

wie könnte ich die kaggle Daten noch bereinigen so das ich so das ich sie gut mit den  
gescraptten Daten vergleichen kann. ich will die Daten einfach nur so bereinigen das sie keine  
Fehler oder so beinhalten: die kaggle Daten sehen immer wie in diesem beispiel aus und ich  
Habe 8 csv datein von kaggle:

serial,player\_name,club,position,assists,corner\_taken,offsides,dribbles,match\_played  
01,Bruno Fernandes,Man. United,Midfielder,7,10,2,7,7 02,Vinícius Júnior,Real  
Madrid,Forward,6,3,4,83,13 02,Sané,Bayern,Midfielder,6,3,3,32,10  
04,Anthony,Ajax,Forward,5,3,4,28,7 05,Alexander-Arnold,Liverpool,Defender,4,36,0,9,9  
05,De Bruyne,Man. City,Midfielder,4,18,0,14,10 05,Modrić,Real  
Madrid,Midfielder,4,10,0,8,13 05,João Mário,Benfica,Midfielder,4,8,0,7,8  
05,Mbappé,Paris,Forward,4,4,8,43,8 05,Gerard Moreno,Villarreal,Forward,4,0,3,9,7  
05,Capoue,Villarreal,Midfielder,4,0,0,17,12 12,Parejo,Villarreal,Midfielder,3,36,0,4,12  
12,Grimaldo,Benfica,Defender,3,11,1,10,10 12,Müller,Bayern,Forward,3,10,8,2,10  
12,Arnold,Wolfsburg,Midfielder,3,10,0,1,6 12,Angeliño,Leipzig,Defender,3,8,0,4,5  
12,Coman,Bayern,Forward,3,4,4,59,9 12,Bernardo Silva,Man. City,Midfielder,3,2,3,18,11  
12,Bellingham,Dortmund,Midfielder,3,1,1,24,6 12,Zapata,Atalanta,Forward,3,0,7,10,6  
12,Lewandowski,Bayern,Forward,3,0,7,1,10 12,João Cancelo,Man. City,Defender,3,0,2,26,9  
12,Cristiano,Sheriff,Defender,3,0,1,7,6 24,Mahrez,Man. City,Midfielder,2,30,5,28,12  
24,Robertson,Liverpool,Defender,2,27,1,9,10 24,Ziyech,Chelsea,Midfielder,2,23,0,12,9  
24,Gündoğan,Man. City,Midfielder,2,16,1,3,10 24,Tsimikas,Liverpool,Defender,2,16,0,4,5  
24,Foden,Man. City,Midfielder,2,13,6,12,11 24,Pjanic,Beşiktaş,Midfielder,2,9,0,4,3  
24,Forsberg,Leipzig,Forward,2,9,0,3,6 24,Bernardeschi,Juventus,Midfielder,2,7,1,7,5  
24,Dahoud,Dortmund,Midfielder,2,7,1,3,4 24,Pedro Gonçalves,Sporting  
CP,Midfielder,2,7,0,10,5 24,Mount,Chelsea,Midfielder,2,7,0,6,7 24,Douglas  
Santos,Zenit,Defender,2,6,1,3,5 24,Rakitskyy,Zenit,Defender,2,6,0,4,6  
24,Tadić,Ajax,Forward,2,5,5,13,7 24,Asensio,Real Madrid,Forward,2,5,0,1,8  
24,Aaronson,Salzburg,Midfielder,2,4,5,9,8 24,Danjuma,Villarreal,Midfielder,2,3,8,31,11  
24,Shaw,Man. United,Defender,2,3,1,1,5 24,Gnabry,Bayern,Forward,2,3,0,12,8  
24,Griezmann,Atlético,Forward,2,2,1,8,9 24,Renan Lodi,Atlético,Defender,2,1,5,9,10  
24,Rodrygo,Real Madrid,Forward,2,1,1,20,11 24,Hudson-Odoi,Chelsea,Midfielder,2,1,1,15,5  
24,Jorginho,Chelsea,Midfielder,2,1,0,1,8 24,André Silva,Leipzig,Forward,2,0,9,5,6  
24,Sterling,Man. City,Forward,2,0,6,14,12 24,Salah,Liverpool,Forward,2,0,5,49,13  
24,Ikoné,LOSC,Midfielder,2,0,1,15,5 24,Werner,Chelsea,Forward,2,0,1,10,5  
24,Jones,Liverpool,Midfielder,2,0,1,0,4 24,Matheus Nunes,Sporting CP,Midfielder,2,0,0,21,6  
24,Davies,Bayern,Midfielder,2,0,0,18,7 24,Mazraoui,Ajax,Defender,2,0,0,13,8  
24,Mendy,Real Madrid,Defender,2,0,0,8,10 24,Keita,Liverpool,Midfielder,2,0,0,6,10  
24,Henderson,Liverpool,Midfielder,2,0,0,2,12 24,Vidal,Inter,Midfielder,2,0,0,2,7 24,Ricardo  
Esgaio,Sporting CP,Defender,2,0,0,1,8 24,Fernandinho,Man. City,Midfielder,2,0,0,1,8  
64,Seiwald,Salzburg,Midfielder,1,18,1,5,8 64,Berghuis,Ajax,Forward,1,15,2,3,8  
64,Gravenberch,Ajax,Midfielder,1,15,1,18,8 64,Lang,Club Brugge,Forward,1,12,5,17,6  
64,Aebischer,Young Boys,Midfielder,1,11,0,6,6 64,Tsygankov,Dynamo  
Kyiv,Midfielder,1,10,0,8,6 64,Oxlade-Chamberlain,Liverpool,Midfielder,1,9,0,6,6 64,S.  
Thill,Sheriff,Midfielder,1,9,0,1,6 64,Di María,Paris,Forward,1,8,1,3,5  
64,Koopmeiners,Atalanta,Midfielder,1,8,0,1,5 64,Neymar,Paris,Forward,1,7,1,27,6  
64,Lemar,Atlético,Midfielder,1,7,0,7,8 64,Rakitić,Sevilla,Midfielder,1,7,0,0,5 64,De  
Ketelaere,Club Brugge,Forward,1,6,10,6,6 64,Malcom,Zenit,Forward,1,6,2,12,6  
64,Vormer,Club Brugge,Midfielder,1,6,1,0,4 64,Rieder,Young Boys,Midfielder,1,6,0,2,6  
64,Reus,Dortmund,Midfielder,1,5,7,5,6 64,Alan Patrick,Shakhtar  
Donetsk,Midfielder,1,4,0,5,4 64,Koke,Atlético,Midfielder,1,4,0,2,9  
64,Milner,Liverpool,Midfielder,1,4,0,1,8 64,Grealish,Man. City,Midfielder,1,3,6,18,7 64,T.  
Hernández,Milan,Defender,1,3,4,10,5 64,Benzema,Real Madrid,Forward,1,2,9,18,12

64,Ilić,Atalanta,Midfielder,1,2,2,13,4 64,Dybala,Juventus,Forward,1,2,0,8,5  
 64, Vanaken,Club Brugge,Midfielder,1,2,0,7,6 64,Bruno Tabata,Sporting  
 CP,Midfielder,1,2,0,2,4 64,João Félix,Atlético,Forward,1,1,3,21,8  
 64,Taremi,Porto,Forward,1,1,2,4,6 64,Ocampos,Sevilla,Midfielder,1,1,1,16,6 64,Moumi  
 Ngamaleu, Young Boys,Midfielder,1,1,0,34,6 64,Zinchenko,Man. City,Defender,1,1,0,3,8  
 64,Aurier,Villarreal,Defender,1,1,0,0,5 64,Mané,Liverpool,Midfielder,1,0,9,17,13  
 64,Paulinho,Sporting CP,Forward,1,0,5,8,8 64,Diogo Jota,Liverpool,Forward,1,0,4,19,11  
 64,Jesus,Man. City,Forward,1,0,4,7,8 64,Rafa Silva,Benfica,Forward,1,0,3,27,9 64,Rafael  
 Leão,Milan,Forward,1,0,3,16,4 64,Valverde,Real Madrid,Midfielder,1,0,3,10,11  
 64,Haller,Ajax,Forward,1,0,3,9,8 64,Greenwood,Man. United,Forward,1,0,3,6,4 64,Alexis  
 Sánchez,Inter,Forward,1,0,3,2,6 64,Luis Díaz,Liverpool,Forward,1,0,2,42,13  
 64,Perišić,Inter,Midfielder,1,0,2,13,8 64,Kristensen,Salzburg,Defender,1,0,2,12,8  
 64,Džeko,Inter,Forward,1,0,2,4,7 64,Dzyuba,Zenit,Forward,1,0,2,1,6  
 64,Pavard,Bayern,Defender,1,0,2,0,10 64, Van Dijk,Liverpool,Defender,1,0,2,0,9

ChatGPT:

ChatGPT

Um die Kaggle-Daten zu bereinigen und sie gut mit den gescrapten Daten zu vergleichen, gibt es verschiedene Schritte, die du unternehmen kannst, um sicherzustellen, dass die Daten korrekt und konsistent sind. Dies umfasst die Bereinigung von Spalten, das Entfernen von Sonderzeichen und die Standardisierung von Namensformaten.

Hier sind die Schritte, die du befolgen kannst:

## 1. Bereinigung der Namen:

Es ist wichtig sicherzustellen, dass die Namen in einem konsistenten Format vorliegen, da Unterschiede wie Akzentzeichen (z. B. „Vinícius“ vs. „Vinicius“) die Übereinstimmung stören können.

- Entfernen von Akzenten (Diakritika).
- Umwandeln der Namen in Kleinbuchstaben für den Vergleich.

## 2. Bereinigung der Sonderzeichen:

Sonderzeichen oder zusätzliche Leerzeichen in den Daten können ebenfalls Probleme bei der späteren Analyse oder beim Vergleich der Datensätze verursachen. Daher solltest du alle Sonderzeichen und Leerzeichen bereinigen.

## 3. Überprüfen auf fehlende Werte:

Überprüfe, ob in den Datensätzen leere oder fehlende Werte vorhanden sind, und entscheide, wie du damit umgehen möchtest (z. B. durch Ersetzen von leeren Werten mit „unbekannt“ oder Entfernen der Zeilen).

## 4. Konvertierung der Spalten in passende Datentypen:

Stelle sicher, dass numerische Spalten als solche erkannt werden und dass keine Texte als Zahlen gespeichert werden (z. B. „7“ als Text anstelle von Zahl).

## 5. Umbenennen der Spalten (falls nötig):

Falls die Namen der Spalten in den Kaggle-Daten und den gescrapten Daten nicht übereinstimmen, kannst du sie umbenennen, um eine spätere Verknüpfung zu vereinfachen.

Hier ist der Python-Code, um diese Schritte umzusetzen:

```
python
KopierenBearbeiten
import pandas as pd
import unicodedata

# Schritt 1: Lade die Kaggle-Daten (ersetze 'path_to_kaggle_data.csv' durch
deinen Dateipfad)
kaggle_df = pd.read_csv('path_to_kaggle_data.csv')

# Schritt 2: Bereinige die Namen (Akzentzeichen entfernen und in
Kleinbuchstaben umwandeln)
def clean_name(name):
    # Entfernt Akzentzeichen und wandelt in Kleinbuchstaben um
    name = unicodedata.normalize('NFD', name).encode('ascii',
'ignore').decode('utf-8')
    name = name.lower().strip() # Entfernt führende und nachfolgende
Leerzeichen
    return name

# Wende die Bereinigung auf die player_name Spalte an
kaggle_df['cleaned_player_name'] =
kaggle_df['player_name'].apply(clean_name)

# Schritt 3: Bereinige andere Spalten, falls notwendig
# Zum Beispiel: Entferne Leerzeichen in den Clubnamen, Positionen, etc.
kaggle_df['club'] = kaggle_df['club'].str.strip()
kaggle_df['position'] = kaggle_df['position'].str.strip()

# Schritt 4: Überprüfe auf fehlende Werte und entscheide, wie du damit
umgehen möchtest
# Fülle leere Werte mit "unbekannt" oder entferne Zeilen mit fehlenden
Werten
kaggle_df.fillna({'club': 'unbekannt', 'position': 'unbekannt', 'assists':
0}, inplace=True)

# Schritt 5: Konvertiere relevante Spalten in den richtigen Datentyp
# Zum Beispiel: Assists sollten als ganze Zahl gespeichert werden
kaggle_df['assists'] = kaggle_df['assists'].astype(int)

# Schritt 6: Überprüfe die Daten
print(kaggle_df.head())

# Optional: Wenn du später mit den gescrapten Daten verknüpfen möchtest,
stelle sicher, dass die Spaltennamen übereinstimmen
# Beispiel: Umbenennen der Spalten, falls nötig
# kaggle_df.rename(columns={'player_name': 'name'}, inplace=True)

# Speichern der bereinigten Daten in einer neuen Datei (optional)
# kaggle_df.to_csv('cleaned_kaggle_data.csv', index=False)
```

**Was wird im Code gemacht:**

1. **Bereinigung der Namen:** Mit der Funktion `clean_name` werden Akzentzeichen entfernt und die Namen werden in Kleinbuchstaben umgewandelt, um eine konsistente Übereinstimmung zu ermöglichen.
2. **Bereinigung von Leerzeichen:** Überflüssige Leerzeichen in den Spalten `club` und `position` werden entfernt.
3. **Fehlende Werte:** In diesem Beispiel wird für leere Werte in den Spalten `club` und `position` der Wert „unbekannt“ zugewiesen. Du kannst auch andere Ansätze wie das Löschen der Zeilen oder die Verwendung von Mittelwerten für numerische Spalten wählen.
4. **Typkonvertierung:** Die Spalte `assists` wird in den Datentyp `int` umgewandelt, um sicherzustellen, dass sie korrekt als Zahl behandelt wird.
5. **Speichern der bereinigten Daten:** Am Ende kannst du die bereinigten Daten optional in einer neuen CSV-Datei speichern.

### Weitere Empfehlungen:

- Wenn du mit den gescrapten Daten arbeitest, stelle sicher, dass auch deren Namen (z. B. im `name`-Feld) bereinigt werden, damit sie mit den Kaggle-Daten übereinstimmen.
- Wenn du viele CSV-Dateien hast, kannst du den Code in einer Schleife ausführen, um alle 8 Dateien zu bereinigen.

Dieser Ansatz stellt sicher, dass die Kaggle-Daten in einem sauberen und konsistenten Format vorliegen und sich problemlos mit den gescrapten Daten vergleichen lassen.

4o mini

Du:

ich will die kaggle Daten nur überprüfen ich will nichts großartig ersetzen

ChatGPT:

ChatGPT

Verstanden! Wenn du die Kaggle-Daten nur auf mögliche Fehler oder Inkonsistenzen überprüfen möchtest, ohne Werte zu ersetzen, kannst du den Fokus auf die Identifikation und Analyse von Problemen wie fehlende Werte, doppelte Einträge und Inkonsistenzen in den Spalten legen. Hier sind einige Schritte, die du zur Überprüfung der Daten durchführen kannst:

### Schritte zur Überprüfung der Kaggle-Daten:

1. **Überprüfen auf fehlende Werte:** Identifizieren von fehlenden Werten (NaN) in den wichtigsten Spalten, wie z.B. `player_name`, `club`, `position` und anderen numerischen Feldern.
2. **Doppelte Einträge:** Überprüfen auf doppelte Zeilen oder Spieler, um sicherzustellen, dass es keine redundanten Daten gibt.
3. **Inhaltliche Konsistenz:** Überprüfen, ob alle Daten in den richtigen Formaten vorliegen, z. B. sollten die numerischen Felder keine Textwerte enthalten.

4. **Unregelmäßige Werte:** Überprüfen auf außergewöhnliche Werte, wie negative Zahlen in Spalten, die keine negativen Werte erlauben sollten (z. B. bei assists, corner\_taken, dribbles).

Hier ist ein einfacher Code, der diese Prüfungen durchführt:

```
python
KopierenBearbeiten
import pandas as pd

# Lade die Kaggle-Daten (ersetze 'path_to_kaggle_data.csv' durch deinen
Dateipfad)
kaggle_df = pd.read_csv('path_to_kaggle_data.csv')

# 1. Überprüfen auf fehlende Werte
missing_values = kaggle_df.isnull().sum()
print("Fehlende Werte pro Spalte:")
print(missing_values)

# 2. Überprüfen auf doppelte Zeilen
duplicates = kaggle_df.duplicated().sum()
print(f"\nDoppelte Zeilen: {duplicates}")

# 3. Überprüfen auf doppelte Spieler (z. B. gleiche Namen)
duplicate_players = kaggle_df['player_name'].duplicated().sum()
print(f"\nDoppelte Spieler (gleiche Namen): {duplicate_players}")

# 4. Überprüfen, ob alle numerischen Werte tatsächlich Zahlen sind (z. B.
keine Texte)
# Konvertiere die numerischen Spalten in Zahlen und prüfe auf Fehler
numerical_columns = ['assists', 'corner_taken', 'offsides', 'dribbles',
'match_played']
for column in numerical_columns:
    # Versuche, die Spalte in numerische Werte zu konvertieren und prüfe
auf Fehler
    kaggle_df[column] = pd.to_numeric(kaggle_df[column], errors='coerce')

    # Identifiziere Zeilen mit ungültigen Zahlen (NaN nach der
Konvertierung)
    invalid_entries = kaggle_df[kaggle_df[column].isna()]
    if not invalid_entries.empty:
        print(f"\nUngültige Werte in der Spalte {column}:")
        print(invalid_entries)

# 5. Überprüfen auf negative Werte in numerischen Spalten, wo es nicht
erlaubt ist
negative_values = kaggle_df[numerical_columns].lt(0).sum()
print("\nAnzahl der negativen Werte in den numerischen Spalten:")
print(negative_values)

# 6. Überprüfen auf inkonsistente Textwerte in den nicht-numerischen
Spalten
# Beispiel: Spielernamen, Club, Position - hier können wir nach führenden
oder nachfolgenden Leerzeichen suchen
inconsistent_text_values = kaggle_df[['player_name', 'club',
'position']].apply(lambda x: x.str.strip()).isna().sum()
print("\nInkonsequente Textwerte in den Spalten 'player_name', 'club' und
'position':")
print(inconsistent_text_values)
```

# Optional: Wenn du eine detaillierte Überprüfung bestimmter Spalten benötigst, kannst du hier hinzufügen.

## Was macht der Code?

1. **Fehlende Werte:** Der Code gibt die Anzahl der fehlenden Werte in jeder Spalte aus. Du kannst so schnell feststellen, in welchen Spalten Werte fehlen.
2. **Doppelte Einträge:** Es wird überprüft, ob es doppelte Zeilen im gesamten DataFrame gibt und wie viele doppelte Spieler vorhanden sind (also Spieler mit denselben Namen).
3. **Ungültige Werte in numerischen Spalten:** Es wird überprüft, ob alle Werte in den numerischen Spalten (`assists`, `corner_taken`, `offsides`, `dribbles`, `match_played`) gültige Zahlen sind. Ungültige Werte werden als `NaN` gekennzeichnet.
4. **Negative Werte:** Der Code identifiziert alle negativen Werte in numerischen Spalten. Da z. B. "Assists" oder "Dribblings" keine negativen Zahlen haben sollten, wird auf solche Werte geprüft.
5. **Inkonsequente Textwerte:** Es wird überprüft, ob Textwerte in den Spalten `player_name`, `club`, und `position` Leerzeichen am Anfang oder Ende haben. Es wird empfohlen, Leerzeichen zu entfernen, um Konsistenz zu gewährleisten.

## Weitere Tipps:

- **Fehlerhafte Daten:** Falls ungültige Werte in den Spalten vorhanden sind, kannst du eine genauere Analyse der spezifischen Zeilen durchführen, um herauszufinden, was mit diesen Werten nicht stimmt.
- **Spaltennamen:** Wenn du mehrere CSV-Dateien hast, solltest du sicherstellen, dass die Spaltennamen in allen Dateien konsistent sind.

Dieser Code überprüft die grundlegenden Konsistenzfragen, sodass du einen Überblick über die Qualität der Daten hast, ohne sie direkt zu ändern.

4o mini

Du:

Hier ist ein einfacher Code, der diese Prüfungen durchführt: python Code kopieren 

```
`import pandas as pd # Lade die Kaggle-Daten (ersetze 'path_to_kaggle_data.csv' durch deinen Dateipfad) kaggle_df = pd.read_csv('path_to_kaggle_data.csv') # 1. Überprüfen auf fehlende Werte missing_values = kaggle_df.isnull().sum() print("Fehlende Werte pro Spalte:") print(missing_values) # 2. Überprüfen auf doppelte Zeilen duplicates = kaggle_df.duplicated().sum() print(f"\nDoppelte Zeilen: {duplicates}") # 3. Überprüfen auf doppelte Spieler (z. B. gleiche Namen) duplicate_players = kaggle_df['player_name'].duplicated().sum() print(f"\nDoppelte Spieler (gleiche Namen): {duplicate_players}") # 4. Überprüfen, ob alle numerischen Werte tatsächlich Zahlen sind (z. B. keine Texte) # Konvertiere die numerischen Spalten in Zahlen und prüfe auf Fehler numerical_columns = ['assists', 'corner_taken', 'offsides', 'dribbles', 'match_played'] for column in numerical_columns: # Versuche, die Spalte in numerische Werte zu konvertieren und prüfe auf Fehler kaggle_df[column] = pd.to_numeric(kaggle_df[column], errors='coerce') # Identifiziere Zeilen mit ungültigen Zahlen (NaN nach der Konvertierung) invalid_entries = kaggle_df[kaggle_df[column].isna()] if not invalid_entries.empty: print(f"\nUngültige Werte in der Spalte {column}:") print(invalid_entries) # 5. Überprüfen auf negative Werte in
```

```
numerischen Spalten, wo es nicht erlaubt ist negative_values =
kaggle_df[numerical_columns].lt(0).sum() print("\nAnzahl der negativen Werte in den
numerischen Spalten:") print(negative_values) # 6. Überprüfen auf inkonsistente Textwerte in
den nicht-numerischen Spalten # Beispiel: Spielername, Club, Position - hier können wir nach
führenden oder nachfolgenden Leerzeichen suchen inconsistent_text_values =
kaggle_df[['player_name', 'club', 'position']].apply(lambda x: x.str.strip()).isna().sum()
print("\nInkonsequente Textwerte in den Spalten 'player_name', 'club' und 'position':")
print(inconsistent_text_values) # Optional: Wenn du eine detaillierte Überprüfung bestimmter
Spalten benötigst, kannst du hier hinzufügen.`
```

wie kann ich den code für 8 dateien schreiben und ich will nicht alle spalten benennen weil die spalten auch in jeder Datei anders sind

ChatGPT:

ChatGPT

Wenn du den Code so erweitern möchtest, dass er für mehrere CSV-Dateien funktioniert, und du nicht manuell alle Spalten benennen möchtest (weil sie in den Dateien variieren können), dann kannst du dynamisch auf alle Spalten zugreifen, anstatt spezifische Spalten zu definieren.

Der Code könnte dann wie folgt aussehen:

## Überprüfungen für mehrere CSV-Dateien

1. **Fehlende Werte**
2. **Doppelte Zeilen**
3. **Doppelte Spieler**
4. **Ungültige numerische Werte**
5. **Negative Werte in numerischen Spalten**
6. **Inkonsequente Textwerte**

Hier ist der vollständige Code, der auf alle CSV-Dateien in einem Verzeichnis anwendbar ist:

```
python
KopierenBearbeiten
import pandas as pd
import os

# Liste der CSV-Dateien in deinem Verzeichnis
directory = 'path_to_your_directory' # Pfad zu deinem Verzeichnis
csv_files = [f for f in os.listdir(directory) if f.endswith('.csv')]

# Funktion zur Überprüfung der Daten
def check_data(file_path):
    kaggle_df = pd.read_csv(file_path)
    print(f"\nÜberprüfung der Datei: {file_path}")

    # 1. Überprüfen auf fehlende Werte
    missing_values = kaggle_df.isnull().sum()
    print("Fehlende Werte pro Spalte:")
    print(missing_values)

    # 2. Überprüfen auf doppelte Zeilen
```

```

duplicates = kaggle_df.duplicated().sum()
print(f"\nDoppelte Zeilen: {duplicates}")

# 3. Überprüfen auf doppelte Spieler (z. B. gleiche Namen)
if 'player_name' in kaggle_df.columns:
    duplicate_players = kaggle_df['player_name'].duplicated().sum()
    print(f"\nDoppelte Spieler (gleiche Namen): {duplicate_players}")

# 4. Überprüfen, ob alle numerischen Werte tatsächlich Zahlen sind (z.
B. keine Texte)
numerical_columns = kaggle_df.select_dtypes(include=['float64',
'int64']).columns
for column in numerical_columns:
    # Versuche, die Spalte in numerische Werte zu konvertieren und
    prüfe auf Fehler
    kaggle_df[column] = pd.to_numeric(kaggle_df[column],
errors='coerce')

    # Identifiziere Zeilen mit ungültigen Zahlen (NaN nach der
Konvertierung)
    invalid_entries = kaggle_df[kaggle_df[column].isna()]
    if not invalid_entries.empty:
        print(f"\nUngültige Werte in der Spalte {column}:")
        print(invalid_entries)

# 5. Überprüfen auf negative Werte in numerischen Spalten, wo es nicht
erlaubt ist
for column in numerical_columns:
    negative_values = kaggle_df[column].lt(0).sum()
    print(f"\nAnzahl der negativen Werte in der Spalte {column}:
{negative_values}")

# 6. Überprüfen auf inkonsistente Textwerte in den nicht-numerischen
Spalten
text_columns = kaggle_df.select_dtypes(include=['object']).columns
for column in text_columns:
    inconsistent_text_values = kaggle_df[column].apply(lambda x:
isinstance(x, str) and x.strip() == '').sum()
    print(f"\nInkonsequente Textwerte in der Spalte '{column}':
{inconsistent_text_values}")

# Überprüfe alle CSV-Dateien im Verzeichnis
for file in csv_files:
    file_path = os.path.join(directory, file)
    check_data(file_path)

```

## Erklärung des Codes:

### 1. Dynamische Spaltenauswahl:

- Der Code verwendet `select_dtypes(include=['float64', 'int64'])`, um automatisch alle numerischen Spalten zu identifizieren.
- Ebenso wird `select_dtypes(include=['object'])` verwendet, um alle Textspalten (z. B. `player_name`, `club`, `position`) zu finden.

### 2. Iteriere über alle CSV-Dateien:

- Der Code sucht nach allen `.csv`-Dateien im angegebenen Verzeichnis (`directory`).
- Für jede Datei wird die Funktion `check_data()` aufgerufen, die dann die oben beschriebenen Prüfungen durchführt.



### 3. Fehlende Werte:

- `kaggle_df.isnull().sum()` gibt die Anzahl der fehlenden Werte in jeder Spalte aus.

### 4. Doppelte Zeilen:

- `kaggle_df.duplicated().sum()` zählt, wie viele doppelte Zeilen es gibt.

### 5. Doppelte Spieler:

- Wenn die Spalte `player_name` existiert, wird überprüft, ob es doppelte Einträge in dieser Spalte gibt.

### 6. Ungültige numerische Werte:

- Die numerischen Spalten werden mithilfe von `pd.to_numeric()` überprüft. Falls ungültige (nicht-numerische) Werte vorhanden sind, werden diese als `NaN` markiert.

### 7. Negative Werte:

- In den numerischen Spalten wird geprüft, ob es negative Werte gibt, die nicht erwartet werden.

### 8. Inkonsistente Textwerte:

- Der Code überprüft, ob in Textspalten leere oder nur aus Leerzeichen bestehende Werte vorhanden sind.

## Nutzung:

- Stelle sicher, dass der Pfad zu deinem Verzeichnis (`directory`) korrekt gesetzt ist.
- Der Code wird alle CSV-Dateien im angegebenen Verzeichnis durchlaufen und für jede Datei die beschriebenen Prüfungen durchführen.

Wenn du diesen Code ausführst, bekommst du eine detaillierte Ausgabe über jede Datei und deren Konsistenzprobleme, ohne dass du die Spalten manuell benennen musst.

40 mini

**Du:**

ich will das mir wenn es doppelte Namen gibt dies angezeigt werden

ChatGPT:

ChatGPT

Um doppelte Spieler zu finden und diese anzuzeigen, kannst du eine zusätzliche Prüfung einfügen, die überprüft, ob es mehr als einen Eintrag für denselben Spielernamen gibt und diese Spieler dann anzeigt.

Hier ist der angepasste Code, der doppelte Spieler mit denselben Namen identifiziert und anzeigt:

## Angepasster Code:

```
python
KopierenBearbeiten
import pandas as pd
import os
```

```

# Liste der CSV-Dateien in deinem Verzeichnis
directory = 'path_to_your_directory' # Pfad zu deinem Verzeichnis
csv_files = [f for f in os.listdir(directory) if f.endswith('.csv')]

# Funktion zur Überprüfung der Daten
def check_data(file_path):
    kaggle_df = pd.read_csv(file_path)
    print(f"\nÜberprüfung der Datei: {file_path}")

    # 1. Überprüfen auf fehlende Werte
    missing_values = kaggle_df.isnull().sum()
    print("Fehlende Werte pro Spalte:")
    print(missing_values)

    # 2. Überprüfen auf doppelte Zeilen
    duplicates = kaggle_df.duplicated().sum()
    print(f"\nDoppelte Zeilen: {duplicates}")

    # 3. Überprüfen auf doppelte Spieler (z. B. gleiche Namen)
    if 'player_name' in kaggle_df.columns:
        duplicate_players = kaggle_df['player_name'].duplicated(keep=False)
    # Markiert alle doppelten Spieler
    duplicated_rows = kaggle_df[duplicate_players]

    if not duplicated_rows.empty:
        print(f"\nDoppelte Spieler (gleiche Namen) in der Datei {file_path}:")
        print(duplicated_rows[['player_name', 'club']]) # Hier kannst du auch weitere Spalten angeben, wenn nötig

    # 4. Überprüfen, ob alle numerischen Werte tatsächlich Zahlen sind (z. B. keine Texte)
    numerical_columns = kaggle_df.select_dtypes(include=['float64', 'int64']).columns
    for column in numerical_columns:
        # Versuche, die Spalte in numerische Werte zu konvertieren und prüfe auf Fehler
        kaggle_df[column] = pd.to_numeric(kaggle_df[column], errors='coerce')

    # Identifiziere Zeilen mit ungültigen Zahlen (NaN nach der Konvertierung)
    invalid_entries = kaggle_df[kaggle_df[column].isna()]
    if not invalid_entries.empty:
        print(f"\nUngültige Werte in der Spalte {column}:")
        print(invalid_entries)

    # 5. Überprüfen auf negative Werte in numerischen Spalten, wo es nicht erlaubt ist
    for column in numerical_columns:
        negative_values = kaggle_df[column].lt(0).sum()
        print(f"\nAnzahl der negativen Werte in der Spalte {column}: {negative_values}")

    # 6. Überprüfen auf inkonsistente Textwerte in den nicht-numerischen Spalten
    text_columns = kaggle_df.select_dtypes(include=['object']).columns
    for column in text_columns:
        inconsistent_text_values = kaggle_df[column].apply(lambda x: isinstance(x, str) and x.strip() == '').sum()
        print(f"\nInkonsequente Textwerte in der Spalte '{column}': {inconsistent_text_values}")

```

```
# Überprüfe alle CSV-Dateien im Verzeichnis
for file in csv_files:
    file_path = os.path.join(directory, file)
    check_data(file_path)
```

## Erklärung der Änderungen:

### 1. Doppelte Spieler mit denselben Namen:

- Der Code verwendet `duplicated(keep=False)`, um alle doppelten Spieler zu markieren, einschließlich der ersten Vorkommen (nicht nur die doppelten nach der ersten Instanz).
- `duplicated_rows = kaggle_df[duplicate_players]` gibt die Zeilen zurück, die doppelte Spieler enthalten.
- Anschließend wird die Ausgabe der doppelten Spieler angezeigt, mit den Spalten `player_name` und `club`, aber du kannst auch andere Spalten angeben, falls gewünscht.

### 2. Ausgabe:

- Wenn doppelte Spieler gefunden werden, werden sie angezeigt, einschließlich ihres Vereins (`club`), damit du die doppelten Spieler leicht identifizieren kannst.

## Nutzung:

- Stelle sicher, dass der Pfad zu deinem Verzeichnis (`directory`) korrekt gesetzt ist.
- Der Code wird alle CSV-Dateien im angegebenen Verzeichnis durchlaufen, doppelte Spieler mit denselben Namen identifizieren und diese in der Ausgabe anzeigen.

4o mini

**Du:**

ich will das überprüft wird ob es bei den spieler Namen Duplikate gibt

**ChatGPT:**

ChatGPT

Um sicherzustellen, dass der Code überprüft, ob es Duplikate bei den Spielernamen gibt, und diese Duplikate anzeigt, kannst du die `duplicated()`-Funktion für die Spalte `player_name` verwenden. Dadurch wird festgestellt, ob ein Name mehr als einmal in einer Datei vorkommt.

Hier ist der Code, der speziell auf Duplikate der Spielernamen prüft und diese ausgibt:

## Angepasster Code:

```
python
KopierenBearbeiten
import pandas as pd
import os

# Liste der CSV-Dateien in deinem Verzeichnis
```

```

directory = 'path_to_your_directory' # Pfad zu deinem Verzeichnis
csv_files = [f for f in os.listdir(directory) if f.endswith('.csv')]

# Funktion zur Überprüfung der Daten
def check_data(file_path):
    kaggle_df = pd.read_csv(file_path)
    print(f"\nÜberprüfung der Datei: {file_path}")

    # 1. Überprüfen auf fehlende Werte
    missing_values = kaggle_df.isnull().sum()
    print("Fehlende Werte pro Spalte:")
    print(missing_values)

    # 2. Überprüfen auf doppelte Zeilen
    duplicates = kaggle_df.duplicated().sum()
    print(f"\nDoppelte Zeilen: {duplicates}")

    # 3. Überprüfen auf doppelte Spieler (gleiche Namen)
    if 'player_name' in kaggle_df.columns:
        duplicate_players = kaggle_df['player_name'].duplicated(keep=False)
    # Markiert alle doppelten Spieler
    duplicated_rows = kaggle_df[duplicate_players]

    if not duplicated_rows.empty:
        print(f"\nDoppelte Spieler (gleiche Namen) in der Datei {file_path}:")
        print(duplicated_rows[['player_name', 'club']]) # Hier kannst du auch andere Spalten angeben

    # 4. Überprüfen, ob alle numerischen Werte tatsächlich Zahlen sind (z. B. keine Texte)
    numerical_columns = kaggle_df.select_dtypes(include=['float64', 'int64']).columns
    for column in numerical_columns:
        # Versuche, die Spalte in numerische Werte zu konvertieren und prüfe auf Fehler
        kaggle_df[column] = pd.to_numeric(kaggle_df[column], errors='coerce')

    # Identifiziere Zeilen mit ungültigen Zahlen (NaN nach der Konvertierung)
    invalid_entries = kaggle_df[kaggle_df[column].isna()]
    if not invalid_entries.empty:
        print(f"\nUngültige Werte in der Spalte {column}:")
        print(invalid_entries)

    # 5. Überprüfen auf negative Werte in numerischen Spalten, wo es nicht erlaubt ist
    for column in numerical_columns:
        negative_values = kaggle_df[column].lt(0).sum()
        print(f"\nAnzahl der negativen Werte in der Spalte {column}: {negative_values}")

    # 6. Überprüfen auf inkonsistente Textwerte in den nicht-numerischen Spalten
    text_columns = kaggle_df.select_dtypes(include=['object']).columns
    for column in text_columns:
        inconsistent_text_values = kaggle_df[column].apply(lambda x: isinstance(x, str) and x.strip() == '').sum()
        print(f"\nInkonsequente Textwerte in der Spalte '{column}': {inconsistent_text_values}")

```

```
# Überprüfe alle CSV-Dateien im Verzeichnis
for file in csv_files:
    file_path = os.path.join(directory, file)
    check_data(file_path)
```

## Erklärung der Änderungen:

- **Überprüfung von Duplikaten in der Spalte `player_name`:**
  - Der Code prüft, ob der Spielername in der Spalte `player_name` mehrfach vorkommt, indem `duplicated(keep=False)` verwendet wird. Dies markiert alle Duplikate (einschließlich des ersten Vorkommens).
  - `duplicated_rows = kaggle_df[duplicate_players]` filtert die Zeilen, die doppelte Spielernamen enthalten.
  - Die doppelten Spieler (mit denselben Namen) werden zusammen mit ihrem Verein (`club`) angezeigt, was dir hilft, Duplikate leicht zu erkennen.

## Nutzung:

- Ersetze `'path_to_your_directory'` durch den tatsächlichen Pfad zu deinem Verzeichnis, in dem sich die CSV-Dateien befinden.
- Der Code wird durch alle CSV-Dateien im Verzeichnis iterieren und die Spieler mit doppelt vorkommenden Namen ausgeben.

Falls du nur die Spielernamen ohne die dazugehörigen Clubs sehen möchtest, kannst du die Zeile `print(duplicated_rows[['player_name', 'club']])` auf `print(duplicated_rows[['player_name']])` ändern.