```
import pandas as pd


# Specify the path to your CSV file
file_path = 'data.csv'  # Change 'your_file.csv' to the actual file name

# Read the CSV file into a DataFrame
df = pd.read_csv(file_path)

# Display the DataFrame
df.head()  # Use df.head() to display the first few rows of the DataFrame
```

	Position ID	Position Status	Time	Time Out	Timecard Hours (as Time)	Pay Cycle Start Date	Pay Cycle End Date	Emp
0	WFS000054	Active	NaN	NaN	NaN	NaN	NaN	S Pra
1	WFS000065	Active	09/12/2023 10:08 AM	09/12/2023 01:53 PM	3:45	09/10/2023	09/23/2023	REsaX
2	WFS000065	Active	09/12/2023 02:23 PM	09/12/2023 07:02 PM	4:39	09/10/2023	09/23/2023	REsaX
3	WFS000065	Active	09/13/2023 10:08 AM	09/13/2023 02:20 PM	4:12	09/10/2023	09/23/2023	REsaX
[illegible]	[illegible]	[illegible]	09/13/2023	09/13/2023	[illegible]	[illegible]	[illegible]	REsaX

```
df.describe()
```

	File Number	Unnamed: 9
count	1484.000000	0.0
mean	437.658356	NaN
std	136.732971	NaN
min	54.000000	NaN
25%	368.000000	NaN
50%	491.000000	NaN
75%	546.500000	NaN
max	591.000000	NaN

```
# Specify the column(s) to be dropped
columns_to_drop = ['Unnamed: 9', 'Unnamed: 10']  # Replace with the actual column names

# Drop the specified columns
df = df.drop(columns=columns_to_drop)


df.head()
```

	Position ID	Position Status	Time	Time Out	Timecard Hours (as Time)	Pay Cycle Start Date	Pay Cycle End Date	Emp
0	WFS000054	Active	NaN	NaN	NaN	NaN	NaN	S Pra
1	WFS000065	Active	09/12/2023 10:08 AM	09/12/2023 01:53 PM	3:45	09/10/2023	09/23/2023	REsaX
2	WFS000065	Active	09/12/2023 02:23 PM	09/12/2023 07:02 PM	4:39	09/10/2023	09/23/2023	REsaX
3	WFS000065	Active	09/13/2023 10:08 AM	09/13/2023 02:20 PM	4:12	09/10/2023	09/23/2023	REsaX

```
df.dtypes
```

```
Position ID                 object
Position Status             object
Time                        object
Time Out                    object
Timecard Hours (as Time)    object
```

```
Pay Cycle Start Date        object
Pay Cycle End Date          object
Employee Name               object
File Number                  int64
dtype: object
```

```
#df = original_df.copy(deep=True)
```

```
df.head()
```

	Position ID	Position Status	Time	Time Out	Timecard Hours (as Time)	Pay Cycle Start Date	Pay Cycle End Date	Emp
0	WFS000054	Active	NaN	NaN	NaN	NaN	NaN	S Pra
1	WFS000065	Active	09/12/2023 10:08 AM	09/12/2023 01:53 PM	3:45	09/10/2023	09/23/2023	REsaX
2	WFS000065	Active	09/12/2023 02:23 PM	09/12/2023 07:02 PM	4:39	09/10/2023	09/23/2023	REsaX
3	WFS000065	Active	09/13/2023 10:08 AM	09/13/2023 02:20 PM	4:12	09/10/2023	09/23/2023	REsaX

```
# Convert 'Time' column to datetime
df['Time'] = pd.to_datetime(df['Time'], errors='coerce')

# Convert 'Time' column to 24-hour format
df['Time'] = df['Time'].dt.strftime('%Y-%m-%d %H:%M:%S')

# Display the updated DataFrame
print(df)
```

```
     Position ID Position Status                 Time             Time Out  \
0      WFS000054          Active                  NaN                  NaN   
1      WFS000065          Active  2023-09-12 10:08:00  09/12/2023 01:53 PM   
2      WFS000065          Active  2023-09-12 14:23:00  09/12/2023 07:02 PM   
3      WFS000065          Active  2023-09-13 10:08:00  09/13/2023 02:20 PM   
4      WFS000065          Active  2023-09-13 14:50:00  09/13/2023 08:44 PM   
...          ...             ...                  ...                  ...   
1479   WFS000589          Active  2023-09-20 09:55:00  09/20/2023 02:30 PM   
1480   WFS000589          Active  2023-09-20 15:00:00  09/20/2023 07:29 PM   
1481   WFS000589          Active  2023-09-21 09:56:00  09/21/2023 02:30 PM   
1482   WFS000589          Active  2023-09-21 15:00:00  09/21/2023 07:16 PM   
1483   WFS000591          Active                  NaN                  NaN   

     Timecard Hours (as Time) Pay Cycle Start Date Pay Cycle End Date  \
0                         NaN                  NaN                NaN   
1                        3:45           09/10/2023         09/23/2023   
2                        4:39           09/10/2023         09/23/2023   
3                        4:12           09/10/2023         09/23/2023   
4                        5:54           09/10/2023         09/23/2023   
...                       ...                  ...                ...   
1479                     4:35           09/10/2023         09/23/2023   
1480                     4:29           09/10/2023         09/23/2023   
1481                     4:34           09/10/2023         09/23/2023   
1482                     4:16           09/10/2023         09/23/2023   
1483                      NaN                  NaN                NaN   

                  Employee Name  File Number  
0                SiWgh, PraGhjEM           54  
1                REsaXiaWE, XAis           65  
2                REsaXiaWE, XAis           65  
3                REsaXiaWE, XAis           65  
4                REsaXiaWE, XAis           65  
...                          ...          ...  
1479             WgAyeW, RayCEWd          589  
1480             WgAyeW, RayCEWd          589  
1481             WgAyeW, RayCEWd          589  
1482             WgAyeW, RayCEWd          589  
1483  ArveXE RECerE, AWdres JesAs          591  

[1484 rows x 9 columns]
```

```
df.head()
```

	Position ID	Position Status	Time	Time Out	Timecard Hours (as Time)	Pay Cycle Start Date	Pay Cycle End Date	Emplo N
0	WFS000054	Active	NaN	NaN	NaN	NaN	NaN	SiV PraGh
1	WFS000065	Active	2023-09-12 10:08:00	09/12/2023 01:53 PM	3:45	09/10/2023	09/23/2023	REsaXia' X
2	WFS000065	Active	2023-09-12 14:23:00	09/12/2023 07:02 PM	4:39	09/10/2023	09/23/2023	REsaXia' X
			2023-					

```
# Convert 'Time Out' column to datetime
df['Time Out'] = pd.to_datetime(df['Time Out'], errors='coerce')

# Convert 'Time Out' column to 24-hour format
df['Time Out'] = df['Time Out'].dt.strftime('%Y-%m-%d %H:%M:%S')
```

```
df.head(10)
```

	Position ID	Position Status	Time	Time Out	Timecard Hours (as Time)	Pay Cycle Start Date	Pay Cycle End Date	Employe Nam
0	WFS000054	Active	NaN	NaN	NaN	NaN	NaN	SiWgl PraGhjEl
1	WFS000065	Active	2023-09-12 10:08:00	2023-09-12 13:53:00	3:45	09/10/2023	09/23/2023	REsaXiaWE XA
2	WFS000065	Active	2023-09-12 14:23:00	2023-09-12 19:02:00	4:39	09/10/2023	09/23/2023	REsaXiaWE XA
3	WFS000065	Active	2023-09-13 10:08:00	2023-09-13 14:20:00	4:12	09/10/2023	09/23/2023	REsaXiaWE XA
4	WFS000065	Active	2023-09-13 14:50:00	2023-09-13 20:44:00	5:54	09/10/2023	09/23/2023	REsaXiaWE XA
5	WFS000065	Active	2023-09-14 10:09:00	2023-09-14 14:30:00	4:21	09/10/2023	09/23/2023	REsaXiaWE XA
6	WFS000065	Active	2023-09-14 15:00:00	2023-09-14 19:14:00	4:14	09/10/2023	09/23/2023	REsaXiaWE XA

```
df.dtypes
```

```
Position ID                 object
Position Status             object
Time                        object
Time Out                    object
Timecard Hours (as Time)    object
Pay Cycle Start Date        object
Pay Cycle End Date          object
Employee Name               object
File Number                  int64
dtype: object
```

```
# Convert 'Time' and 'Time Out' columns to datetime
df['Time'] = pd.to_datetime(df['Time'], errors='coerce')
df['Time Out'] = pd.to_datetime(df['Time Out'], errors='coerce')

# Convert 'Pay Cycle Start Date' and 'Pay Cycle End Date' columns to date
df['Pay Cycle Start Date'] = pd.to_datetime(df['Pay Cycle Start Date']).dt.date
df['Pay Cycle End Date'] = pd.to_datetime(df['Pay Cycle End Date']).dt.date

# Display the updated DataFrame
df.head(15)
```

	Position ID	Position Status	Time	Time Out	Timecard Hours (as Time)	Pay Cycle Start Date	Pay Cycle End Date	Employee Name	Fi Numb
0	WFS000054	Active	NaT	NaT	NaN	NaT	NaT	SiWgh, PraGhjEM	
1	WFS000065	Active	2023-09-12 10:08:00	2023-09-12 13:53:00	3:45	2023-09-10	2023-09-23	REsaXiaWE, XAis	
2	WFS000065	Active	2023-09-12 14:23:00	2023-09-12 19:02:00	4:39	2023-09-10	2023-09-23	REsaXiaWE, XAis	
3	WFS000065	Active	2023-09-13 10:08:00	2023-09-13 14:20:00	4:12	2023-09-10	2023-09-23	REsaXiaWE, XAis	
4	WFS000065	Active	2023-09-13 14:50:00	2023-09-13 20:44:00	5:54	2023-09-10	2023-09-23	REsaXiaWE, XAis	
5	WFS000065	Active	2023-09-14 10:09:00	2023-09-14 14:30:00	4:21	2023-09-10	2023-09-23	REsaXiaWE, XAis	
6	WFS000065	Active	2023-09-14 15:00:00	2023-09-14 19:14:00	4:14	2023-09-10	2023-09-23	REsaXiaWE, XAis	
7	WFS000065	Active	2023-09-15 10:11:00	2023-09-15 14:41:00	4:30	2023-09-10	2023-09-23	REsaXiaWE, XAis	
8	WFS000065	Active	2023-09-15 15:11:00	2023-09-15 19:05:00	3:54	2023-09-10	2023-09-23	REsaXiaWE, XAis	
9	WFS000065	Active	2023-09-16 09:51:00	2023-09-16 14:50:00	4:59	2023-09-10	2023-09-23	REsaXiaWE, XAis	
10	WFS000065	Active	2023-09-16	2023-09-16	4:59	2023-	2023-	REsaXiaWE,	

```
df.dtypes
```

```
Position ID                         object
Position Status                     object
Time                        datetime64[ns]
Time Out                    datetime64[ns]
Timecard Hours (as Time)            object
Pay Cycle Start Date                object
Pay Cycle End Date                  object
Employee Name                       object
File Number                          int64
dtype: object
```

```
# Check null values in each column
null_values_per_column = df.isnull().sum()

# Display the null values count for each column
print(null_values_per_column)
```

```
Position ID                  0
Position Status              0
Time                        10
Time Out                    14
Timecard Hours (as Time)    10
Pay Cycle Start Date        10
Pay Cycle End Date          10
Employee Name                0
File Number                  0
dtype: int64
```

```
#creating backup
original_df = df.copy(deep=True)
```

```
# Create a filter for 'Time Out' column not being null
not_null_time_out_filter = df['Time Out'].notna()

# Update the original DataFrame with the filtered data
df = df[not_null_time_out_filter]
```

```
# Display the filtered DataFrame
df.count()
```

```
Position ID                 1470
Position Status             1470
Time                        1470
Time Out                    1470
Timecard Hours (as Time)    1470
Pay Cycle Start Date        1470
Pay Cycle End Date          1470
Employee Name               1470
File Number                 1470
dtype: int64
```

```
# Display the original DataFrame
original_df.count()
```

```
Position ID                 1484
Position Status             1484
Time                        1474
Time Out                    1470
Timecard Hours (as Time)    1474
Pay Cycle Start Date        1474
Pay Cycle End Date          1474
Employee Name               1484
File Number                 1484
dtype: int64
```

```
# Check null values in each column
null_values_per_column = df.isnull().sum()

# Display the null values count for each column
print(null_values_per_column)
```

```
Position ID                 0
Position Status             0
Time                        0
Time Out                    0
Timecard Hours (as Time)    0
Pay Cycle Start Date        0
Pay Cycle End Date          0
Employee Name               0
File Number                 0
dtype: int64
```

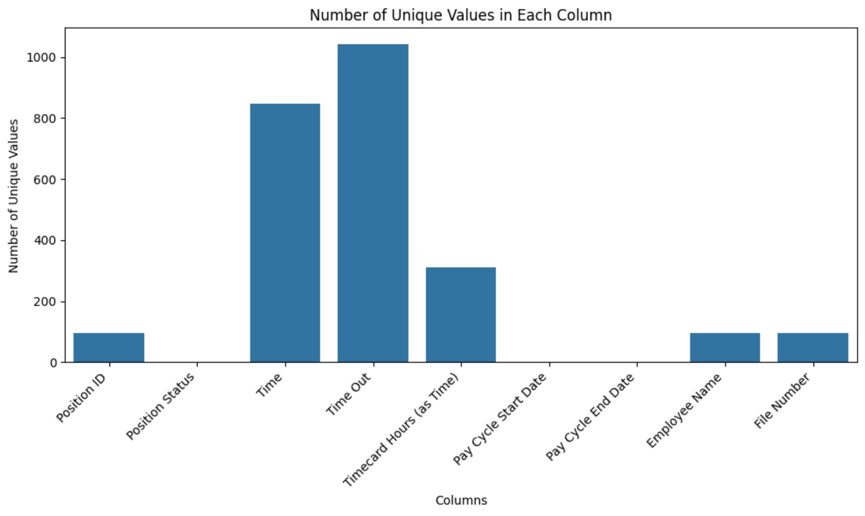
```
df.head(25)
```

5	WFS000065	Active	09-14 10:09:00	09-14 14:30:00	4:21	09-10	09-23	XAis
6	WFS000065	Active	2023-09-14 15:00:00	2023-09-14 19:14:00	4:14	2023-09-10	2023-09-23	REsaXiaWE, XAis
7	WFS000065	Active	2023-09-15 10:11:00	2023-09-15 14:41:00	4:30	2023-09-10	2023-09-23	REsaXiaWE, XAis
8	WFS000065	Active	2023-09-15 15:11:00	2023-09-15 19:05:00	3:54	2023-09-10	2023-09-23	REsaXiaWE, XAis
9	WFS000065	Active	2023-09-16 09:51:00	2023-09-16 14:50:00	4:59	2023-09-10	2023-09-23	REsaXiaWE, XAis
10	WFS000065	Active	2023-09-16 15:20:00	2023-09-16 20:19:00	4:59	2023-09-10	2023-09-23	REsaXiaWE, XAis
11	WFS000065	Active	2023-09-19 09:55:00	2023-09-19 13:56:00	4:01	2023-09-10	2023-09-23	REsaXiaWE, XAis
12	WFS000065	Active	2023-09-19 14:26:00	2023-09-19 18:29:00	4:03	2023-09-10	2023-09-23	REsaXiaWE, XAis
13	WFS000065	Active	2023-09-20 09:54:00	2023-09-20 14:12:00	4:18	2023-09-10	2023-09-23	REsaXiaWE, XAis
14	WFS000065	Active	2023-09-20 14:42:00	2023-09-20 18:46:00	4:04	2023-09-10	2023-09-23	REsaXiaWE, XAis
15	WFS000065	Active	2023-09-21 09:55:00	2023-09-21 14:03:00	4:08	2023-09-10	2023-09-23	REsaXiaWE, XAis
16	WFS000065	Active	2023-09-21 14:33:00	2023-09-21 18:47:00	4:14	2023-09-10	2023-09-23	REsaXiaWE, XAis
17	WFS000065	Active	2023-09-22 09:55:00	2023-09-22 14:50:00	4:55	2023-09-10	2023-09-23	REsaXiaWE, XAis
18	WFS000065	Active	2023-09-22 15:20:00	2023-09-22 20:26:00	5:06	2023-09-10	2023-09-23	REsaXiaWE, XAis
19	WFS000065	Active	2023-09-23 09:55:00	2023-09-23 15:47:00	6:52	2023-09-10	2023-09-23	REsaXiaWE, XAis
			2023-	2023-				

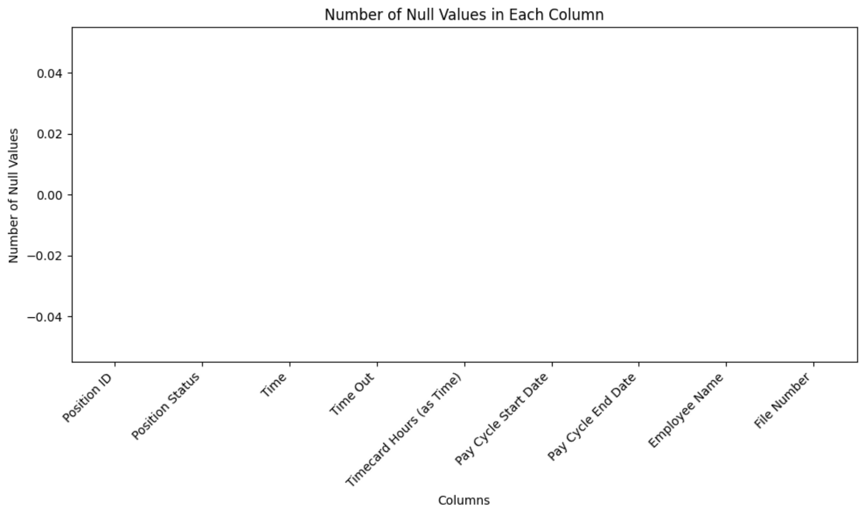
```
import matplotlib.pyplot as plt
import seaborn as sns


# Check unique values and null values
unique_values = df.nunique()
null_values = df.isnull().sum()

# Plot unique values
plt.figure(figsize=(10, 6))
sns.barplot(x=unique_values.index, y=unique_values.values)
plt.title('Number of Unique Values in Each Column')
plt.xlabel('Columns')
plt.ylabel('Number of Unique Values')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```



```
# Plot null values
plt.figure(figsize=(10, 6))
sns.barplot(x=null_values.index, y=null_values.values)
plt.title('Number of Null Values in Each Column')
plt.xlabel('Columns')
plt.ylabel('Number of Null Values')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```



```
# Specify the columns of interest
columns_of_interest = ['Position Status', 'Employee Name', 'File Number']

# Display count and unique values for each specified column
for column in columns_of_interest:
    unique_values = df[column].value_counts()
    print(f"\nColumn: {column}")
    print(f"Total Unique Values: {len(unique_values)}")
    print(unique_values)
```

```
Column: Position Status
Total Unique Values: 1
Active    1470
Name: Position Status, dtype: int64

Column: Employee Name
Total Unique Values: 96
REdrigAez, GraWdEW AWgeX       22
GAeWdia, JAaW CarXEs           22
Xee, XaCar                     22
SiWgXeMEW, REger               21
MraW, WiXXiaC Ha               21
                               ..
CeWdEza, Erik                   7
CharXes, EGadiah SEraccE Jr     5
REdrigAez, AXexis GAMierrez     5
RECerE, SergiE                  3
CarMer, XyWWeXX DejAaW Jr       2
Name: Employee Name, Length: 96, dtype: int64

Column: File Number
Total Unique Values: 96
550    22
473    22
200    22
426    21
345    21
       ..
465     7
566     5
505     5
420     3
576     2
Name: File Number, Length: 96, dtype: int64
```

```
# Filter the DataFrame to include only rows where 'Time' is null
records_with_null_Time = df[df['Time'].isnull()]

# Display the records where 'Time' is null
records_with_null_Time.head(30)
```

Position ID	Position Status	Time	Time Out	Timecard Hours (as Time)	Pay Cycle Start	Pay Cycle End	Employee Name	File Number

```
# Filter the DataFrame to include only rows where 'Time Out' is null
records_with_null_Time_Out = df[df['Time Out'].isnull()]

# Display the records where 'Time Out' is null
records_with_null_Time_Out.head(30)
```

Position ID	Position Status	Time	Time Out	Timecard Hours (as Time)	Pay Cycle Start	Pay Cycle End	Employee Name	File Number

```
# Filter the DataFrame to include only rows where 'Timecard Hours (as Time)' is 0:00
records_with_null_Timecard = df[df['Timecard Hours (as Time)']=='0:00']

# Display the records where 'Timecard Hours (as Time)'' is 0:00
records_with_null_Timecard.head(30)
```

Position ID	Position Status	Time	Time Out	Timecard Hours (as Time)	Pay Cycle Start Date	Pay Cycle End Date	Employee Name	F Nun

```
# Filter the DataFrame to include only rows where 'Pay Cycle Start Date' is null
records_with_null_Pay_Cycle_Start_Date = df[df['Pay Cycle Start Date'].isnull()]

# Display the records where 'Pay Cycle Start Date' is null
records_with_null_Pay_Cycle_Start_Date.head(30)
```

```
                                                     Pay     Pay
                                        Timecard   Cycle   Cycle   Employee     File
   Position  Position   Time    Time   Hours (as   Start     End       Name   Number
         ID    Status            Out       Time)
```

```
# Filter the DataFrame to include only rows where 'Pay Cycle End Date' is null
records_with_null_Pay_Cycle_End_Date = df[df['Pay Cycle End Date'].isnull()]

# Display the records where 'Pay Cycle End Date' is null
records_with_null_Pay_Cycle_End_Date.head(30)
```

```
                                                     Pay     Pay
                                        Timecard   Cycle   Cycle   Employee     File
   Position  Position   Time    Time   Hours (as   Start     End       Name   Number
         ID    Status            Out       Time)
```

```
# Filter the DataFrame to include only rows where 'File Number' is null
records_with_null_File_Number = df[df['File Number'].isnull()]

# Display the records where 'Employee Name' is null
records_with_null_File_Number.head(30)
```

```
                                                     Pay     Pay
                                        Timecard   Cycle   Cycle   Employee     File
   Position  Position   Time    Time   Hours (as   Start     End       Name   Number
         ID    Status            Out       Time)
```

```
# Filter the DataFrame to include only rows where 'Employee Name' is null
records_with_null_Employee_Name = df[df['Employee Name'].isnull()]

# Display the records where 'Employee Name' is null
records_with_null_Employee_Name.head(30)
```

```
                                                     Pay     Pay
                                        Timecard   Cycle   Cycle   Employee     File
   Position  Position   Time    Time   Hours (as   Start     End       Name   Number
         ID    Status            Out       Time)
```

```
# Convert 'Time' column to datetime if not already done
df['Time'] = pd.to_datetime(df['Time'])

# Create a new column 'Date' to store only the date part of 'Time'
df['Date'] = df['Time'].dt.date
```

a) Employees who have worked for 7 consecutive days:

```
import pandas as pd


# Convert the 'Date' column to datetime type
df['Date'] = pd.to_datetime(df['Date'])

# Sort the DataFrame by 'Employee Name' and 'Date'
df = df.sort_values(['Employee Name', 'Date'])

# Calculate the difference between consecutive dates for each employee
df['Days Difference'] = df.groupby('Employee Name')['Date'].diff().dt.days

# Identify employees who have worked for 7 or more consecutive days without any gaps
consecutive_days_condition = (df['Days Difference'] == 1) | (df['Days Difference'].isnull())
consecutive_days = df[consecutive_days_condition].groupby('Employee Name')['Date'].agg(['first', 'last', 'count'])

# Function to exclude potential holidays
def exclude_potential_holidays(row):
    consecutive_dates = pd.date_range(start=row['first'], end=row['last'])

    # Identify potential holidays (dates with a gap)
    potential_holidays = df[(df['Employee Name'] == row.name) & (df['Days Difference'] > 1)]['Date']

    # Update the 'first' date to the first working day after a potential holiday
    for potential_holiday in potential_holidays:
        after_holiday = consecutive_dates[consecutive_dates > potential_holiday]
        if len(after_holiday) > 0:
            row['first'] = after_holiday.min()

    return pd.Series({'first': row['first'], 'last': row['last'], 'count': (row['last'] - row['first']).days + 1})

# Apply the function to exclude potential holidays
filtered_consecutive_days = consecutive_days.apply(exclude_potential_holidays, axis=1)

# Filter only those employees with a count of 7 or more days
result = filtered_consecutive_days[filtered_consecutive_days['count'] >= 6]

# Display the result
print(result)
```

```
                          first       last  count
Employee Name
CEreira Jr, JEse     2023-09-17 2023-09-23      7
GAeWdia, JAaW CarXEs 2023-09-17 2023-09-23      7
Sparks, KeWWeMh      2023-09-17 2023-09-23      7
```

b) Employees with less than 10 hours between shifts but greater than 1 hour:

```
# Convert 'Time' and 'Time Out' columns to datetime if not already done
df['Time'] = pd.to_datetime(df['Time'])
df['Time Out'] = pd.to_datetime(df['Time Out'])

# Calculate the time difference between shifts
shift_time_difference = (df['Time'] - df['Time Out'].shift()).dt.total_seconds() / 3600

# Filter employees with less than 10 hours between shifts but greater than 1 hour
employees_between_shifts = df[(shift_time_difference < 10) & (shift_time_difference > 1)]

# Display the result
print("Employees with less than 10 hours between shifts but greater than 1 hour:")
print(employees_between_shifts[['Employee Name', 'Position Status']].drop_duplicates())
```

```
Employees with less than 10 hours between shifts but greater than 1 hour:
                        Employee Name Position Status
592                  CEreira Jr, JEse          Active
54                   CaMaXaWE, CeghaW          Active
277              De Xa Cerda, IgWaciE          Active
1037  DeXgadiXXE REdarMe, ChrisMiaW S          Active
1075            HaCiXMEW, DeaWMe DevEW          Active
312                  MraW, WiXXiaC Ha          Active
1162         REdrigAez, GraWdEW AWgeX          Active
163                        Xee, XaCar          Active
```

c) Employees who have worked for more than 14 hours in a single shift:

```
# Convert 'Time' and 'Time Out' columns to datetime if not already done
df['Time'] = pd.to_datetime(df['Time'])
df['Time Out'] = pd.to_datetime(df['Time Out'])

# Calculate the duration of each shift
shift_duration = (df['Time Out'] - df['Time']).dt.total_seconds() / 3600

# Filter employees who have worked for more than 14 hours in a single shift
employees_more_than_14_hours = df[shift_duration > 14]

# Display the result
print("Employees who have worked for more than 14 hours in a single shift:")
print(employees_more_than_14_hours[['Employee Name', 'Position Status']].drop_duplicates())
```

```
Employees who have worked for more than 14 hours in a single shift:
```

```
# Filter the DataFrame to include only rows where 'Employee name' is XiWW, JAsMiW
records_with_null_Employee_Name = df[df['Employee Name']=='Arias, FeXipe']

# Display the records where 'Employee name' 'is XiWW, JAsMiW'
records_with_null_Employee_Name.head(100)
```

	Position ID	Position Status	Time	Time Out	Timecard Hours (as Time)	Pay Cycle Start Date	Pay Cycle End Date	Employee Name	File Number
71	WFS000170	Active	2023-09-10 02:00:00	2023-09-10 05:27:00	3:27	2023-09-10	2023-09-23	Arias, FeXipe	170
72	WFS000170	Active	2023-09-10 05:57:00	2023-09-10 08:40:00	2:43	2023-09-10	2023-09-23	Arias, FeXipe	170
73	WFS000170	Active	2023-09-11 00:49:00	2023-09-11 05:18:00	4:29	2023-09-10	2023-09-23	Arias, FeXipe	170
74	WFS000170	Active	2023-09-11 05:48:00	2023-09-11 10:38:00	4:50	2023-09-10	2023-09-23	Arias, FeXipe	170
75	WFS000170	Active	2023-09-13 01:25:00	2023-09-13 05:04:00	3:39	2023-09-10	2023-09-23	Arias, FeXipe	170