```python
# == searchForTheWorst.ipynb ===

#%%
# Import Dependencies
import pandas as pd

#%%
# Create reference to CSV file
csv_path = "Resources/Soccer2018Data.csv"

# ---------------------------------------------------------------------------
# low_memory
# Internally process the file in chunks, resulting in lower memory use while
# parsing, but possibly mixed type inference. To ensure no mixed types either
# set False, or specify the type with the dtype parameter.
# Note: that the entire file is read into a single DataFrame regardless, use
# the chunksize or iterator parameter to return the data in chunks.
# Only valid with C parser
# https://pandas.pydata.org/pandas-docs/stable/generated/pandas.read_csv.html
# ---------------------------------------------------------------------------

# Import the CSV into a pandas DataFrame
soccer_2018_df = pd.read_csv(csv_path, low_memory=False)
soccer_2018_df

#%%
# Collect a list of all the unique values in "Preferred Position"
soccer_2018_df["Preferred Position"].unique()

#%%
# ---------------------------------------------------------------------------
# To filter in only those players that play in a particular position, use
# `df.loc()` and have it collect only those rows where "Preferred Position"
# is equal to the position desired.# Looking only at strikers (ST) to start
# ---------------------------------------------------------------------------
strikers_2018_df = soccer_2018_df.loc[soccer_2018_df[
    "Preferred Position"] == "ST", :]

strikers_2018_df.head()

#%%
# Sort the DataFrame by the values in the "ST" column to find the worst
strikers_2018_df = strikers_2018_df.sort_values("ST")

# Reset the index so that the index is now based on the sorting locations
strikers_2018_df = strikers_2018_df.reset_index(drop=True)

strikers_2018_df.head()

#%%
# Save all of the information collected on the worst striker
worst_striker = strikers_2018_df.loc[0, :]
worst_striker

#%%
```