
Indexes

drivers
driverId INT
driverRef TEXT
number INT
code TEXT
surname TEXT
dob TEXT
nationality TEXT
Indexes

races
raceId INT
year INT
round INT
circuitId INT
name TEXT
date TEXT
time TEXT
url TEXT
Indexes

results
resultId INT
raceId INT
driverId INT
constructorId INT
number INT
grid INT
position INT
points INT
laps INT
time TEXT
fastestLap INT
rank INT
statusId INT
Indexes

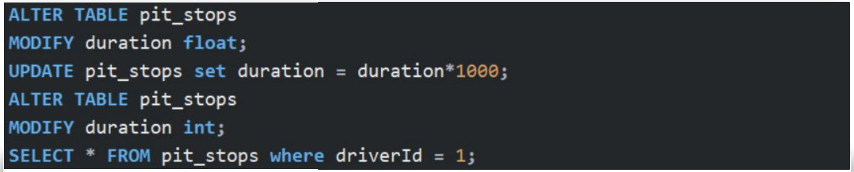
status1
statusId INT
status TEXT
Indexes

pit_stops
raceId INT
driverId INT
stop INT
lap INT
time TEXT
duration DOUBLE
Indexes

circuits
circuitId INT
circuitRef TEXT
name TEXT
location TEXT
country TEXT
Indexes

FINAL RACE: GO!

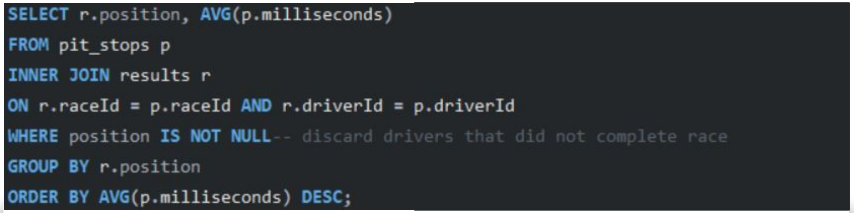
During analysis we were most interested in looking at metrics relating to team performance, in order to secure as many points as possible in future Grands Prix.



```sql
ALTER TABLE pit_stops
MODIFY duration float;
UPDATE pit_stops set duration = duration*1000;
ALTER TABLE pit_stops
MODIFY duration int;
SELECT * FROM pit_stops where driverId = 1;
```

DDL + DML

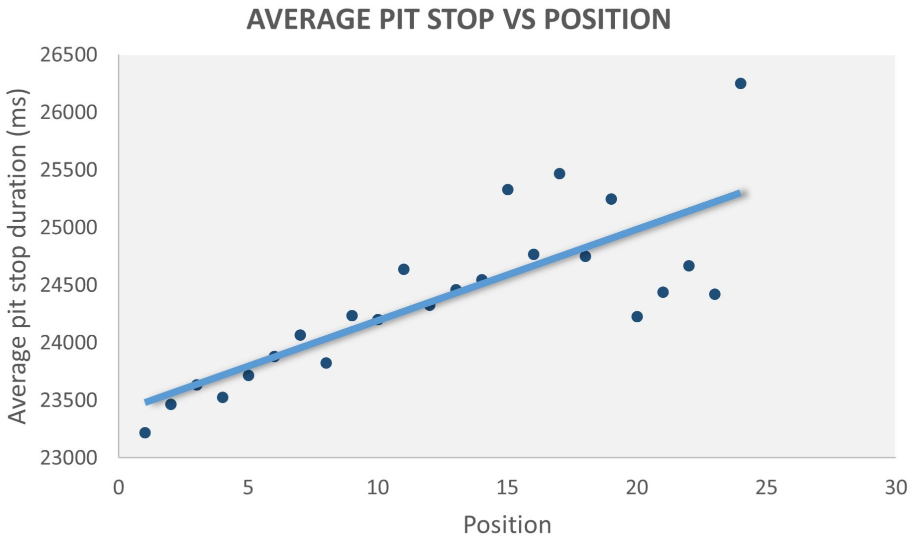
1. Before extracting pit stop data, the pit stop duration was first converted to milliseconds using a combination of *DDL* and *DML,* to make easier any numerical manipulation.



```sql
SELECT r.position, AVG(p.milliseconds)
FROM pit_stops p
INNER JOIN results r
ON r.raceId = p.raceId AND r.driverId = p.driverId
WHERE position IS NOT NULL-- discard drivers that did not complete race
GROUP BY r.position
ORDER BY AVG(p.milliseconds) DESC;
```

AVG()

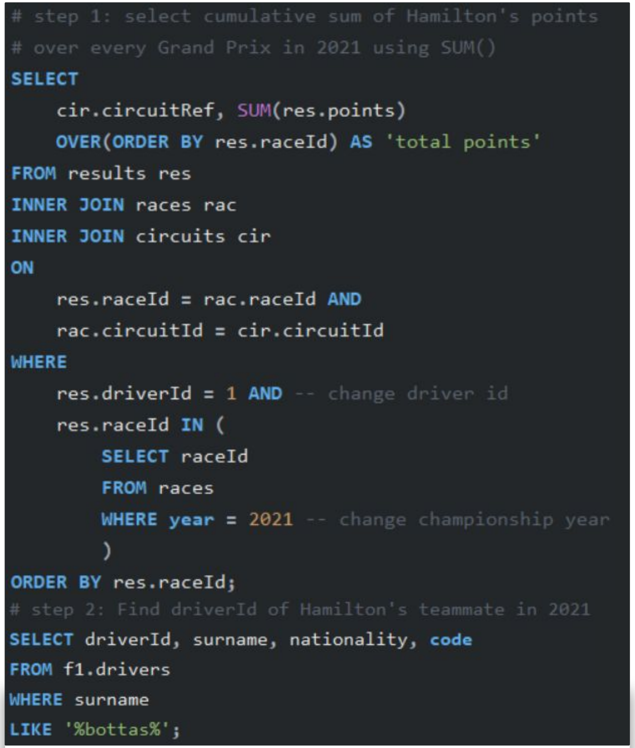
2. Then, using an *aggregate function*, the average pit stop duration grouped with the end position was selected, allowing for the graph (right) to be produced.



From this scatter plot we can deduce that we should be aiming for a pit stop time of less than 24 seconds in order to secure a top 10 position, meaning guaranteed points.

This graph only shows correlation between pit stop duration and end position, but we could also look at, for example, grid position or qualifier time and how these variables correlate with the end position. Then comparing all these factors helps to see which ones are better indicators of winning performance, and therefore which areas to prioritise and give more attention to before a race.

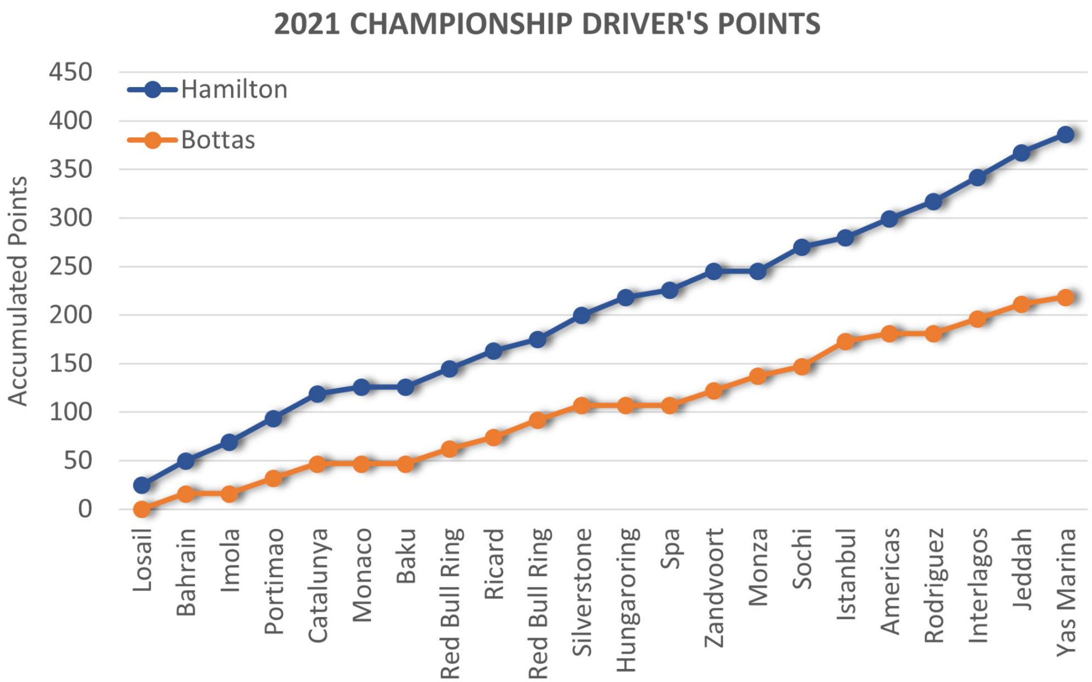
FINAL RACE: GO!



```
# step 1: select cumulative sum of Hamilton's points
# over every Grand Prix in 2021 using SUM()
SELECT
    cir.circuitRef, SUM(res.points)
    OVER(ORDER BY res.raceId) AS 'total points'
FROM results res
INNER JOIN races rac
INNER JOIN circuits cir
ON
    res.raceId = rac.raceId AND
    rac.circuitId = cir.circuitId
WHERE
    res.driverId = 1 AND -- change driver id
    res.raceId IN (
        SELECT raceId
        FROM races
        WHERE year = 2021 -- change championship year
        )
ORDER BY res.raceId;
# step 2: Find driverId of Hamilton's teammate in 2021
SELECT driverId, surname, nationality, code
FROM f1.drivers
WHERE surname
LIKE '%bottas%';
```

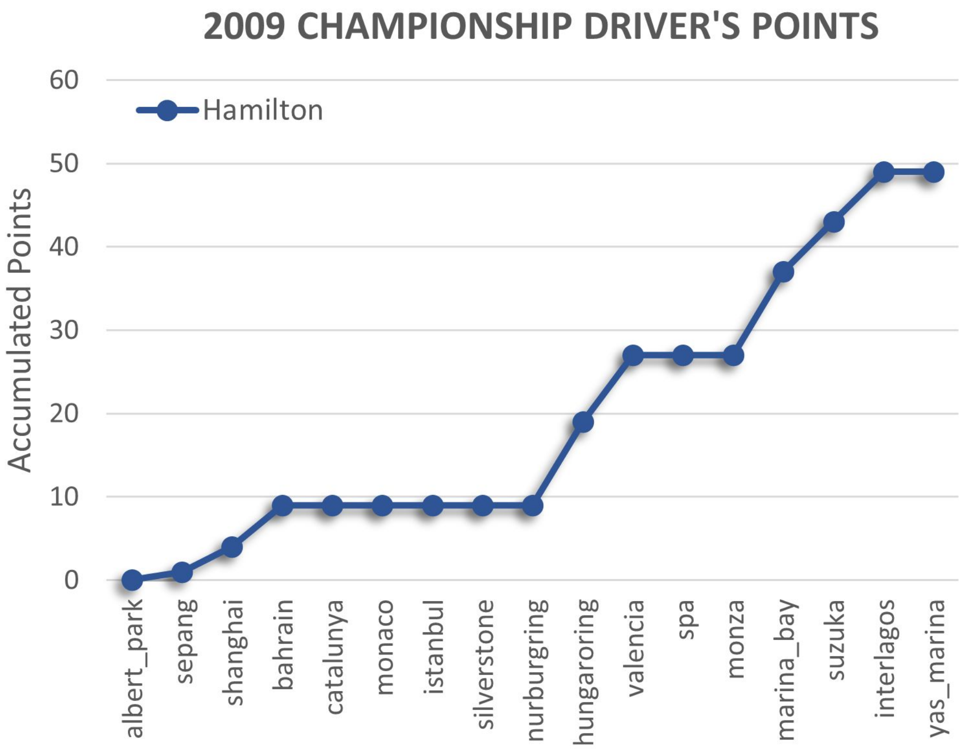
SUM()

The *SUM()* function can be used to find the cumulative number of points each driver earns as the season progresses.

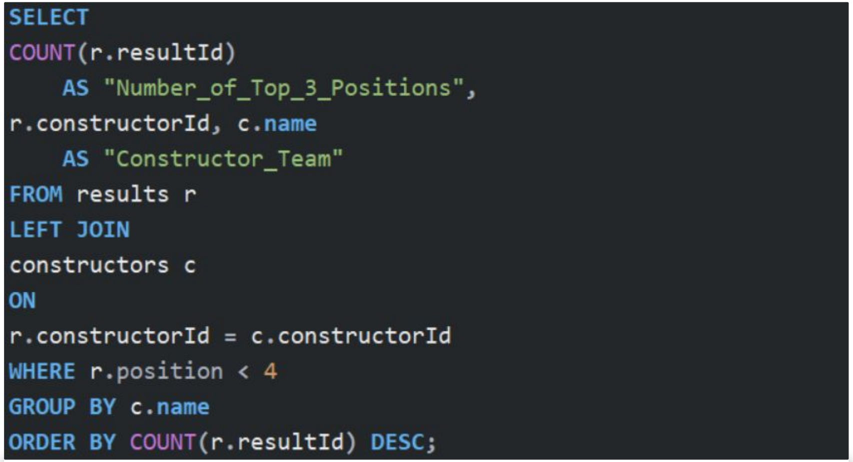


The above graph shows the two teammates, Hamilton and Bottas', points haul across last year's races. From this we can identify the weaker circuits for the two, so that we can start to pinpoint areas where one of them might be lacking. By comparing this season's points climb to previous season's, we can also track their progress across the years. Aside from a couple of races, the graph shows a pretty consistent and steady points climb…

FINAL RACE: GO!



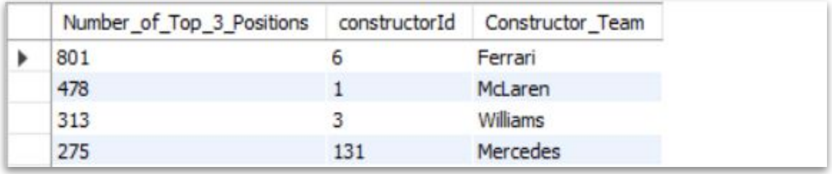
…which is an improvement to some of the more stagnant previous years.



```sql
SELECT
COUNT(r.resultId)
    AS "Number_of_Top_3_Positions",
r.constructorId, c.name
    AS "Constructor_Team"
FROM results r
LEFT JOIN
constructors c
ON
r.constructorId = c.constructorId
WHERE r.position < 4
GROUP BY c.name
ORDER BY COUNT(r.resultId) DESC;
```

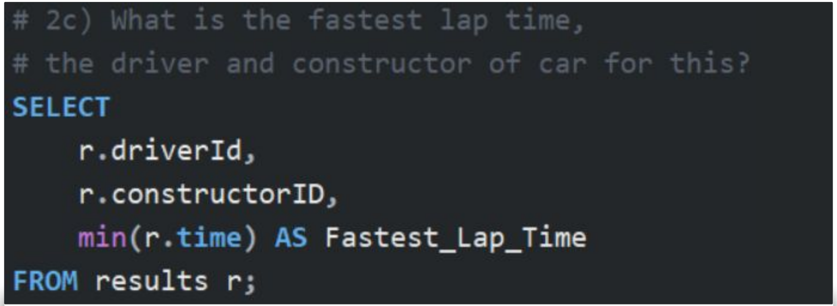
COUNT()

Use of the *COUNT()* function allows for a list of the frequency at which a team is placing top 3 at the final race, ordered from highest to lowest. As a competitor to these teams, it is always useful to look to what the most successful teams are doing and identifying the teams is the first step in doing so. The output is shown below



	Number_of_Top_3_Positions	constructorId	Constructor_Team
▶	801	6	Ferrari
	478	1	McLaren
	313	3	Williams
	275	131	Mercedes

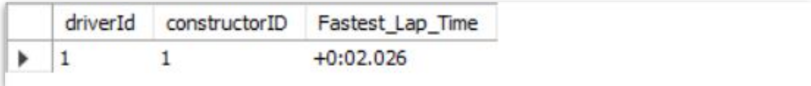
FINAL RACE: GO!



```
# 2c) What is the fastest lap time,
# the driver and constructor of car for this?
SELECT
    r.driverId,
    r.constructorID,
    min(r.time) AS Fastest_Lap_Time
FROM results r;
```

MIN()

Using the aggregate function *MIN()* a general query can be created that returns the fastest lap time. If we wanted to get into the specifics we would use *WHERE* to look at the fastest lap times for a particular driver, or for a particular race, to query more specific figures that can be used as benchmarks for how well Hamilton is performing. The code output is shown below.



	driverId	constructorID	Fastest_Lap_Time
▶	1	1	+0:02.026

These are just some of the ways we imagine the coding we have done can be used strategically between races for improvements, and to increase the chances of earning more points.