Wirtschaftsinformatik 2

Pandas JOIN DataFrames

Prof. Dr. Christian Bockermann, Prof. Dr. Volker Klingspor

Hochschule Bochum

Wintersemester 2026/2027

Vorbemerkungen zum Block Pandas JOIN DataFrames

In den letzten zwei Foliensätzen ging es um die Berechnung, Filterung und das Gruppieren *einer* Tabelle (bzw. eines DataFrames). Daten liegen häufig nicht in einer einzigen Tabelle vor. In vielen Fällen ist es sogar sinnvoll, die vorliegenden Daten mit Daten aus anderen Quellen (z.B. Wetterdaten) zu ergänzen.
In *Wirtschaftsinformatik 1* haben wir die Verbindung von mehreren Tabellen in einer Datenbank über den `JOIN` Befehl realisiert. Das zugrundeliegende Konzept hat Fremdschlüsselbeziehungen benutzt, die aus dem ER-Modell abgeleitet wurden. In ähnlicher Weise unterstützt auch Pandas das Zusammenführen von mehreren Tabellen über Fremdschlüssel. Der wesentliche Unterschied ist hier nun, dass wir in der Regel kein vorab modelliertes Datenmodell vorliegen haben.
In diesem Block geht es um die einfache Verbindung von zwei DataFrames. Das ist keine vollständige Behandlung des Themas, sondern soll nur die grundsätzlichen Ideen aufzeigen.

Beispiel: Kuchenverkauf in einem Café

Wir betrachten als Beispiel ein kleines Café in Bochum. Dort werden jeden Tag unterschiedliche Kuchen angeboten, die Sie als Betreiber von einer Konditorei bestellen. Im folgenden Datensatz finden Sie einen Auszug aus der Kasse mit den verkauften Kuchen je Bon:

BonNr	Datum	Apfelkuchen	Kirschkuchen	Zitronenkuchen	Sacher
77	2023-10-12	0	0	0	0
78	2023-10-12	0	0	0	0
79	2023-10-12	1	0	0	0
80	2023-10-12	0	0	0	0

Aus betriebswirtschaftlicher Sicht, möchten Sie natürlich nicht zuviel Kuchen bestellen, der dann ggf. entsorgt werden muss. Zu wenig Kuchen kann ein Ärgernis für Kunden sein, die leer ausgehen. Daher stellt sich die Frage:

Wieviel Kuchen müssen Sie für die nächste Woche bestellen?

Beispiel: Kuchenverkauf in einem Café

Die Beispiel-Daten können Sie direkt einlesen. Die Bon Daten selbst stellen jede Bestellung einzeln dar. Zur Beantwortung unserer Fragen müssen die Daten daher noch *pro Datum* zusammengefasst werden:

```
kuchen = pd.read_csv('https://data.hsbo.de/kuchen.csv')
kuchenProTag = kuchen.groupby('Datum').sum()

# Gesamt-Spalte berechnen:
kuchenProTag['Gesamt'] = kuchenProTag.sum(axis=1)
```

	Apfelkuchen	Kirschkuchen	Zitronenkuchen	Sacher	Gesamt
2023-10-11	6	12	13	14	45
2023-10-12	4	0	0	2	6
2023-10-13	8	4	0	2	14
2023-10-16	0	1	0	0	1

Beispiel: Kuchenverkauf in einem Café

Die Anzahl der verkauften Kuchen hängt natürlich von vielen Faktoren ab. Wenn das Café geschlossen ist, wird kein Kuchen verkauft, saisonal bedingt gibt es bestimmte Kuchen, die z.B. im Sommer häufiger verkauft werden, usw.
Ein Faktor, den wir hier näher betrachten wollen ist das Wetter. Bei warmen Temperaturen und ohne Regen haben Sie draußen noch mehr Plätze die belegt sind und dementsprechend mehr Gäste.
Wetterdaten hatten wir in der Vorlesung, z.B. schonmal über die Bochumer Smarten Bänke betrachtet (vgl. Plenum-Folien zu Block 5), hier als Ausschnitt:

Zeitpunkt	Temperatur	Luftfeuchtigkeit	Regenmenge/Std	Wind (m/s)	Luftdruck
2024-04-12 14:09:36+02:00	18.5	64.0	0.0	1.3	1020.1
2024-04-12 14:16:00+02:00	19.0	61.0	0.0	0.7	1020.0
2024-04-12 14:20:16+02:00	19.3	60.0	0.0	0.7	1020.0
2024-04-12 21:09:52+02:00	16.2	77.0	0.0	4.1	1015.2
2024-04-12 21:14:08+02:00	16.2	77.0	0.0	4.2	1015.2

Beispiel: Kuchenverkauf in einem Café

Im Auszug der Wetterdaten erkennt man, dass die Daten mehrmals am Tag erhoben werden – teilweise mehrere Messungen pro Stunde. Das ist für unseren Fall natürlich nicht so geeignet, daher bietet es sich auch hier an, die Daten zu gruppieren und zu Aggregieren:

```
wetter = pd.read_csv('..')

# Zeitpunkt konvertieren und Spalte 'Datum' erzeugen
wetter['Zeitpunkt'] = pd.to_datetime(wetter['Zeitpunkt'])
wetter['Datum'] = wetter['Zeitpunkt'].dt.date

# Nur die Spalten Datum, Temperatur und Regenmenge/Std
wetter = wetter['Datum','Temperatur','Regenmenge/Std']

# Mittelwerte (Temperatur, Regenmente) pro Tag berechnen:
daten = wetter.groupby('Datum').mean()

# alle Zellen auf 3 Nachkommastellen runden
daten = daten.round(3)
```

Beispiel: Kuchenverkauf in einem Café

Aggregiert sehen die Wetterdaten dann folgendermaßen aus:

	Temperatur	Regen
2024-04-12	16.073	0.0
2024-04-13	18.16	0.0
2024-04-14	14.437	0.0
2024-04-15	8.017	2.401
2024-04-16	6.648	5.032

DataFrames verbinden

DataFrames verbinden

DataFrames sind die Datentypen, die in Pandas Tabellen darstellen. Sie bestehen aus einem Zeilenindex (`.index`) und Spaltenindex (`.columns`), über die sich alle Zeilen, Spalten und Zellen referenzieren lassen.

Spalten-Index

`df.columns`

	a1	a2	a3	a4
0	**4**	**1**	**2**	**9**
1	**5**	**1**	**3**	**6**
2	**3**	**8**	**7**	**4**

Zeilen-Index

`df.index`

DataFrame `.merge(..)`

Der einfachste Weg, einen DataFrame um die Daten eines anderen DataFrame zu erweitern ist über die `.merge(..)` Methode. Wichtig dabei ist, dass es eine Spalte in beiden DataFrames gibt, über die die Zuordnung der Zeilen zueinander erfolgen kann.
Wir betrachten die folgenden DataFrames `kuchenVerkauf` und `wetter`:

	Datum	Anzahl
0	2024-11-30	23
1	2024-12-01	8
2	2024-12-02	33
3	2024-12-03	28
4	2024-12-04	12

`kuchenVerkauf`

	Datum	Temperatur	Regen
0	2024-11-27	16.1	0.0
1	2024-11-28	18.2	0.0
2	2024-11-29	14.4	0.0
3	2024-11-30	8.0	2.4
4	2024-12-01	6.6	5.0
5	2024-12-02	4.9	1.4
6	2024-12-03	7.5	0.3

`wetter`

DataFrame .merge(..)

	Datum	Anzahl
0	2024-11-30	23
1	2024-12-01	8
2	2024-12-02	33
3	2024-12-03	28
4	2024-12-04	12

	Datum	Temperatur	Regen
0	2024-11-27	16.1	0.0
1	2024-11-28	18.2	0.0
2	2024-11-29	14.4	0.0
3	2024-11-30	8.0	2.4
4	2024-12-01	6.6	5.0
5	2024-12-02	4.9	1.4
6	2024-12-03	7.5	0.3

`kuchenVerkauf.merge( wetter )` → `daten`

	Datum	Anzahl	Temperatur	Regen
0	2024-11-30	23	8.0	2.4
1	2024-12-01	8	6.6	5.0
2	2024-12-02	33	4.9	1.4
3	2024-12-03	28	7.5	0.3

```
daten = kuchenVerkauf.merge(wetter)
```

DataFrame .merge(..)

	Datum	Anzahl
0	2024-11-30	23
1	2024-12-01	8
2	2024-12-02	33
3	2024-12-03	28
4	2024-12-04	12

	Datum	Temperatur	Regen
0	2024-11-27	16.1	0.0
1	2024-11-28	18.2	0.0
2	2024-11-29	14.4	0.0
3	2024-11-30	8.0	2.4
4	2024-12-01	6.6	5.0
5	2024-12-02	4.9	1.4
6	2024-12-03	7.5	0.3

`kuchenVerkauf.merge( wetter )` → `daten`

	Datum	Anzahl	Temperatur	Regen
0	2024-11-30	23	8.0	2.4
1	2024-12-01	8	6.6	5.0
2	2024-12-02	33	4.9	1.4
3	2024-12-03	28	7.5	0.3

```
daten = kuchenVerkauf.merge(wetter)
```

DataFrame .merge(..)

	Datum	Anzahl
0	2024-11-30	23
1	2024-12-01	8
2	2024-12-02	33
3	2024-12-03	28
4	2024-12-04	12

	Datum	Temperatur	Regen
0	2024-11-27	16.1	0.0
1	2024-11-28	18.2	0.0
2	2024-11-29	14.4	0.0
3	2024-11-30	8.0	2.4
4	2024-12-01	6.6	5.0
5	2024-12-02	4.9	1.4
6	2024-12-03	7.5	0.3

`kuchenVerkauf.merge( wetter )` → `daten`

	Datum	Anzahl	Temperatur	Regen
0	2024-11-30	23	8.0	2.4
1	2024-12-01	8	6.6	5.0
2	2024-12-02	33	4.9	1.4
3	2024-12-03	28	7.5	0.3

```
daten = kuchenVerkauf.merge(wetter)
```

DataFrame .merge(..)

	Datum	Anzahl
0	2024-11-30	23
1	2024-12-01	8
2	2024-12-02	33
3	2024-12-03	28
4	2024-12-04	12

	Datum	Temperatur	Regen
0	2024-11-27	16.1	0.0
1	2024-11-28	18.2	0.0
2	2024-11-29	14.4	0.0
3	2024-11-30	8.0	2.4
4	2024-12-01	6.6	5.0
5	2024-12-02	4.9	1.4
6	2024-12-03	7.5	0.3

`kuchenVerkauf.merge( wetter )` → `daten`

	Datum	Anzahl	Temperatur	Regen
0	2024-11-30	23	8.0	2.4
1	2024-12-01	8	6.6	5.0
2	2024-12-02	33	4.9	1.4
3	2024-12-03	28	7.5	0.3

```
daten = kuchenVerkauf.merge(wetter)
```

DataFrame `.merge(..)`

Die Methode `.merge(..)` sucht nach Spalten, die in beiden DataFrames vorkommen und gleicht damit die Zeilen ab. Es wird ein neuer DataFrame zurückgegeben, der alle Zeilen enthält, für die es in den gemeinsamen Spalten den gleichen Wert gibt.

Häufig will man aber die Spalte, die in beiden für das zusammenführen benutzt werden soll, explizit angeben, in diesem Fall *Datum*:

```
daten = kuchenVerkauf.merge(wetter, on='Datum')
```

Damit fallen z.B. die Zeilen mit dem Datum 2024-11-15 raus, weil es dieses Datum nicht in allen DataFrames gibt.

DataFrame .merge(..) – Left Join

Wie bei SQL (W-Inf1) auch, gibt es die Möglichkeit, alle Zeilen der linken Seite zu nehmen und mit den entsprechenden Zeilen aus dem anderen DataFrame aufzufüllen:

```
daten = kuchenVerkauf.merge(wetter, on='Datum',
                            how='left')
```

	Datum	Anzahl	Temperatur	Regen
0	2024-11-30	23	8.0	2.4
1	2024-12-01	8	6.6	5.0
2	2024-12-02	33	4.9	1.4
3	2024-12-03	28	7.5	0.3
4	2024-12-04	12	nan	nan

Zellen, die aus der zweiten Tabelle nicht gefüllt werden können, werden mit NaN (not-a-number) gefüllt.

DataFrame `.merge(..)` – how?

Weitere Möglichkeiten für den Parameter **how** sind

- `outer` – Alle Zeilen aus linker und rechter Tabelle, Rest mit NaN auffüllen.
- `right` – Alle Zeilen von linkem DataFrame, ggf. werden die Felder des rechten Frames aufgefüllt.

	Datum	Anzahl	Temperatur	Regen
0	2024-11-30	23.0	8.0	2.4
1	2024-12-01	8.0	6.6	5.0
2	2024-12-02	33.0	4.9	1.4
3	2024-12-03	28.0	7.5	0.3
4	2024-12-04	12.0	nan	nan
5	2024-11-27	nan	16.1	0.0
6	2024-11-28	nan	18.2	0.0

```
...merge(wetter, how='outer')
```

	Datum	Anzahl	Temperatur	Regen
0	2024-11-27	nan	16.1	0.0
1	2024-11-28	nan	18.2	0.0
2	2024-11-29	nan	14.4	0.0
3	2024-11-30	23.0	8.0	2.4
4	2024-12-01	8.0	6.6	5.0
5	2024-12-02	33.0	4.9	1.4
6	2024-12-03	28.0	7.5	0.3

```
...merge(wetter, how='right')
```

Weitere Details zu `.merge(..)`

Manchmal enthalten die Tabellen Spalten, mit denen Sie verbunden werden sollen, allerdings heissen diese ggf. nicht gleich. Über die Parameter `left_on` und `right_on` lassen sich die Spalten der linken und rechten Tabellen festlegen, über die die Daten zusammengeführt werden sollen:

Datum	BestellungNr	KundeNr	Betrag
2024-06-10	211	211	0.15
2024-06-10	215	215	0.1
2024-06-10	219	219	0.15
2024-06-10	222	222	0.15
2024-06-11	229	229	0.05

`bestellungen`

ID	Altersgruppe	Kundengruppe
211	0-5	KigaKind
215	31-40	Mitarbeiter
219	6-10	Schulkind
222	0-5	KigaKind
229	41-50	Mitarbeiter

`kunden`

```
bestellungen.merge(kunden, left_on='KundeNr', right_on='ID')
```

Weitere Details zu `.merge(..)`

Manchmal enthalten die Tabellen Spalten, mit denen Sie verbunden werden sollen, allerdings heissen diese ggf. nicht gleich. Über die Parameter `left_on` und `right_on` lassen sich die Spalten der linken und rechten Tabellen festlegen, über die die Daten zusammengeführt werden sollen:

Datum	BestellungNr	KundeNr	Betrag
2024-06-10	211	211	0.15
2024-06-10	215	215	0.1
2024-06-10	219	219	0.15
2024-06-10	222	222	0.15
2024-06-11	229	229	0.05

`bestellungen`

ID	Altersgruppe	Kundengruppe
211	0-5	KigaKind
215	31-40	Mitarbeiter
219	6-10	Schulkind
222	0-5	KigaKind
229	41-50	Mitarbeiter

`kunden`

```
bestellungen.merge(kunden, left_on='KundeNr', right_on='ID')
```

Weitere Literatur...

Das Thema join/merge ist recht vielschichtig und erfordert in der Praxis in vielen Fällen eine Reihe von vorbereitenden Schritten (z.B. Filtern und Gruppieren). Da wir uns in dieser Vorlesung auf die grundlegenden Konzepte fokussieren, führt dies weit über die Inhalte der Vorlesung hinaus.

An dieser Stelle sei für interessierte Leser auf das Kapitel 8.2 des Buches *Datenanalyse mit Python* verwiesen, das ein umfassendes Bild aller Möglichkeiten zum Verbinden von Datentabellen bietet.