Hochschule Luzern – Wirtschaft

Master of Science in Applied Information and Data Science

Data Collection, Integration, Preprocessing – Projektarbeit (LN2)

ETL – Airlines Daten USA 2015

16. April 2020

FS 20

Stefan Berchtold

Marius Gisler

Remo Oehninger

Gianni Pinelli

Hochschule Luzern – Wirtschaft

Master of Science in Applied Information and Data Science

Modul:	Data Collection, Integration, Preprocessing
Art der Arbeit:	Projektarbeit (LN2)
Titel:	ETL – Airlines Daten USA 2015
Datum:	Luzern, 16. April 2020

Verfasser:

Stefan Berchtold	stefan.berchtold@stud.hslu.ch
Marius Gisler	marius.gisler.01@stud.hslu.ch
Remo Oehninger	remo.oehninger@stud.hslu.ch
Gianni Pinelli	gianni.pinelli@stud.hslu.ch

Dozent:

Erwin Mathis	erwin.mathis@hslu.ch

Tabellenverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

1.1 Aufgabestellung

Als Teil des Moduls Data Collection, Integration, Preprocessing (CIP) wurde eine Extract-Transform-Load-Prozess (ETL-Prozess) in einer Gruppenarbeit erarbeitet. Dieser Prozess wird in diesem Dokument beschrieben und zusätzlich als PowerPoint-Präsentation mit Tonspur erklärt.

Für das Projekt sollen Daten aus zwei verschiedenen Datenquellen ausgewählt werden und anschliessend in verschiedenen Tools bearbeitet, bereinigt und zusammengeführt werden. Zum Schluss werden die Daten auf eine Datenbank importiert und die von der Gruppe gestellte Fragestellung beantwortet.

1.2 Fragestellung

Für das Projekt wurden Daten von Inlandflügen der USA aus dem Jahr 2015 verwendet. Dazu wurden drei Datentabellen aus dem Internet verwendet, welche aufbereitet und analysiert werden sollen. Zudem wird mittels eines selbst programmierten Website-Crawlers in Python eine weitere Datentabelle erstellt, welche mit den Datentabellen aus dem Internet gemappt wird.

Die abschliessende Analyse der Daten soll aufzeigen, welche der Airlines die meisten verspäteten, pünktlichen, annullierten und umgeleiteten Flüge verbucht haben.

1.3 Projektziele

Folgende Projektzeile wurden zusätzlich definiert:

- Wir gewinnen Wissen rund um die Datenextraktion, Datentransformation und das Bereitstellen von Daten.
- Wir crawlen Websites.
- Wir bereinigen und vereinen die gecrawlten Daten mit Python Pandas.
- Wir bereinigen und vereinen heruntergeladene Datensätze mit Tableau Prep.
- Wir laden die Daten auf eine Datenbank.
- Wir präsentieren unser Projekt und die gewonnenen Ergebnisse mittels eines Videos.
- Wir erstellen eine interessante und aufschlussreiche Projektbeschreibung.
- Wir erreichen die gesteckten Ziele und haben Spass bei der Erarbeitung des Projekts.

Auf die oben erwähnten Projektziele wird am Schluss dieser Arbeit eingegangen und als Gruppe darüber reflektiert.

2. Datenquellen und Daten

Um unsere Fragestellung zu beantworten, nutzen die Autoren zwei unabhängige Datenquellen. Die erste Quelle ist die Website kaggle.com. Seit 2017 ist die Plattform für Data Science Anwender ein Tochterunternehmen von Alphabet Inc. und hat mittlerweile über 1 Million Nutzer[1]. Unter «https://www.kaggle.com/usdot/flight-delays» sind die CSV-Dateien «flights.csv», «airline.csv» und «airport.csv» bezogen worden.

Die Datei «flights.csv» beinhaltet Informationen zu allen Inlandflugbewegungen, die in der USA im Jahr 2015 von den 14 ausgewählten amerikanischen Airlines durchgeführt wurden. Der Datensatz enthält unter anderem den Abflugflughafen und den Ankunftsflughafen, allfällige Verspätungen aufgrund des Wetters und den Grund annullierter Flüge. Die zweite Datei «airline.csv» liefert Informationen bezüglich der Airline, wie den IATA-Erkennungscode. Die dritte Datei «airports.csv» beinhaltet Informationen zu den Flughäfen, wie zum Beispiel die Breiten- und Längengrade.

Die zweite Datenquelle ist Wikipedia. Die Wikipedia-Tabelle «https://de.wikipedia.org/wiki/Liste_von_Fluggesellschaften» stellt Informationen über weltweit operierende Airlines zur Verfügung. Als Erweiterung der bereits bezogenen Daten wurde die Spalte 'Rufzeichen' und den 'IATA-Code' der ausgewählten amerikanischen Airlines extrahiert.

Anschliessend wurde diese Information in Python mit Pandas bereinigt. Die Formatierung wurde so gewählt, dass sie einstimmig formatiert ist wie die Datei «airlines.csv» von kaggle. Weiter wurde Pandas eingesetzt, um die beiden Dateien zu vereinen (engl. merge). Daraus ergab sich für unser Projekt die dritte Datei, «airlines_merged.csv». Diese Dateien wurden im Tableau Prep zu der Datei «Inlandfluege_usa.csv» zusammengeführt.

[1] https://techcrunch.com/2017/03/07/google-is-acquiring-data-science-community-kaggle/

3. Software / Verwendete Tools

In der Projektarbeit wurde verschiedenste Software eingesetzt. In diesem Kapitel wird die Software kurz beschrieben. Alle beschriebene Software wurde auf einer Virtuellen Maschine der Hochschule Luzern eingesetzt. Näher auf die Softwarenutzung wird in den darauffolgenden Kapiteln eingegangen.

3.1 Python

Python ist eine der meist genutzten interpretierbaren Programmiersprachen weltweit[2]. Die Autoren nutzten die Programmiersprache, um anhand eines Webcrawlers Daten zu sammeln und diese anschliessend zu bereinigen und zu vereinen.

Innerhalb der Programmiersprache arbeiteten die Autoren mit PyCharm und dem Project Jupyter Notebook. Beide Programme eignen sich für das Interpretieren und dem Coden von Python Programmen.

Als hauptsächliche Programmbibliotheken nutzten die Autoren Selenium, BeautifulSoup und Pandas. Selenium wurde eingesetzt, um einen Chromedriver des Webbrowsers Chrome automatisch zu steuern. BeautifulSoup ist ein Paket, das sich zum Parsen von HTML Webseiten eignet. Pandas ist ebenfalls ein Paket, das mit Python genutzt werden kann. Der Nutzen von Pandas liegt vor allem in der Visualisierung und Aufbereitung von grossen Datensätzen.

3.2 Tableau

Zur Aufbereitung und Visualisierung der heruntergeladenen und gecrawlten Daten nutzten die Autoren Software von Tableau Software, Inc. In einem ersten Schritt diente Tableau Prep Builder als Aufbereitungstool und in einem zweiten Schritt Tableau Desktop als Visualisierungstool.

3.3 Microsoft SQL-Server

Der Microsoft SQL-Server diente dem Projekt als Datenbank-Management-System. Der Server wurde genutzt, um Daten nach dem Bereinigen mit Tableau Prep hochzuladen, um später von Tableau Desktop herunterzuladen.

[2] https://www.zdnet.com/article/programming-languages-python-developers-now-outnumber-java-ones/

4. ETL Prozess / Lösungsschritte

Der ETL Prozess, kurz für Extraktion, Transformation und Load beschreibt den ganzen Vorgang bei dem verschiedene Datenquellen extrahiert, bereinigt und in eine Datenbank geladen werden. In Abbildung 1 wird dieser Prozess mit den eingesetzten Tools veranschaulicht. Zusätzlich zum ETL-Prozess ist hier der Visualisierungsschritt abgebildet.

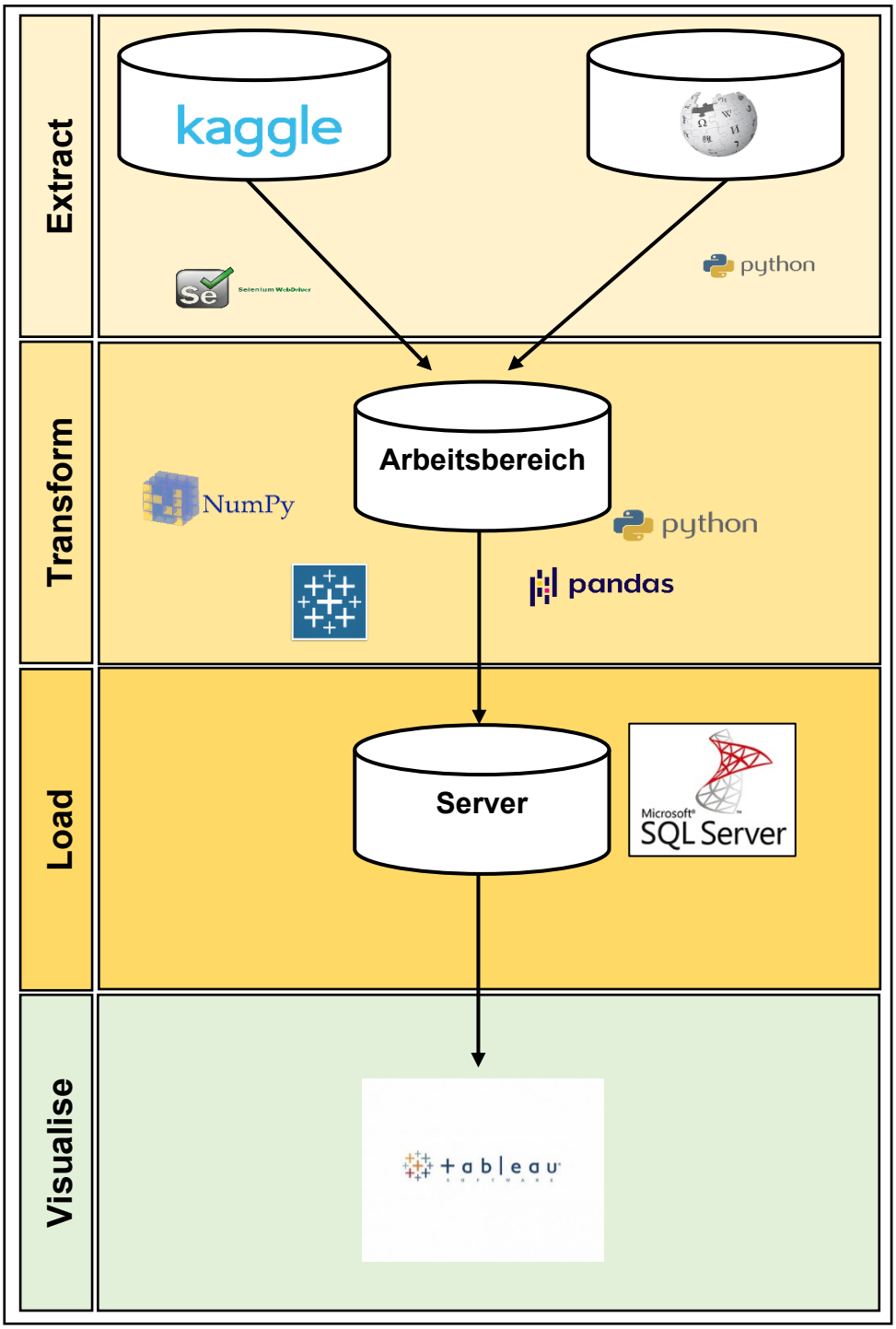

Abbildung 1: ETL mit Visualisierung Prozess

4.1 Extract - Python

Wie im Kapitel 3 beschrieben, diente Python zur Daten-Extraktion, dem Bearbeiten und dem Vereinen der Daten. Alle Python Codes sind im Anhang B ersichtlich.

4.1.1 Crawler_Kaggle.py

Die Daten von kaggle wurden mithilfe eines Webcrawlers von der Website gecrawlt, dazu wurde ein «Google Chrome Chromedriver» mit dem Modul Selenium als Webdriver genutzt. Der Chromedriver ist nicht standardgemäss auf der Virtuellen Maschine geladen. Der Download kann unter «https://chromedriver.chromium.org/» getätigt werden. Anschliessend muss die Chromedriver Datei auf der gewünschten Python-Umgebung abgelegt werden.

Der Chromedriver rief ein Browser-Fenster auf und steuerte diesen anhand des Python Codes (Abbildung 2&3). Der Chromedriver öffnete die Website kaggle.com loggte sich mit den Anmelde-Credentials ein und startete den Download des gewünschten Datensets.

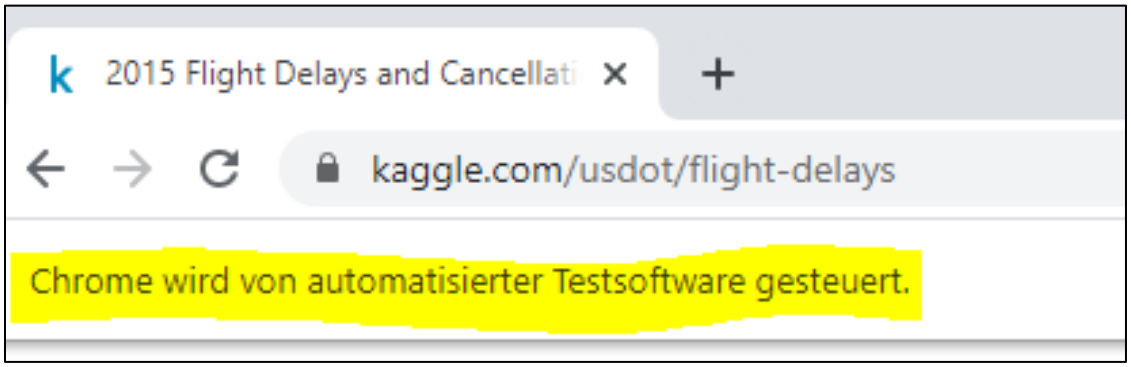

Abbildung 2: Webdriver steuert Chrome Browser

Abbildung 3: Webdriver leitet Download ein

Die Daten wurden als Zip-Ordner heruntergeladen, entpackt und im Dateiordner «Data_Kaggle» als CSV-Dateien abgespeichert.

4.1.2 Crawler_Wikipedia.py

Die Daten aus der Wikipedia-Tabelle IATA-Liste von der Webseite «https://de.wikipedia.org/wiki/Liste_von_Fluggesellschaften» wurden mithilfe von BeautifoulSoup gecrawlt und anschliessend als CSV-Datei «crawler_wikipedia_output.csv» im Ordner «Data_Wikipedia» abgespeichert.

Name	IATA	ICAO	Rufzeichen
ABC Air Hungary	–	AHU	ABC HUNGARY
Abelag Aviation	W9	AAB	ABG
ABS Jets	–	ABP	BAIR
Abu Dhabi Aviation	–	AXU	–
ABX Air	GB	ABX	ABEX
ACM Air Charter	–	BVR	BAVARIAN
ACT Airlines	9T	RUN	CARGO TURK
Aegean Airlines	A3	AEE	AEGEAN
Aer Lingus	EI	EIN	SHAMROCK
Alpine Flight Service	–	–	–
Aero Africa	–	RFC	AERO AFRICA
Aero Charter	DW	UCR	CHARTER UKRAINE
Aero Contractors (Nigeria)	AJ	NIG	AEROLINE
Aero Contractors	–	–	–
Aero-Dienst	–	ADN	AERODIENST
Aero-Service	BF	RSR	CONGOSERV
Avianca Cargo	QT	TPA	TAMPA
Avianca Ecuador	2K	GLG	AVIANCA ECUADOR
Aerogaviota	KG	GTV	AEROGAVIOTA
Aero Mongolia	M0	MNG	AERO MONGOLIA
AeroRescue	–	RES	Search

Abbildung 4: Wikipedia Tabelle

4.1.3 Cleaner.py

Die CSV-Datei wurde anschliessend mittels Pandas gereinigt, fehlende Daten hinzugefügt, um mit einem Datensatz aus dem kaggle-Download zu vereinen.

Der Vereinigungsprozess «Merge» wurde anhand eines Left-Joins auf dem Primary Key «IATA-CODE» durchgeführt. Somit wurden nur die Airline Daten der gewünschten Airlines als Pandas Dataframe abgelegt.

Das Dataframe wurde als CSV-Datei im Ordner «Data_Python» als «airlines_merged.csv» abgespeichert.

4.1.4 Crawler_Airport_ID.py

Das Team hat zu einem späteren Zeitpunkt des Projekts bemerkt, dass einige Daten im Datensatz «flights.csv» für die Verknüpfungen in Tableau Prep nicht geeignet waren. Folglich wurde festgestellt, dass die Daten des ganzen Monats Oktober fehlten. Der ganze Monat Oktober enthielt sogenannte sequenzielle IDs in den Spalten «ORIGIN_AIRPORT» und «DESTINATION_AIRPORT», anstatt der herkömmlichen IATA-Codes.

Um den kaggle-Datensatz «flights.csv» zu vervollständigen wurde ein weiterer Crawler mit Selenium erstellt. Die gecrawlten Daten von der Webseite «https://www.transtats.bts.gov/DL_SelectFields.asp?Table_ID=236&DB_Short_Name=On-Time» wurden in ein zweites CSV-File abgespeichert und anschliessend in einem weiteren Python-File bearbeitet.

4.1.5 Data_kaggle_cleaner.py

In dieser Python-Datei wurde die CSV-Datei «flights.csv» angepasst. Die sequenziellen IDs wurden im Monat Oktober durch IATA-Codes ersetzt und anschliessend als CSV-Datei «flights_new.csv» im Ordner «Data_Kaggle» abgespeichert.

4.2 Transform - Tableau Prep

Nach dem Extract Prozess in Python haben die Autoren in Tableau Prep alle Dateien zu einer Datei «Inlandfluege_usa.csv» zusammengeführt. Die Daten wurden als CSV-Dateien in die Tableau Prep geladen.

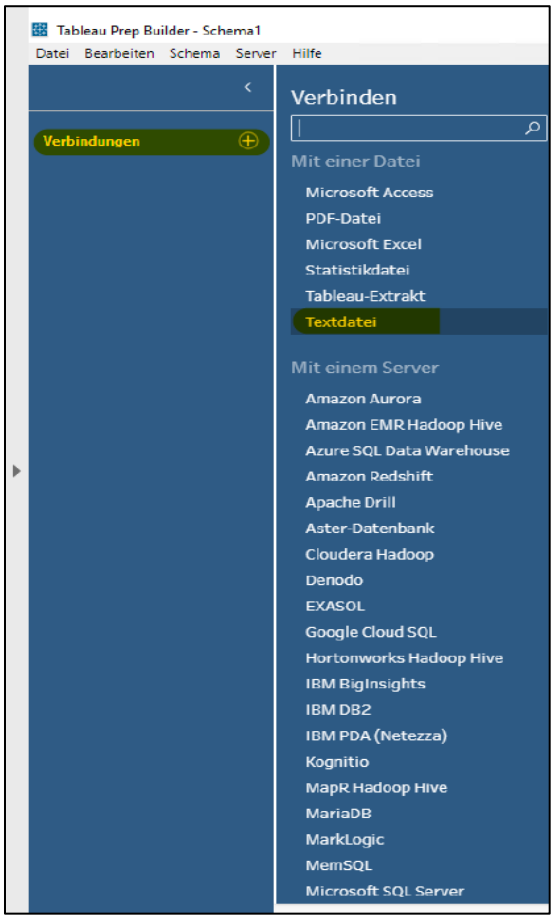

Abbildung 5: Einlesen von Dateien ins Tableau Prep

Als nächstes wurde mit der Funktion “Schritt hinzufügen” die CSV-Dateien «flights_new.csv» und «airports.csv» der erste Aufbereitungsschritt getätigt.

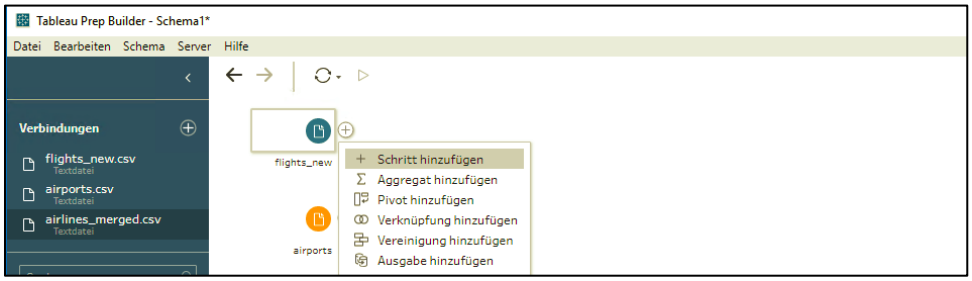

Abbildung 6: Schritt hinzufügen

Durch Klick auf «airports.csv» und per “Drag-and-drop" auf «flights_new.csv» wurden die beiden Dateien miteinander verknüpft.

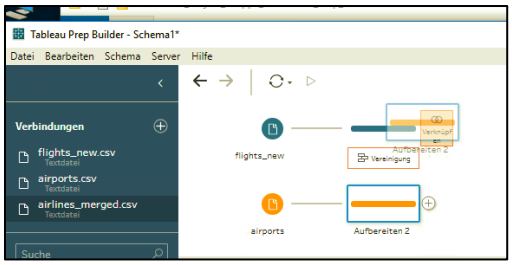

Abbildung 7: Verknüpfung erstellen

Dafür wählten die Autoren die Left-Join-Funktion mit den Spalten «ORIGIN_AIRPORT» und «IATA_CODE» als Join-Klausel.

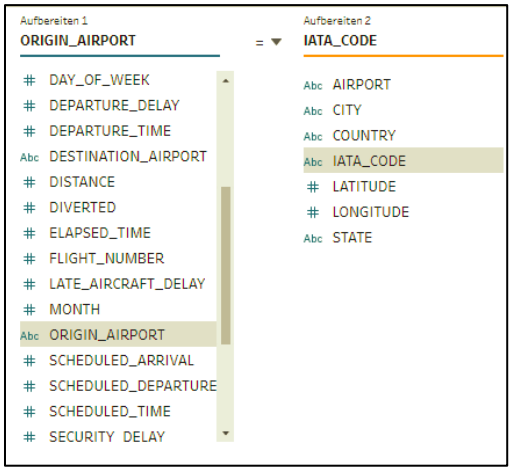

Abbildung 8: Auswahl Spalten für den Left-Join 1

Nach dem Left-Join wurden die im unteren Bild ersichtlichen Änderungen vorgenommen.

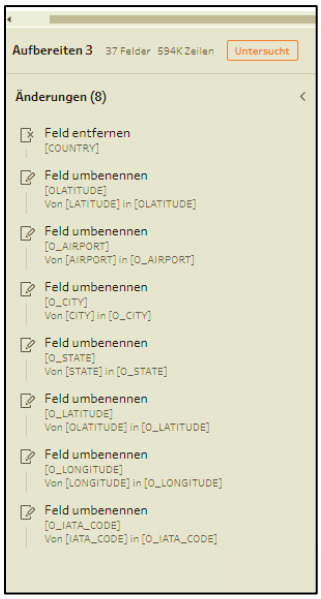

Abbildung 9: Vorgenommene Änderungen 1

Als nächster Schritt wurde wieder eine Left-Join-Verknüpfung zwischen den beiden Dateien «flights_new.csv» und «airports.csv» gemacht.

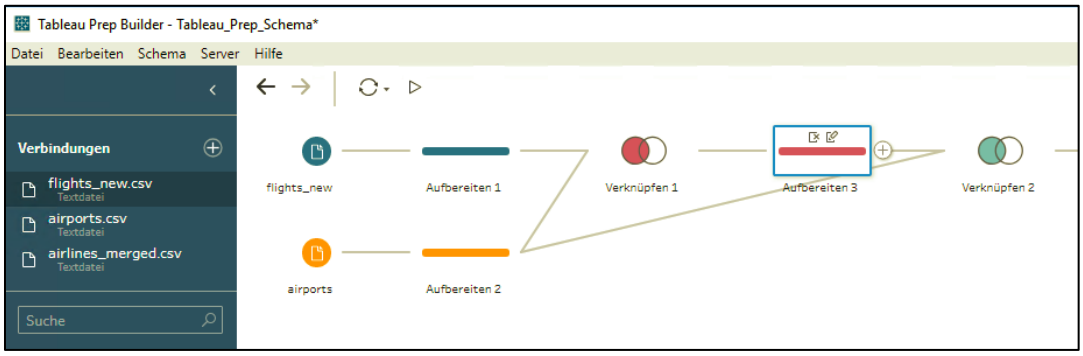

Abbildung 10: Zweite Verknüpfung

Für den zweiten Left-Join wurden die Spalten «DESTINATION_AIRPORT» und «IATA_CODE» als Join-Klausel gewählt.

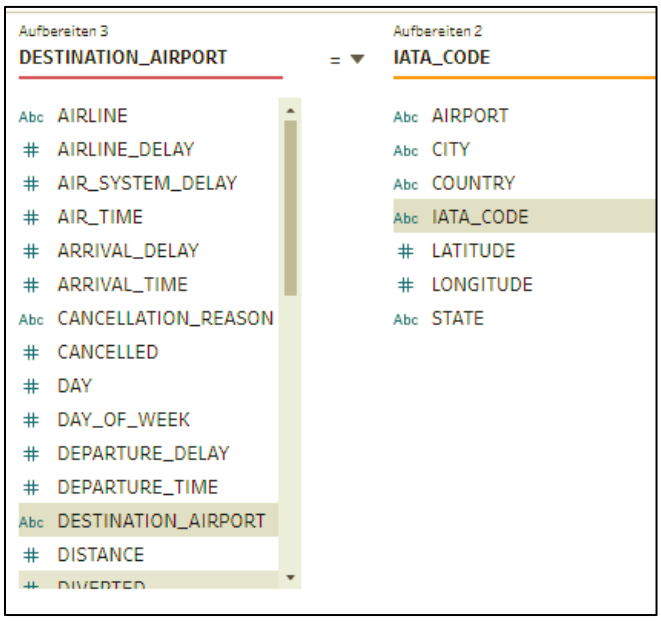

Abbildung 11: Auswahl Spalten für Left-Join 2

Auch hier haben die Autoren mehrere Änderungen vorgenommen.

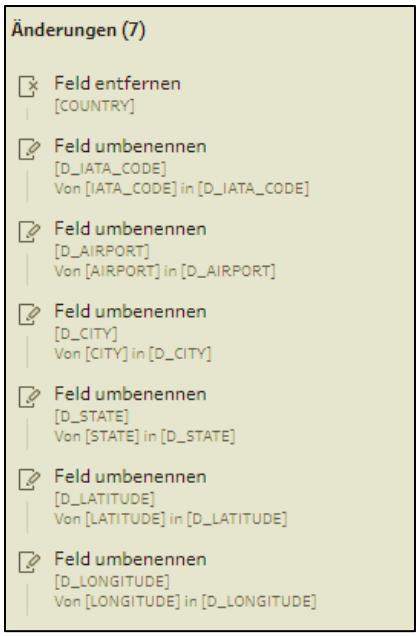

Abbildung 12: Vorgenommene Änderungen 2

Als weiteren Schritt wurde ein Left-Join mit der Datei airlincs_merged.csv getätigt.

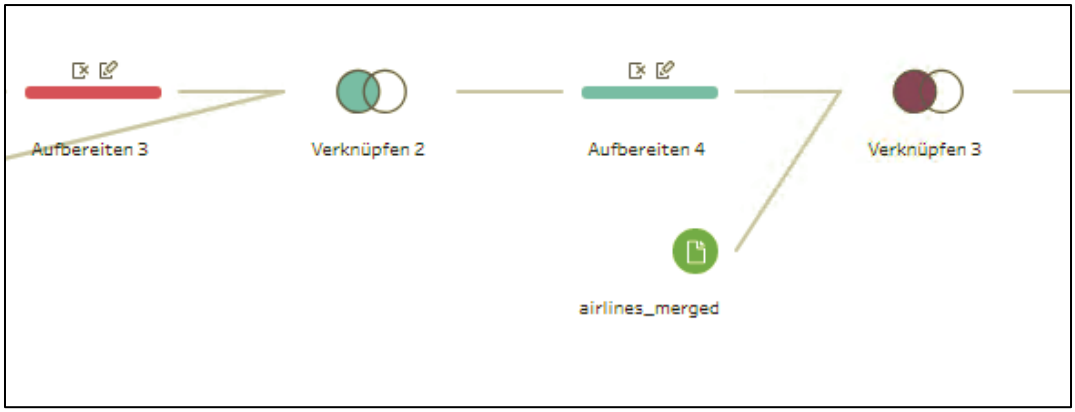

Abbildung 13: Verknüpfung mit airlines_merged.csv

Dafür wurden die Spalten «AIRLINE» und «IATA_CODE» als Join-Klausel festgelegt.

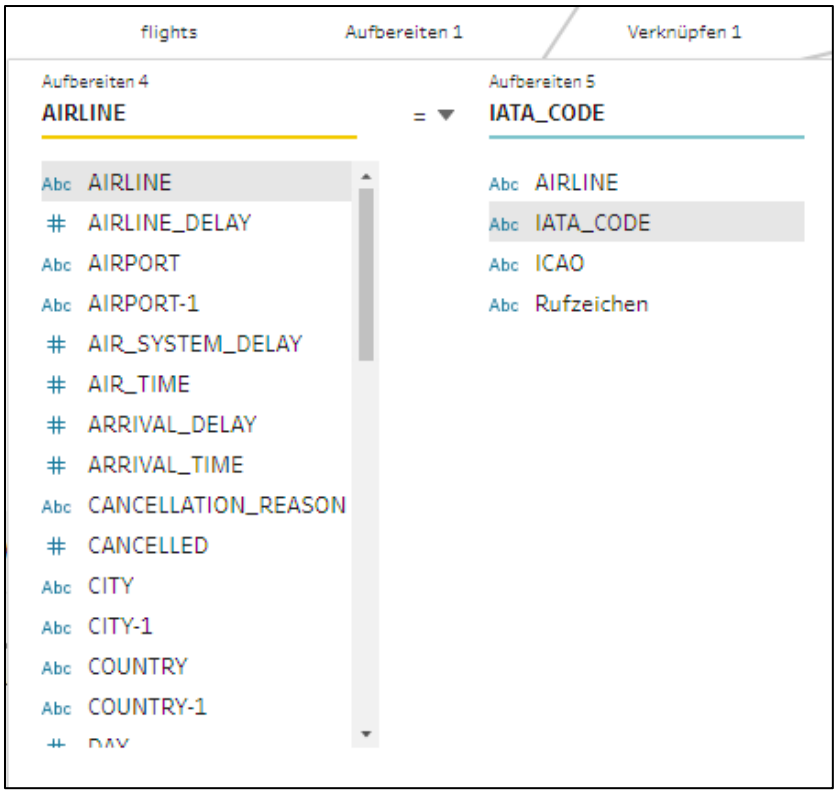

Abbildung 14: Auswahl Spalten Left-Join 3

Im dritten Join wurden folgende Änderungen vorgenommen.

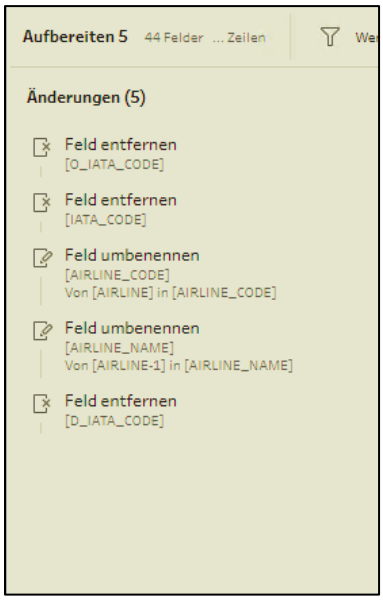

Abbildung 15: Vorgenommene Änderungen 3

Unter Durchsuchen wurde der Speicherort ausgewählt. Der Ausgabetyp CSV wurde bei der Auswahl des Dateiformates selektiert. Danach wurde das Schema ausgeführt. Die Ausführung des Schemas hat bei dieser Grösse ca. sieben Minuten in Anspruch genommen.

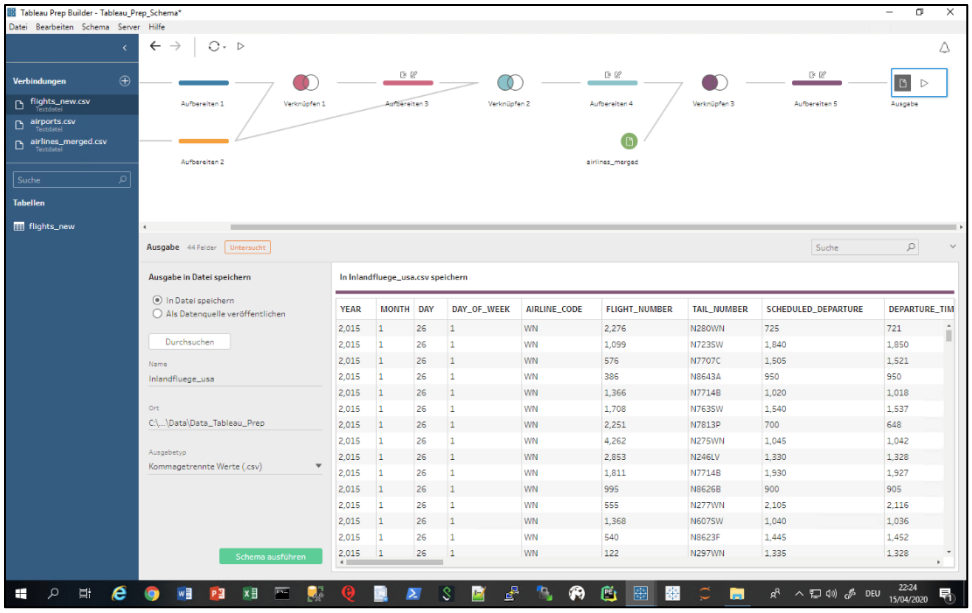

Abbildung 16: Abschluss

4.3 Load – Microsoft SQL-Server

Nach dem Verknüpfen und Bereinigen der Datensätze in Tableau Prep wurden die Daten in eine Microsoft-SQL-Server-Datenbank importiert. Dazu wurde zuerst die Datenbank «CIP» erstellt und anschliessend die Daten über den Wizard in die Datenbank hochgeladen.

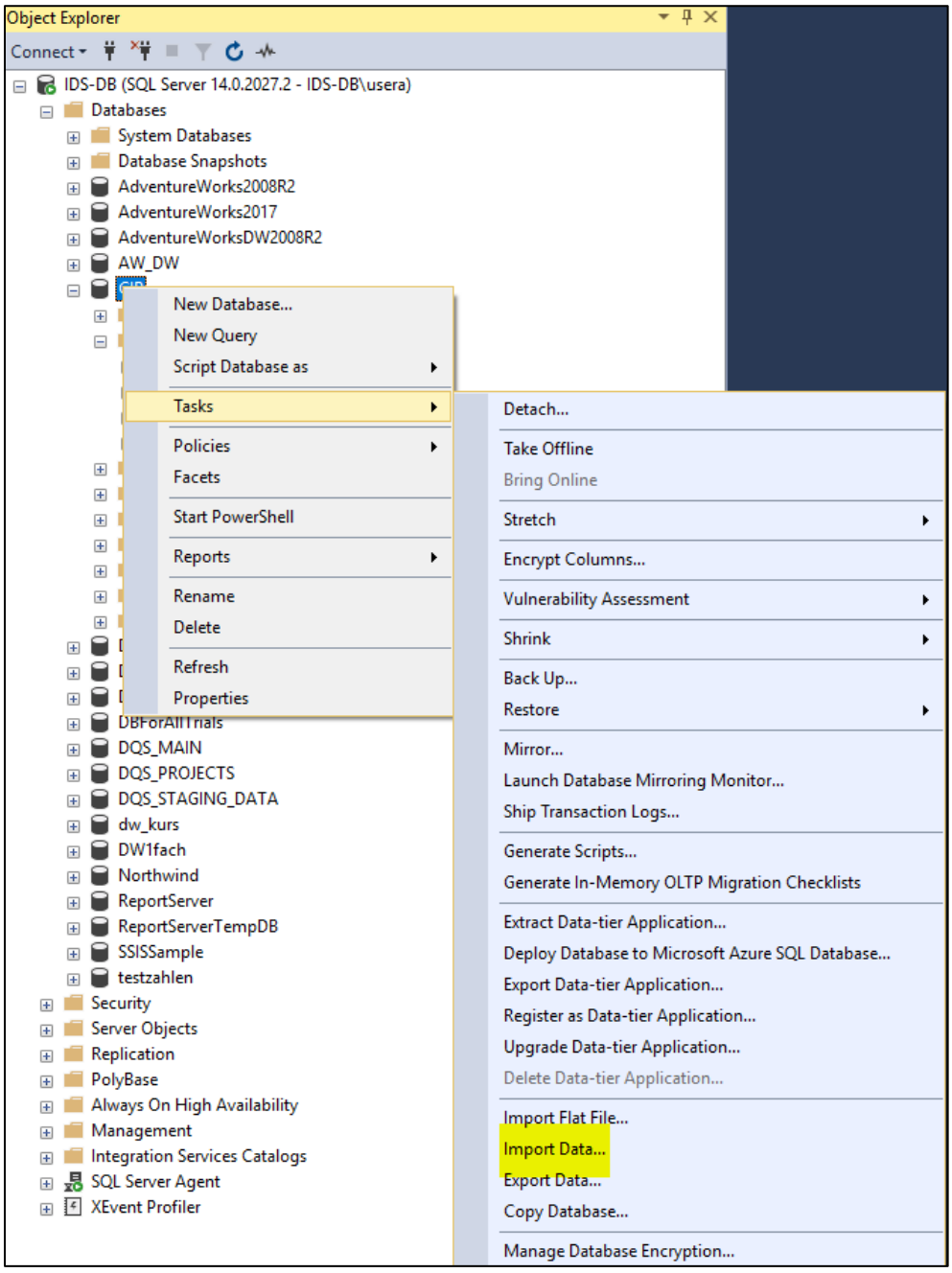

Abbildung 17: Import der Tabelle auf die Datenbank CIP auf dem SQL-Server

Im ersten Schritt des Wizards wird die Datenquelle und der Textqualifier mit dem Doppelhochkomma festgelegt, wie die Abbildung 18 zeigt.

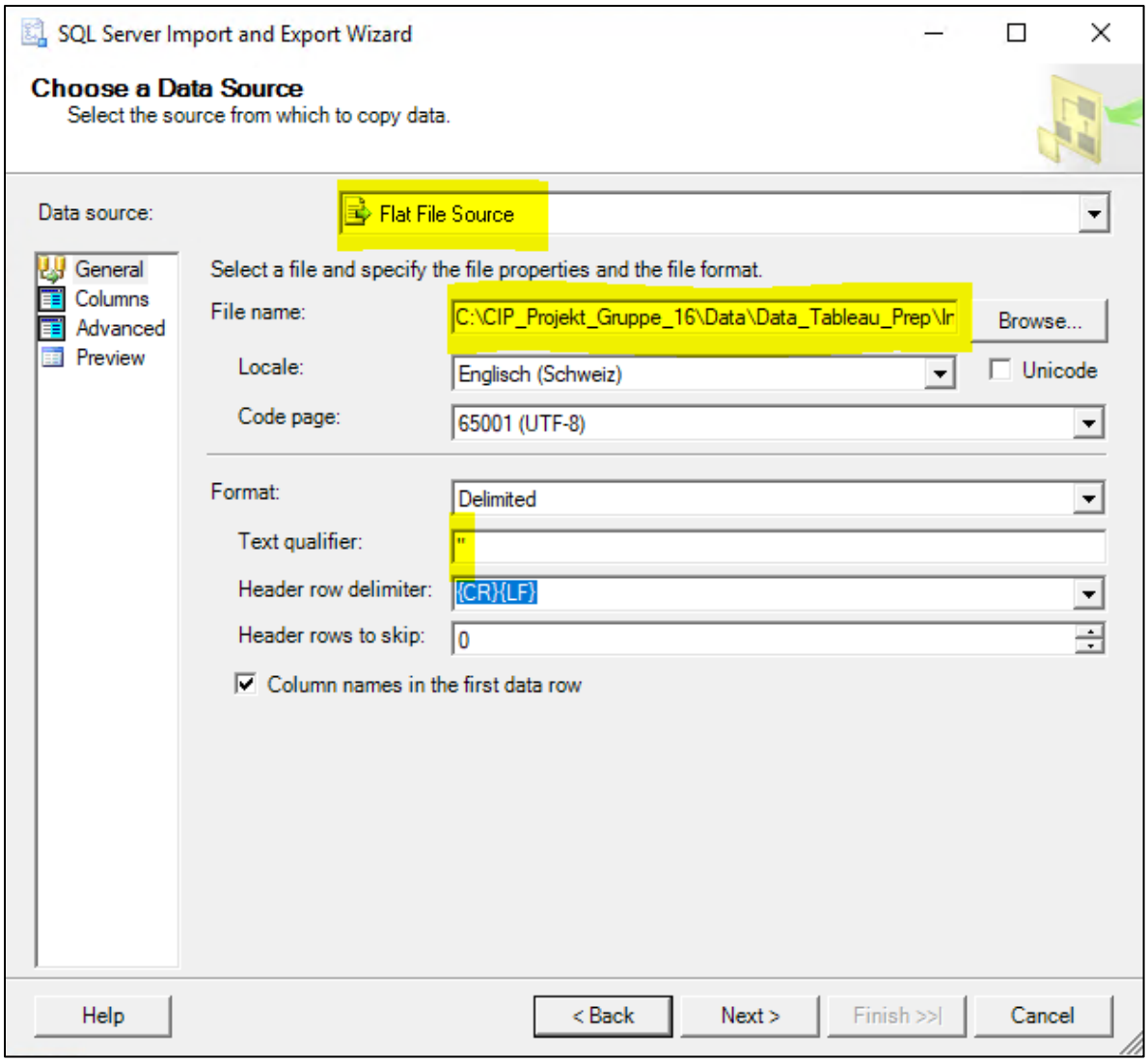

Abbildung 18: Datenimport aus SQL-Server-Datenbank mit dem Wizard

Beim Einspielen der Tabelle definierte der Wizard alle Spalten als Strings, resp. varchars welche in den Advanced-Einstellungen angepasst werden mussten. Deshalb wurden alle nötigen Spalten, welche Zahlen enthalten auf den Datentyp «Integer» umgestellt und zudem wurden bei den Spalten «AIRPORT_NAME», «D_AIRPORT» und «O_AIRPORT» die Feldlänge auf 255 erweitert.

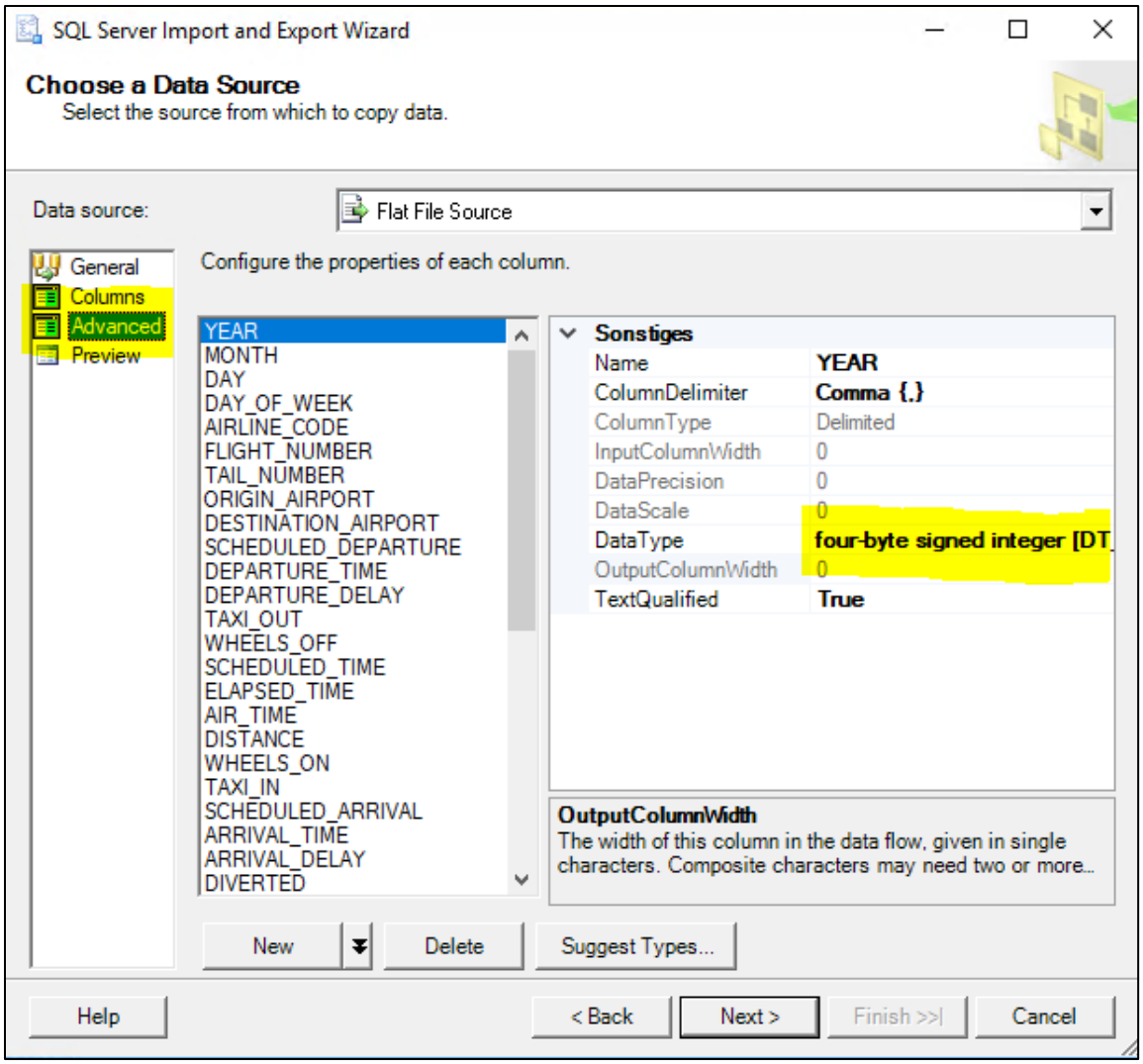

Abbildung 19: Advanced-Einstellungen im Wizard

Folgende Tabelle zeigt die Spaltenüberschriften der Tabelle und die dazugehörigen Datentypen.

Tabelle 1: Spaltenüberschriften und Datentypen auf dem SQL-Server

Spaltenüberschrift	Datentyp	Spaltenüberschrift	Datentyp
YEAR	int	ARRIVAL_DELAY	int
MONTH	int	DIVERTED	int
DAY	int	CANCELLED	int
DAY_OF_WEEK	int	CANCELLATION_REASON	varchar(50)
AIRLINE_CODE	varchar(50)	AIR_SYSTEM_DELAY	int
FLIGHT_NUMBER	int	SECURITY_DELAY	int
TAIL_NUMBER	varchar(50)	AIRLINE_DELAY	int
ORIGIN_AIRPORT	varchar(50)	LATE_AIRCRAFT_DELAY	int
DESTINATION_AIR-PORT	varchar(50)	WEATHER_DELAY	int
SCHEDULED_DEPAR-TURE	int	D_AIRPORT	varchar(255)
DEPARTURE_TIME	int	D_CITY	varchar(50)
DEPARTURE_DELAY	int	D_STATE	varchar(50)
TAXI_OUT	int	D_LATITUDE	varchar(50)
WHEELS_OFF	int	D_LONGITUDE	varchar(50)
SCHEDULED_TIME	int	O_AIRPORT	varchar(255)
ELAPSED_TIME	int	O_CITY	varchar(50)
AIR_TIME	int	O_STATE	varchar(50)
DISTANCE	int	O_LATITUDE	varchar(50)
WHEELS_ON	int	O_LONGITUDE	varchar(50)
TAXI_IN	int	AIRLINE_NAME	varchar(255)
SCHEDULED_ARRIVAL	int	ICAO	varchar(50)
ARRIVAL_TIME	int	Rufzeichen	varchar(50)

Nach den Advanced-Einstellungen wurde der Speicherort für die importierte Tabelle definiert. Dazu wurde der SQL-Server Native Client 11.0 ausgewählt, auf dem die Datenbank CIP zu Beginn dieses Unterkapitels erstellt worden ist.

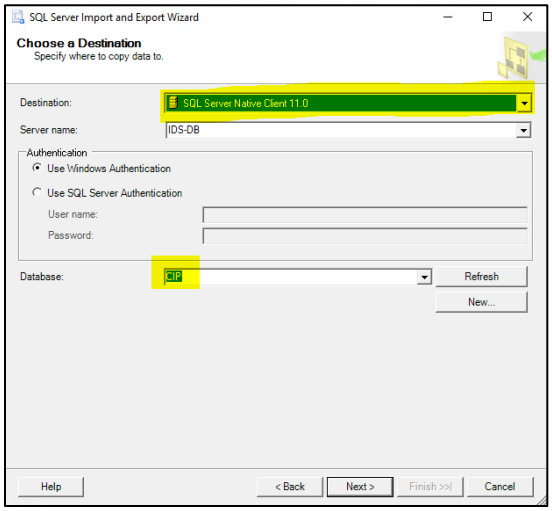

Abbildung 20: Definition des Speicherorts

Zuletzt wurde der Import gestartet und erfolgreich abgeschlossen, wie man im folgenden Bild sieht.

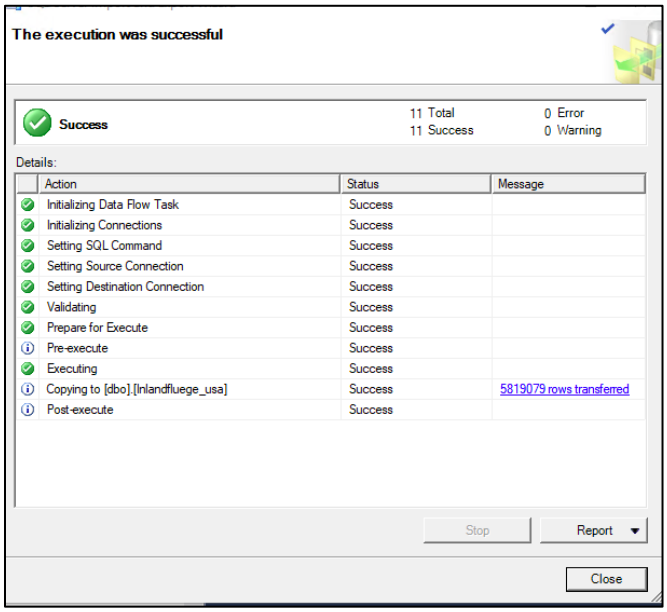

Abbildung 21: Erfolgreicher Import der Tabelle in die Datenbank CIP

In der folgenden Abbildung ist ersichtlich, dass die Tabelle «Inlandfluege_usa» in die Datenbank CIP importiert wurde und kann nun von hier aus direkt in Tableau Desktop geladen werden.

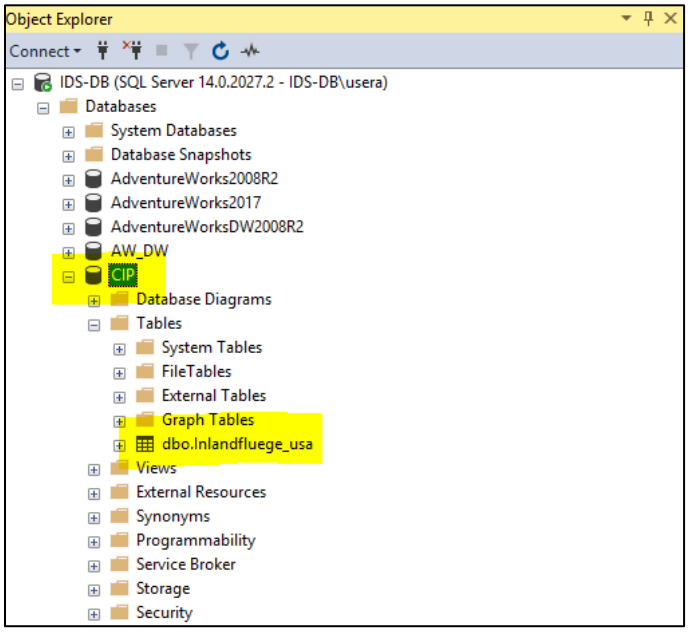

Abbildung 22: Struktur der Datenbank auf dem SQL-Server

5. Analyse - Tableau Desktop

5.1 Aufbereitung Grafiken in Tableau Desktop

Die Daten wurden direkt von der Datenbank in die Tableau Desktop Umgebung geladen, um die Fragestellung welche Airlines die meisten pünktlichen, verspäteten, annullierten und umgeleiteten Flüge aufweisen zu beantworten.

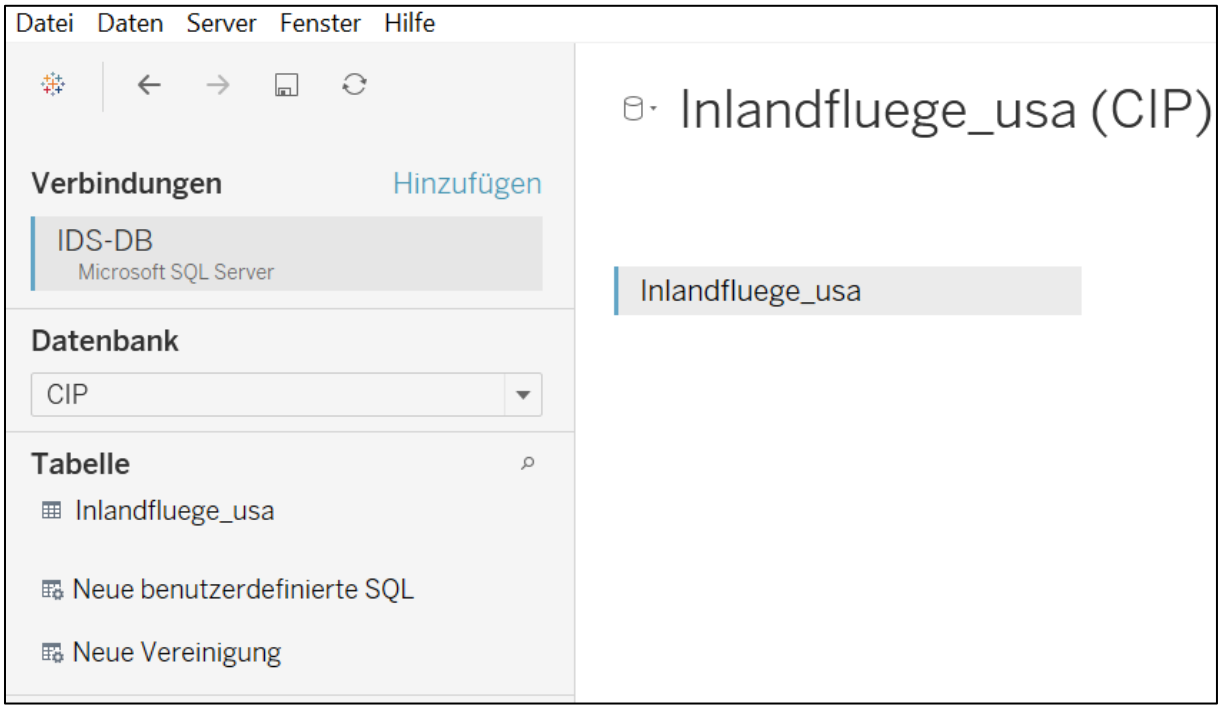

Abbildung 23: Laden der Daten von SQL-Server in Tableau Desktop

Um die absolute Zahl der Verspätungen pro Airline zu visualisieren wurde als erstes per Drag & Drop die Dimension «AIRLINE_NAME» als Spalte und die Kennzahl «FLIGHTS_DELAY» als Zeile in ein Arbeitsblatt eingefügt, dabei wird «FLIGHTS_DELAY» automatisch von Tableau aufsummiert. Dieser Schritt wurde für die Kennzahlen «FLIGHTS_ONPOINT», «DIVERTED» und «CANCELLED» wiederholt, um die absoluten Zahlen der pünktlichen, umgeleiteten und annullierten Flüge pro Airline zu bekommen.

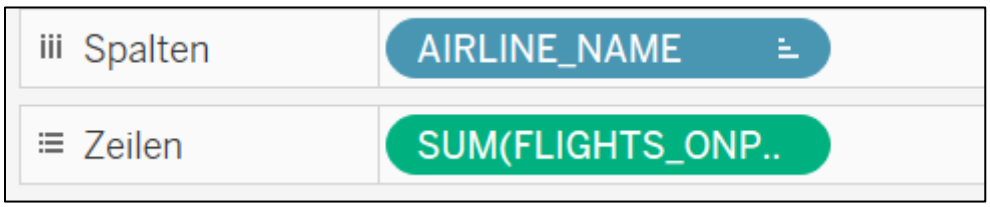

Abbildung 24: Beispiel Datenverschmelzung

In Tableau-Formelschreibweise, die erkennbar wird durch einen Doppelklick auf die Spalte oder Zeile, wurde dieser Schritt folgendermassen dargestellt: [AIRLINE_NAME] als Spalte SUM([FLIGHTS_DELAY]) bzw. SUM([FLIGHTS_ONPOINT]), SUM([DIVERTED]), SUM([CANCELLED]) als Zeile.

Jedoch ist eine Analyse absoluter Zahlen selten aussagekräftig, deshalb musste in einem nächsten Schritt die Anzahl geflogener Flüge aller Airlines berücksichtigt werden. Dazu wurde ein weiteres Arbeitsblatt erstellt, bei dem als erstes «FLIGHTS_DELAY» als Spalte eingefügt wurde. In einem zweiten Schritt wurde «AIRLINE_NAME», als Kennzahl (Anzahl) als Spalte hinzugefügt.

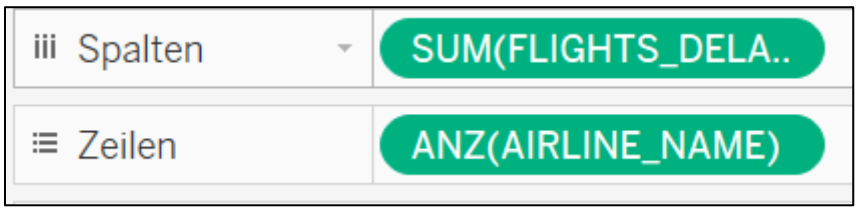

Abbildung 25: Beispiel Datenverschmelzung Anzahl Flüge berücksichtigt

Um grafisch, zwischen den verschiedenen Airlines unterscheiden zu können, wurde «AIRLINE_NAME» sowohl als Text-, wie auch als Farbmarkierung verwendet.

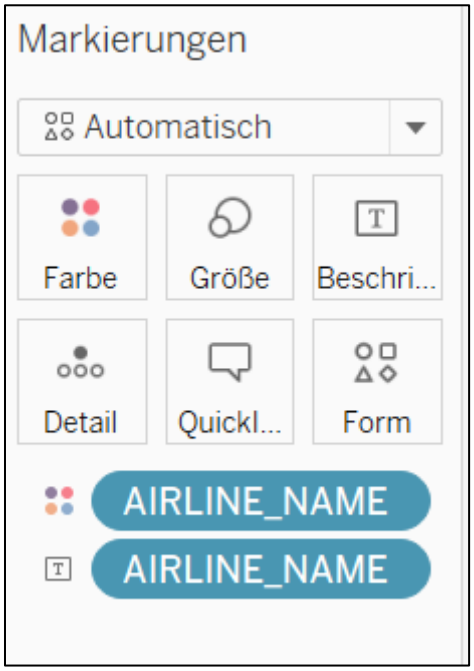

Abbildung 26: Einfügen Text- und Farbmarkierung

Um Prozentzahlen zu generieren, wurde in einem letzten Schritt die Spalte durch die Anzahl Flüge pro Airline geteilt. Diese Schritte wurden für alle zu untersuchenden Kennzahlen wiederholt. In Tableau-Formelschreibweise sehen die Spalten der einzelnen Arbeitsblätter wie folgt aus:

SUM([FLIGHTS_DELAY])/COUNT([AIRLINE_NAME])

SUM([FLIGHTS_ONPOINT])/COUNT([AIRLINE_NAME])

SUM([DIVERTED])/COUNT([AIRLINE_NAME])

SUM([CANCELLED])/COUNT([AIRLINE_NAME])

Die Zeile in den vier Arbeitsblättern ist für alle untersuchten Kennzahlen gleich: COUNT([AIRLINE_NAME]).

Zuletzt wurde in den vier Grafiken mit Prozentzahlen Mediane mit unterem und oberem Quartil eingefügt, wobei der Median der Anzahl Flüge irrelevant für die Analyse war und somit entfernt wurde.

Abbildung 27: Einfügen Median mit Quartilen

5.2 Analyse mit Tableau Desktop

Um die Fragestellung welche Airlines die meisten verspäteten, pünktlichen, annullierten und umgeleiteten Flüge aufweisen zu beantworten, wurden die im vorherigen Kapitel generieten Grafiken analysiert. Zu jeder Frage wurde sowohl die absoluten wie auch die relativen Zahlen analysiert.

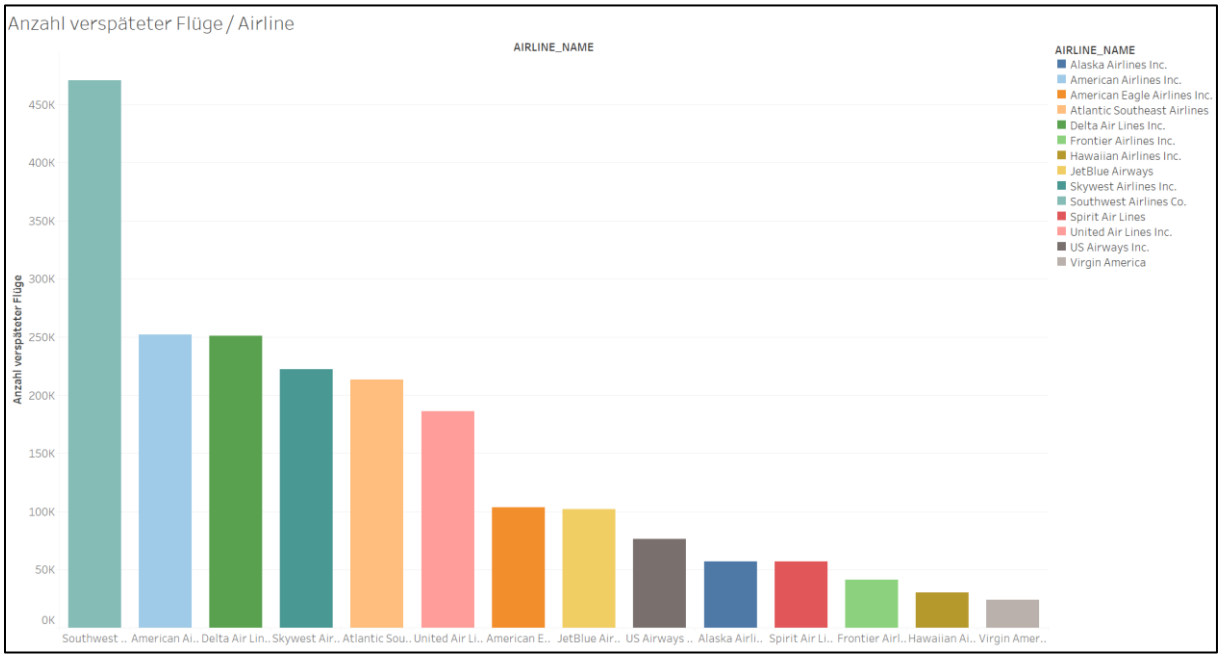

Abbildung 28: Anzahl verspäteter Flüge pro Airline

In Abbildung 28 ist klar zu erkennen, dass die die Airline Southwest Airlines Co. mit etwa 470'000 Verspätungen, die meisten verspäteten Flüge aufwies. Wohingegen die Airline Virgin America mit 24’000 Verspätungen, die wenigsten aufwies.

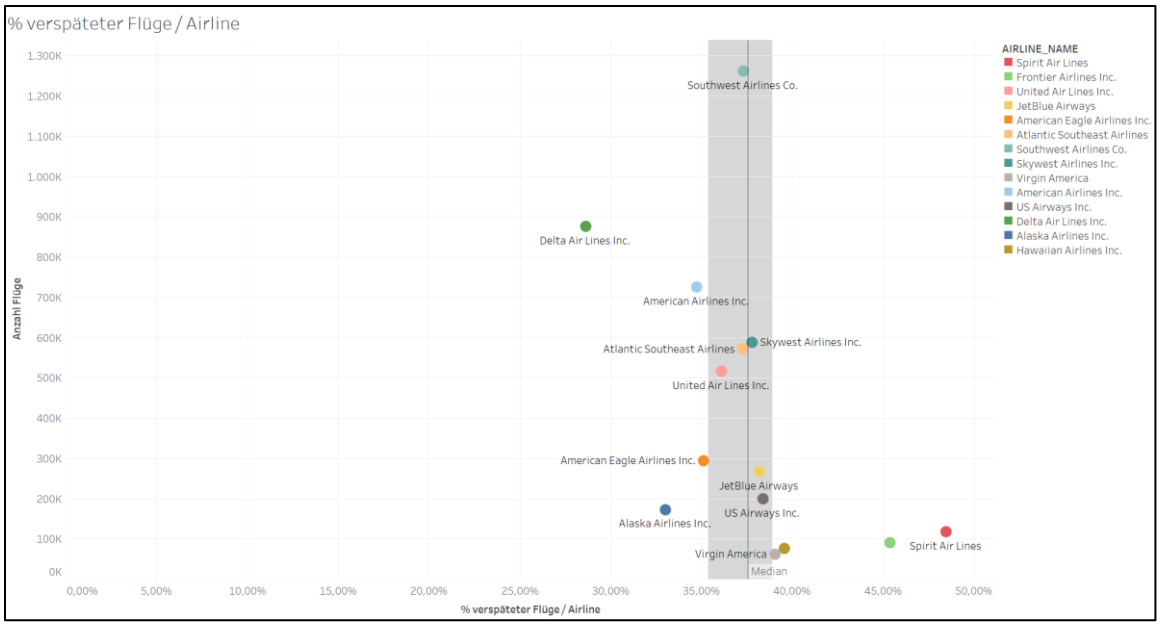

Abbildung 29: Prozentsatz verspäteter Flüge pro Airline

Vergleicht man Abbildung 28 mit Abbildung 29 ist erkennbar, dass Southwest Airlines Co. nicht mehr die am meisten verspätete Airline war, wenn man die Anzahl der geflogenen Flüge miteinbezieht. Etwa 37% der Flüge der Southwest Airlines Co. waren zu spät, was circa dem Wert, der in Abbildung 28 am wenigsten verspäteten Airline Virgin America entspricht. Bei der Hälfte der untersuchten Airlines waren zwischen 35% und 38% der Flüge verspätet. Die höchste Verspätungsrate hatten die Airlines Spirit Air Lines, mit 48% und Frontier Airlines mit 45%. Die relativ am wenigsten verspätete Airline war Delta Air Lines Inc. bei welcher 29% aller Flüge verspätet waren.

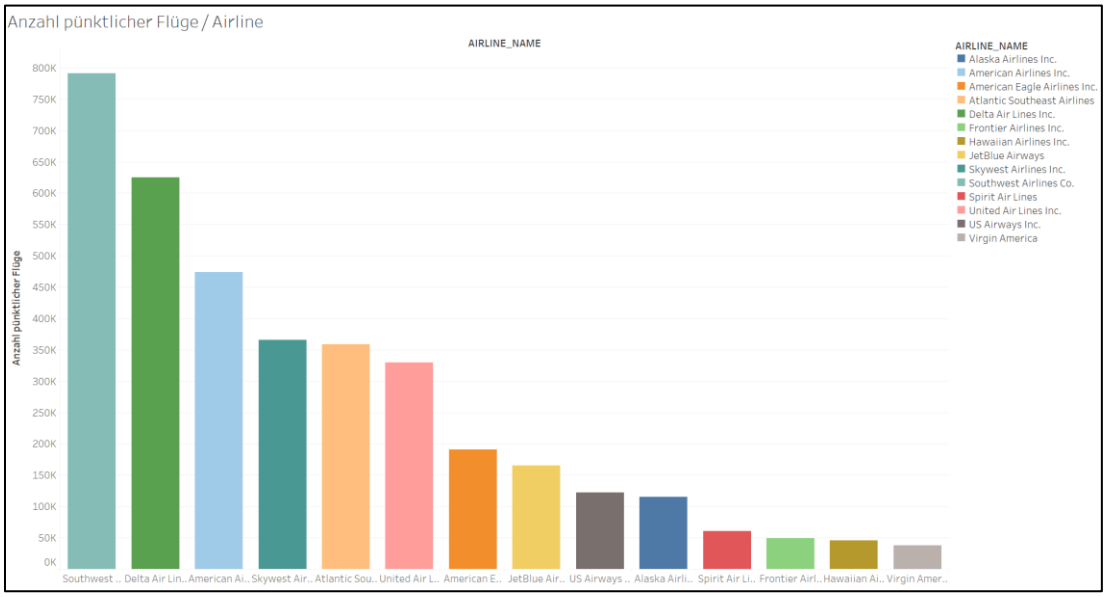

Abbildung 30: Anzahl pünktlicher Flüge pro Airline

Sowohl die am meisten verspäteten, wie auch die pünktlichsten Airlines waren in etwa dieselben. Dies ist nicht weiter überraschend, da dies mit der Anzahl geflogener Flüge erklärbar ist.

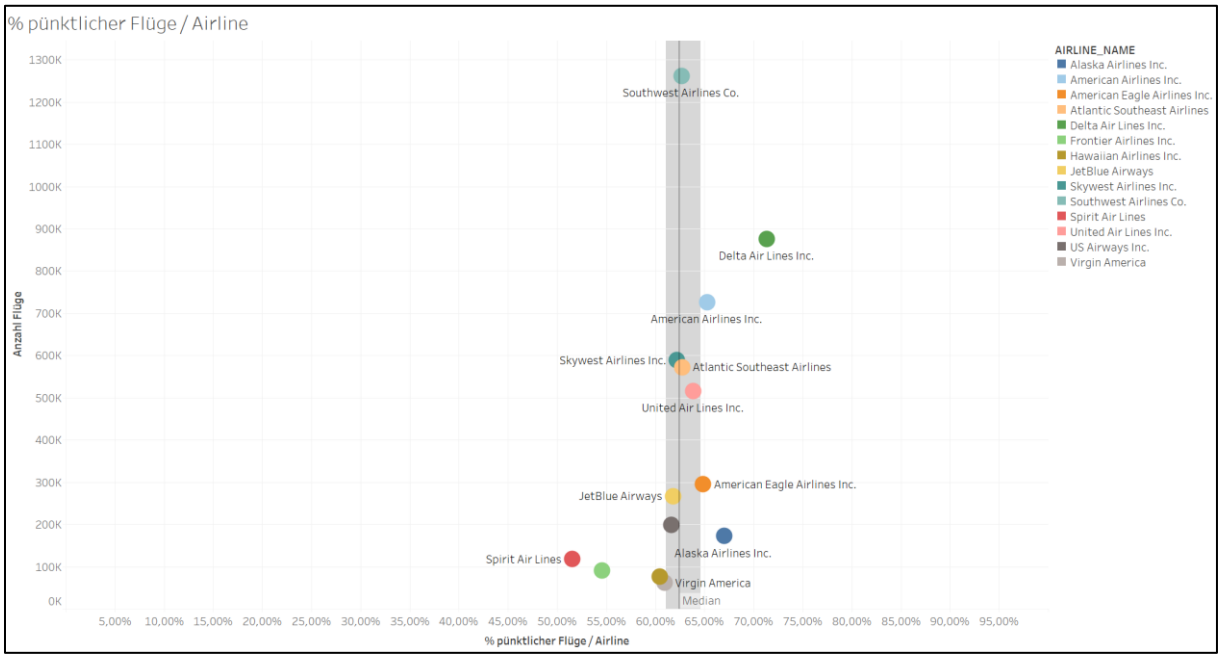

Abbildung 31: Prozentsatz pünktlicher Flüge pro Airline

Aus den relativen Zahlen ergibt sich ein ähnliches Muster wie bei den verspäten Flügen bloss mit umgekehrten Vorzeichen. Die am meisten verspäteten Airlines waren dieselben wie die am wenigsten pünktlichen und vice versa.

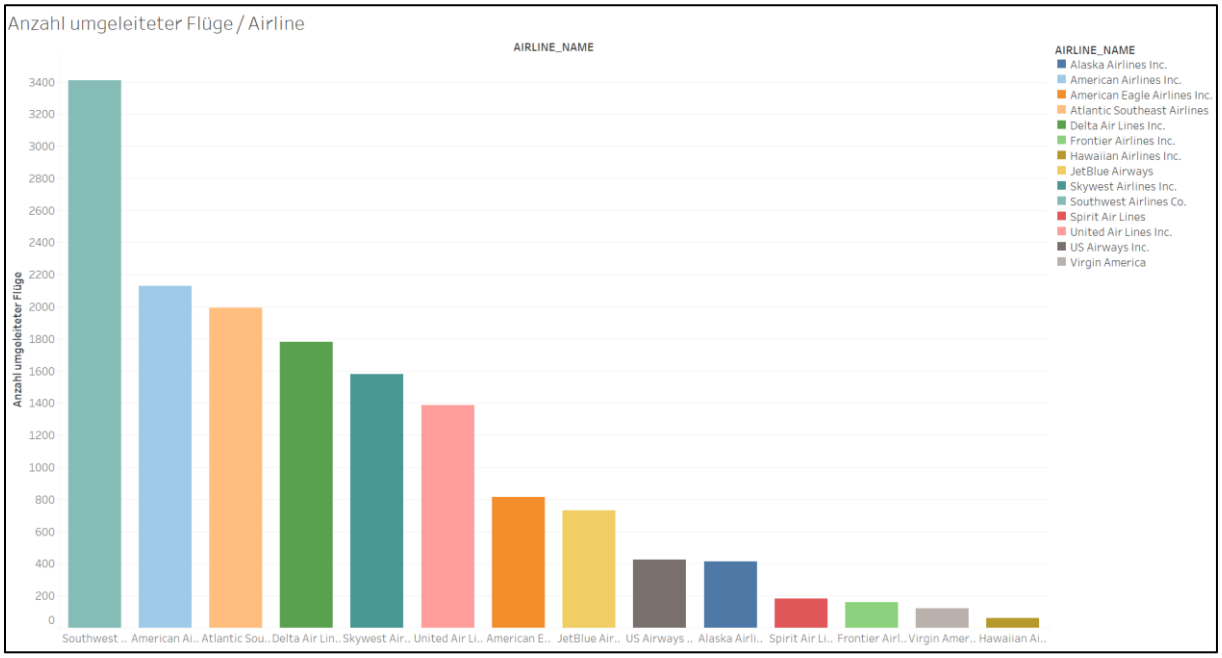

Abbildung 32: Anzahl umgeleiteter Flüge pro Airline

Man erkennt, dass im Vergleich zu verspäteten und pünktlichen Flügen, nur sehr wenige Flüge umgeleitet wurden. Auch in dieser Grafik erkennt man, dass die Airlines mit den meisten Flügen auch am meisten umgeleitet wurden.

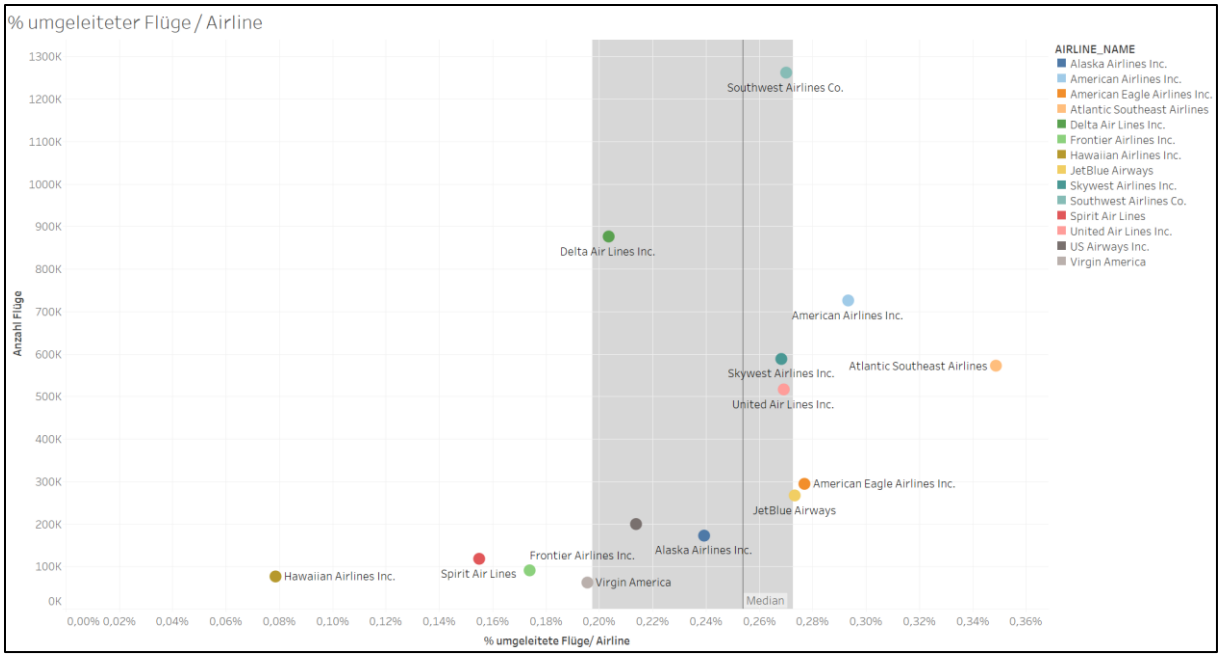

Abbildung 33: Prozentsatz umgeleiteter Flüge pro Airline

In Abbildung 33 ist erkennbar, dass bei der Hälfte der untersuchten Airlines zwischen 0.2% und 0.27% der Flüge umgeleitet wurden. Bei der Hawaiian Airlines werden nur 0.08% aller Flüge umgeleitet, wohingegen es bei der Atlantic Southeast Airlines 0.35% sind.

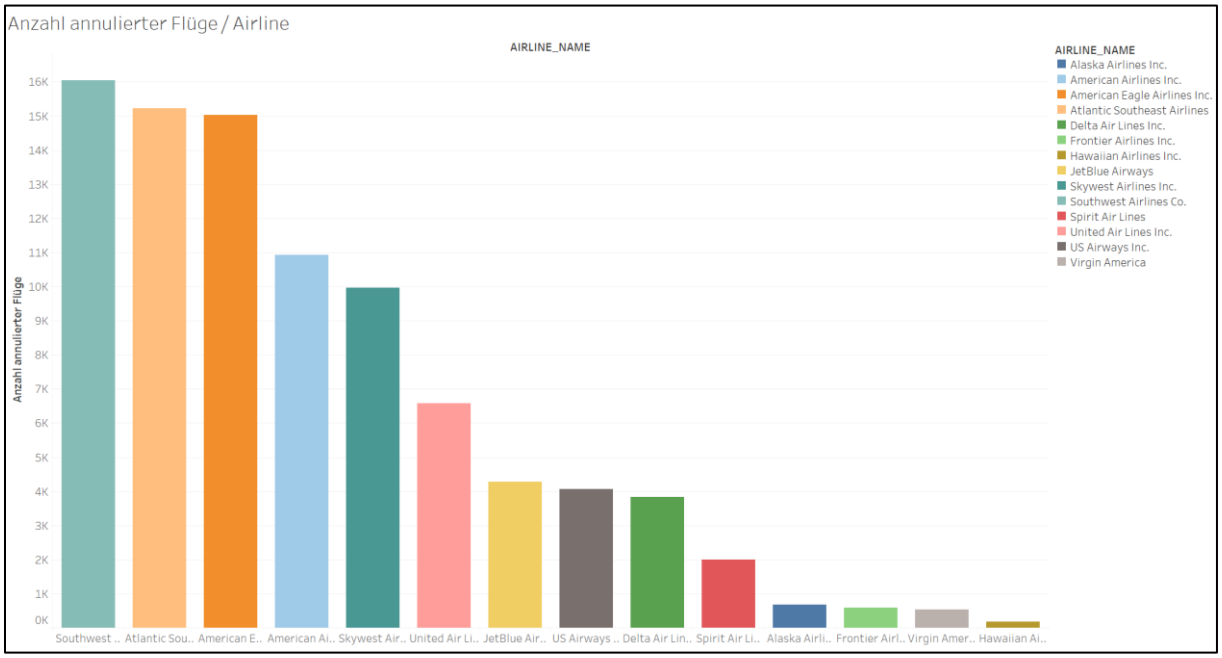

Abbildung 34: Anzahl annullierter Flüge pro Airline

Bei der Anzahl annullierter Flüge ergibt sich ein ähnliches Bild wie bei der Analyse der verspäteten, pünktlichen und umgeleiteten Flüge. Southwest Airlines Co. hatte die meisten annullierten Flüge, Hawaiian Airlines und Virgin America die wenigsten. Hervorzuheben ist die Delta Air Lines Inc., die trotz vieler Flüge sehr wenige Annullierungen vorzuweisen hatte. Die Annullierungen erkennt man auch in der relativen Betrachtung, obwohl Delta Airlines Inc. 875'000 Flüge durchführte, wurden nur 0.44% annulliert, wobei die Hälfte der untersuchten Airlines zwischen 0.7% und 1.7% der Flüge annullierten. Am besten schnitt ein weiteres Mal die Hawaiian Airlines ab, bei welcher nur 0.22% der Flüge annulliert wurden. Im Gegensatz dazu wurden mit 5.1% bei der American Eagle Airlines Inc. prozentual die meisten Flüge annulliert.

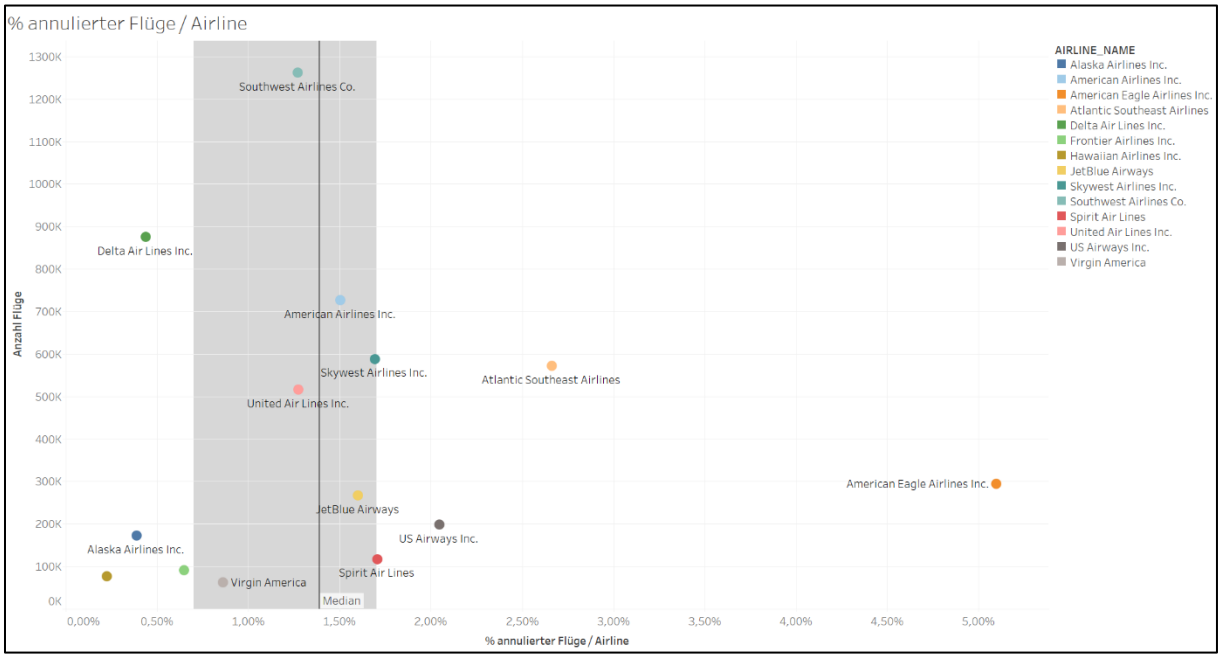

Abbildung 35: Prozentsatz annullierte Flüge pro Airline

6. Schlusswort

Zusammenfassend kann man sagen, dass bei den absoluten Zahlen die Southwest Airlines Co. bei allen untersuchten Variablen führend war, dies ist jedoch mit der hohen Anzahl Flüge zu erklären. Bei der relativen Betrachtung liegt die Airline immer in der Hälfte der untersuchten Airlines. Betrachtet man die relativen Zahlen bezüglich verspätete/pünktliche Flüge, schnitt die Spirit Air Lines am schlechtesten ab, jedoch hatte sie relativ wenig umgeleitete Flüge und lag bei den annullierten Flügen im Mittel. Ähnlich ist es bei der Frontier Airlines Inc.. American Eagle Airlines Inc. hatte relativ viele annullierte Flüge, lag jedoch bei den anderen untersuchten Variablen im oder nahe am Mittelmass. Die Hawaiian Airlines hatte die wenigsten annullierten und umgeleiteten Flügen und schnitt auch bezüglich Pünktlichkeit besser als die Hälfte der untersuchten Airlines ab. Jedoch war ihre Anzahl an Flügen sehr gering, wohingegen die Delta Airlines Inc. mit einer hohen Anzahl an Flügen am pünktlichsten war. Des Weiteren war sie bezüglich umgeleiteter Flüge im unteren Quartil des Medians angesiedelt und hatte weniger Annullierungen als die Hälfte der untersuchten Airlines.

Es hätten weitere Themen untersucht werden können, insbesondere die Analyse der Differenz der geplanten Flugzeit und der effektiven Flugzeit sowie der Zeitgewinn/Verlust während des Fluges. Des Weiteren wäre auch eine geografische Analyse möglich gewesen, beispielsweise in welcher Flugrichtung es die meisten Verspätungen gab und welche Städte die effizientesten Flughäfen in der Betrachtungsperiode hatten. Auf alldem lag jedoch nicht der Fokus dieser Arbeit und kann bei zukünftigen Analysen untersucht werden.

7. Reflexion

Die Gruppenarbeit im Modul Data Collection, Integration and Preprocessing war herausfordernd und zeitintensives zugleich, aber dennoch ein sehr lehrreiches Projekt. Neben der Theorie zum ETL-Prozess in den Lernveranstaltungen, wurde dieser direkt an einem eigenen Prozess angewendet und mit verschiedenen Tools geschaffen.

Durch das Erstellen eines Website-Crawlers und die anschliessende Analyse mit Python wurde der Praxisbezug zum Programmieralltag geschaffen. Dadurch war das ganze Team gezwungen sich unvorhergesehenen Problemen zu stellen und diese individuell oder als Gruppe zu lösen. Zudem wurden komplett neue Tools kennengelernt, welche via z.B. YouTube-Tutorials in den groben Zügen erlernt und in den ETL-Prozess integriert wurden. Als sehr nützlich hat sich die Seite «https://stackoverflow.com» erwiesen. Auf der Plattform von stackoverflow.com wurde fast jede technische Herausforderung des Projekts bereits einmal behandelt.

Herausfordernd war vor allem der Abgleich auf den verschiedenen Virtual Machines und die grosse Menge an Daten, welche für einzelne Prozesse jeweils mehrere Minuten gebraucht haben. Dazu kam ein Fehler im Datensatz bei der vermeintlichen abschliessenden Analyse, welcher erheblichen zusätzlichen Mehraufwand generiert hat.

Die Aufgabenteilung diskutierte man jeweils während weniger Team-Meetings, damit man individuell an einzelnen Aufgaben arbeiten konnte. Zudem wurde jeweils der Abgleich der VM, sowie bisherige Resultate besprochen.

Abschliessend war dies ein anspruchsvolles Projekt, das einiges an Zeit in Anspruch genommen hat. Jedoch entstand während der Arbeit viel neues Wissen und die Lernkurve in der Gruppe ist stetig gestiegen.

8. Anhang

A: Zeitplan:

Prozess	Deadline	Arbeitsschritt	Bemerkung	Teilnehmer
Kick-Off	DI, 10.03.	- Projektübersicht erstellen - Ziele definieren - Datensatz auswählen - Datensatz explorieren (individuell)	- Dokument in Word-Form an Dozenten per E-Mail	- Alle
Besprechung mit Dozenten	FR, 13.03.	- Besprechung CIP-Projekt mit Dozenten		- Alle
Teammeeting	FR, 13.03.	- Besprechung weiterer Prozess gem. Feedback Dozent	- VM-Verzeichnis für alle synchronisieren	- Alle
Wissensaufbau und Crawler bauen	SA, 14.03. – SO, 15.03.	- Tableau Prep - Tableau - SQL - Erstellung Crawler		- Gianni, Marius (Crawler) - Remo (SQL) - Stephan (Tableau)
Extract	MO, 16.03. – SA, 21.03.	- Daten crawlen mit BeautifulSoup 4		- Alle
Teammeeting	FR, 20.03.	- Besprechung Datensammlung Crawler	- VM-Verzeichnis für alle synchronisieren	- Alle
Transform	SO, 22.03. – FR, 27.03.	- Daten transformieren, bereinigen und Erkenntnisse gewinnen	- Pandas - Tableau Prep	- Alle
Teammeeting	FR, 27.03.	- Besprechung Datentransformation	- VM-Verzeichnis für alle synchronisieren	- Alle
Hackathon	FR, 27.03-30.03		- Anpassung Zeitplan	
Individuelle Arbeiten	MO, 31.03-SA 11.04	- Individuelle Arbeit an zugewiesenen Aufträgen		- Alle
Load	SO, 12.04	- Bereinigte Daten auf SQL-Datenbank laden		- Alle (individuelle VM)
Visualise	MO, 13.04	- Daten mittels Tableau Desktop analysieren und darstellen		- Alle (lead: Stefan)
Wrap-Up	MI, 15.04.	- Erarbeitung Projektbeschreibung / Videoaufnahme		- Alle
Abgabe	FR, 17.04.	- Abgabe Projektarbeit	- Videoaufnahme - Zip-File - Projektbeschreibung - Zeiterfassung - A4 Datenstruktur	- Alle

B: Python Code

Code: Crawler_Kaggle.py

```
from selenium import webdriver
import time
from zipfile import ZipFile


def fetch():
    #Webdriver wird gestartet
    driver = webdriver.Chrome(executable_path=r"chromedriver.exe")

    driver.set_window_size(1400,1200)

    driver.get("https://www.kaggle.com/account/login?phase=startSignInTab&return
    Url=https%3A%2F%2Fwww.kaggle.com%2F")

    time.sleep(5)

    #Login Prozess

    login_button = driver.find_element_by_xpath('//*[@id="site-contai
    ner"]/div[1]/div/form/div[2]/div/div[2]/a/li/span')
    login_button.click()

    mail_input = driver.find_ele
    ment_by_xpath('/html/body/main/div/div[1]/div/form/div[2]/div[1]/div/div/in
    put')
    mail_input.send_keys('testseleniumhslu@hotmail.com')


    mail_paswortd = driver.find_ele
    ment_by_xpath('/html/body/main/div/div[1]/div/form/div[2]/div[2]/div/div/in
    put')
    time.sleep(2)
    mail_paswortd.click()
    mail_paswortd.send_keys('TestHSLU)

    time.sleep(2)

    #Webdriver öffnet Seite mit geüwünschtem Dataset

    sign_in = driver.find_ele
    ment_by_xpath('/html/body/main/div/div[1]/div/form/div[2]/div[3]/div/but
    ton/span')
    sign_in.click()

    time.sleep(2)

    driver.get('https://www.kaggle.com/usdot/flight-delays')


    #Driver startet Download
    download_button = driver.find_ele
    ment_by_xpath('/html/body/main/div[1]/div/div[5]/div[2]/div[1]/div/div/div[2]/
    div[2]/div[1]/div[2]/a[1]/div/span')

    download_button.click()
```

```
    time.sleep(10)

    driver.quit()

    #Die heruntergeladenen Dateien werden entzippt und in Ordner Data kaggle ge
    speichert
    with ZipFile('C:/Users/usera/Downloads/flight-delays.zip', 'r') as zipObj:
        zipObj.extractall('C:/CIP_Daten/CIP_Projekt_Gruppe_16/Data/Data_kaggle')


fetch()

print('Daten in Ordner Data_kaggle gespeichert und entzippt. Nächster Schritt:
Crawler Wikipedia.')
```

Code: Crawler_Wikipedia.py

```
import requests
import time
from bs4 import BeautifulSoup
import csv


print('Daten von Kaggle wurden gecrawlt. Jetzt wird Wikipedia Seite gecrawlt Ruf-
zeichen der Airlines zu erhalten.')


###Klassen Definition Airline
class Airline():
    def __init__(self, name,iatacode,icao,rufzeichen):
        self.name = name
        self.iatacode = iatacode
        self.icao = icao
        self.rufzeichen = rufzeichen


###Klassen Definition Data
class Data():
    def fetch(self):
        time.sleep(1)
        r = requests.get("https://de.wikipedia.org/wiki/Liste_von_Fluggesellschaf
    ten")

        airline = []

        doc = BeautifulSoup(r.text, "html.parser")
        doc.find_all("table", class_="wikitable sortable")
        My_table = doc.find("table", {"class":"wikitable sortable"})
        tr_table = My_table.findAll("tr")

        for i in range(1, len(tr_table)):
            columns = tr_table[i].text
            items = columns.split('\n')
            t1 = Airline(items[1], items[3], items[5], items[7])
            airline.append(t1)
```

```
        return airline


fetcher = Data()

with open('C:/CIP_Daten/CIP_Projekt_Gruppe_16/Data/Data_Wikipedia/crawler_wikipe
    dia_output.csv', 'w', newline='',encoding='utf-8') as csvfile:
    articlewriter = csv.writer(csvfile, delimiter=';', quotechar='"', quo
    ting=csv.QUOTE_MINIMAL)
    for t in fetcher.fetch():
        articlewriter.writerow([t.name, t.iatacode, t.icao, t.rufzeichen])


print('Daten in Ordner Data_wikipedia gespeichert. Nächster Schritt: Cleaner.')
```

Code: Cleaner.py

```
import pandas as pd

print('Daten von Kaggle und Wikipedia wurden gecrawlt. Jetzt werden die Daten be-
reinigt und erste Vereinigungen vorgenommen.')

def clean():

    df1 = pd.read_csv('C:/CIP_Projekt_Gruppe_16/Data/Data_Wikipedia/crawler_wi-
kipedia_output.csv', sep=";")
    df1.columns = ['Name','IATA_CODE','ICAO','Rufzeichen']

    ###Drop all NA's
    df1 = df1.dropna()

    ###Update Index
    transform = lambda x: x+1
    df1.index = df1.index.map(transform)

    ### Load downloaded csv-file
    df2 = pd.read_csv('C:/CIP_Projekt_Gruppe_16/Data/Data_kaggle/airlines.csv',
sep=',')
    df2.index = df2.index.map(transform)

    ### Merge csv df with df2 to create df3
    df3 = pd.merge(df2,df1[['IATA_CODE','ICAO','Rufzeichen']],how="left",on = "I-
ATA_CODE")
    df3.index = df3.index.map(transform)

    #Manually Add missing values to df3
    #US Airways IACO: AWE und Rufzeichen Cactus
    values_us_airways = {'ICAO':'AWE','Rufzeichen':'CACTUS'}
    df3 =df3.fillna(value=values_us_airways, limit=1)

    #Atlantic Southeast Airlines ICAO: ASQ und Rufzeichen ACEY
    values_atlantic_southeast = {'ICAO':'ASQ','Rufzeichen':'ACEY'}
    df3 = df3.fillna(value=values_atlantic_southeast, limit=1)

    #Virgin America ICAO: VRD und Rufzeicehn ALASKA
    values_virgin_america = {'ICAO':'VRD','Rufzeichen':'ALASKA'}
    df3 = df3.fillna(value=values_virgin_america, limit=1)
```

```
    ### Save as csv file
    df3.to_csv("C:/CIP_Projekt_Gruppe_16/Data/Data_Python/airlines_merged.csv",in-
dex = False, header=True)


clean()

print('Daten bereinigt und in Ordner Data_Python als CSV-Datei airlines_merged ab-
gespeichert. Nächster Schritt: Crawler_Airport_ID.py')
```

Code: Crawler_Airport_ID.py

```
from selenium import webdriver
import time
import pandas as pd


def fetch():
    #Webdriver wird gestartet
    driver = webdriver.Chrome(executable_path=r"chromedriver.exe")

    driver.get("https://www.transtats.bts.gov/DL_Select-
Fields.asp?Table_ID=236&DB_Short_Name=On-Time")

    origin_airport = driver.find_ele-
ment_by_xpath('/html/body/div[3]/div[3]/table[1]/tbody/tr/td[2]/table[4]/tbody/tr[
16]/td[4]/a')
    origin_airport.click()

    time.sleep(5)
    origin = driver.find_element_by_xpath('//*[@id="con-
tent"]/table[1]/tbody/tr/td[2]/table[4]/tbody/tr[19]/td[4]/a')
    origin.click()

    time.sleep(5)
    driver.quit()


def clean():
    df1 = pd.read_csv('C:/Users/usera/Downloads/L_AIRPORT.csv_')
    df1.to_csv(r'C:/CIP_Projekt_Gruppe_16/Data/Data_kaggle/AIRPORT.csv', index =
False)
    df2 = pd.read_csv('C:/Users/usera/Downloads/L_AIRPORT_ID.csv_')
    df2.to_csv(r'C:/CIP_Projekt_Gruppe_16/Data/Data_kaggle/AIRPORT_ID.csv', index
= False)


fetch()
clean()

print('Die beiden CSV-Files AIRPORT.csv und AIRPORT_ID.csv wurden heruntergeladen
und im Ordner data_kaggle abgelegt. Nächster Schritt: data_kaggle_cleaner.py')
```

Code: data_kaggle_cleaner

```
import pandas as pd
import numpy as np

# Import Datensatz flights.csv
df = pd.read_csv(r'C:/CIP_Projekt_Gruppe_16/Data/Data_kaggle/flights.csv')
print("Datensatz flights.csv wurde geladen.")
print("Der Datensatz enthält " + str(len(df)) + " Datensätze.")
print('----------------------')

# Ausgabe Kopf des Datensatzes
#print(df.head())

# Ausgabe Spalten des Datensatzes
# pritn(df.columns)

# ------------------------------------------------------------------------------
-------
# Laden der airports.csv
df_airports = pd.read_csv(r'C:/CIP_Projekt_Gruppe_16/Data/Data_kaggle/air-
ports.csv')
print('airports.csv wurde geladen.')
print("Der Datensatz airports.csv enthält " + str(len(df_airports)) + " Datens-
ätze.")
print('----------------------')
# print(df_airports.head())
# print(len(df_airports))
# Output: 322

# ------------------------------------------------------------------------------
-------
# Import Datensatz AIRPORT_ID.csv
airport_id = pd.read_csv(r'C:/CIP_Projekt_Gruppe_16/Data/Data_kaggle/AIR-
PORT_ID.csv')
# airport_id.head()
# airport_id.index

print("Der Datensatz AIRPORT_ID.csv wurde geladen. Er enthält " + str(len(air-
port_id)) + " Datensätze.")
print('----------------------')
# ------------------------------------------------------------------------------
-------
# Import Datensatz AIRPORT.csv
airport = pd.read_csv(r'C:/CIP_Projekt_Gruppe_16/Data/Data_kaggle/AIRPORT.csv')
print("Der Datensatz AIRPORT.csv wurde geladen. Er enthält " + str(len(airport)) +
" Datensätze.")
print('----------------------')
# airport.head()
# airport.index

# ------------------------------------------------------------------------------
-------
# Merge von airports.csv mit der AIRPORT.csv
a_1 = df_airports.merge(airport, left_on = 'IATA_CODE', right_on = 'Code', how =
'left')
# a_1.head()
print('Merge von airports.csv und AIRPORT.csv. Die verknüpfte Datei a_1 hat eine
Länge von ' + str(len(a_1)))
print('----------------------')
```

```
# Merge von a_1 mit der AIRPORT_ID.csv
a_2 = a_1.merge(airport_id, left_on = 'Description', right_on = 'Description', how
= 'left')
print('Merge von a_1 und AIRPORT_ID.csv. Die verknüpfte Datei a_2 hat eine Länge
von ' + str(len(a_2)))
print('----------------------')
# ------------------------------------------------------------------------------
-------
# Prüfen auf doppelte Inhalte:
v = a_2['Description'].duplicated()
dupl = (a_2[v])
# print(dupl)

# a_2.tail()
print('321 und 322 haben denselben IATA_Code. Die IDs sind 16218 und 13758.')
print('----------------------')
# ------------------------------------------------------------------------------
-------
# Überprüfen, ob die Daten in den Spalten des Datensatzes "df" ORIGIN_AIRPORT und
DESTINATON_AIRPORT enthalten sind.
z = df.loc[df['ORIGIN_AIRPORT'] == 16218]
#print(z)
z = df.loc[df['ORIGIN_AIRPORT'] == 13758]
#print(z)

z = df.loc[df['DESTINATION_AIRPORT'] == 13785]
#print(z)
z = df.loc[df['DESTINATION_AIRPORT'] == 16218]
#print(z)

print('Es ist nur die ID 16218 im Datensatz "df" enthalten. Also wird die Zeile
mit der ID 13758 gelöscht.')
print('----------------------')

# ------------------------------------------------------------------------------
-------
# Die Zeile mit der ID = 13785 ist nicht im Datensatz "df" enthalten, deshalb wird
diese gelöscht.
# Diese liegt im Datensatz df_airports im Index 321.
a_2.drop(index=321)
print('Zeile mit dem Index 321 wurde gelöscht.')
print('----------------------')

# ------------------------------------------------------------------------------
-------
# Erstellen eines Dataframes mit IATA-Codes und IDs aus dem Dataframe a_2.
list_2 = []

for i in a_2.to_numpy().tolist():
    list_1 = []
    vari = str(i[9])
    list_1.append(vari)
    list_1.append(i[7])
    list_2.append(list_1)
#print(list_2)

codes = pd.DataFrame(list_2, columns = ['ID','IATA'])
# df_test.head()
```

```
# ------------------------------------------------------------------------------
-------
# Series erstellen, welche anschliessend an den Datensatz "df" angehängt werden.
flights_1 = df['ORIGIN_AIRPORT']
flights_2 = df['DESTINATION_AIRPORT']
#print(len(flights_1))
#f1 = flights_1.head(10)
#print(f1)

# ------------------------------------------------------------------------------
-------
# Die beiden Series werden nun an den Datensatz "df" angehängt.
df['Origin'] = flights_1
df['Destination'] = flights_2
#print(df_h)
#print(type(df_h))

# ------------------------------------------------------------------------------
-------
# Verknüpfen der von df und codes
merge_origin = df.merge(codes, left_on = 'ORIGIN_AIRPORT', right_on = 'ID', how =
'left')
# print(merge)
# merge_origin.head()

# Ersetzten der ID durch den IATA-Code in der Spalte "Origin"
merge_origin.loc[(merge_origin['ORIGIN_AIRPORT']) == (merge_origin['ID']), 'Ori-
gin'] = merge_origin.IATA

# Löschen der folgenden Spalten, da diese nicht mehr gebraucht werden.
del merge_origin['ORIGIN_AIRPORT']
del merge_origin['ID']
del merge_origin['IATA']

# Umbenennen der Spalte Origin zu ORIGIN_AIRPORT.
merge_origin.rename(columns={'Origin': 'ORIGIN_AIRPORT'}, inplace=True)

# TESTING merge_origin
# is_october = merge_origin['MONTH'] == 10
# a = merge_origin[is_october]
# a.head(10)
# print(len(merge))

# ------------------------------------------------------------------------------
-------
# Verknüpfen der von merge_origin und codes
merge_dest = merge_origin.merge(codes, left_on = 'DESTINATION_AIRPORT', right_on =
'ID', how = 'left')
# merge_dest.head()

# Ersetzten der ID durch den IATA-Code in der Spalte "Destination"
merge_dest.loc[(merge_dest['DESTINATION_AIRPORT']) == (merge_dest['ID']), 'Desti-
nation'] = merge_dest.IATA

# Löschen der folgenden Spalten, da diese nicht mehr gebraucht werden.
del merge_dest['DESTINATION_AIRPORT']
del merge_dest['ID']
del merge_dest['IATA']

# Umbenennen der Spalte Origin zu DESTINATION_AIRPORT.
```

```
merge_dest.rename(columns={'Destination': 'DESTINATION_AIRPORT'}, inplace=True)

# is_october = merge_dest['MONTH'] == 10
# b = merge_dest[is_october]
# b.head(10)

# ------------------------------------------------------------------------------
-------
# Prüfen auf NAs:
print(merge_dest['ORIGIN_AIRPORT'].isna().sum())
print(merge_dest['DESTINATION_AIRPORT'].isna().sum())
print('----------------------')

# ------------------------------------------------------------------------------
-------
# Prüfen, ob neuer und alter Datensatz dieselbe länge haben
print('Differenz zwischen alten und neuem Datensatz: ' + str(len(merge_dest) -
len(df)))
print('----------------------')

# ------------------------------------------------------------------------------
-------
# Ausgabe als CSV-Datei
print('CSV wird  erstellt.')
print('----------------------')
merge_dest.to_csv(r'C:/CIP_Projekt_Gruppe_16/Data/Data_kaggle/flights_new.csv',
index = False)
print('CSV wurde erstellt. Datenaufbereitung ist abgeschlossen!')
```

C: Eidesstattliche Erklärung

Hiermit erklären wir, dass die vorliegende Arbeit von uns selbst und ohne unerlaubte Mithilfe Dritter verfasst wurde. Die verwendeten Quellen sind im Literaturverzeichnis angegeben und die Urheberrechtsbestimmungen der Hochschule Luzern wurden respektiert.

Unterschrift

Remo Oehninger 

Gianni Pinelli 

Marius Gisler 

Stefan Berchtold 

Ort, Datum Luzern, 16. April 2020