



Universität Regensburg

**Philosophische Fakultät III**  
**Sprach- , Literatur- und Kulturwissenschaften**  
**Institut für Information und Medien, Sprache und Kultur (I:IMSK)**  
**Lehrstuhl für Medieninformatik**

---

**Kurs: Analysieren und Visualisieren mit Python**  
Veranstaltungsnummer: 36671a  
Wintersemester 15/16  
**Leitung: Müller, Manuel Tonio**

## **Crime Statistics**

- Dokumentation -

**Verfasser: Benedikt Hierl, Markus Guder, Christian Weber, Philipp Mai**

Abgegeben am 13.01.2016

# INHALT

<b>1</b>	<b>INFRASTRUKTURELLER AUFBAU .....</b>	<b>1</b>
1.1	ERKLÄRUNG UI.....	1
1.2	ERKLÄRUNG SERVER .....	2
<b>2</b>	<b>AUFTEILUNG DATENSATZ .....</b>	<b>2</b>
<b>3</b>	<b>ERKLÄRUNG PYTHON-DATEIEN .....</b>	<b>2</b>
3.1	„CSV_N_TABLE_CREATION.PY“ .....	2
3.2	“CREATE_ARREST_TABLE.PY“ .....	5
3.3	“CREATE_COMMON_CRIME_TABLE.PY“ .....	5
3.4	“CREATE_LOCATION_TABLE.PY“ .....	7
3.5	“CREATE_TIME_TABLE.PY“ .....	8
3.6	“TIMECRIME.PY“ .....	9

## Abbildungen

Abbildung 1: Infrastruktureller Aufbau .....	1
Abbildung 2: Ansicht aller zu erzeugenden Tabellen.....	4
Abbildung 3: Darstellung Tabelle ‚arrest‘ .....	5
Abbildung 4: Darstellung Tabelle ‚commonc‘ .....	6
Abbildung 5: Hinzufügen zur entsprechenden Liste .....	6
Abbildung 6: Darstellung ‚loc2007‘ Tabelle, repräsentativ für ‚locX‘ .....	7
Abbildung 7: Darstellung Heatmap auf zwei Nachkommastellen gerundet.....	8
Abbildung 8: Darstellung Heatmap auf drei Nachkommastellen gerundet .....	8
Abbildung 9: Darstellung ‚time‘ Tabelle.....	9

## 1 Infrastruktureller Aufbau

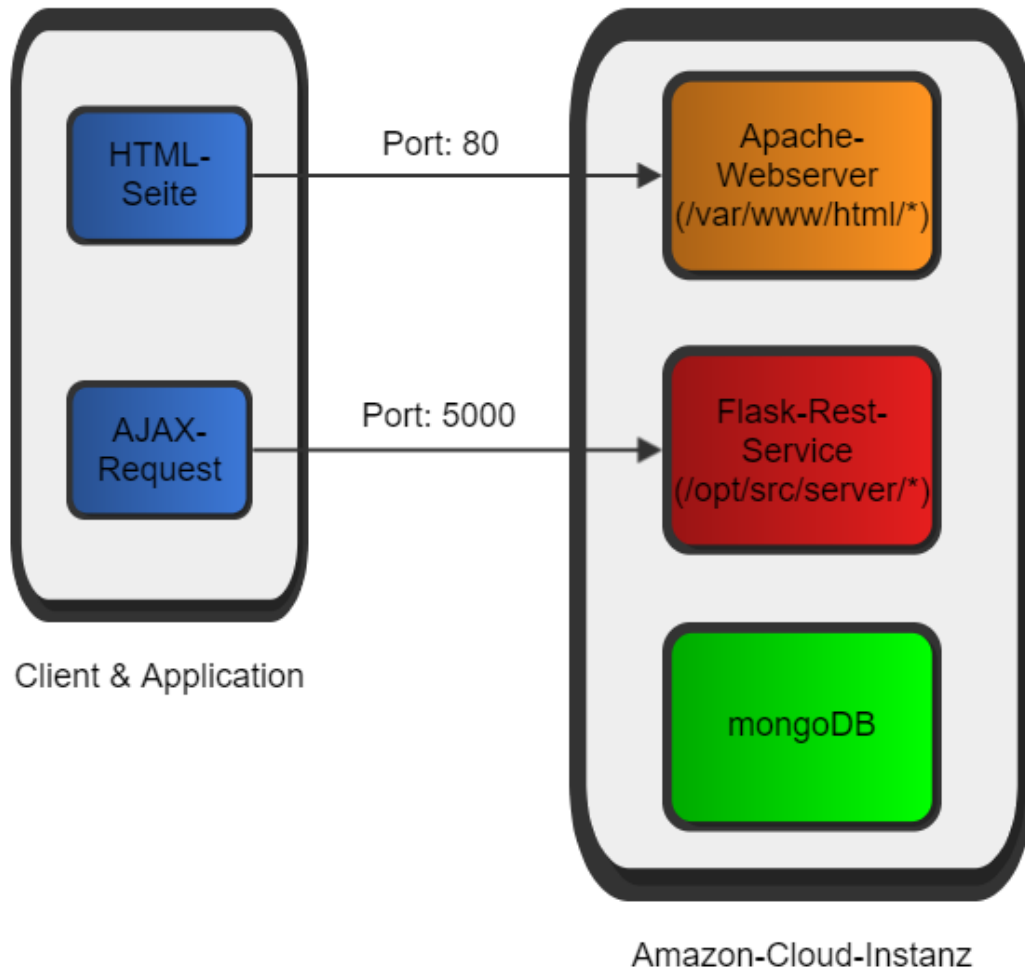


Abbildung 1: Infrastruktureller Aufbau

### 1.1 Erklärung UI

Der Apache Webserver stellt die Website zur Verfügung, diese ist unter [ec2-52-29-118-210.eu-central-1.compute.amazonaws.com](http://ec2-52-29-118-210.eu-central-1.compute.amazonaws.com) abrufbar, alternativ auch unter der blanken IP [52.29.118.210](http://52.29.118.210). Will man seine lokalen Änderungen auf den Server bringen, geht man mit `cd /var/www` in das Verzeichnis und schreibt **git pull origin master**, um sich den aktuellen Stand vom Server, den man gerade gepushed hat, zu holen und schon werden die Änderungen auf der Website angezeigt. Dies sollte in der Regel ohne einen Neustart des Apache Servers zu schaffen sein.

Der UI-Teil und der Server-Teil sind im Repository in den Ordnern **/html** und **/src** vertreten. Diese sind auch auf dem Server nur teilweise ausgecheckt und strukturell voneinander getrennt. Das bedeutet, dass die Sourcen des Rest-Servers aus sicherheitstechnischen Gründen nicht im Apache-Verzeichnis liegen und umgekehrt.

## 1.2 Erklärung Server

Der Flask-Server stellt den Rest-Service zur Verfügung. Will man seine lokalen Änderungen auf den Server bringen, geht man mit `cd /opt` in das Verzeichnis und schreibt **git pull origin master**, um sich den aktuellen Stand vom Server zu holen. Nun muss der Rest Service mit **python /opt/src/server/server.py** neu gestartet werden.

## 2 Aufteilung Datensatz

Der zu verwendende Datensatz wurde von nachfolgender Website bezogen:

<https://data.cityofchicago.org/Public-Safety/Crimes-2001-to-present/ijzp-q8t2>

Um die Ausführbarkeit unseres Projekts zu gewährleisten, haben wir den Datensatz entsprechend in Jahre unterteilt, was sich an den korrespondierenden .csv-Dateien im Ordner **.\datasets\** zeigt, da das Einlesen des Gesamtdatensatzes mit einer Größe von ~ 1,4 GB Schwierigkeiten auf manchen Systemen verursacht hat.

Die Aufteilung des Datensatzes erfolgt auf der Website, da jene eine explizite Filterung nach Jahren erlaubt.

Nach entsprechender Filterung wurden die .csv-Dateien heruntergeladen.

## 3 Erklärung Python-Dateien

### 3.1 „csv\_n\_table\_creation.py“

Diese .py-Datei wird initial nach Einrichtung der Mongo-Datenbank ausgeführt. Jene Datei erzeugt die Tabellen mit der Bezeichnung ‚2001‘ bis ‚2014‘ (sofern noch nicht existent) korrespondierend zur gerade verarbeiteten .csv-Datei.

Zur Verarbeitung der .csv-Dateien wird das **Framework ‚pandas‘** verwendet, welches es im Vorfeld zu installieren gilt. Die Verwendung von ‚pandas‘ wurde bedingt durch

den enormen Geschwindigkeitsvorteil gegenüber normalen Iterationen innerhalb Pythons gewählt.

Unter vorheriger Definition wird nur eine bestimmte Anzahl an Spalten in unsere Datenbank übernommen, was in der Beschleunigung der Verarbeitungsgeschwindigkeit begründet ist.

Zusätzlich werden, um die Reaktionsgeschwindigkeit der Website zu erhöhen und unnötigen Traffic zu vermeiden, eigens für die Website, extra Tabellen erzeugt, wie jene mit der Bezeichnung ‚arrest‘, ‚time‘, ‚commonc‘, ‚locX‘, was nachfolgend noch erläutert wird. Dies hat den Zweck, dass Anfragen nicht über die ganze Datenbank gehen, da es sehr viel Zeit und Bandbreite beansprucht, sondern nur explizit auf kleine Datensätze innerhalb der Datenbank.

Bei Überführung der Datensätze in die Datenbank wurde nicht zwingend der Grundsatz der Normalisierung eingehalten.

Nachfolgend ein Screenshot, welcher alle, durch dieses Script, zu generierenden Tabellen, aufzeigt:

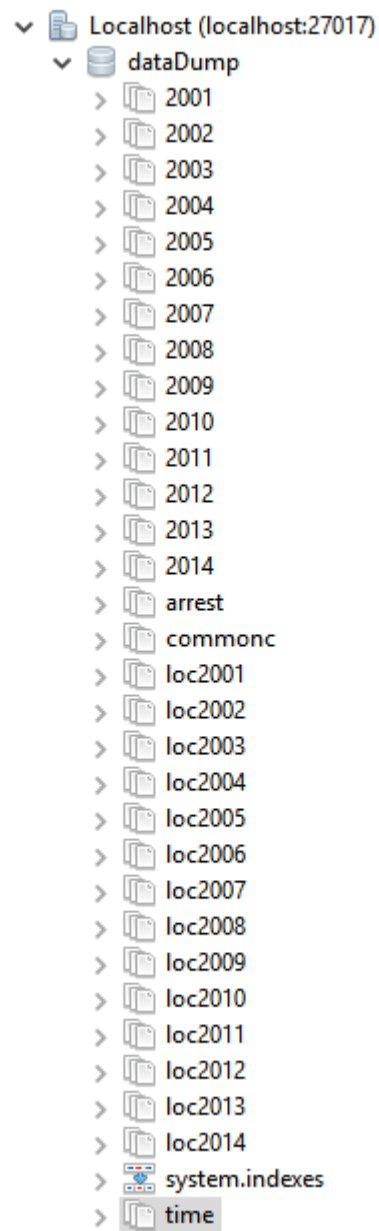
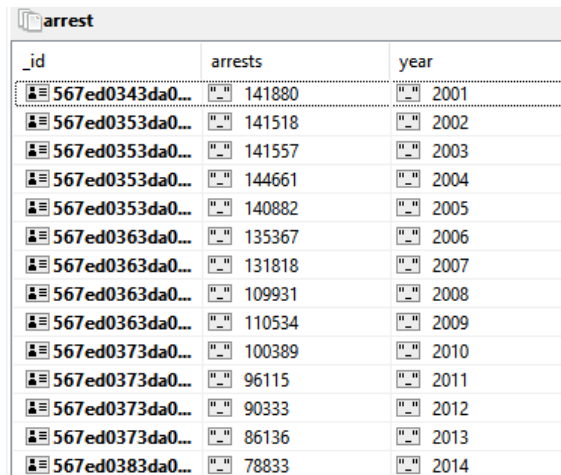


Abbildung 2: Ansicht aller zu erzeugenden Tabellen

### 3.2 “create\_arrest\_table.py”

Diese Klasse überprüft bei Ihrer Ausführung, ob die Tabelle ‚arrest‘ bereits besteht. Ist dem nicht so, wird jene erzeugt. Die Tabelle ‚arrest‘ beinhaltet die Anzahl aller Verbrechen - in Summe - pro Jahr, die zu einer Verhaftung geführt haben. Anbei ein Screenshot der Datenbank, welcher den Aufbau der Tabelle verdeutlicht:



_id	arrests	year
567ed0343da0...	141880	2001
567ed0353da0...	141518	2002
567ed0353da0...	141557	2003
567ed0353da0...	144661	2004
567ed0353da0...	140882	2005
567ed0363da0...	135367	2006
567ed0363da0...	131818	2007
567ed0363da0...	109931	2008
567ed0363da0...	110534	2009
567ed0373da0...	100389	2010
567ed0373da0...	96115	2011
567ed0373da0...	90333	2012
567ed0373da0...	86136	2013
567ed0383da0...	78833	2014

Abbildung 3: Darstellung Tabelle ‚arrest‘

Zur Bewerkstelligung dieser Ansicht wird der Datensatz fürs jedes Jahr aus der Datenbank bezogen. Dieser Datensatz wird nach den Werten , „Arrest“: True , überprüft, und anschließend die Anzahl jener Ergebnisse mit **selector.count()** in Kombination mit dem Jahr in die Datenbank geschrieben.

### 3.3 “create\_common\_crime\_table.py”

Diese Klasse überprüft bei Ihrer Ausführung ob die Tabelle ‚commonc ‘ bereits besteht. Ist dem nicht so, wird jene erzeugt.

Diese Tabelle beinhaltet die Anzahl von Verbrechenstypen - mit Ihrer Auftrittshäufigkeit für jedes Jahr. Anbei ein Screenshot der Datenbank, welcher den Aufbau der Tabelle verdeutlicht:



ROBBERY	SEX OFFENSE	STALKING	THEFT	WEAPONS VIOLATI...	year
18441	2225	203	99263	4274	2001
18522	2166	200	98326	4281	2002
17332	2065	247	98875	4211	2003
15978	1800	215	95463	4297	2004
16047	1799	192	85684	4106	2005
15968	1558	186	86236	3821	2006
15450	1515	213	85154	3554	2007
16703	1476	190	88430	3877	2008
15931	1239	167	80791	4144	2009
14273	1106	189	76746	3704	2010
13980	1059	181	75132	3879	2011
13483	1031	207	75447	3904	2012
11819	1004	153	71509	3246	2013
9800	907	140	61488	3110	2014

Abbildung 4: Darstellung Tabelle ‚common‘

Damit die Tabelle ihr oben dargestelltes Erscheinungsbild hat, wird für jede Jahres-Collection nach der Spalte „Primary Type“ gefiltert, und deren Ergebnisse werden vorerst vom Format Unicode nach String umgewandelt, um Vergleichsoperationen fehlerfrei durchführen zu können.

Anschließend wird überprüft, ob jene Verbrechenart, über welche im Moment gerade iteriert wird, in einer Iteration bereits vorhanden war oder nicht, zu einer entsprechenden Liste hinzugefügt.

```
# This is a faster index checker than usual...
try:
    b=types_Array.index(type_Var)

# If the crime type does not yet exist in the list
except ValueError:
    types_Array.extend([type_Var])          # adds that type into the list
    types_Array_Count.extend([1])          # adds a counter into the count
list

# If crime type does already exist
else:
    # raise counter by one, depending on index
    types_Array_Count[types_Array.index(type_Var)] +=1
```

Abbildung 5: Hinzufügen zur entsprechenden Liste

Hierbei werden zwei Listen zeitgleich gepflegt, wobei eine Liste die Art des Verbrechens beinhaltet („types\_Array“) und die andere Liste die Anzahl ihrer Erscheinungshäufigkeit („types\_Array\_Count“). Hierbei wird mit gleichen Indexes gearbeitet, sprich „types\_Array\_Count[3]“ bezieht sich auf „types\_Array[3]“. Dies ermöglicht eine vereinfachte Generierung eines Dictionaries im Nachhinein und erleichtert den Schreibaufwand in die Datenbank.

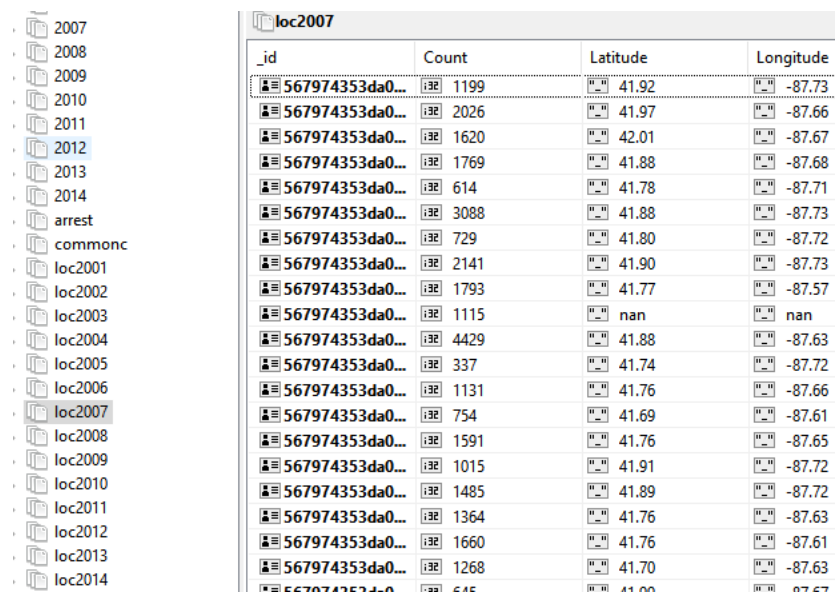
### 3.4 “create\_location\_table.py”

Diese Klasse überprüft bei Ihrer Ausführung, ob die Tabelle ‚locX‘ bereits besteht – X steht korrespondierend für das Jahr. Ist dem nicht so, werden jene Tabellen erzeugt.

Diese beinhalten Paare von Längen- und Breitengrad, auf die dritte Nachkommastelle gerundet, mit der Anzahl ihrer Erscheinungshäufigkeit.

Wir haben uns für das Runden der Geokoordinaten entschieden, da Anfragen an die Datenbank nicht korrekt beantwortet können. Dies zeigt sich darin, dass bei Rundung auf die zweite Nachkommastelle die Heatmaps sehr ungenau werden und ein enorm großes Raster aufweise. Dies hat uns dazu veranlasst, auf die dritte Nachkommastelle zu runden, was allerdings zur Folge hatte, dass das zurückgegebene Datenpaket nun nicht mehr ~ 0,1 MB sondern ~ 4 MB groß ist. Würde man auf die vierte oder mehr Nachkommastellen runden, würde die Datengröße exponentiell steigen und eine Bedienung und performante Darstellbarkeit der Website könnte nicht mehr gewährleistet werden.

Der Aufbau der locX-Tabellen ist nachfolgend dargestellt:



_id	Count	Latitude	Longitude
567974353da0...	1199	41.92	-87.73
567974353da0...	2026	41.97	-87.66
567974353da0...	1620	42.01	-87.67
567974353da0...	1769	41.88	-87.68
567974353da0...	614	41.78	-87.71
567974353da0...	3088	41.88	-87.73
567974353da0...	729	41.80	-87.72
567974353da0...	2141	41.90	-87.73
567974353da0...	1793	41.77	-87.57
567974353da0...	1115	nan	nan
567974353da0...	4429	41.88	-87.63
567974353da0...	337	41.74	-87.72
567974353da0...	1131	41.76	-87.66
567974353da0...	754	41.69	-87.61
567974353da0...	1591	41.76	-87.65
567974353da0...	1015	41.91	-87.72
567974353da0...	1485	41.89	-87.72
567974353da0...	1364	41.76	-87.63
567974353da0...	1660	41.76	-87.61
567974353da0...	1268	41.70	-87.63

Abbildung 6: Darstellung ‚loc2007‘ Tabelle, repräsentativ für ‚locX‘

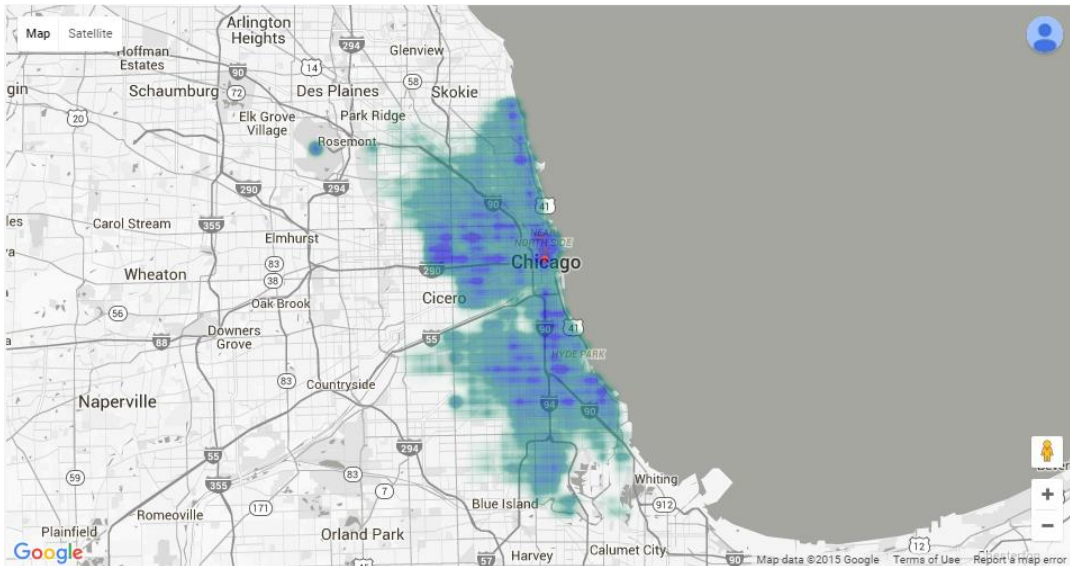


Abbildung 7: Darstellung Heatmap auf zwei Nachkommastellen gerundet

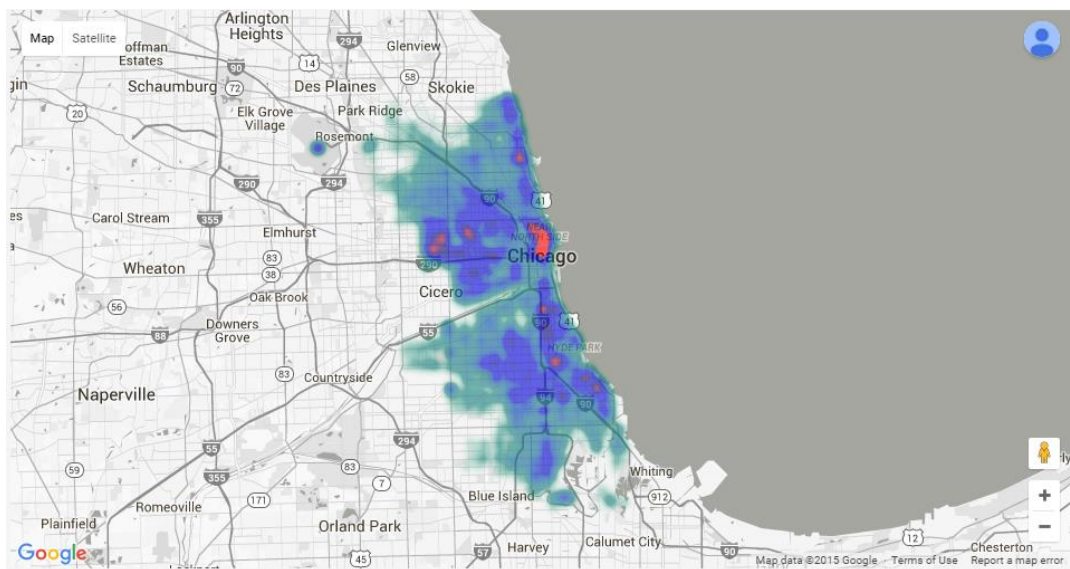


Abbildung 8: Darstellung Heatmap auf drei Nachkommastellen gerundet

### 3.5 “create\_time\_table.py”

Diese Klasse überprüft bei ihrer Ausführung, ob die Tabelle ‚time‘ bereits besteht. Ist dem nicht so, wird die Tabelle erzeugt.

Diese Tabelle beinhaltet die Anzahl aller Verbrechen - sortiert nach Tageszeit im 24h-Format. Die Schwierigkeit bei dieser Tabellenerzeugung war, dass im Ursprungsdatensatz die dritte Normalform nicht eingehalten wurde, was sich daran zeigt, dass als Key eine Kombination von Datum mit Zeitstempel im 12h-Format angegeben wird „02/26/2001 07:15:00 PM“ angegeben wird.

In Folge dessen wird jede Jahrescollection zuerst nach „Date“ gefiltert, jenes Ergebnis von Unicode zu String umgewandelt und anschließend mehrfach gesplittet, um an die nackte Stundenzahl zu gelangen.

Hierbei wurde ein großes IF-ELSEIF-Statement verwendet um, bedingt durch die Unterteilung in AM/PM ein 24h-Stundenformat gewährleisten zu können.

Die Unterteilung geschieht in der Art, dass beispielsweise die Zeiten 01:00:00 bis einschließlich 01:59:59 zur Stunde 1 (1 Uhr) gezählt werden.

Der Aufbau der Tabelle ‚time‘ ist nachfolgend dargestellt:

6	7	8	9	year
6808	10067	15404	18942	2001
7183	10590	15926	19589	2002
7065	10411	15635	18494	2003
7294	10381	15254	18844	2004
6887	9940	14869	18491	2005
7165	10460	15206	18581	2006
6633	10055	14723	18263	2007
6341	9441	14575	18219	2008
5969	8644	12852	17026	2009
5919	8372	12747	16484	2010
5897	8220	12808	16022	2011
5571	8083	11905	15700	2012
4980	7206	10810	14524	2013
4716	6283	9148	12553	2014

Abbildung 9: Darstellung ‚time‘ Tabelle

### 3.6 “timeCrime.py”

Die „main\_Method“ dieser Datei wird im Zuge des REST-Services ausgeführt. Diese Methode holt sich aus der Datenbank die Tabelle ‚time‘ und bereitet jene dergestalt auf, dass diese für das entsprechende **HighCharts-Diagramm** (jedes Diagramm möchte die Daten anders aufbereitet haben) einlesbar ist.