```python
""" Udacity Programming for Data Scienarchive
    Create: 30-May 2019 Last Mod: 16-June 2019
    Python file name: bikeshare.py
Onport Libary    """
import time
import pandas as pd
import numpy as np

""" Data definition  """
CITY_DATA  = { 'Chicago': 'chicago.csv',
              'New York City': 'new_york_city.csv',
              'Washington': 'washington.csv' }
CITIES     = ['Chicago', 'New York City', 'Washington']
MONTHS     = ['All','January', 'February', 'March', 'April', 'May', 'June']
DAYS       = ['All','Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday' ]
sleeptime  = 2 #secounds
""" Progamm  Start  """
def get_filters():
    """  Input User  city, month, and day to analyze.
         Output (str) city, (str) month  (str) day   """
    print('Explore some US bikeshare data')
        # Input City
    print('Which data of city like you: ',CITIES ,'?')
    city=input('City? ').title()
    while city not in CITIES: city=input('Please again input City: ')
        # Input Month
    print('Which month? ', MONTHS ,'?')
    month=input('Month? ').title()
    while month not in MONTHS:  month=input('Please again input Month: ')
         # Input Day
    print('Which day ', DAYS, '?' )
    day=input('Day? ').title()
    while day not in DAYS:  day=input('Please again input Day:')

    print('-'*48)
    print('Now we will analyze US bikeshare data for')
    print('City: ' ,city, 'in month: ',month , 'on days: ', day)
    print('-'*48)
    return city, month, day

def load_data(city, month, day):
    """
   Inpit  data city, month, day from get_filters():
   Output df - Pandas DataFrame    """

    df = pd.read_csv(CITY_DATA[city])
    df['Start Time'] = pd.to_datetime(df['Start Time'])

    df['month'] = df['Start Time'].dt.month
    df['day_of_week'] = df['Start Time'].dt.weekday_name
        # Filter by Month
    if month != 'All':
        months = ['January', 'February', 'March', 'April', 'May', 'June']
        month = months.index(month) + 1
        df = df[df['month'] == month]
        # Filter by Day
    if day != 'All':
        df = df[df['day_of_week'] == day.title()]
    return df

def time_stats(df,city,month,day):
    """ Input data city, month, day from get_filters(); df from load_data()
        Output Displays statistics on the most frequent times of travel."""
    print ('\nCalculating The Most Frequent Times of Travel')
    print("for City, Month, Day:  %s ,%s ,%s"%(city,month,day))
    start_time = time.time()
        # Display the most common month
    common_month=df['month'].value_counts().head(1)
    print("\ncommon month and count:   %s ."%(common_month))
           # Display the most common day of week
    common_day=df['day_of_week'].value_counts().head(1)
    print("\ncommon day and count:   %s"%(common_day))
         # Display the most common start hour
    df['hour']=df['Start Time'].dt.hour
    common_hour=df['hour'].value_counts().head(1)
    print("\ncommon hour and cout:   %s"%(common_hour))

    print("\nThis took %s seconds." % (time.time() - start_time))
    print('-'*40)

def station_stats(df,city,month,day):
    """
    Input df from  load_data() city,month,day from get_filters()
    Output Displays statistics on the most popular stations and trip."""

    print( '\nCalculating The Most Popular Stations and Trip')
    print("for City, Month, Day:  %s ,%s ,%s"%(city,month,day))
    start_time = time.time()
        # Display most commonly used start station
    common_start_station=df['Start Station'].value_counts().head(1)
    print("\ncommon start station and count: %s"%(common_start_station))
```

```
          # display most commonly used end station
    common_end_station=df['End Station'].value_counts().head(1)
    print("\ncommon start station and count: %s"%(common_end_station))
        #  display most frequent combination of start station and end station trip
    df_group=df['Start Station']+df['End Station']
    frequent_combination=df_group.value_counts().head(1)
    print("\nthe most frequent combination:%s"%(frequent_combination))

    print("\nThis took %s seconds." % (time.time() - start_time))
    print('-'*40)

def trip_duration_stats(df,city,month,day):
    """
    Input: df from  load_data() city,month,day from
    Output: Displays statistics on the total and average trip duration."""
    print('\nCalculating Trip Duration')
    print("for City, Month, Day:  %s ,%s ,%s"%(city,month,day))
    start_time = time.time()
        # Display total travel time
    total_time=df['Trip Duration'].sum()
    print("\ntotal travel time:%s's"%(total_time))
        # Display mean travel time
    mean_time=df['Trip Duration'].mean()
    print("\nmean travel time:%s's"%(mean_time))

    print("\nThis took %s seconds." % (time.time() - start_time))
    label: end
    print('-'*40)

def user_short (df,city,month,day):
    """ Input df from load_data()
        Output: Displays statistics on bikeshare users."""
    # Data of with out gender burth information
    print('\nCalculating User Stats')
    print("for City, Month, Day:  %s ,%s ,%s"%(city,month,day))
    start_time = time.time()
         # Display counts of user types
    user_types=df['User Type'].value_counts()
    print("\ncounts of user types:%s"%(user_types))
         # Display earliest, most recent, and most common year of birth
    print ('\nSory. Data about gender/birth are \nnot avabile for '+ city)

    print("\nThis took %s seconds." % (time.time() - start_time))
    print('-'*40)
```

```python

def user_stats(df,city,month,day):
    """ Input df from load_data()
        Output: Displays statistics on bikeshare users."""
    print('\nCalculating User Stats')
    print("for City, Month, Day:  %s ,%s ,%s"%(city,month,day))
    start_time = time.time()
         # Display counts of user types
    user_types=df['User Type'].value_counts()
    print("\ncounts of user types:%s"%(user_types))
         # Display counts of gender
    count_gender=df['Gender'].value_counts()
    print("\ncounts of gender:%s"%(count_gender))
        # Display earliest, most recent, and most common year of birth
    earliest=df['Birth Year'].min()
    most_recent=df['Birth Year'].max()
    most_commmon=df['Birth Year'].mode()

    print("\nthe earliest year:%s"%(earliest))
    print("the most recent year:%s"%(most_recent))
    print("the most common year:%s"%(most_commmon))

    print("\nThis took %s seconds." % (time.time() - start_time))
    print('-'*40)

def   view_raw_data(df,city,month,day):
    """
    Input: df
    Output: Displays the raw datapython  """
    print('\nView Raw Data')

    df = df.drop(['month', 'day_of_week'], axis = 1)
    rowIndex = 0
    seeData = input("\nWould you see the raw data of stats?\nPlease write [y] Yes [n] No: ").lower()
    while True:
        if seeData == 'n':
            return
        if seeData == 'y':
            print(df[rowIndex: rowIndex + 5])
            rowIndex = rowIndex + 5
        seeData = input("\nWould you see next five more rows?\nPlease write [y] Yes [n] No: ").lower()

def main():
    while True:
        city, month, day = get_filters()
        df = load_data(city, month, day)


        time_stats(df,city,month,day)
        time.sleep(sleeptime)
        station_stats(df,city,month,day)
        time.sleep(sleeptime)

        trip_duration_stats(df,city,month,day)
        time.sleep(sleeptime)
            # sort solution city =='Washington' have no gender data
        if city =='Washington':
            user_short (df,city,month,day)
        if city !='Washington':
            user_stats(df,city,month,day)

        view_raw_data(df,city,month,day)
        print('-'*40)

        restart = input('\nWould you like to restart?\nEnter [y] Yes [n] No: ')
        if restart.lower() != 'y':  break

if __name__ == "__main__":
    main()

```