Load: NBA Data

```
# load data
nba = pd.read_csv("nba.csv")
```

1. Find number of unique values in College, and Position Columns.

```
def get_unique_column(ds, name):
    idx_nan = ds[name].isnull()
    return ds[name][~idx_nan].unique()


unique_colleges = get_unique_column(nba, name="College")
unique_position = get_unique_column(nba, name="Position")

print("Unique values in College: ", len(unique_colleges))
print("Unique values in Position: ", len(unique_position))
```

2. Count and display the number of teams under each college.

```
count_teams_by_college = nba[ ["College", "Team"] ].groupby("College"
).count()

print(count_teams_by_college)
```

3. For each distinct value of Age, compute the following statistics :
    a. For Height column – minimum, and maximum value

```
res= nba[["Age", "Height"]].groupby("Age").agg({"Height": ["min", "max"]})
print(res)
```

    b. For Weight column – minimum, and maximum value

```
res= nba[["Age", "Weight"]].groupby("Age").agg({"Weight": ["min", "max"]})
print(res)
```

    c. For Salary column – sum (total), mean, and count.

```
res = nba[["Age", "Salary"]].groupby("Age").agg({"Salary": ["sum", "mean",
"count"]})
print(res)
```

4. For Texas College, find minimum Salary for every Position.

```
nba_texas_college = nba[nba["College"] == "Texas"]
res = nba_texas_college[["Position", "Salary"]].groupby("Position").min()
print(res)
```

5. Count number of players under every age ctegory, who play for 'Boston Celtics' Team.

```
boston_celtics = nba[nba["Team"] == "Boston Celtics"]

players_by_age = boston_celtics[["Age", "Name"]].groupby("Age").count()
players_by_age.columns = ["Players"]

print(players_by_age)
```

Load: Tips Data

```
tips = pd.read_csv("tips.csv")
```

1. Find how many female, and male customers smoke.

```
smokers = tips[tips.smoker == "Yes"]

smokers_gender = smokers[["gender", "smoker"]].groupby("gender").count()
print(smokers_gender)
```

2. Display the records of male customers who had dinner on Saturday.

```
res = tips[(tips.gender == "Male") & (tips.time == "Dinner") & (tips.day ==
"Sat")]
print(res)
```

3. Display the records of female customers who had meal either on Saturdays, or Sundays.

```
res =tips[(tips.gender == "Female") & (tips.day == "Sat") | (tips.day ==
"Sun")]
print(res)
```

4. For each day, print only the total_bill, and tip information (use for loop if necessary, to iterate through different keys/groups).

```
for day, data in tips.groupby("day"):
    print("Day: ", day)
    print(data.total_bill, data.tip)
```

5. For evey day, display number of males and females under smoking, nonsmoking categories.

```
for day, data in tips.groupby("day"):
    smokers = data[data.smoker == "Yes"]
    non_smokers = data[data.smoker == "No"]
    res = smokers[["gender", "smoker"]].groupby("gender").count()
    res["non smokers"] = (
        non_smokers[["gender", "smoker"]].groupby("gender").count()
    )
    print("Day: ", day)
    print(res)
    print("")
```

6. For each day and time, find the following statistics :
    a. For total_bill column – minimum, maximum, mean, and sum.

```
for (day, time), data in tips.groupby(["day", "time"]):
    res = data.agg(
        {
            "total_bill": ["min", "max", "mean", "sum"],
        },
    )
    print(day, time)
    print(res)
```

    b. For tip column – minimum, maximum, mean, and sum.

```
for (day, time), data in tips.groupby(["day", "time"]):
    res = data.agg(
        {
            "tip": ["min", "max", "mean", "sum"],
        },
    )
    print(day, time)
    print(res)
```

7. Find total number of non-smoker males and females

```
non_smokers = tips[tips.smoker == "No"]

count_non_smokers = non_smokers[ ["gender", "smoker"]
].groupby("gender").count()
count_non_smokers.columns = ["non-smoker"]
print(count_non_smokers)
```

8. Find average tip given by/per gender.

```
res = tips[["gender", "tip"]].groupby("gender").mean()
print(res)
```

9. Find how many male and female customers who smoke had dinner on Friday.

```
smokers = tips[(tips.smoker == "Yes") & (tips.day == "Fri") & (tips.time ==
"Dinner")]

res = smokers[["gender", "smoker"]].groupby("gender").count()
print(res)
```

10. Find the count/number of tips above amount $5.

```
count_tips_above_5 = len(tips[tips.tip > 5.0])
print(count_tips_above_5)
```

Load : States (population, area, abbreviation) Data

```
populations = pd.read_csv("state-population.csv")
abbreviations = pd.read_csv("state-abbreviations.csv")
area = pd.read_csv("state-areas.csv")
```

1. Perform left join between population and abbreviation data. What type of cardinality is involved between these datasets ?

```
pd.merge(populations, abbreviations, how="left", left_on="state/region",
right_on="abbreviation")
```

The cardinality between these dataset is many to one as the polulation data set contains many states, and abbreviations contains individual state information.

2. Perform right join between population and abbreviation data.

```
pd.merge(populations, abbreviations, how="right", left_on="state/region",
right_on="abbreviation")
```

3. What type of cardinality is involved between the above two datasets ? Also, explain why the number of rows returned in the above two queries (1,2) are different.

The cardinality between the above two dataset is one to many. Since we are doing right join, only the values from abbreviation data are considered. Hence, some of the rows from population data such as state/region labeled as USA are left out.

4. Perform inner join between area and abbreviation data.

```
area_abbreviation_inner = pd.merge(area, abbreviations, how="inner")
```

5. Perform outer join between area and abbreviation data.

```
area_abbreviation_outer = pd.merge(area, abbreviations, how="outer")
```

6. What type of cardinality is involved between the above two datasets ? Also, explain why the number of rows returned in the above two queries (4, 5) are different

The cardinality between the above two dataset is one to one. The number of rows returned are different because in inner join, we are only considering only the common values between area and abbreviation. This causes value Puerto Rico to be left out. On the other hand, for outer join we are consider all the elements both in area and abbreviation dataset.

7. Find the state, and it's abbrevation having minimum area (in square miles).

```
area_abbreviation = pd.merge(area, abbreviations, how="left", on="state")
min_area = area_abbreviation["area (sq. mi)"].min()
state_min_area = area_abbreviation[area_abbreviation["area (sq. mi)"] ==
min_area]
print(state_min_area)
```

8. Count total number of population in 2010 across all states.

```
populations_2010 = populations[populations.year == 2010]
poplulation_state = populations_2010[["state/region",
"population"]].groupby("state/region").sum()
print(poplulation_state)
print(poplulation_state["population"].sum())
```