

ich will jetzt meine gescripteten Daten mit der csv Datei key_stats.csv verknüpfen. ich will dafür die spalten player_name und club benutzen. wenn es beim player name mehrere gleiche Namen gibt soll überprüft werden ob der Vorname aus der first_name spalte in der player_name spalte ist oder ob der anfangs Buchstabe des Vornamens mit einem Punkt dahinter in der player_name spalte ist. wenn ja dann kann die Daten aus der Zeile verknüpfen. schreibe mir dafür einen code. so sehen meine cleanen gescripteten Daten aus:

```
player_name,market_value,club,first_name Courtois,60.0,Real Madrid,Thibaut
Lunin,2.5,Real Madrid,Andriy López,0.3,Real Madrid,Luis Fuidias,0.3,Real Madrid,Toni
Militão,60.0,Real Madrid,Éder Alaba,55.0,Real Madrid,David Fernández,7.0,Real
Madrid,Nacho Vallejo,3.0,Real Madrid,Jesús Gila,0.5,Real Madrid,Mario Marín,0.4,Real
Madrid,Rafa Mendy,40.0,Real Madrid,Ferland Gutiérrez,6.0,Real Madrid,Miguel
Marcelo,3.0,Real Madrid,Marcelo Carvajal,18.0,Real Madrid,Daniel Vázquez,12.0,Real
Madrid,Lucas Odriozola,12.0,Real Madrid,Álvaro Santos,0.25,Real Madrid,Sergio
Casemiro,40.0,Real Madrid,Casemiro Blanco,5.0,Real Madrid,Antonio Valverde,70.0,Real
Madrid,Federico Camavinga,55.0,Real Madrid,Eduardo Kroos,20.0,Real Madrid,Toni
Ceballos,14.0,Real Madrid,Dani Modric,10.0,Real Madrid,Luka Isco,7.5,Real Madrid,Isco
Junior,100.0,Real Madrid,Vinicius Hazard,12.0,Real Madrid,Eden Rodrygo,60.0,Real
Madrid,Rodrygo Asensio,40.0,Real Madrid,Marco Bale,3.0,Real Madrid,Gareth
Federico,1.0,Real Madrid,Peter Benzema,30.0,Real Madrid,Karim Jović,16.0,Real
Madrid,Luka Díaz,6.0,Real Madrid,Mariano Latasa,1.0,Real Madrid,Juanmi
Alisson,50.0,Liverpool,Alisson Kelleher,8.0,Liverpool,Caoimhín Karius,1.5,Liverpool,Loris
Adrián,1.0,Liverpool,Adrián Pitaluga,0.1,Liverpool,Marcelo Van dijk,55.0,Liverpool,Virgil
Konaté,40.0,Liverpool,Ibrahima Gomez,18.0,Liverpool,Joe Matip,18.0,Liverpool,Joel
Phillips,8.0,Liverpool,Nat Van den berg,5.0,Liverpool,Sepp Williams,4.0,Liverpool,Rhys
Davies,2.5,Liverpool,Ben Koumetio,0.8,Liverpool,Billy Robertson,65.0,Liverpool,Andrew
Tsimikas,15.0,Liverpool,Konstantinos Beck,0.4,Liverpool,Owen Alexander-
arnold,80.0,Liverpool,Trent Williams,8.0,Liverpool,Neco Bradley,1.0,Liverpool,Conor
Fabinho,60.0,Liverpool,Fabinho Morton,3.0,Liverpool,Tyler Jones,28.0,Liverpool,Curtis
Keita,25.0,Liverpool,Naby Elliott,22.0,Liverpool,Harvey Alcántara,20.0,Liverpool,Thiago
Oxlade-chamberlain,16.0,Liverpool,Alex Henderson,15.0,Liverpool,Jordan
Milner,2.0,Liverpool,James Dixon-bonner,0.3,Liverpool,Elijah Woodburn,1.2,Liverpool,Ben
Mané,70.0,Liverpool,Sadio Díaz,65.0,Liverpool,Luis Minamino,12.0,Liverpool,Takumi
Salah,90.0,Liverpool,Mohamed Ojo,2.0,Liverpool,Sheyi Gordon,1.5,Liverpool,Kaide
Blair,0.3,Liverpool,Harvey Jota,60.0,Liverpool,Diogo Firmino,32.0,Liverpool,Roberto
Origi,12.0,Liverpool,Divock Woltman,0.3,Liverpool,Max Ederson,45.0,Man. City,Ederson
Steffen,6.0,Man. City,Zack Carson,0.3,Man. City,Scott Dias,75.0,Man. City,Rúben
Laporte,45.0,Man. City,Aymeric Stones,28.0,Man. City,John Aké,25.0,Man. City,Nathan
Sandler,2.0,Man. City,Philippe Egan-riley,0.8,Man. City,CJ Mbete,0.6,Man. City,Luke
Burns,0.3,Man. City,Finley Zinchenko,25.0,Man. City,Oleksandr Wilson-esbrand,0.4,Man.
City,Josh Cancelo,65.0,Man. City,João Walker,18.0,Man. City,Kyle Couto,5.0,Man. City,Yan
Rodri,80.0,Man. City,Rodri Lavia,2.0,Man. City,Roméo Fernandinho,1.5,Man.
City,Fernandinho Gomes,1.0,Man. City,Claudio Gündoğan,25.0,Man. City,İlkay
Doyle,2.5,Man. City,Tommy De bruyne,85.0,Man. City,Kevin Silva,80.0,Man. City,Bernardo
Palmer,6.0,Man. City,Cole Mcatee,3.0,Man. City,James Grealish,70.0,Man. City,Jack
Edozie,0.8,Man. City,Samuel Foden,90.0,Man. City,Phil Sterling,70.0,Man. City,Raheem
Torres,45.0,Man. City,Ferran Mahrez,35.0,Man. City,Riyad Kayky,14.0,Man. City,Kayky
Jesus,50.0,Man. City,Gabriel Delap,1.5,Man. City,Liam Rulli,6.0,Villarreal,Gerónimo
Asenjo,4.0,Villarreal,Sergio Álvarez,0.3,Villarreal,Iker Jørgensen,0.3,Villarreal,Filip
Torres,50.0,Villarreal,Pau Cuenca,8.0,Villarreal,Jorge Mandi,6.0,Villarreal,Aïssa
Gaspar,3.0,Villarreal,Mario Albiol,3.0,Villarreal,Raúl De la fuente,0.6,Villarreal,Adrián
Estupiñán,20.0,Villarreal,Pervis Pedraza,18.0,Villarreal,Alfonso
```

Moreno,5.0,Villarreal,Alberto Tasende,0.5,Villarreal,Dani Foyth,25.0,Villarreal,Juan Aurier,10.0,Villarreal,Serge Peña,2.5,Villarreal,Rubén Lanchi,0.2,Villarreal,Lanchi Coquelin,5.0,Villarreal,Francis Capoue,3.0,Villarreal,Étienne Iborra,2.0,Villarreal,Vicente Loscelso,22.0,Villarreal,Giovani Trigueros,14.0,Villarreal,Manu Parejo,9.0,Villarreal,Dani Morlanes,5.0,Villarreal,Manu García,0.4,Villarreal,Carlo Pacheco,0.2,Villarreal,Antonio Danjuma,50.0,Villarreal,Arnaut Gómez,8.0,Villarreal,Moi Iosifov,1.2,Villarreal,Nikita Pino,40.0,Villarreal,Yérémy Chukwueze,20.0,Villarreal,Samuel Raba,0.9,Villarreal,Dani Moreno,40.0,Villarreal,Gerard Dia,12.0,Villarreal,Boulaye Alcácer,9.0,Villarreal,Paco Niño,3.0,Villarreal,Fer Jackson,1.0,Villarreal,Nicolas Carlos arana,0.8,Villarreal,Juan

ChatGPT:

ChatGPT

Um die beiden Datensätze zu verknüpfen, basierend auf den Spalten `player_name` und `club`, und dabei zu berücksichtigen, dass Spieler mit denselben Namen eindeutig durch den Vornamen (`first_name`) identifiziert werden können, hier der Code:

Verknüpfung der Datensätze

```
python
KopierenBearbeiten
import pandas as pd

# CSV-Dateien laden
cleaned_data_file = "clean01_ucl_top4_teams_market_values.csv"
key_stats_file = "key_stats.csv"

cleaned_df = pd.read_csv(cleaned_data_file)
key_stats_df = pd.read_csv(key_stats_file)

# Funktion, um Player Name und Club abzugleichen
def match_player(row, cleaned_df):
    # Filter nach Club
    candidates = cleaned_df[cleaned_df['club'] == row['club']]

    # Genau übereinstimmende Namen finden
    exact_match = candidates[candidates['player_name'] ==
row['player_name']]
    if not exact_match.empty:
        return exact_match.iloc[0]

    # Prüfung mit Vorname in player_name
    contains_first_name = candidates[
        candidates['player_name'].str.contains(fr"\b{row['first_name']}\b",
case=False, na=False)
    ]
    if not contains_first_name.empty:
        return contains_first_name.iloc[0]

    # Prüfung mit Initial des Vornamens
    initial = row['first_name'][0].upper() if pd.notna(row['first_name'])
else ""
    initial_match = candidates[
        candidates['player_name'].str.startswith(f"{initial}.", na=False)
    ]
    if not initial_match.empty:
        return initial_match.iloc[0]
```

```

        return None

# Daten zusammenführen
matched_rows = []
for _, row in key_stats_df.iterrows():
    matched = match_player(row, cleaned_df)
    if matched is not None:
        combined_row = row.to_dict()
        combined_row.update(matched.to_dict()) # Werte aus beiden
DataFrames zusammenführen
        matched_rows.append(combined_row)

# Neues DataFrame erstellen
combined_df = pd.DataFrame(matched_rows)

# Ergebnis speichern
output_file = "merged_data.csv"
combined_df.to_csv(output_file, index=False)

print(f"Die verknüpften Daten wurden in '{output_file}' gespeichert.")

```

Erklärung des Codes

1. **Datenfilterung:**
 - Zuerst wird der `club` abgeglichen, um die Suche nach Kandidaten einzugrenzen.
 - Spieler mit exakt passendem `player_name` werden direkt übernommen.
2. **Vorname in `player_name`:**
 - Es wird geprüft, ob der Vorname (`first_name`) irgendwo im `player_name` enthalten ist, um die Zeile eindeutig zuzuordnen.
3. **Initial des Vornamens:**
 - Falls keine exakte Übereinstimmung oder kein Vorname gefunden wird, prüft der Code, ob der Initialbuchstabe des Vornamens gefolgt von einem Punkt (x.) mit dem `player_name` übereinstimmt.
4. **Verknüpfung der Daten:**
 - Bei einem Treffer werden die Werte aus den Zeilen beider DataFrames (`key_stats_df` und `cleaned_df`) kombiniert.
5. **Ergebnis:**
 - Die verknüpften Daten werden in eine neue CSV-Datei gespeichert.

Beispielausgabe

Eingangsdaten

- `cleaned_df`: Bereinigte Spieler-Daten mit den Spalten `player_name`, `market_value`, `club`, `first_name`.
- `key_stats_df`: Schlüsselstatistiken mit denselben Spielern, möglicherweise aber ohne eindeutige Zuordnung.

Ausgabe

Die Datei `merged_data.csv` enthält:

- Daten aus beiden DataFrames, korrekt verknüpft anhand von `player_name` und `club`.
- Falls der Name mehrdeutig ist, werden zusätzliche Prüfungen wie der Vorname und Initiale verwendet.

Typischer Inhalt von `merged_data.csv`:

player_name	market_value	club	first_name	stat1	stat2	...
Courtois	60.0	Real Madrid	Thibaut
Williams	8.0	Liverpool	Neco

40

Du:

dieser error kommt bei mir: -----
 KeyError Traceback (most recent call last) File /opt/anaconda3/lib/python3.11/site-packages/pandas/core/indexes/base.py:3791, in Index.get_loc(self, key) 3790 try: -> 3791 return self._engine.get_loc(casted_key) 3792 except KeyError as err: File index.pyx:152, in pandas._libs.index.IndexEngine.get_loc() File index.pyx:181, in pandas._libs.index.IndexEngine.get_loc() File pandas/_libs/hashtable_class_helper.pxi:7080, in pandas._libs.hashtable.PyObjectHashTable.get_item() File pandas/_libs/hashtable_class_helper.pxi:7088, in pandas._libs.hashtable.PyObjectHashTable.get_item() KeyError: 'first_name' The above exception was the direct cause of the following exception: KeyError Traceback (most recent call last) Cell In[33], line 42 40 matched_rows = [] 41 for _, row in key_stats_df.iterrows(): --> 42 matched = match_player(row, cleaned_df) 43 if matched is not None: 44 combined_row = row.to_dict() Cell In[33], line 24, in match_player(row, cleaned_df) 20 return exact_match.iloc[0] 22 # Prüfung mit Vorname in player_name 23 contains_first_name = candidates[---> 24 candidates['player_name'].str.contains(fr"\b{row['first_name']}\b", case=False, na=False) 25] 26 if not contains_first_name.empty: 27 return contains_first_name.iloc[0] File /opt/anaconda3/lib/python3.11/site-packages/pandas/core/series.py:1040, in Series.__getitem__(self, key) 1037 return self._values[key] 1039 elif key_is_scalar: -> 1040 return self._get_value(key) 1042 # Convert generator to list before going through hashable part 1043 # (We will iterate through the generator there to check for slices) 1044 if is_iterator(key): File /opt/anaconda3/lib/python3.11/site-packages/pandas/core/series.py:1156, in Series._get_value(self, label, takeable) 1153 return self._values[label] 1155 # Similar to Index.get_value, but we do not fall back to positional -> 1156 loc = self.index.get_loc(label) 1158 if is_integer(loc): 1159 return self._values[loc] File /opt/anaconda3/lib/python3.11/site-packages/pandas/core/indexes/base.py:3798, in Index.get_loc(self, key) 3793 if isinstance(casted_key, slice) or (3794 isinstance(casted_key, abc.Iterable) 3795 and any(isinstance(x, slice) for x in casted_key) 3796): 3797 raise InvalidIndexError(key) -> 3798 raise KeyError(key) from err 3799 except TypeError: 3800 # If we have a listlike key, _check_indexing_error will raise 3801 # InvalidIndexError. Otherwise we fall through and re-raise 3802 # the TypeError. 3803 self._check_indexing_error(key) KeyError: 'first_name'

ChatGPT:

ChatGPT

Der Fehler tritt auf, weil in den Daten aus der CSV-Datei `key_stats.csv` möglicherweise die Spalte `first_name` nicht vorhanden ist. Es könnte sein, dass die CSV-Datei diese Spalte nicht enthält, oder sie hat einen anderen Namen.

Hier ist, was wir tun können:

1. **Überprüfen der Spaltennamen in `key_stats.csv`:**
 - Es ist wichtig sicherzustellen, dass die Spalte `first_name` in der Datei `cleaned_df` korrekt vorhanden ist und dass die Spalten in der Datei `key_stats_df` ebenfalls die erwarteten Namen haben.
2. **Feldnamen anpassen:**
 - Falls in `key_stats.csv` keine `first_name`-Spalte vorhanden ist, sollten wir das entsprechend anpassen.

Vorgehensweise

1. **Spaltennamen überprüfen:** Führe diesen Code aus, um die Spaltennamen in beiden CSV-Dateien zu überprüfen:

```
python
KopierenBearbeiten
# Überprüfe die Spaltennamen in beiden DataFrames
print("Spalten in cleaned_df:")
print(cleaned_df.columns)

print("Spalten in key_stats_df:")
print(key_stats_df.columns)
```

2. **Wenn der `first_name` in `cleaned_df` korrekt ist, aber nicht in `key_stats_df` vorhanden ist,** sollten wir den Code anpassen, um diese Spalte zu verwenden, wenn sie im `cleaned_df` vorhanden ist, aber sicherstellen, dass wir nur mit den Spalten aus `key_stats_df` arbeiten.

Falls der `first_name` nur in `cleaned_df` vorhanden ist und nicht in `key_stats_df`, kann der Fehler durch Hinzufügen einer Überprüfung verhindert werden, ob der `first_name`-Wert in `cleaned_df` auch vorhanden ist.

3. **Code anpassen:** Hier eine angepasste Version, die sicherstellt, dass die `first_name`-Spalte aus `cleaned_df` verwendet wird und auch mögliche Fehler abgefangen werden, falls diese nicht existiert.

Angepasster Code

```
python
KopierenBearbeiten
import pandas as pd

# CSV-Dateien laden
cleaned_data_file = "clean01_ucl_top4_teams_market_values.csv"
key_stats_file = "key_stats.csv"

cleaned_df = pd.read_csv(cleaned_data_file)
key_stats_df = pd.read_csv(key_stats_file)
```

```

# Überprüfe, ob 'first_name' in cleaned_df existiert
if 'first_name' not in cleaned_df.columns:
    raise KeyError("Die Spalte 'first_name' fehlt in cleaned_df!")

# Funktion, um Player Name und Club abzugleichen
def match_player(row, cleaned_df):
    # Filter nach Club
    candidates = cleaned_df[cleaned_df['club'] == row['club']]

    # Genau übereinstimmende Namen finden
    exact_match = candidates[candidates['player_name'] ==
row['player_name']]
    if not exact_match.empty:
        return exact_match.iloc[0]

    # Vorname in player_name
    contains_first_name = candidates[
        candidates['player_name'].str.contains(fr"\b{row['first_name']}\b",
case=False, na=False)
    ]
    if not contains_first_name.empty:
        return contains_first_name.iloc[0]

    # Initial des Vornamens
    initial = row['first_name'][0].upper() if pd.notna(row['first_name'])
else ""
    initial_match = candidates[
        candidates['player_name'].str.startswith(f"{initial}.", na=False)
    ]
    if not initial_match.empty:
        return initial_match.iloc[0]

    return None

# Daten zusammenführen
matched_rows = []
for _, row in key_stats_df.iterrows():
    matched = match_player(row, cleaned_df)
    if matched is not None:
        combined_row = row.to_dict()
        combined_row.update(matched.to_dict()) # Werte aus beiden
DataFrames zusammenführen
        matched_rows.append(combined_row)

# Neues DataFrame erstellen
combined_df = pd.DataFrame(matched_rows)

# Ergebnis speichern
output_file = "merged_data.csv"
combined_df.to_csv(output_file, index=False)

print(f"Die verknüpften Daten wurden in '{output_file}' gespeichert.")

```

Erklärung der Änderungen:

1. Prüfung auf `first_name`:

- Es wird überprüft, ob die `first_name`-Spalte in `cleaned_df` vorhanden ist. Wenn nicht, wird ein `KeyError` ausgelöst, um den Fehler deutlich zu machen.

2. Anpassungen in der Funktionslogik:

- In der Funktion `match_player` wird weiterhin der `first_name` aus `cleaned_df` verwendet, um die Zuordnung vorzunehmen.
3. **Fehlerbehandlung:**
- Falls `first_name` fehlt, wird eine Fehlermeldung ausgegeben, anstatt einen Fehler zu werfen, wenn der Code später läuft.

Nächste Schritte:

1. **Spaltennamen prüfen:** Überprüfe, ob der Fehler weiterhin auftritt, indem du die Spaltennamen der DataFrames aus gibst.
2. **CSV-Datenformat überprüfen:** Stelle sicher, dass die CSV-Dateien korrekt geladen werden und die relevanten Spalten vorhanden sind.

40 mini

Du:

ja die key stats datei enthält die spalte nicht.

ChatGPT:

ChatGPT

Da die `key_stats.csv`-Datei die Spalte `first_name` nicht enthält, müssen wir sicherstellen, dass wir beim Abgleich der Spieler die `first_name`-Spalte nur aus den gescrapten (bereinigten) Daten verwenden. Wir passen den Code an, sodass wir die `first_name`-Spalte nur zur Verknüpfung der Daten aus den bereinigten Daten verwenden, aber nicht aus `key_stats.csv`.

Die Hauptidee bleibt, die Spieler anhand des `player_name` und des `club` zu matchen, und für den Fall, dass der `player_name` nicht exakt übereinstimmt, verwenden wir den Vornamen oder die Initialen des Vornamens. Das bedeutet, wir prüfen auch die Fälle, bei denen der Vorname mit einem Initial (z.B. "N." für "Nico") übereinstimmt.

Angepasster Code:

Hier ist der angepasste Code, der sicherstellt, dass die `first_name`-Spalte nur aus den `cleaned_df`-Daten verwendet wird und die `key_stats.csv`-Datei nicht beeinträchtigt wird.

```
python
KopierenBearbeiten
import pandas as pd

# CSV-Dateien laden
cleaned_data_file = "clean01_ucl_top4_teams_market_values.csv"
key_stats_file = "key_stats.csv"

cleaned_df = pd.read_csv(cleaned_data_file)
key_stats_df = pd.read_csv(key_stats_file)

# Funktion, um Player Name und Club abzugleichen
def match_player(row, cleaned_df):
    # Filter nach Club
```

```

candidates = cleaned_df[cleaned_df['club'] == row['club']]

# Genau übereinstimmende Namen finden
exact_match = candidates[candidates['player_name'] ==
row['player_name']]
if not exact_match.empty:
    return exact_match.iloc[0]

# Vorname in player_name (genauer Name im player_name)
contains_first_name = candidates[
    candidates['player_name'].str.contains(fr"\b{row['first_name']}\b",
case=False, na=False)
]
if not contains_first_name.empty:
    return contains_first_name.iloc[0]

# Initial des Vornamens (Prüfung, ob der Name mit einem Initial
übereinstimmt)
initial = row['first_name'][0].upper() if pd.notna(row['first_name'])
else ""
initial_match = candidates[
    candidates['player_name'].str.startswith(f"{initial}.", na=False)
]
if not initial_match.empty:
    return initial_match.iloc[0]

return None

# Daten zusammenführen
matched_rows = []
for _, row in key_stats_df.iterrows():
    matched = match_player(row, cleaned_df)
    if matched is not None:
        combined_row = row.to_dict()
        combined_row.update(matched.to_dict()) # Werte aus beiden
DataFrames zusammenführen
        matched_rows.append(combined_row)

# Neues DataFrame erstellen
combined_df = pd.DataFrame(matched_rows)

# Ergebnis speichern
output_file = "merged_data.csv"
combined_df.to_csv(output_file, index=False)

print(f"Die verknüpften Daten wurden in '{output_file}' gespeichert.")

```

Änderungen:

- **Verwendung der first_name-Spalte aus cleaned_df:**
Da key_stats_df keine first_name-Spalte hat, verwenden wir nur die first_name-Spalte aus cleaned_df (den gescrapten Daten).
- **Keine Änderung an key_stats_df:** Der Code verknüpft die Daten ausschließlich basierend auf den player_name und club-Spalten. Die first_name-Spalte wird nur zur Bestimmung des Vornamens oder Initials für den Abgleich verwendet.

Ablauf:

1. **match_player-Funktion:**

- Der Spieler wird anhand des `player_name` und `club` aus beiden DataFrames gesucht.
 - Wenn der exakte Name gefunden wird, wird der Spieler abgeglichen.
 - Wenn der Vorname in der `player_name`-Spalte vorhanden ist, wird der Spieler ebenfalls abgeglichen.
 - Wenn die Initiale des Vornamens (z.B. "N." für "Nico") im `player_name` zu finden ist, wird auch dieser Spieler abgeglichen.
2. **Daten zusammenführen:**
- Für jeden Spieler aus der `key_stats_df`-Datei wird der passende Spieler aus `cleaned_df` gesucht und die entsprechenden Daten werden zusammengeführt.
3. **Ausgabe:**
- Die verknüpften Daten werden in eine neue Datei `merged_data.csv` geschrieben.

Überprüfen und Ausführen:

- Stelle sicher, dass die CSV-Dateien im richtigen Format vorliegen und dass die Spalten `player_name`, `club`, und `first_name` in `cleaned_df` korrekt sind.
- Wenn der Code immer noch nicht funktioniert, überprüfe, ob es eventuell Tippfehler oder abweichende Spaltenbezeichner in den CSV-Dateien gibt.

40 mini

Du:

```
----- KeyError Traceback (most
recent call last) File /opt/anaconda3/lib/python3.11/site-
packages/pandas/core/indexes/base.py:3791, in Index.get_loc(self, key) 3790 try: -> 3791
return self._engine.get_loc(casted_key) 3792 except KeyError as err: File index.pyx:152, in
pandas._libs.index.IndexEngine.get_loc() File index.pyx:181, in
pandas._libs.index.IndexEngine.get_loc() File pandas/_libs/hashtable_class_helper.pxi:7080,
in pandas._libs.hashtable.PyObjectHashTable.get_item() File
pandas/_libs/hashtable_class_helper.pxi:7088, in
pandas._libs.hashtable.PyObjectHashTable.get_item() KeyError: 'first_name' The above
exception was the direct cause of the following exception: KeyError Traceback (most recent
call last) Cell In[37], line 46 44 matched_rows = [] 45 for _, row in key_stats_df.iterrows(): --
-> 46 matched = match_player(row, cleaned_df) 47 if matched is not None: 48 combined_row
= row.to_dict() Cell In[37], line 28, in match_player(row, cleaned_df) 24 return
exact_match.iloc[0] 26 # Vorname in player_name 27 contains_first_name = candidates[ --->
28 candidates['player_name'].str.contains(fr"\b{row['first_name']}\b", case=False, na=False)
29 ] 30 if not contains_first_name.empty: 31 return contains_first_name.iloc[0] File
/opt/anaconda3/lib/python3.11/site-packages/pandas/core/series.py:1040, in
Series.__getitem__(self, key) 1037 return self._values[key] 1039 elif key_is_scalar: -> 1040
return self._get_value(key) 1042 # Convert generator to list before going through hashable
part 1043 # (We will iterate through the generator there to check for slices) 1044 if
is_iterator(key): File /opt/anaconda3/lib/python3.11/site-packages/pandas/core/series.py:1156,
in Series._get_value(self, label, takeable) 1153 return self._values[label] 1155 # Similar to
Index.get_value, but we do not fall back to positional -> 1156 loc = self.index.get_loc(label)
1158 if is_integer(loc): 1159 return self._values[loc] File /opt/anaconda3/lib/python3.11/site-
packages/pandas/core/indexes/base.py:3798, in Index.get_loc(self, key) 3793 if
```

```
isinstance(casted_key, slice) or ( 3794 isinstance(casted_key, abc.Iterable) 3795 and
any(isinstance(x, slice) for x in casted_key) 3796 ): 3797 raise IndexError(key) ->
3798 raise KeyError(key) from err 3799 except TypeError: 3800 # If we have a listlike key,
_check_indexing_error will raise 3801 # IndexError. Otherwise we fall through and re-
raise 3802 # the TypeError. 3803 self._check_indexing_error(key) KeyError: 'first_name'
```

ChatGPT:

ChatGPT

Der Fehler besagt, dass die Spalte `first_name` in deinem `cleaned_df` DataFrame nicht gefunden wird. Es scheint, als ob entweder der Name der Spalte falsch geschrieben ist oder die Spalte in der `cleaned_df`-Datei nicht existiert.

Lösungsansätze:

1. **Überprüfe die Spaltennamen:** Du kannst sicherstellen, dass die `first_name`-Spalte in `cleaned_df` korrekt vorhanden ist, indem du die Spaltennamen des DataFrames überprüfst:

```
python
KopierenBearbeiten
print(cleaned_df.columns)
```

2. **Korrigiere den Spaltennamen:** Falls der Spaltenname in `cleaned_df` einen anderen Namen hat, wie zum Beispiel `firstName`, `First Name` oder etwas anderes, musst du den Code so anpassen, dass der korrekte Spaltenname verwendet wird.

Beispiel:

Wenn die Spalte in `cleaned_df` einen anderen Namen hat, zum Beispiel `first_name_full` oder `first`, dann musst du den Code entsprechend ändern.

Falls du den Spaltennamen herausgefunden hast, kannst du den Code anpassen, indem du den richtigen Namen verwendest:

```
python
KopierenBearbeiten
# Beispiel: Annahme, dass die richtige Spalte 'first' statt 'first_name'
heißt
contains_first_name = candidates[
    candidates['player_name'].str.contains(fr"\b{row['first']}\b",
case=False, na=False)
]
```

Schritt-für-Schritt-Überprüfung:

1. Lade den DataFrame `cleaned_df`.
2. Führe den `print(cleaned_df.columns)` Befehl aus und überprüfe die Spalten.
3. Ändere den Code, um sicherzustellen, dass der richtige Name für den Vornamen verwendet wird.

Wenn du den Fehler mit den Spaltennamen behoben hast, sollte der Code ohne das `KeyError`-Problem weiterlaufen.

4o mini

Du:

```
----- KeyError Traceback (most recent call last) File /opt/anaconda3/lib/python3.11/site-packages/pandas/core/indexes/base.py:3791, in Index.get_loc(self, key) 3790 try: -> 3791 return self._engine.get_loc(casted_key) 3792 except KeyError as err: File index.py:152, in pandas._libs.index.IndexEngine.get_loc() File index.py:181, in pandas._libs.index.IndexEngine.get_loc() File pandas/_libs/hashtable_class_helper.pxi:7080, in pandas._libs.hashtable.PyObjectHashTable.get_item() File pandas/_libs/hashtable_class_helper.pxi:7088, in pandas._libs.hashtable.PyObjectHashTable.get_item() KeyError: 'first_name' The above exception was the direct cause of the following exception: KeyError Traceback (most recent call last) Cell In[38], line 42 40 matched_rows = [] 41 for _, row in key_stats_df.iterrows(): --> 42 matched = match_player(row, cleaned_df) 43 if matched is not None: 44 combined_row = row.to_dict() Cell In[38], line 24, in match_player(row, cleaned_df) 20 return exact_match.iloc[0] 22 # Vorname in player_name (genauer Name im player_name) 23 contains_first_name = candidates[ ---> 24 candidates['player_name'].str.contains(fr"\b{row['first_name']}\b", case=False, na=False) 25 ] 26 if not contains_first_name.empty: 27 return contains_first_name.iloc[0] File /opt/anaconda3/lib/python3.11/site-packages/pandas/core/series.py:1040, in Series.__getitem__(self, key) 1037 return self._values[key] 1039 elif key_is_scalar: -> 1040 return self._get_value(key) 1042 # Convert generator to list before going through hashable part 1043 # (We will iterate through the generator there to check for slices) 1044 if is_iterator(key): File /opt/anaconda3/lib/python3.11/site-packages/pandas/core/series.py:1156, in Series._get_value(self, label, takeable) 1153 return self._values[label] 1155 # Similar to Index.get_value, but we do not fall back to positional -> 1156 loc = self.index.get_loc(label) 1158 if is_integer(loc): 1159 return self._values[loc] File /opt/anaconda3/lib/python3.11/site-packages/pandas/core/indexes/base.py:3798, in Index.get_loc(self, key) 3793 if isinstance(casted_key, slice) or ( 3794 isinstance(casted_key, abc.Iterable) 3795 and any(isinstance(x, slice) for x in casted_key) 3796 ): 3797 raise InvalidIndexError(key) -> 3798 raise KeyError(key) from err 3799 except TypeError: 3800 # If we have a listlike key, _check_indexing_error will raise 3801 # InvalidIndexError. Otherwise we fall through and re-raise 3802 # the TypeError. 3803 self._check_indexing_error(key) KeyError: 'first_name'
```

ChatGPT:

ChatGPT

Der Fehler `KeyError: 'first_name'` deutet darauf hin, dass die Spalte `'first_name'` in deinem `key_stats_df` DataFrame nicht existiert. Wenn du versuchst, auf die Spalte `first_name` zuzugreifen, aber sie nicht vorhanden ist, könnte es daran liegen, dass der DataFrame `key_stats_df` diese Spalte nicht hat.

Um das Problem zu lösen, führe die folgenden Schritte durch:

1. Überprüfe die Spalten von `key_stats_df`

Lass uns sicherstellen, dass die `first_name`-Spalte tatsächlich im `key_stats_df` DataFrame existiert. Gib einfach die Spaltennamen aus:

```
python
KopierenBearbeiten
print(key_stats_df.columns)
```

2. Anpassung der Logik

Falls die `first_name`-Spalte nicht vorhanden ist, kannst du die `cleaned_df`-Spalte verwenden, die den Vorname enthält (und die auch als Referenz für den Vergleich dient).

3. Beispielanpassung

Wenn `key_stats_df` keine `first_name`-Spalte hat, kannst du den Vorname aus `cleaned_df` verwenden, um den Vergleich anzustellen. Hier ist ein Beispiel:

```
python
KopierenBearbeiten
def match_player(row, cleaned_df):
    # Finde Spieler in cleaned_df, der den gleichen Namen und Verein hat
    candidates = cleaned_df[cleaned_df['club'] == row['club']]

    # Überprüfe, ob der Spielername exakt übereinstimmt
    exact_match = candidates[candidates['player_name'] ==
row['player_name']]
    if not exact_match.empty:
        return exact_match.iloc[0]

    # Wenn der Vorname im player_name enthalten ist
    contains_first_name =
candidates[candidates['player_name'].str.contains(fr"\b{row['first_name']}\b",
case=False, na=False)]
    if not contains_first_name.empty:
        return contains_first_name.iloc[0]

    # Überprüfe, ob der Initial des Vornamens mit einem Punkt im
player_name enthalten ist
    first_initial = row['first_name'][0].upper() + "."
    contains_initial =
candidates[candidates['player_name'].str.contains(fr"\b{first_initial}\b",
case=False, na=False)]
    if not contains_initial.empty:
        return contains_initial.iloc[0]

    return None

# Iteriere über die Zeilen von key_stats_df und versuche einen passenden
Spieler zu finden
matched_rows = []
for _, row in key_stats_df.iterrows():
    matched = match_player(row, cleaned_df)
    if matched is not None:
        combined_row = row.to_dict()
        for col in matched.index:
            combined_row[col] = matched[col]
        matched_rows.append(combined_row)

# Erstelle ein neues DataFrame aus den verknüpften Zeilen
```

```
combined_df = pd.DataFrame(matched_rows)
```

Erklärung:

- **Spaltennamen überprüfen:** Wenn der Fehler weiterhin auftritt, überprüfe, ob die Spalte `first_name` in `key_stats_df` existiert, und stelle sicher, dass sie korrekt benannt ist.
- **Vergleich anpassen:** Wenn es keine `first_name`-Spalte gibt, kannst du die Logik ändern, um entweder den Namen direkt zu vergleichen oder anhand anderer Spalten zu prüfen, welche Spieler zugeordnet werden sollen.

Vergewissere dich also, dass du entweder die richtige Spalte im `key_stats_df` verwendest oder die Logik entsprechend anpasst.

4o mini

Du:

```
Index(['player_name', 'market_value', 'club', 'first_name'], dtype='object')
```

ChatGPT:

ChatGPT

Es scheint, dass der Fehler durch einen Tippfehler oder durch das Fehlen von Werten in der `first_name`-Spalte verursacht wird, obwohl sie existiert. Da du bestätigt hast, dass die `first_name`-Spalte in `key_stats_df` vorhanden ist, gibt es einige Dinge, die wir überprüfen sollten:

1. Sicherstellen, dass die `first_name`-Spalte korrekt geschrieben ist

Da Pandas Groß- und Kleinschreibung bei Spaltennamen berücksichtigt, vergewissere dich, dass du den richtigen Namen verwendest.

2. Überprüfen, ob Werte in der `first_name`-Spalte fehlen

Es könnte sein, dass einige Zeilen in der `first_name`-Spalte leere oder `NaN`-Werte haben. Um das zu überprüfen, kannst du den folgenden Code ausführen:

```
python
KopierenBearbeiten
# Zeige alle Zeilen an, bei denen 'first_name' fehlt
print(key_stats_df[key_stats_df['first_name'].isnull()])
```

Wenn leere Werte vorhanden sind, kannst du die Zeilen entweder aus der Analyse ausschließen oder sicherstellen, dass sie korrekt behandelt werden.

3. Vorname korrekt extrahieren

Falls alles in Ordnung ist, stellen wir sicher, dass der Vorname korrekt extrahiert wird. Wenn es immer noch zu Problemen kommt, versuchen wir, den `first_name`-Wert explizit zu überprüfen und sicherzustellen, dass der Wert vorhanden ist.

Hier ist der angepasste Code, der sowohl `first_name` als auch den gesamten `player_name` berücksichtigt:

Codebeispiel:

```
python
KopierenBearbeiten
def match_player(row, cleaned_df):
    # Prüfe, ob 'first_name' existiert und nicht NaN ist
    if pd.isna(row['first_name']):
        return None

    # Suche nach Kandidaten im cleaned_df mit dem gleichen Club
    candidates = cleaned_df[cleaned_df['club'] == row['club']]

    # Exakte Übereinstimmung des Spielernamens
    exact_match = candidates[candidates['player_name'] ==
row['player_name']]
    if not exact_match.empty:
        return exact_match.iloc[0]

    # Vorname im player_name (genauer Vergleich)
    contains_first_name =
candidates[candidates['player_name'].str.contains(fr"\b{row['first_name']}\
b", case=False, na=False)]
    if not contains_first_name.empty:
        return contains_first_name.iloc[0]

    # Initial des Vornamens im player_name (z.B. "J." für "John")
    first_initial = row['first_name'][0].upper() + "."
    contains_initial =
candidates[candidates['player_name'].str.contains(fr"\b{first_initial}\b",
case=False, na=False)]
    if not contains_initial.empty:
        return contains_initial.iloc[0]

    return None

# Iteriere durch key_stats_df und versuche passende Spieler zu finden
matched_rows = []
for _, row in key_stats_df.iterrows():
    matched = match_player(row, cleaned_df)
    if matched is not None:
        combined_row = row.to_dict()
        for col in matched.index:
            combined_row[col] = matched[col]
        matched_rows.append(combined_row)

# Erstelle ein neues DataFrame mit den kombinierten Zeilen
combined_df = pd.DataFrame(matched_rows)
```

Erklärung:

- **Fehlende `first_name`-Werte:** Wir überprüfen nun explizit, ob der `first_name` fehlt (`pd.isna(row['first_name'])`) und überspringen solche Zeilen.
- **Genauere Übereinstimmung und Initialen:** Wir führen den Vergleich wie zuvor durch, jedoch sicherer, indem wir auch mit Initialen (z.B. "J." für "John") arbeiten.

Wenn der Fehler weiterhin auftritt:

Falls der Fehler weiterhin auftritt, könnte es hilfreich sein, den genauen Wert zu überprüfen, der für `row['first_name']` verwendet wird. Dafür kannst du den Wert in der Schleife ausdrucken:

```
python
KopierenBearbeiten
print(row['first_name']) # Ausgabe des Vornamens, um sicherzustellen, dass
er korrekt ist
```

Wenn du diese Schritte befolgst, solltest du in der Lage sein, das Problem zu diagnostizieren und zu lösen.

4o mini

Du:

bei dem code: `import pandas as pd` # CSV-Dateien laden `cleaned_data_file = "clean01_ucl_top4_teams_market_values.csv"` `key_stats_file = "/Users/gegenbhardt/Documents/Dev/Python/Datenmodellierung/Projektarbeit_DIS08/ucl-202122-folder/key_stats.csv"` `cleaned_df = pd.read_csv(cleaned_data_file)` `key_stats_df = pd.read_csv(key_stats_file)` `print(cleaned_df.columns)` `print(key_stats_df.columns)` kommt das : `Index(['player_name', 'market_value', 'club', 'first_name'], dtype='object')`
`Index(['player_name', 'club', 'position', 'minutes_played', 'match_played', 'goals', 'assists', 'distance_covered'], dtype='object')`

ChatGPT:

ChatGPT

Vielen Dank für die Klarstellung der Spaltennamen! Ich sehe jetzt, dass `cleaned_df` die Spalte `first_name` enthält, aber `key_stats_df` diese Spalte nicht hat. Die Spalten von `key_stats_df` sind:

- `player_name`
- `club`
- `position`
- `minutes_played`
- `match_played`
- `goals`
- `assists`
- `distance_covered`

Da `key_stats_df` keine `first_name`-Spalte hat, müssen wir die Logik so anpassen, dass wir den `first_name` aus `cleaned_df` verwenden, um Übereinstimmungen zwischen den beiden DataFrames zu finden.

Lösung

Da in `key_stats_df` der `player_name` bereits vorhanden ist und wir versuchen, Spieler in `cleaned_df` zu finden, können wir den `first_name` aus `cleaned_df` extrahieren und mit dem `player_name` in `key_stats_df` abgleichen. Dies bedeutet, dass wir den Spielernamen im `cleaned_df` in Vorname und Nachname aufteilen müssen.

Hier ist der angepasste Code, um das zu tun:

```
python
KopierenBearbeiten
import pandas as pd

# CSV-Dateien laden
cleaned_data_file = "clean01_ucl_top4_teams_market_values.csv"
key_stats_file =
"/Users/gingebhardt/Documents/Dev/Python/Datenmodellierung/Projektarbeit_DI
S08/ucl-202122-folder/key_stats.csv"

cleaned_df = pd.read_csv(cleaned_data_file)
key_stats_df = pd.read_csv(key_stats_file)

# Sicherstellen, dass 'first_name' aus cleaned_df extrahiert wird
cleaned_df['first_name'] = cleaned_df['player_name'].apply(lambda x:
x.split()[0] if isinstance(x, str) else '')

# Funktion zur Spielerabgleichung
def match_player(row, cleaned_df):
    # Extrahiere den Spielernamen
    full_name = row['player_name']
    first_name = full_name.split()[0] # Vorname extrahieren

    # Suche nach dem Spieler in cleaned_df
    candidates =
cleaned_df[cleaned_df['first_name'].str.contains(fr"\b{first_name}\b",
case=False, na=False)]

    # Überprüfen, ob es Übereinstimmungen gibt
    if not candidates.empty:
        matched_player = candidates.iloc[0] # Wähle den ersten gefundenen
Spieler
        return matched_player
    return None

# Iteriere durch key_stats_df und versuche passende Spieler zu finden
matched_rows = []
for _, row in key_stats_df.iterrows():
    matched = match_player(row, cleaned_df)
    if matched is not None:
        combined_row = row.to_dict()
        for col in matched.index:
            combined_row[col] = matched[col]
        matched_rows.append(combined_row)

# Erstelle ein neues DataFrame mit den kombinierten Zeilen
combined_df = pd.DataFrame(matched_rows)

# Zeige das kombinierte DataFrame
print(combined_df.head())
```

Was passiert im Code:

1. **Extraktion des Vornamens:** Wir teilen den `player_name` aus `cleaned_df` auf und extrahieren den ersten Teil als `first_name`. Dies wird mit einer `lambda`-Funktion gemacht, die den Namen in Vorname und Nachname unterteilt.

2. **Anpassung der match_player-Funktion:** Anstatt direkt auf `first_name` in `key_stats_df` zuzugreifen (was nicht existiert), suchen wir nach dem `first_name` in `cleaned_df` und vergleichen diesen mit dem `player_name`.
3. **Durchlaufen und Matching:** Wir iterieren durch `key_stats_df`, verwenden `match_player`, um Übereinstimmungen zu finden, und kombinieren die Daten aus beiden DataFrames.
4. **Kombiniertes DataFrame:** Alle kombinierten Zeilen werden in `matched_rows` gespeichert und in ein neues DataFrame `combined_df` übertragen.

Hinweise:

- Diese Lösung geht davon aus, dass der `player_name` im `cleaned_df` aus Vorname und Nachname besteht, wobei der Vorname immer der erste Teil des Namens ist.
- Es wird davon ausgegangen, dass die Übereinstimmungen durch den Vornamen gefunden werden. Wenn der Vorname nicht eindeutig genug ist, könnte es helfen, auch nach dem Nachnamen oder der vollständigen Übereinstimmung des `player_name` zu suchen.

Lass mich wissen, wenn du zusätzliche Anpassungen benötigst!

4o mini

Du:

wie mache ich den data frame zur csv datei

ChatGPT:

ChatGPT

Um einen DataFrame in eine CSV-Datei zu speichern, kannst du die `to_csv`-Methode von Pandas verwenden. Hier ist ein einfaches Beispiel, wie du das machen kannst:

```
python
KopierenBearbeiten
# Speichern des kombinierten DataFrames als CSV-Datei
combined_df.to_csv('combined_data.csv', index=False)
```

Erklärung:

- `combined_df.to_csv('combined_data.csv', index=False):`
 - `'combined_data.csv'`: Der Name der CSV-Datei, in die der DataFrame gespeichert wird.
 - `index=False`: Verhindert, dass der Index (Zeilenbezeichner) des DataFrames in der CSV-Datei gespeichert wird. Wenn du den Index mit speichern möchtest, kannst du `index=True` oder das Argument weglassen.

Du kannst den Pfad zur Datei auch anpassen, um sie an einem spezifischen Speicherort zu speichern, z. B.:

```
python
KopierenBearbeiten
```

```
combined_df.to_csv('/path/to/directory/combined_data.csv', index=False)
```

Damit wird die CSV-Datei an dem angegebenen Ort gespeichert.

Lass mich wissen, wenn du weitere Fragen hast!

4o mini

Du:

okay es hat geklappt. ich will aber das geguckt wird ob players_name und club von clean01_ucl_top4_teams_market_values.csv gleich sind mit players_name und club von key_stats.csv wenn ja soll verknüpft werden. und es soll dann die spalte players_name von players_name übernommen werden. wenn der players_name von clean01_ucl_top4_teams_market_values.csv in der spalte players_name von key_stats.csv zwar gefunden wird aber davor noch etwas steht, soll mit dem first_name von clean01_ucl_top4_teams_market_values.csv geguckt werden ob er vor dem players_name steht oder ob eine Abkürzung des first_name wie N. davor steht. weil mein Problem ist das in vor manchen Namen noch der vor name oder eine Abkürzung des Vornamens steht, ich aber in clean01_ucl_top4_teams_market_values.csv die Vornamen in einer eignen spalte habe. die Namen bei key_stats.csv sind nicht immer einheitlich formatiert. ich will aber trotzdem die passenden Verknüpfungen finden.

ChatGPT:

ChatGPT

Um das gewünschte Verhalten zu erreichen, kannst du den Code so anpassen, dass sowohl der player_name als auch der first_name berücksichtigt werden, um die Spieler korrekt zu verknüpfen. Wenn der Name in der key_stats.csv-Datei mit einem Vornamen oder einer Abkürzung des Vornamens beginnt, soll dieser Name mit dem first_name aus clean01_ucl_top4_teams_market_values.csv verglichen werden.

Hier ist eine detaillierte Schritt-für-Schritt-Anleitung, wie du vorgehen kannst:

Vorgehen:

- Verknüpfung anhand von player_name und club:** Du prüfst zuerst, ob der Spieler in beiden DataFrames (bei clean01_ucl_top4_teams_market_values.csv und key_stats.csv) mit dem gleichen player_name und club existiert.
- Verwendung des first_name:** Falls der player_name aus clean01_ucl_top4_teams_market_values.csv in key_stats.csv gefunden wird, aber noch ein Vorname oder eine Abkürzung davor steht, prüfst du, ob der first_name aus clean01_ucl_top4_teams_market_values.csv vor dem Nachnamen in player_name steht. Du kannst auch überprüfen, ob eine Abkürzung des first_name (z. B. "N." für "Niklas") davor steht.

Beispiel-Code:

```
python  
KopierenBearbeiten
```

```

import pandas as pd
import re

# CSV-Dateien laden
cleaned_data_file = "clean01_ucl_top4_teams_market_values.csv"
key_stats_file =
"/Users/gengebhardt/Documents/Dev/Python/Datenmodellierung/Projektarbeit_DI
S08/ucl-202122-folder/key_stats.csv"

cleaned_df = pd.read_csv(cleaned_data_file)
key_stats_df = pd.read_csv(key_stats_file)

# Funktion, um Namen zu verknüpfen, basierend auf player_name und club
def match_player(row, key_stats_df):
    player_name = row['player_name']
    first_name = row['first_name']
    club = row['club']

    # Zuerst nach exakter Übereinstimmung von player_name und club suchen
    exact_match = key_stats_df[(key_stats_df['player_name'] == player_name)
& (key_stats_df['club'] == club)]
    if not exact_match.empty:
        return exact_match.iloc[0]

    # Wenn keine exakte Übereinstimmung, nach einem Abgleich mit Vorname
(first_name) suchen
    # Prüfen, ob der player_name bei key_stats mit first_name beginnt (oder
einer Abkürzung davon)
    first_name_pattern = r"^(\\s)" + re.escape(first_name) + r"(\\.|\\s)"
    contains_first_name = key_stats_df[
        key_stats_df['player_name'].str.contains(first_name_pattern,
case=False, na=False)
    ]

    # Wenn ein Spieler mit passendem Vorname gefunden wurde
    if not contains_first_name.empty:
        matched_player = contains_first_name[contains_first_name['club'] ==
club]
        if not matched_player.empty:
            return matched_player.iloc[0]

    return None

# Liste für die verknüpften Spieler
matched_rows = []

# Alle Zeilen aus cleaned_df durchgehen und verknüpfen
for _, row in cleaned_df.iterrows():
    matched = match_player(row, key_stats_df)
    if matched is not None:
        # Wenn eine Übereinstimmung gefunden wird, den Spielernamen aus
clean_df übernehmen
        combined_row = row.to_dict()
        # Alle anderen relevanten Daten aus matched Spieler hinzufügen
        combined_row.update(matched.to_dict())
        matched_rows.append(combined_row)

# Die verknüpften Daten in ein neues DataFrame umwandeln
combined_df = pd.DataFrame(matched_rows)

# Speichern der kombinierten Daten als CSV
combined_df.to_csv('combined_data.csv', index=False)

```

Erklärung der Schritte:

1. `match_player` Funktion:

- Zuerst wird nach einer exakten Übereinstimmung von `player_name` und `club` gesucht. Wenn sie gefunden wird, wird die Zeile aus `key_stats_df` zurückgegeben.
- Falls keine exakte Übereinstimmung gefunden wird, wird überprüft, ob der `first_name` (oder eine Abkürzung davon) im `player_name` in der `key_stats_df` enthalten ist. Der reguläre Ausdruck `first_name_pattern` stellt sicher, dass der Vorname entweder ganz oder mit einer Abkürzung (z. B. "N.") vor dem Nachnamen erscheint.
- Es wird auch darauf geachtet, dass der `club` in beiden DataFrames übereinstimmt, falls ein Spielernamen mit Vorname gefunden wird.

2. Iterieren über `cleaned_df`: Die `match_player`-Funktion wird für jede Zeile in `cleaned_df` angewendet. Wenn eine Übereinstimmung gefunden wird, wird die Zeile mit den entsprechenden Daten aus `key_stats_df` kombiniert.

3. Speichern der verknüpften Daten: Nachdem alle Spieler verknüpft wurden, wird der kombinierte DataFrame als CSV-Datei gespeichert.

Anmerkungen:

- Der reguläre Ausdruck (`re.escape(first_name)`) stellt sicher, dass der Vorname richtig behandelt wird, auch wenn Sonderzeichen im Namen enthalten sind.
- Der Code prüft nicht nur die exakte Übereinstimmung der Namen, sondern auch, ob der Vorname mit einer Abkürzung übereinstimmt, wenn er im `player_name` von `key_stats.csv` auftaucht.
- Das Ziel ist es, die beste Übereinstimmung basierend auf den Spalten `player_name` und `club` zu finden und korrekt zu kombinieren.

Probier den Code aus und schau, ob er für deine Daten funktioniert. Lass mich wissen, falls es noch Anpassungen oder Fragen gibt!

4o mini

Du:

und was ist mit der Abkürzung des Vornamens? wie bei Nico Williams wird zu N. Williams

ChatGPT:

ChatGPT

Um auch Abkürzungen des Vornamens, wie beispielsweise „N. Williams“ für „Nico Williams“, korrekt zu behandeln, müssen wir den Code so erweitern, dass er auch Abkürzungen des Vornamens berücksichtigt.

Vorgehen:

1. Wir müssen prüfen, ob der `first_name` als Abkürzung in `player_name` auftaucht. Das bedeutet, dass wir nach einem Muster suchen müssen, bei dem der erste Buchstabe des Vornamens gefolgt von einem Punkt steht, wie in „N. Williams“.

2. Der reguläre Ausdruck (Regex) für diese Prüfung muss daher auch eine Abkürzung des Vornamens berücksichtigen, z.B. N. für Nico.

Anpassung des Codes:

Der reguläre Ausdruck muss so angepasst werden, dass er sowohl den vollständigen `first_name` als auch eine Abkürzung des `first_name` (z.B. „N.“) erkennt.

Hier ist die angepasste Version des Codes:

```
python
KopierenBearbeiten
import pandas as pd
import re

# CSV-Dateien laden
cleaned_data_file = "clean01_ucl_top4_teams_market_values.csv"
key_stats_file =
"/Users/gengebhardt/Documents/Dev/Python/Datenmodellierung/Projektarbeit_DI
S08/ucl-202122-folder/key_stats.csv"

cleaned_df = pd.read_csv(cleaned_data_file)
key_stats_df = pd.read_csv(key_stats_file)

# Funktion, um Namen zu verknüpfen, basierend auf player_name und club
def match_player(row, key_stats_df):
    player_name = row['player_name']
    first_name = row['first_name']
    club = row['club']

    # Zuerst nach exakter Übereinstimmung von player_name und club suchen
    exact_match = key_stats_df[(key_stats_df['player_name'] == player_name)
& (key_stats_df['club'] == club)]
    if not exact_match.empty:
        return exact_match.iloc[0]

    # Wenn keine exakte Übereinstimmung, nach einem Abgleich mit Vorname
(first_name) suchen
    # Prüfen, ob der player_name bei key_stats mit first_name beginnt (oder
einer Abkürzung davon)
    first_name_pattern = r"(^|\s)" + re.escape(first_name) + r"(\.|\s)"
    # Abkürzung des Vornamens, z.B. 'N.' für 'Nico'
    abbreviation_pattern = r"(^|\s)" + re.escape(first_name[0]) + r"\.|\s"

    contains_first_name = key_stats_df[
        key_stats_df['player_name'].str.contains(first_name_pattern,
case=False, na=False)
        | key_stats_df['player_name'].str.contains(abbreviation_pattern,
case=False, na=False)
    ]

    # Wenn ein Spieler mit passendem Vorname oder Abkürzung gefunden wurde
if not contains_first_name.empty:
        matched_player = contains_first_name[contains_first_name['club'] ==
club]
        if not matched_player.empty:
            return matched_player.iloc[0]

    return None
```

```
# Liste für die verknüpften Spieler
matched_rows = []

# Alle Zeilen aus cleaned_df durchgehen und verknüpfen
for _, row in cleaned_df.iterrows():
    matched = match_player(row, key_stats_df)
    if matched is not None:
        # Wenn eine Übereinstimmung gefunden wird, den Spielernamen aus
        clean_df übernehmen
        combined_row = row.to_dict()
        # Alle anderen relevanten Daten aus matched Spieler hinzufügen
        combined_row.update(matched.to_dict())
        matched_rows.append(combined_row)

# Die verknüpften Daten in ein neues DataFrame umwandeln
combined_df = pd.DataFrame(matched_rows)

# Speichern der kombinierten Daten als CSV
combined_df.to_csv('combined_data.csv', index=False)
```

Erklärung der Änderungen:

1. **Abkürzung des Vornamens:** Der reguläre Ausdruck `abbreviation_pattern` sucht nach dem ersten Buchstaben des Vornamens gefolgt von einem Punkt, z. B. "N." für "Nico". Dies ermöglicht es, sowohl den vollständigen Vornamen als auch seine Abkürzung zu erkennen.
 - o Der reguläre Ausdruck `r"^(^|\s)" + re.escape(first_name[0]) + r"\.\s"` sucht nach dem ersten Buchstaben des Vornamens (`first_name[0]`) gefolgt von einem Punkt (und dann einem Leerzeichen oder Wortgrenze).
2. **contains_first_name-Abfrage:** Wir verwenden nun den `str.contains()`-Aufruf sowohl für den vollständigen Vornamen als auch für die Abkürzung des Vornamens. Falls einer der beiden gefunden wird, prüfen wir weiter, ob der `club` übereinstimmt.

Beispiele für gesuchte Muster:

- Wenn der `first_name` "Nico" ist, sucht der reguläre Ausdruck:
 - o Nach "Nico" in `player_name` (z. B. "Nico Williams").
 - o Nach "N." in `player_name` (z. B. "N. Williams").

Resultat:

Jetzt sollte der Code auch dann eine korrekte Zuordnung vornehmen, wenn der `player_name` in der `key_stats.csv` Datei nur mit einer Abkürzung des Vornamens ("N.") und nicht mit dem vollständigen Vornamen ("Nico") vor dem Nachnamen erscheint.

Probiere diesen Code aus und prüfe, ob er für deine Daten korrekt funktioniert! Wenn es noch Anpassungen benötigt, lass es mich wissen.

4o mini

Du:

das ist gut so. ich will jetzt nur noch das bei den kombinierten Daten bei der spalte `player_name` nicht mehr die Abkürzungen der Vornamen wie N. bei Nico Williams stehen

und ich will das bei dem Vergleich also bei der Suche groß und klein Schreibung nicht beachtet wird also das zum Beispiel der Nachname Bruyne und bruyne übereinstimmen

ChatGPT:

ChatGPT

Um sicherzustellen, dass:

1. Bei der Ausgabe der kombinierten Daten die Abkürzungen des Vornamens (wie „N.“) nicht mehr erscheinen und stattdessen der vollständige Name (z. B. „Nico Williams“) übernommen wird.
2. Der Vergleich und die Suche nach Namen ohne Berücksichtigung der Groß- und Kleinschreibung erfolgt (z. B. „Bruyne“ und „bruyne“ als gleiche Namen betrachtet werden).

kann der Code folgendermaßen angepasst werden:

Anpassungen:

1. **Groß- und Kleinschreibung ignorieren:** Beim Vergleichen der Namen wird jetzt `case=False` hinzugefügt, um die Groß- und Kleinschreibung zu ignorieren.
2. **Vornamen abkürzen und auf vollständigen Namen ersetzen:** Nach der Identifikation eines Spielers mit Abkürzung des Vornamens wird der vollständige Name aus der `cleaned_df` Tabelle übernommen, nicht der Abkürzungsname.

Hier ist der angepasste Code:

```
python
KopierenBearbeiten
import pandas as pd
import re

# CSV-Dateien laden
cleaned_data_file = "clean01_ucl_top4_teams_market_values.csv"
key_stats_file =
"/Users/gingebhardt/Documents/Dev/Python/Datenmodellierung/Projektarbeit_DI
S08/ucl-202122-folder/key_stats.csv"

cleaned_df = pd.read_csv(cleaned_data_file)
key_stats_df = pd.read_csv(key_stats_file)

# Funktion, um Namen zu verknüpfen, basierend auf player_name und club
def match_player(row, key_stats_df):
    player_name = row['player_name']
    first_name = row['first_name']
    club = row['club']

    # Zuerst nach exakter Übereinstimmung von player_name und club suchen
    (Groß-/Kleinschreibung ignorieren)
    exact_match = key_stats_df[
        (key_stats_df['player_name'].str.lower() == player_name.lower()) &
        (key_stats_df['club'].str.lower() == club.lower())
    ]
    if not exact_match.empty:
        return exact_match.iloc[0]
```

```

    # Wenn keine exakte Übereinstimmung, nach einem Abgleich mit Vorname
    (first_name) suchen
    # Prüfen, ob der player_name bei key_stats mit first_name beginnt (oder
    einer Abkürzung davon)
    first_name_pattern = r"(^|\s)" + re.escape(first_name) + r"(\.|\s)"
    # Abkürzung des Vornamens, z.B. 'N.' für 'Nico'
    abbreviation_pattern = r"(^|\s)" + re.escape(first_name[0]) + r"\.\s"

    contains_first_name = key_stats_df[
        key_stats_df['player_name'].str.contains(first_name_pattern,
        case=False, na=False)
        | key_stats_df['player_name'].str.contains(abbreviation_pattern,
        case=False, na=False)
    ]

    # Wenn ein Spieler mit passendem Vorname oder Abkürzung gefunden wurde
    if not contains_first_name.empty:
        matched_player =
contains_first_name[contains_first_name['club'].str.lower() ==
club.lower()]
        if not matched_player.empty:
            return matched_player.iloc[0]

    return None

# Liste für die verknüpften Spieler
matched_rows = []

# Alle Zeilen aus cleaned_df durchgehen und verknüpfen
for _, row in cleaned_df.iterrows():
    matched = match_player(row, key_stats_df)
    if matched is not None:
        # Wenn eine Übereinstimmung gefunden wird, den Spielernamen aus
clean_df übernehmen
        combined_row = row.to_dict()
        # Alle anderen relevanten Daten aus matched Spieler hinzufügen
        combined_row.update(matched.to_dict())

        # Übernahme des vollständigen Namens (ohne Abkürzungen) für
player_name
        combined_row['player_name'] = row['player_name']

        matched_rows.append(combined_row)

# Die verknüpften Daten in ein neues DataFrame umwandeln
combined_df = pd.DataFrame(matched_rows)

# Speichern der kombinierten Daten als CSV
combined_df.to_csv('combined_data.csv', index=False)

```

Änderungen im Detail:

1. Groß-/Kleinschreibung ignorieren:

- Beim Vergleichen der Namen und Clubs wird `str.lower()` verwendet, um die Suche unabhängig von der Groß- und Kleinschreibung zu machen. Dies passiert in der Zeile:

```
python
KopierenBearbeiten
```



```
exact_match =
key_stats_df[(key_stats_df['player_name'].str.lower() ==
player_name.lower()) & (key_stats_df['club'].str.lower() ==
club.lower())]
```

und ebenfalls beim Abgleich von club:

```
python
KopierenBearbeiten
matched_player =
contains_first_name[contains_first_name['club'].str.lower() ==
club.lower()]
```

2. **Abkürzung des Vornamens:** Wenn eine Übereinstimmung auf Grundlage des Abkürzungs-Vornamens (z.B. „N.“) gefunden wird, wird beim Kombinieren des Datensatzes der vollständige Name aus `cleaned_df` übernommen:

```
python
KopierenBearbeiten
combined_row['player_name'] = row['player_name']
```

Diese Zeile sorgt dafür, dass bei der Ausgabe der CSV-Datei der vollständige `player_name` aus `cleaned_df` verwendet wird und nicht die abgekürzte Form aus `key_stats_df`.

Ergebnis:

- **Groß-/Kleinschreibung wird ignoriert**, was bedeutet, dass „Bruyne“ und „bruyne“ als gleich betrachtet werden.
- **Der vollständige Name wird übernommen**, auch wenn in `key_stats_df` eine Abkürzung des Vornamens (z. B. „N. Williams“ statt „Nico Williams“) steht.

Probier diesen Code aus und schau, ob er deinen Anforderungen entspricht! Wenn du noch Anpassungen benötigst, lass es mich wissen.

4o mini

Du:

* Bei der Ausgabe der kombinierten Daten die Abkürzungen des Vornamens (wie „N.“) nicht mehr erscheinen und stattdessen der vollständige Name (z. B. „Nico Williams“) übernommen wird.

nein es soll dann zum beispiel das N. entfernt werden

ChatGPT:

ChatGPT

Verstanden! Du möchtest also, dass bei der Ausgabe der kombinierten Daten die Abkürzungen des Vornamens (wie „N.“) entfernt werden, und stattdessen der vollständige Vorname (z. B. „Nico“ statt „N.“) verwendet wird.

Dazu können wir eine zusätzliche Logik einbauen, die überprüft, ob der `player_name` in `key_stats_df` eine Abkürzung enthält und diese durch den vollständigen Namen ersetzt. Wenn ein Name mit einer Abkürzung des Vornamens übereinstimmt, wird der vollständige Vorname aus `cleaned_df` verwendet.

Hier ist der angepasste Code, der dies berücksichtigt:

Anpassungen:

- Nach der Erkennung eines Namens mit einer Abkürzung des Vornamens (z. B. "N. Williams"), wird der vollständige Vorname aus `cleaned_df` übernommen.
- Der Vorname wird durch das Ersetzen der Abkürzung entfernt, falls vorhanden.

Angepasster Code:

```
python
KopierenBearbeiten
import pandas as pd
import re

# CSV-Dateien laden
cleaned_data_file = "clean01_ucl_top4_teams_market_values.csv"
key_stats_file =
"/Users/gingebhardt/Documents/Dev/Python/Datenmodellierung/Projektarbeit_DI
S08/ucl-202122-folder/key_stats.csv"

cleaned_df = pd.read_csv(cleaned_data_file)
key_stats_df = pd.read_csv(key_stats_file)

# Funktion, um Namen zu verknüpfen, basierend auf player_name und club
def match_player(row, key_stats_df):
    player_name = row['player_name']
    first_name = row['first_name']
    club = row['club']

    # Zuerst nach exakter Übereinstimmung von player_name und club suchen
    (Groß-/Kleinschreibung ignorieren)
    exact_match = key_stats_df[
        (key_stats_df['player_name'].str.lower() == player_name.lower()) &
        (key_stats_df['club'].str.lower() == club.lower())
    ]
    if not exact_match.empty:
        return exact_match.iloc[0]

    # Wenn keine exakte Übereinstimmung, nach einem Abgleich mit Vorname
    (first_name) suchen
    # Prüfen, ob der player_name bei key_stats mit first_name beginnt (oder
    einer Abkürzung davon)
    first_name_pattern = r"^(|\s)" + re.escape(first_name) + r"(\.|\s)"
    # Abkürzung des Vornamens, z.B. 'N.' für 'Nico'
    abbreviation_pattern = r"^(|\s)" + re.escape(first_name[0]) + r"\.\s"

    contains_first_name = key_stats_df[
        key_stats_df['player_name'].str.contains(first_name_pattern,
        case=False, na=False)
        | key_stats_df['player_name'].str.contains(abbreviation_pattern,
        case=False, na=False)
    ]
```

```

        # Wenn ein Spieler mit passendem Vorname oder Abkürzung gefunden wurde
        if not contains_first_name.empty:
            matched_player =
contains_first_name[contains_first_name['club'].str.lower() ==
club.lower()]
            if not matched_player.empty:
                return matched_player.iloc[0]

    return None

# Liste für die verknüpften Spieler
matched_rows = []

# Alle Zeilen aus cleaned_df durchgehen und verknüpfen
for _, row in cleaned_df.iterrows():
    matched = match_player(row, key_stats_df)
    if matched is not None:
        # Wenn eine Übereinstimmung gefunden wird, den Spielernamen aus
clean_df übernehmen
        combined_row = row.to_dict()
        # Alle anderen relevanten Daten aus matched Spieler hinzufügen
        combined_row.update(matched.to_dict())

        # Übernahme des vollständigen Namens (ohne Abkürzungen) für
player_name
        # Entfernen der Abkürzung des Vornamens, falls vorhanden
        combined_player_name = combined_row['player_name']
        # Falls die Abkürzung des Vornamens wie 'N.' vorhanden ist,
entfernen wir sie
        combined_player_name = re.sub(r"^(\\s) [A-Z]\\.\s", r"\\1",
combined_player_name)

        # Den bereinigten player_name in die finale Zeile setzen
        combined_row['player_name'] = combined_player_name

    matched_rows.append(combined_row)

# Die verknüpften Daten in ein neues DataFrame umwandeln
combined_df = pd.DataFrame(matched_rows)

# Speichern der kombinierten Daten als CSV
combined_df.to_csv('combined_data.csv', index=False)

```

Erklärung der Änderungen:

1. Entfernen der Abkürzung des Vornamens:

- Es wird eine Regex verwendet, um Namen wie "N. Williams" in "Williams" zu ändern.
- Der reguläre Ausdruck `r"^(\\s) [A-Z]\\.\s"` erkennt einen Großbuchstaben (der den Vornamen abkürzt) gefolgt von einem Punkt und einem Leerzeichen. Dieser Teil wird entfernt:

```

python
KopierenBearbeiten
combined_player_name = re.sub(r"^(\\s) [A-Z]\\.\s", r"\\1",
combined_player_name)

```

Hierbei wird der Vorname (wie „N.“) entfernt und nur der Nachname (z. B. „Williams“) bleibt übrig.

2. Groß-/Kleinschreibung ignorieren:

- Der Vergleich zwischen `player_name` und den Namen in `key_stats_df` wird weiterhin ohne Berücksichtigung der Groß-/Kleinschreibung durchgeführt, durch `str.lower()`:

```
python
KopierenBearbeiten
exact_match =
key_stats_df[(key_stats_df['player_name'].str.lower() ==
player_name.lower()) & (key_stats_df['club'].str.lower() ==
club.lower())]
```

Ergebnis:

- In der Ausgabe der kombinierten Daten wird der vollständige Vorname verwendet, und es wird kein „Abkürzungs-Vorname“ wie „N.“ mehr angezeigt.
- Der Vergleich berücksichtigt weiterhin keine Groß-/Kleinschreibung.

Versuche den Code und prüfe, ob er deinen Anforderungen entspricht! Falls du noch Anpassungen benötigst, stehe ich gerne zur Verfügung.

4o mini

Du:

es sollen dann die `players_name` werte von `clean01_ucl_top4_teams_market_values.csv` Bei der ausgabe der kombinierten werden genommen werden. also muss der schritt wo die Vornamen Abkürzungen entfernt werden wie N. bei Nico Williams gar nicht gemacht werden.

ChatGPT:

ChatGPT

Verstanden! Wenn du bei der Ausgabe der kombinierten Daten die `player_name` Werte aus `cleaned_df` (also aus der Datei `clean01_ucl_top4_teams_market_values.csv`) übernehmen möchtest und keine Änderungen an den Vornamen wie "N." vorgenommen werden sollen, dann müssen wir den Schritt, bei dem Abkürzungen des Vornamens entfernt werden, weglassen.

Änderungen:

- Wir werden in der `combined_row` direkt den `player_name` aus `cleaned_df` übernehmen, ohne die Vornamenabkürzung zu entfernen.
- Der Schritt, bei dem die Abkürzungen wie "N." entfernt werden, entfällt komplett.

Hier ist der angepasste Code:

Angepasster Code:

```

python
KopierenBearbeiten
import pandas as pd
import re

# CSV-Dateien laden
cleaned_data_file = "clean01_ucl_top4_teams_market_values.csv"
key_stats_file =
"/Users/gengebhardt/Documents/Dev/Python/Datenmodellierung/Projektarbeit_DI
S08/ucl-202122-folder/key_stats.csv"

cleaned_df = pd.read_csv(cleaned_data_file)
key_stats_df = pd.read_csv(key_stats_file)

# Funktion, um Namen zu verknüpfen, basierend auf player_name und club
def match_player(row, key_stats_df):
    player_name = row['player_name']
    first_name = row['first_name']
    club = row['club']

    # Zuerst nach exakter Übereinstimmung von player_name und club suchen
    (Groß-/Kleinschreibung ignorieren)
    exact_match = key_stats_df[
        (key_stats_df['player_name'].str.lower() == player_name.lower()) &
        (key_stats_df['club'].str.lower() == club.lower())
    ]
    if not exact_match.empty:
        return exact_match.iloc[0]

    # Wenn keine exakte Übereinstimmung, nach einem Abgleich mit Vorname
    (first_name) suchen
    # Prüfen, ob der player_name bei key_stats mit first_name beginnt (oder
    einer Abkürzung davon)
    first_name_pattern = r"(^|\s)" + re.escape(first_name) + r"(\.|\s)"
    # Abkürzung des Vornamens, z.B. 'N.' für 'Nico'
    abbreviation_pattern = r"(^|\s)" + re.escape(first_name[0]) + r"\.\s"

    contains_first_name = key_stats_df[
        key_stats_df['player_name'].str.contains(first_name_pattern,
        case=False, na=False)
        | key_stats_df['player_name'].str.contains(abbreviation_pattern,
        case=False, na=False)
    ]

    # Wenn ein Spieler mit passendem Vorname oder Abkürzung gefunden wurde
    if not contains_first_name.empty:
        matched_player =
contains_first_name[contains_first_name['club'].str.lower() ==
club.lower()]
        if not matched_player.empty:
            return matched_player.iloc[0]

    return None

# Liste für die verknüpften Spieler
matched_rows = []

# Alle Zeilen aus cleaned_df durchgehen und verknüpfen
for _, row in cleaned_df.iterrows():
    matched = match_player(row, key_stats_df)
    if matched is not None:
        # Wenn eine Übereinstimmung gefunden wird, den Spielernamen aus
        clean_df übernehmen

```

```

combined_row = row.to_dict()
# Alle anderen relevanten Daten aus matched Spieler hinzufügen
combined_row.update(matched.to_dict())

# Hier wird der player_name aus cleaned_df übernommen, ohne
Änderungen oder Entfernen von Abkürzungen
combined_row['player_name'] = row['player_name']

matched_rows.append(combined_row)

# Die verknüpften Daten in ein neues DataFrame umwandeln
combined_df = pd.DataFrame(matched_rows)

# Speichern der kombinierten Daten als CSV
combined_df.to_csv('combined_data.csv', index=False)

```

Was wurde geändert:

- **Übernahme des player_name aus cleaned_df ohne Änderung:** Statt den player_name zu bereinigen (d.h. Abkürzungen wie "N." zu entfernen), wird einfach der player_name aus cleaned_df übernommen:

```

python
KopierenBearbeiten
combined_row['player_name'] = row['player_name']

```

Das bedeutet, dass der Name in der Ausgabe der kombinierten Daten exakt so bleibt, wie er in cleaned_df erscheint, einschließlich der Abkürzungen wie „N. Williams“ für „Nico Williams“.

Ergebnis:

- Der player_name aus cleaned_df wird ohne Änderungen übernommen und erscheint in der kombinierten CSV-Datei.
- Der Name wird nicht verändert, und Abkürzungen wie "N." werden nicht entfernt.

Probier den Code aus und schau, ob es deinen Anforderungen entspricht! Wenn du noch weitere Anpassungen benötigst, stehe ich gerne zur Verfügung.

4o mini