```
In [1]: import os
        import pandas as pd
        from bs4 import BeautifulSoup
        from io import StringIO
        SCORE_DIR = "data/scores"
```

```
In [2]: box_scores = os.listdir(SCORE_DIR)
        box_scores = [os.path.join(SCORE_DIR, f) for f in box_scores if f.endswith("
```

```
In [3]: def parse_html(box_score):
            with open(box_score) as f:
                html = f.read()

            soup = BeautifulSoup(html)
            [s.decompose() for s in soup.select("tr.over_header")]
            [s.decompose() for s in soup.select("tr.thead")]

            return soup
```

```
In [4]: def read_season_info(soup):
            nav = soup.select("#bottom_nav_container")[0]
            hrefs = [a["href"] for a in nav.find_all('a')]
            season = os.path.basename(hrefs[1]).split("_")[0]
            return season
```

```
In [5]: def read_line_score(soup):
            line_score = pd.read_html(StringIO(str(soup)), attrs={'id': 'line_score'
            cols = list(line_score.columns)
            cols[0] = "team"
            cols[-1] = "total"
            line_score.columns = cols

            line_score = line_score[["team", "total"]]
            return line_score
```

```
In [6]: def read_stats(soup, team, stat):
            df = pd.read_html(StringIO(str(soup)), attrs = {'id': f'box-{team}-game-
            df = df.apply(pd.to_numeric, errors="coerce")
            return df
```

```
In [7]: games = []
        base_cols = None

        for box_score in box_scores:
            soup = parse_html (box_score)
            line_score = read_line_score(soup)
            teams = list(line_score["team"])

            summaries = []
            for team in teams:
                basic = read_stats(soup, team, "basic")
                advanced = read_stats(soup, team, "advanced")
```

```python
        totals = pd.concat([basic.iloc[-1,:], advanced.iloc[-1,:]])
        totals.index = totals.index.str.lower()

        maxes = pd.concat([basic.iloc[:-1].max(), advanced.iloc[:-1].max()])
        maxes.index = maxes.index.str.lower() + "_max"

        summary = pd.concat([totals, maxes])

        if base_cols is None:
            base_cols = list(summary.index.drop_duplicates(keep="first"))
            base_cols = [b for b in base_cols if "bpm" not in b]
        summary = summary[base_cols]
        summaries.append(summary)

    summary = pd.concat(summaries, axis=1).T

    game = pd.concat([summary, line_score], axis=1)

    game["home"] = [0,1]

    game_opp = game.iloc[::-1].reset_index()
    game_opp.columns += "_opp"

    full_game = pd.concat([game, game_opp], axis=1)
    full_game["season"] = read_season_info(soup)
    full_game["date"] = os.path.basename(box_score)[:8]
    full_game["date"] = pd.to_datetime(full_game["date"], format="%Y%m%d")
    full_game["won"] = full_game["total"] > full_game["total_opp"]
    games.append(full_game)

    if len(games) % 100 == 0:
        print(f"{len(games)} / {len(box_scores)}")
```

```
100 / 4874
200 / 4874
300 / 4874
400 / 4874
500 / 4874
600 / 4874
700 / 4874
800 / 4874
900 / 4874
1000 / 4874
1100 / 4874
1200 / 4874
1300 / 4874
1400 / 4874
1500 / 4874
1600 / 4874
1700 / 4874
1800 / 4874
1900 / 4874
2000 / 4874
2100 / 4874
2200 / 4874
2300 / 4874
2400 / 4874
2500 / 4874
2600 / 4874
2700 / 4874
2800 / 4874
2900 / 4874
3000 / 4874
3100 / 4874
3200 / 4874
3300 / 4874
3400 / 4874
3500 / 4874
3600 / 4874
3700 / 4874
3800 / 4874
3900 / 4874
4000 / 4874
4100 / 4874
4200 / 4874
4300 / 4874
4400 / 4874
4500 / 4874
4600 / 4874
4700 / 4874
4800 / 4874
```

```
In [16]: games_df = pd.concat(games, ignore_index = True)
```

```
In [28]: games_df.to_csv('nba_games_data.csv')
```

```
In [9]:
```

```
Out[9]: []
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```