```
#importing necessary libraries and loading the dataset:
```

```
import numpy as np
import pandas as pd

# Load dataset
df = pd.read_csv("/assists-turnovers (1).csv")

# Display first few rows
print(df.head())

```

```
   Unnamed: 0  GP  Games played  GS  Games started  MPG  Minutes Per Game  \
0           0               19                  0                 21.3
1           1               19                  2                 18.1
2           2               20                  3                 23.6
3           3               18                  1                 14.8
4           4               19                  0                 13.7

   OREB  Offensive Rebounds  DREB  Defensive Rebounds  REB  Total Rebounds  \
0                       6.0                    31.0                 37.0
1                       4.0                    31.0                 35.0
2                       6.0                    30.0                 36.0
3                       5.0                    28.0                 33.0
4                       8.0                    26.0                 34.0

   RPG  Rebounds Per Game  AST  Total Assists  APG  Assists Per Game  \
0                    1.9                NaN                    NaN
1                    1.8                NaN                    NaN
2                    1.8                NaN                    NaN
3                    1.8                NaN                    NaN
4                    1.8                NaN                    NaN

   TO  Turnovers  TOPG  Turnovers Per Game  A/TO  Assists Per Turnover  \
0           NaN                      NaN                        NaN
1           NaN                      NaN                        NaN
2           NaN                      NaN                        NaN
3           NaN                      NaN                        NaN
4           NaN                      NaN                        NaN

             Player Position Team
0       George Hill       PG  MIL
1  Danuel House Jr.       SF  PHI
2        Tyus Jones       PG  MEM
3       Moses Moody       SF   GS
4      Derrick Rose       PG   NY
```

```
#UNIT I – NumPy Operations
```

```
# Clean column names
df.columns = df.columns.str.strip()

# Rename necessary columns
df = df.rename(columns={
    'AST  Total Assists': 'Assists',
    'TO  Turnovers': 'Turnovers'
})

# Check it's all good
print(df[['Player', 'Assists', 'Turnovers']].head())

```

```
             Player  Assists  Turnovers
0       George Hill      NaN        NaN
1  Danuel House Jr.      NaN        NaN
2        Tyus Jones      NaN        NaN
3       Moses Moody      NaN        NaN
4      Derrick Rose      NaN        NaN
```

```
#fixed type arrays
import numpy as np

# Create array with fixed data type
fixed_array = np.array([1, 2, 3, 4], dtype=np.int32)
print("Fixed type array:", fixed_array)

```

```
Fixed type array: [1 2 3 4]
```

```
#Creating Arrays
zeros_array = np.zeros(5)
ones_array = np.ones((2, 3))
range_array = np.arange(0, 10, 2)

print("Zeros:", zeros_array)
print("Ones:", ones_array)
print("Range:", range_array)

```

```
Zeros: [0. 0. 0. 0. 0.]
Ones: [[1. 1. 1.]
 [1. 1. 1.]]
Range: [0 2 4 6 8]
```

```
#Array Indexing
arr = np.array([10, 20, 30, 40, 50])
print("Element at index 2:", arr[2])

```

```
Element at index 2: 30
```

```
#Array Slicing
print("Slice [1:4]:", arr[1:4])

```

```
Slice [1:4]: [20 30 40]
```

```
#Reshaping Arrays
reshaped = np.arange(12).reshape(3, 4)
print("Reshaped (3x4):\n", reshaped)

```

```
Reshaped (3x4):
 [[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]
```

```
#Array Concatenation and Splitting
a = np.array([1, 2, 3])
b = np.array([4, 5, 6])
concat = np.concatenate([a, b])
split = np.split(np.arange(6), 3)

print("Concatenated:", concat)
print("Split arrays:", split)

```

```
Concatenated: [1 2 3 4 5 6]
Split arrays: [array([0, 1]), array([2, 3]), array([4, 5])]
```

```
#Universal Functions
x = np.array([1, 2, 3])
y = np.array([4, 5, 6])

print("Add:", np.add(x, y))
print("Multiply:", np.multiply(x, y))

```

```
Add: [5 7 9]
Multiply: [ 4 10 18]
```

```
#Aggregations
print("Sum:", np.sum(x))
print("Mean:", np.mean(x))
print("Max:", np.max(x))

```

```
Sum: 6
Mean: 2.0
Max: 3
```

```
#Broadcasting Rules
arr = np.array([1, 2, 3])
broadcasted = arr + 5
print("Broadcast add:", broadcasted)

```

```
Broadcast add: [6 7 8]
```

```
#Comparisons
compare = arr > 2
print("Greater than 2:", compare)

```

```
Greater than 2: [False False  True]
```

```
#Fancy Indexing
values = np.array([4, 1, 3, 2])
indices = [3, 1, 0]
print("Fancy indexing:", values[indices])

```

```
Fancy indexing: [2 1 4]
```

```
# Fast Sorting using np.sort and np.argsort
print("Sorted:", np.sort(values))
print("Indices for sort:", np.argsort(values))

```

```
Sorted: [1 2 3 4]
Indices for sort: [1 3 2 0]
```

```
#Partial Sorting
arr = np.array([7, 2, 5, 1, 9, 3])
top3 = np.partition(arr, -3)[-3:]
print("Top 3 values (unsorted):", top3)

```

```
Top 3 values (unsorted): [5 7 9]
```

```
#Structured Arrays
data = np.array([('Alice', 5.0), ('Bob', 7.2)],
                dtype=[('name', 'U10'), ('score', 'f4')])
print("Structured:", data)
print("Bob's score:", data[1]['score'])

```

```
Structured: [('Alice', 5. ) ('Bob', 7.2)]
Bob's score: 7.2
```

```
#Compound Types / Record Arrays
record = np.rec.array([('Tom', 4.5), ('Jerry', 6.3)],
                      dtype=[('name', 'U10'), ('grade', 'f4')])
print("Record array:", record)
print("Tom's grade:", record[0].grade)

```

```
Record array: [('Tom', 4.5) ('Jerry', 6.3)]
Tom's grade: 4.5
```

```
#UNIT II: Pandas
```

```
import pandas as pd

# Create a simple Series
players = pd.Series(['George Hill', 'Danuel House Jr.', 'Tyus Jones'])
print(players)

```

```
0         George Hill
1    Danuel House Jr.
2          Tyus Jones
dtype: object
```

```
#DataFrame Object
# Use your existing dataset
df = pd.read_csv("/assists-turnovers (1).csv")
df.columns = df.columns.str.strip()
df = df.rename(columns={'AST  Total Assists': 'Assists', 'TO  Turnovers': 'Turnovers'})

# Display DataFrame
print(df.head())

```

```
   Unnamed: 0  GP  Games played  GS  Games started  MPG  Minutes Per Game  \
0           0                19                  0                   21.3
1           1                19                  2                   18.1
2           2                20                  3                   23.6
3           3                18                  1                   14.8
4           4                19                  0                   13.7

   OREB  Offensive Rebounds  DREB  Defensive Rebounds  REB  Total Rebounds  \
0                       6.0                      31.0                 37.0
1                       4.0                      31.0                 35.0
2                       6.0                      30.0                 36.0
3                       5.0                      28.0                 33.0
4                       8.0                      26.0                 34.0

   RPG  Rebounds Per Game  Assists  APG  Assists Per Game  Turnovers  \
0                    1.9      NaN                    NaN        NaN
1                    1.8      NaN                    NaN        NaN
2                    1.8      NaN                    NaN        NaN
3                    1.8      NaN                    NaN        NaN
4                    1.8      NaN                    NaN        NaN

   TOPG  Turnovers Per Game  A/TO  Assists Per Turnover            Player  \
0                       NaN                         NaN       George Hill
1                       NaN                         NaN  Danuel House Jr.
2                       NaN                         NaN        Tyus Jones
3                       NaN                         NaN       Moses Moody
4                       NaN                         NaN      Derrick Rose

  Position Team
0       PG  MIL
1       SF  PHI
2       PG  MEM
3       SF   GS
4       PG   NY
```

```
#Data Indexing & Selecting (Series)
# Accessing Series values
print("First player:", df['Player'].iloc[0])

```

```
First player: George Hill
```

```
#Data Indexing & Selecting (DataFrame)
# Row selection with .iloc and .loc
print("Third row:\n", df.iloc[2])
print("Player and Team columns:\n", df[['Player', 'Team']].head())

```

```
Third row:
 Unnamed: 0                           2
GP  Games played                    20
GS  Games started                    3
MPG  Minutes Per Game             23.6
OREB  Offensive Rebounds           6.0
DREB  Defensive Rebounds          30.0
REB  Total Rebounds               36.0
RPG  Rebounds Per Game             1.8
Assists                            NaN
APG  Assists Per Game              NaN
Turnovers                          NaN
TOPG  Turnovers Per Game           NaN
A/TO  Assists Per Turnover         NaN
Player                       Tyus Jones
Position                             PG
Team                                MEM
Name: 2, dtype: object
Player and Team columns:
              Player Team
0       George Hill  MIL
```

```
1  Danuel House Jr.  PHI
2        Tyus Jones  MEM
3       Moses Moody   GS
4      Derrick Rose   NY
```

```
#Universal Functions with Index Preservation
# Adding 10 to assists using a ufunc
df['Adjusted_Assists'] = df['Assists'].fillna(0) + 10
print(df[['Assists', 'Adjusted_Assists']].head())

```

```
   Assists  Adjusted_Assists
0      NaN              10.0
1      NaN              10.0
2      NaN              10.0
3      NaN              10.0
4      NaN              10.0
```

```
#Index Alignment
# Aligning two Series by index
assists = df['Assists'].fillna(0)
turnovers = df['Turnovers'].fillna(1)

assist_to_turnover = assists / turnovers
print("A/TO ratio:\n", assist_to_turnover.head())

```

```
A/TO ratio:
 0    0.0
1    0.0
2    0.0
3    0.0
4    0.0
dtype: float64
```

```
#Operations Between Series and DataFrames
# Subtract mean of assists from all values
df['Assist_Deviation'] = df['Assists'] - df['Assists'].mean()
print(df[['Player', 'Assist_Deviation']].head())

```

```
             Player  Assist_Deviation
0       George Hill               NaN
1  Danuel House Jr.               NaN
2        Tyus Jones               NaN
3       Moses Moody               NaN
4      Derrick Rose               NaN
```

```
#Handling Missing Data
# Check for nulls
print(df.isnull().sum())

# Fill missing assists with 0
df['Assists'].fillna(0, inplace=True)

```

```
Unnamed: 0                      0
GP  Games played                0
GS  Games started               0
MPG  Minutes Per Game         250
OREB  Offensive Rebounds      250
DREB  Defensive Rebounds      250
REB  Total Rebounds           250
RPG  Rebounds Per Game        250
Assists                        29
APG  Assists Per Game          29
Turnovers                      29
TOPG  Turnovers Per Game       29
A/TO  Assists Per Turnover     29
Player                          0
Position                        0
Team                            0
Adjusted_Assists                0
Assist_Deviation               29
dtype: int64
<ipython-input-37-cb5479a41725>:6: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignm
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting valu
```

```
For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].me


  df['Assists'].fillna(0, inplace=True)
```

```
#Operating on Null Values
# Drop rows with any nulls
df_cleaned = df.dropna()
print("Shape after dropping nulls:", df_cleaned.shape)

# Replace nulls in 'Turnovers' with column mean
df['Turnovers'].fillna(df['Turnovers'].mean(), inplace=True)

```

```
Shape after dropping nulls: (0, 18)
<ipython-input-38-a625d63297b7>:7: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignm
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting valu

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].me


  df['Turnovers'].fillna(df['Turnovers'].mean(), inplace=True)
```

```
#Hierarchical Indexing
# Create a multi-index from Team and Position
df.set_index(['Team', 'Position'], inplace=True)
print(df.head())

# Reset back to default index if needed
df.reset_index(inplace=True)

```

```
               Unnamed: 0  GP  Games played  GS  Games started  \
Team Position
MIL  PG                 0                19                  0
PHI  SF                 1                19                  2
MEM  PG                 2                20                  3
GS   SF                 3                18                  1
NY   PG                 4                19                  0

               MPG  Minutes Per Game  OREB  Offensive Rebounds  \
Team Position
MIL  PG                        21.3                       6.0
PHI  SF                        18.1                       4.0
MEM  PG                        23.6                       6.0
GS   SF                        14.8                       5.0
NY   PG                        13.7                       8.0

               DREB  Defensive Rebounds  REB  Total Rebounds  \
Team Position
MIL  PG                           31.0                 37.0
PHI  SF                           31.0                 35.0
MEM  PG                           30.0                 36.0
GS   SF                           28.0                 33.0
NY   PG                           26.0                 34.0

               RPG  Rebounds Per Game  Assists  APG  Assists Per Game  \
Team Position
MIL  PG                          1.9      0.0                    NaN
PHI  SF                          1.8      0.0                    NaN
MEM  PG                          1.8      0.0                    NaN
GS   SF                          1.8      0.0                    NaN
NY   PG                          1.8      0.0                    NaN

               Turnovers  TOPG  Turnovers Per Game  \
Team Position
MIL  PG           29.092                      NaN
PHI  SF           29.092                      NaN
MEM  PG           29.092                      NaN
GS   SF           29.092                      NaN
NY   PG           29.092                      NaN

               A/TO  Assists Per Turnover            Player  Adjusted_Assists  \
Team Position
MIL  PG                               NaN       George Hill              10.0
PHI  SF                               NaN  Danuel House Jr.              10.0
MEM  PG                               NaN        Tyus Jones              10.0
GS   SF                               NaN       Moses Moody              10.0
```

```
NY   PG                              NaN      Derrick Rose              10.0

                Assist_Deviation
Team Position
MIL  PG                      NaN
PHI  SF                      NaN
MEM  PG                      NaN
GS   SF                      NaN
NY   PG                      NaN
```

```
#UNIT III: Combining & Aggregating
```

```
#Concatenation
import pandas as pd

df = pd.read_csv("/assists-turnovers (1).csv")
df.columns = df.columns.str.strip()
df = df.rename(columns={'AST  Total Assists': 'Assists', 'TO  Turnovers': 'Turnovers'})

# Create a copy to simulate multiple data sources
df_copy = df.copy()

# Concatenate vertically (stack one after another)
concat_df = pd.concat([df, df_copy], ignore_index=True)
print("Shape after concat:", concat_df.shape)

```

```
Shape after concat: (558, 16)
```

```
#merge
# Simulate merge using a DataFrame with player & team
teams_df = df[['Player', 'Team']].copy()
teams_df['League'] = 'NBA'

merged_df = pd.merge(df, teams_df, on=['Player', 'Team'])
print("Merged DataFrame:\n", merged_df[['Player', 'Team', 'League']].head())

```

```
Merged DataFrame:
                Player Team League
0       George Hill  MIL    NBA
1       George Hill  MIL    NBA
2  Danuel House Jr.  PHI    NBA
3  Danuel House Jr.  PHI    NBA
4        Tyus Jones  MEM    NBA
```

```
#join
# Create two DataFrames with the same index
df1 = df[['Player', 'Assists']].set_index('Player')
df2 = df[['Player', 'Turnovers']].set_index('Player')

joined_df = df1.join(df2)
print("Joined DataFrame:\n", joined_df.head())

```

```
Joined DataFrame:
                     Assists  Turnovers
Player
George Hill           NaN        NaN
George Hill           NaN       14.0
Danuel House Jr.      NaN        NaN
Danuel House Jr.      NaN       11.0
Tyus Jones            NaN        NaN
```

```
# Inner Join
merged_inner = pd.merge(df1, df2, on='Player', how='inner')

```

```
#Left Join
merged_left = pd.merge(df1, df2, on='Player', how='left')

```

```
#Right Join
merged_right = pd.merge(df1, df2, on='Player', how='right')
```

```
#Outer Join (Full Join)
merged_outer = pd.merge(df1, df2, on='Player', how='outer')
```

```
#Aggregation with GroupBy
# Total assists by team
assists_by_team = df.groupby('Team')['Assists'].sum()
print("Total assists by team:\n", assists_by_team)

# Mean turnovers by team
mean_turnovers = df.groupby('Team')['Turnovers'].mean()
print("Mean turnovers by team:\n", mean_turnovers)
```

```
DAL    391.0
DEN    477.0
DET    336.0
GS     562.0
HOU    404.0
IND    520.0
LAC    466.0
LAL    339.0
MEM    396.0
MIA    353.0
MIL    441.0
MIN    532.0
NO     512.0
NY     437.0
OKC    478.0
ORL    269.0
PHI    302.0
PHO    400.0
POR    372.0
SA     475.0
SAC    515.0
TOR    305.0
UTA    593.0
WAS    423.0
Name: Assists, dtype: float64
Mean turnovers by team:
 Team
ATL    26.300000
BKN    29.250000
BOS    26.666667
CHA    27.500000
CHI    29.444444
CLE    32.375000
DAL    26.625000
DEN    25.888889
DET    31.833333
GS     35.625000
HOU    35.333333
IND    30.222222
LAC    34.000000
LAL    26.500000
MEM    26.750000
MIA    29.142857
MIL    29.625000
MIN    35.111111
NO     26.300000
NY     29.375000
OKC    23.909091
ORL    26.500000
PHI    24.142857
PHO    25.625000
POR    31.571429
SA     29.111111
SAC    26.800000
TOR    26.666667
UTA    35.222222
WAS    30.857143
Name: Turnovers, dtype: float64
```

```
#Multiple Aggregations
# Aggregating assists and turnovers
agg_stats = df.groupby('Team')[['Assists', 'Turnovers']].agg(['sum', 'mean'])
print("Aggregated stats:\n", agg_stats)
```

```
Aggregated stats:
      Assists            Turnovers
         sum       mean       sum       mean
Team
ATL    484.0  48.400000     263.0  26.300000
BKN    451.0  56.375000     234.0  29.250000
BOS    544.0  60.444444     240.0  26.666667
CHA    373.0  46.625000     220.0  27.500000
CHI    460.0  51.111111     265.0  29.444444
CLE    457.0  57.125000     259.0  32.375000
DAL    391.0  48.875000     213.0  26.625000
DEN    477.0  53.000000     233.0  25.888889
DET    336.0  56.000000     191.0  31.833333
GS     562.0  70.250000     285.0  35.625000
HOU    404.0  44.888889     318.0  35.333333
IND    520.0  57.777778     272.0  30.222222
LAC    466.0  51.777778     306.0  34.000000
LAL    339.0  42.375000     212.0  26.500000
MEM    396.0  49.500000     214.0  26.750000
MIA    353.0  50.428571     204.0  29.142857
MIL    441.0  55.125000     237.0  29.625000
MIN    532.0  59.111111     316.0  35.111111
NO     512.0  51.200000     263.0  26.300000
NY     437.0  54.625000     235.0  29.375000
OKC    478.0  43.454545     263.0  23.909091
ORL    269.0  33.625000     212.0  26.500000
PHI    302.0  43.142857     169.0  24.142857
PHO    400.0  50.000000     205.0  25.625000
POR    372.0  53.142857     221.0  31.571429
SA     475.0  52.777778     262.0  29.111111
SAC    515.0  51.500000     268.0  26.800000
TOR    305.0  50.833333     160.0  26.666667
UTA    593.0  65.888889     317.0  35.222222
WAS    423.0  60.428571     216.0  30.857143
```

```
#Pivot Table
# Pivot: Mean assists per team
pivot = df.pivot_table(values='Assists', index='Team', aggfunc='mean')
print("Pivot Table (Assists):\n", pivot.head())

```

```
Pivot Table (Assists):
         Assists
Team
ATL   48.400000
BKN   56.375000
BOS   60.444444
CHA   46.625000
CHI   51.111111
```

```
#Pivot Table with Multiple Aggregations
pivot_multi = df.pivot_table(values=['Assists', 'Turnovers'], index='Team', aggfunc=['mean', 'sum'])
print("Pivot Table (Multi):\n", pivot_multi.head())

```

```
Pivot Table (Multi):
             mean                sum
        Assists  Turnovers Assists Turnovers
Team
ATL   48.400000  26.300000   484.0     263.0
BKN   56.375000  29.250000   451.0     234.0
BOS   60.444444  26.666667   544.0     240.0
CHA   46.625000  27.500000   373.0     220.0
CHI   51.111111  29.444444   460.0     265.0
```

```
Start coding or generate with AI.
```