|  |  |
| --- | --- |
| *Alexandra Speer*  *Nicolas Henzel*  *Gruppe 2*  *Hochschule der Medien*  *Sommersemester 23* |  |
|  |  |

Data Warehouse Workshop  
Prof. Dr. Klaus Freyburger

Inhaltsverzeichnis

[Daten 2](#_Toc132531475)

[Logisches Modell 3](#_Toc132531476)

[Design der Umsetzung 4](#_Toc132531477)

[Data Wrangling 4](#_Toc132531478)

[Beispielhafte Auswertungen 5](#_Toc132531479)

[Implementierung 7](#_Toc132531480)

[Auswertung to-do 10](#_Toc132531481)

[Anmerkungen für Vorgehen 14](#_Toc132531482)

[Quellen 15](#_Toc132531483)

# Daten

Die Datensätze stammen von Kaggle[[1]](#footnote-1). Inhalt sind die Weltmeisterschaften der Formel 1 im Zeitraum von 1950 bis 2023. Die Daten sind durch die Ergast Developer API[[2]](#footnote-2) zusammengestellt worden, die Renndaten für nicht kommerzielle Zwecke zur Verfügung stellt.

Die Formel 1 (oft auch F1) ist eine Formelserie, die durch den Automobil Dachverband Fédération Internationale de l’Automobile (FIA) autorisiert ist. Formelserie bedeutet hierbei, dass bestimmte Regeln (Formeln) auf technischer Ebene für die Leistungsfähigkeit der Fahrzeuge festgelegt wurden, um einen Wettkampf unter gleichbleibenden Bedingungen zu ermöglichen. Die Formel 1 Weltmeisterschaft fand erstmals 1950 statt und besteht pro Saison aus bis zu 23 Grand Prix (französisch für „Großer Preis) Rennen. Dies sind Einzelrennen auf ausgewählten Rennstrecken in jeweils unterschiedlichen Ländern. Dabei sammeln die Fahrer abhängig von ihrer Endposition bei diesen Rennen Punkte. Am Ende der Saison gewinnt der Fahrer mit den meisten Punkten.  
Außerdem erhalten die Konstrukteure der Wagen Punkte, die ebenfalls am Ende der Saison ausgewertet werden.[[3]](#footnote-3)

Die Daten beinhalten alle Informationen der Formel 1 Weltmeisterschaften seit 1950:  
Rennen, Fahrer, Konstrukteure, Qualifizierungen, Rennstrecken, Rundenzeiten, Boxenstopps und Ergebnisse. Diese Informationen sind in 14 Datensätze aufgeteilt. Eine Übersicht der Daten in Python findet sich im DataExploration.ipynb.  
Die wichtigsten Erkenntnisse aus dieser Übersicht sind:

* Zellen, bei denen kein sinnvoller Wert eingetragen ist, sind mit \N markiert.  
  Das wird im Data Wrangling behoben
* Die Daten sind bereits „tidy“:  
  Jede Zeile beinhaltet eine Beobachtung,   
  jede Spalte beschreibt eine Variable,   
  in jeder Zelle ist genau ein Wert (und nicht mehrere).[[4]](#footnote-4)  
  Ausnahme hierzu bilden die Tabellen „races“ und „qualifying“, in der einige Werte fehlen.
* Die Datensätze besitzen mindestens eine Id Spalte (Primärschlüssel). wodurch sich die Daten leicht referenzieren lassen. Manche Datensätze haben weitere Id Spalten aufgelistet, die als Fremdschlüssel verwendet werden können.

Zusätzlich haben wir eine Tabelle zum Mapping der Länder und Nationalitäten genutzt.  
Die Daten dazu haben wir von einem öffentlichen GitHub Repository heruntergeladen:  
<https://github.com/Imagin-io/country-nationality-list/blob/master/countries.csv>

# Logisches Modell



# Design der Umsetzung

## Data Wrangling

Die Daten wurden im ersten Schritt im DataExploration.ipynb untersucht und in einer flachen Form zusammengeführt. Dabei haben wir uns auf die Fahrer Daten fokussiert und die Id Spalten genutzt, um die unterschiedlichen Datensätze miteinander zu joinen. Die Konstrukteur Daten sind im ER-Modell enthalten. Die Struktur dient zur ersten Übersicht, sowie beispielhaften Verbindung der Datensätze zu einem finalen Datensatz. Damit wurden zusätzlich Auswertungen beispielhaft im Python Skript umgesetzt.

Die Datensätze, die genutzt wurden, sind:

* circuits.csv
* driver\_standings.csv
* drivers.csv
* lap\_times.csv
* pit\_stops.csv
* results.csv
* races.csv
* status.csv
* constructors.csv
* constructor\_standings.csv
* constructor\_results.csv
* qualifying.csv
* sprint\_results.csv

Der seasons.csv Datensatz wurde nicht genutzt, da hier keine neuen Informationen enthalten sind.

Zum initialen Laden der Tabellen in DWC wurde eine JSON-Datei erstellt (180\_FormulaOne.json), die die Tabellen und Verbindungen anlegt.  
Folgende Anpassungen haben wir vorgenommen:

* Driver\_standings.csv - positionText Spalte entfernt
* Pit\_stops.csv - duration Spalte entfernt
* Results.csv - positon, positionText Spalte entfernt
* Constructor\_standings.csv - positionText Spalte entfernt
* Sprint\_results.csv - positon, positionText Spalte entfernt
* Alle „URL“ Spalten in jedem Datensatz entfernt (für die Auswertung nicht relevant)
* Werte in allen mit \N markierten Zellen werden entfernt, damit beim Import in DWC korrekt NULL eingetragen werden kann

Die entfernten Spalten haben keinen Mehrwert geboten, da alle Informationen bereits in anderen Spalten vorhanden waren.Die konkreten Schritte sind im DataWrangling.ipynb nachzuvollziehen.   
Die beispielhaften Auswertungen werden im Anschluss erläutert.

## Beispielhafte Auswertungen

Vorschläge zur Auswertung finden sich in der DataExploration.ipynb Datei.  
Wir wollen damit folgende Fragen beantworten:

* Wie ist das Verhältnis zwischen Start und Endposition?

Chart, bar chart

Description automatically generated

Chart, bar chart

Description automatically generated

* **Welcher Fahrer hat die meisten Grand Prix Rennen gewonnen?**

Chart, bar chart

Description automatically generated

* Welcher Hersteller hat die meisten Weltmeisterschaften gewonnen?
* Welcher Fahrer hat die meisten Weltmeisterschaften gewonnen?
* In welcher Saison sind die meisten Unfälle passiert?
* Welcher Hersteller hat die meisten Defekte / Ausfälle / Unfälle?
* Welche Strecken werden mit den kürzesten, welche mit den längsten Zeiten gefahren?

# Implementierung

Die initiale Implementierung der Tabellen wurde mit der manuell erstellten JSON-Datei durchgeführt, wodurch das E/R Modell per Import in DWC angelegt wurde. Alle weiteren Anpassungen sind in DWC direkt durchgeführt worden.

## Datenqualität für die Analyse

Im Abschnitt Daten haben wir angemerkt, dass in der Tabelle „races“ (in DWC 180\_FormulaOneRaces) einige Werte fehlen. Konkret sind die Spalten I – R (fp1\_date, fp1\_time, …, sprint\_time) nur mit 67 Werten oder weniger befüllt. Aus diesem Grund haben wir hier keine Auswertungen erstellt, da nicht ausreichend Daten vorhanden sind.   
Ähnliches gilt für die Tabelle „qualifying“ und „sprint\_results“ – da hier viele Datenpunkte fehlen, oder zu wenige vorhanden sind (sprint\_results) werden mit diesen Daten keine Auswertungen durchgeführt.

In der Tabelle „results“ ist die Zeit in zwei unterschiedlichen Formaten enthalten. Für den ersten Platz eines Rennens ist in Zeitformat in der Form HH:MM:SS.MS angegeben, alle weiteren Zeiten des gleichen Rennens in der Form +S.MS. Da die Transformation dieser Daten innerhalb von DWC nicht möglich ist, haben wir uns entschieden diese Daten nicht zu verwerten (siehe Anhang).

Beim Importieren der Daten sind einige Inhalte in Spalten, die einen Zeitwert enthalten, nicht korrekt erkannt worden, da diese nicht im H:MM:SS Format vorliegen, sondern M:SS.MS.



## Transformationen und Views

Aus diesem Grund haben wir diese Spalten als String importiert, nachdem die Fehlermeldung in DWC aufgetreten ist. Diese werden anschließend in DWC in das Format H:MM:SS transformiert, indem eine Berechnung genutzt wird:  
TO\_TIME(CONCAT('00:0',SUBSTRING(SPALTENNAME,1,4)))

Die Zeitwerte sind zusätzlich als Millisekunden in den entsprechenden Spalten in Integer Werten vorhanden. Diese konnten auch nach mehrfachen Versuchen nicht in das Format H:MM:SS transformiert werden, da die in DWC vorhandenen Funktionen[[5]](#footnote-5) nicht ausreichend waren, um die Transformation durchzuführen. Die konkreten Schritte sind im Anhang aufgelistet.  
  
Zusätzlich gibt es Spalten mit Zahlen, die auf den ersten Blick wie Integer aussehen, nach dem Import jedoch Float sind, was wir auch anpassen mussten (z.B. points Spalte in results.csv).

Zuerst haben wir die SQL Views 180\_FormulaOneLapTimesGroupedView und 180\_FormulaOnePitStopsGroupedView erstellt. In der ersten View haben wir pro Fahrer je Rennen die einzelnen Rundenzeiten und Rundenpositionen als Mittelwert aggregiert:  




Hierbei konnten wir leider nur die durchschnittlichen Rundenzeiten in Millisekunden angeben, nicht im Format HH:MM:SS, da die Transformation nicht erfolgreich war (siehe oben). Als eine weitere Idee haben wir versucht die vorhandenen Zeitwerte in Millisekunden umzurechnen, diese Werte zu aggregieren und die Transformation rückgängig zu machen. Leider reicht dafür der Umfang der Funktionen in SAP Datasphere nicht aus (siehe Anhang).

In der zweiten sind pro Fahrer je Rennen die Boxenstopps und durchschnittliche Dauer aggregiert:  


Table

Description automatically generated

Anschließend haben wir folgende Views für die Erstellung von Stories angelegt:

* 180\_FormulaOneLapTimeAnalysis – Auflistung der einzelnen Rundenzeiten und Positionen für jeden Fahrer je Rennen
* 180\_FormulaOnePitStopAnalysis - Auflistung der einzelnen Pit Stops und Stop Dauer für jeden Fahrer je Rennen
* 180\_FormulaOneDriverStandingsAnalysis – Auflistung der finalen Auswertungen der Rennergebnisse für jeden Fahrer je Rennen von 1950 bis 2023
* 180\_FormulaOneDriverStandingsAnalysis – Eingrenzung der oberen View auf FinalPosition = 1 (Fahrer, die erster wurden im jeweiligen Rennen)
* 180\_FormulaOneRaceResultsAnalysis – Analyse aller Rennergebnisse pro Fahrer und Konstrukteur auf jeder Strecke ab 1950 bis 2023
* 180\_FormulaOneAVGLapTimeAnalysis – Analyse der Durchschnittlichen Rundenzeit pro Fahrer pro Strecke. Auflistung der Jahre, um Werte vergleichbar zu machen.

**Unser E/R Modell in DWC sieht folgendermaßen aus (180\_FormularOne):**

Diagram, schematic

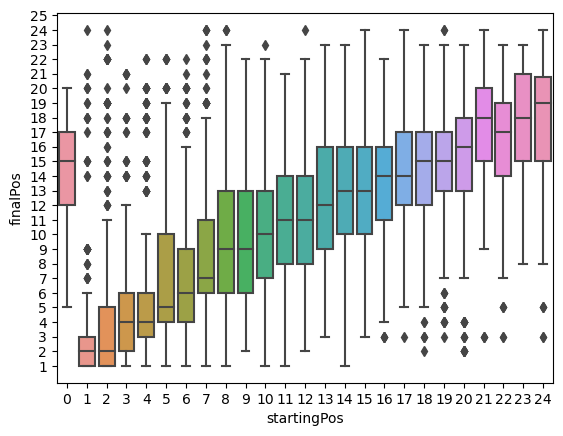
Description automatically generated

# Auswertung to-do

Vorschläge zur Auswertung finden sich in der DataExploration.ipynb Datei.  
Wir wollen damit folgende Fragen beantworten:

* Wie ist das Verhältnis zwischen Start und Endposition?

Anm.: Vergleich zu Python, Start- und Endpositionen in keiner View mehr enthalten?



* Siege nach Startposition



* **Welcher Fahrer hat die meisten Grand Prix Rennen gewonnen?**



* Gewonnene Rennen nach Konstrukteur



* Siege nach Nation des Fahrers (Dafuer mappe ich noch Nationalitaet mit Land)



Hierarchie damit Drill Down moeglich auf Fahrer pro Nationailtaet

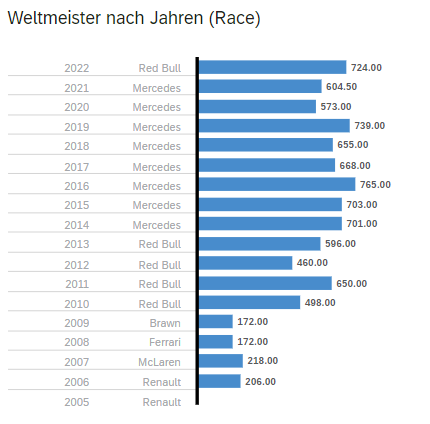
Problem: Semantic Use muss auf “Dimension” gestellt werden, damit eine Hierarchie erstellt werden kann

* Siege nach Nation des Herstellers



* Welcher Hersteller hat die meisten Weltmeisterschaften gewonnen?

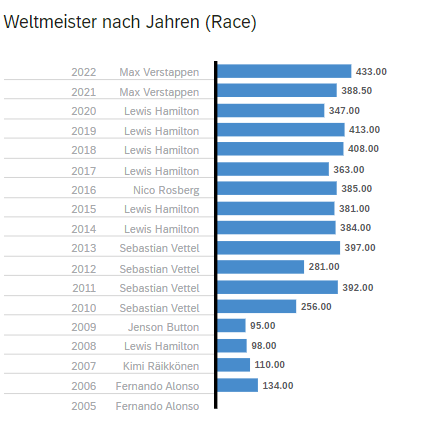
*Wer hat die meisten Punkte? Erst seit 1958 für Hersteller.*

**

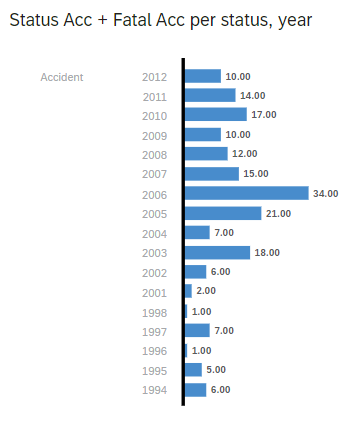
*Vgl: https://www.formula1.com/en/results.html/2022/drivers.html*

* Welcher Fahrer hat die meisten Weltmeisterschaften gewonnen?

*Wer hat die meisten Punkte?*

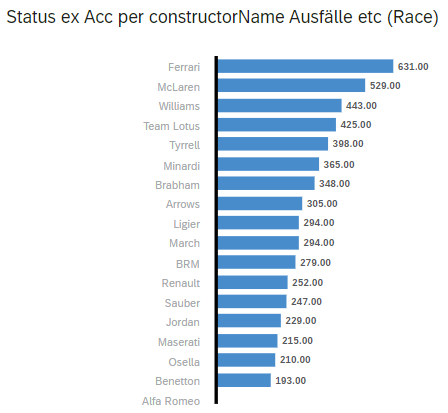


* In welcher Saison sind die meisten Unfälle passiert?



* Welcher Hersteller hat die meisten Defekte / Ausfälle / Unfälle?

Anm.: Evtl. Drill down zu Status? 🡪 Hierarchie s.o.

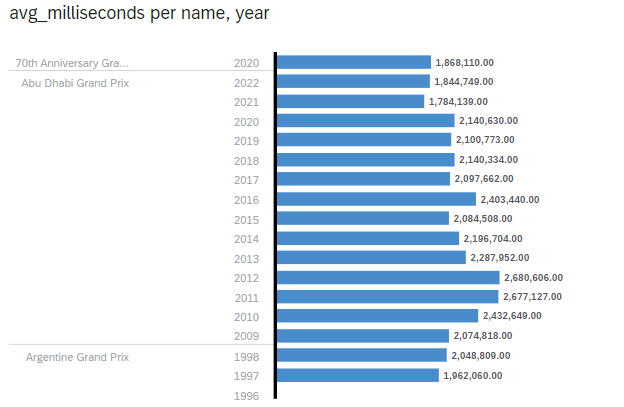


* Welche Strecken werden mit den kürzesten, welche mit den längsten Zeiten gefahren?

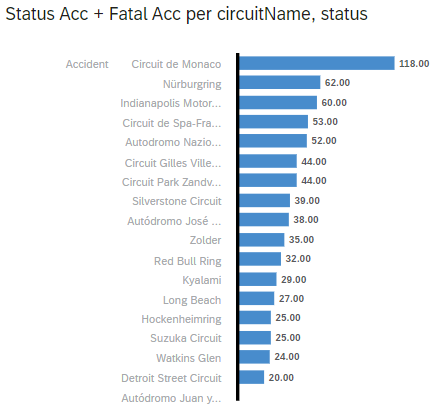


* Wie veraendern sich die Zeiten im Verlauf der Jahre  
  GGF auch als Drill Down abbilden (Hierarchie Jahre hinzufuegen)

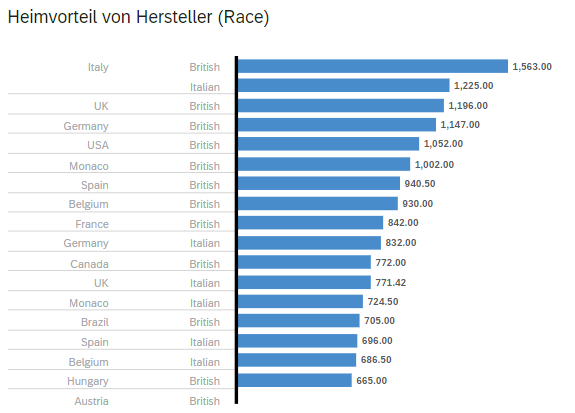
Anm.: Können wir aus dem Jahr noch ein Datumsdimension machen? Dann könnte man evtl. als Diagramm eine Time-Series nehmen und wir hätten man eine andere Art.

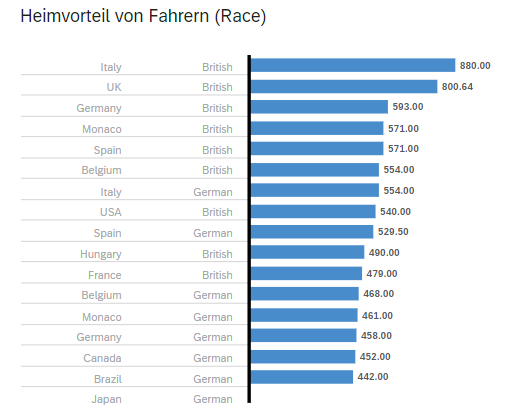


* Was sind die gefaehrlichsten Stecken (mit den meisten unfaellen / Ausscheidungen)



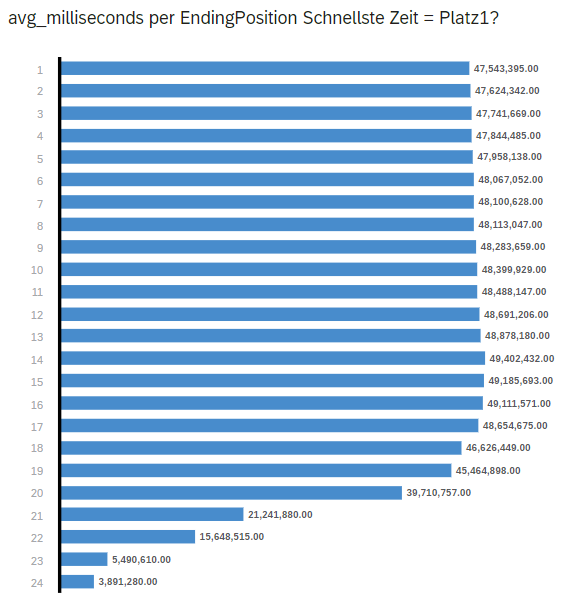
* Gibt es Fahrer/Konstrukteure, die Heimvorteile haben (Mapping Nationality und Country hilfreich)





* Schnellste Zeit auch auf Platz 1?

Anm.: Kannst du dir die Zeiten auf den letzten Startplätzen erklären? Wurde nicht zu Ende gefahren?



# Anmerkungen für Vorgehen

Nächste Schritte:

Nico

Alexandra

Beide

* Views bauen  
  - Aggregation von lap times zu grouped lap times in SQL View nachbauen   
  - View in ERM einbinden  
  - Rennzeiten Format in View bearbeiten, um HH:MM:SS zu erhalten  
  - Offen:
* LapTimes Aggregieren – Durchschnittliche Rundenzeit errechnen pro Fahrer pro Rennstrecke pro Jahr und Final Position anhaengen  
  - Views zur Beantwortung der Fragen bauen (z.B. Gruppierung der Wins pro Fahrer)
* Hierarchien auf Ebene Nationalität / Land, um Auswertungen für Fahrer, Konstrukteure, Strecken zu ermöglichen  
  - Ausprobieren wie die Hierarchie aufgebaut werden muss
* Stories bauen, um Auswertungen darzustellen – iterative Anpassungen an Datengrundlage, um Auswertungen korrekt abzubilden

# Quellen

<https://www.kaggle.com/datasets/rohanrao/formula-1-world-championship-1950-2020>  
Zuletzt besucht am 30.03.2023 19 Uhr

<http://ergast.com/mrd/>  
Zuletzt besucht am 30.03.2023 19 Uhr

<https://de.wikipedia.org/wiki/Formel_1>  
Zuletzt besucht am 30.03.2023 19 Uhr

<https://cran.r-project.org/web/packages/tidyr/vignettes/tidy-data.html>  
Zuletzt besucht am 30.03.2023 19 Uhr

# Anhang

Versuch zur Umwandlung von Millisekunden in HH:MM:SS Format per SQL View:

1. SELECT CONVERT(varchar(8), DATEADD(ms, "milliseconds", CAST('00:00:00' as time)), 108) as mytime

FROM "180\_FormulaOneLapTimes"

Graphical user interface, text, application

Description automatically generated

1. SELECT TIME\_FORMAT(SEC\_TO\_TIME("milliseconds" / 1000), '%H:%i:%s') AS my\_time

FROM "180\_FormulaOneLapTimes"

Text

Description automatically generated with low confidence

Versuch Umwandlung Time Format in Milliskeunden, anschließende Aggregation und rückwärts Transformation per SQL View:

1. SELECT "driverId", "raceId",

AVG(UNIX\_TIMESTAMP(TO\_TIME(CONCAT('00:', SUBSTRING("time", 1, 2), ':', SUBSTRING("time", 4, 2), '.000')))) AS "lapTime"

FROM "180\_FormulaOneLapTimes"

GROUP BY "driverId", "raceId"

Text

Description automatically generated

1. SELECT "driverId", "raceId",

SEC\_TO\_TIME(AVG(TIME\_TO\_SEC(TO\_TIME("lapTime")))\*1000) AS "AVGlapTime"

FROM "180\_FormulaOneConvertedLapTimes"

GROUP BY "driverId", "raceId"

Table

Description automatically generated with medium confidence

1. <https://www.kaggle.com/datasets/rohanrao/formula-1-world-championship-1950-2020> [↑](#footnote-ref-1)
2. <http://ergast.com/mrd/> [↑](#footnote-ref-2)
3. <https://de.wikipedia.org/wiki/Formel_1>  
    [↑](#footnote-ref-3)
4. https://cran.r-project.org/web/packages/tidyr/vignettes/tidy-data.html [↑](#footnote-ref-4)
5. https://help.sap.com/docs/SAP\_HANA\_PLATFORM/4fe29514fd584807ac9f2a04f6754767/20a61f29751910149f99f0300dd95cd9.html [↑](#footnote-ref-5)