In [ ]:

```
import time
import pandas as pd
import numpy as np
import calendar

CITY_DATA = { 'chicago': 'chicago.csv',
              'new york city': 'new_york_city.csv',
              'washington': 'washington.csv' }

def get_filters():
    """
    Asks user to specify a city, month, and day to analyze.

    Returns:
        (str) city - name of the city to analyze
        (str) month - name of the month to filter by, or "all" to apply no month fi
        (str) day - name of the day of week to filter by, or "all" to apply no day
    """
    print('Hello! Let\'s explore some US bikeshare data!')
    # get user input for city (chicago, new york city, washington). HINT: Use a wh
    while True:
        cities = ['chicago', 'new york city', 'washington']
        city = input("enter the city from (chicago, new york city, washington): ")
        if city not in cities:
            print('please enter a valid city name from chicago, new york city or wa
        else:
            break

    # get user input for month (all, january, february, ... , june)
    months = ['january', 'february', 'march', 'april', 'may', 'june', 'all']
    months_abbr = ['jan', 'feb', 'mar', 'apr', 'may', 'jun']
    while True:
        month = input('Enter the month\'s name (all, january, february, ... , june)
        if month not in months + months_abbr:
            print('please enter a valid month\'s name from the specific range or al
        else:
            if month in months_abbr:
                month = months[months_abbr.index(month)]
            break

    # get user input for day of week (all, monday, tuesday, ... sunday)
    days = [x.lower() for x in list(calendar.day_name)]
    days.append('all')
    days_abbr = [x.lower() for x in list(calendar.day_abbr)]

    while True:
        day = input('Enter the day\'s name (all, monday, tuesday, ... sunday): ').l
        if day not in days + days_abbr:
            print('please enter a valid day\'s name or all')
        else:
            if day in days_abbr:
                day = days[days_abbr.index(day)]
            break

    print('-'*40)
    return city, month, day


def load_data(city, month, day):
    """
    Loads data for the specified city and filters by month and day if applicable.
```

```
    Args:
        (str) city - name of the city to analyze
        (str) month - name of the month to filter by, or "all" to apply no month fi
        (str) day - name of the day of week to filter by, or "all" to apply no day
    Returns:
        df - Pandas DataFrame containing city data filtered by month and day
    """

    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.day_name()


    if month != 'all':
        months = ['january', 'february', 'march', 'april', 'may', 'june']
        month = months.index(month.lower()) + 1
        df = df[df['month'] == month]

    if day != 'all':
        df = df[df['day_of_week'] == day.title()]

    return df


def time_stats(df):
    """Displays statistics on the most frequent times of travel."""

    print('\nCalculating The Most Frequent Times of Travel...\n')
    start_time = time.time()

    # display the most common month
    months = ['january', 'february', 'march', 'april', 'may', 'june']
    most_month = months[df['month'].mode()[0]-1]
    print(f'The most common month is {most_month}')

    # display the most common day of week
    most_day = df['day_of_week'].mode()[0]
    print(f'The most common day is {most_day}')


    # display the most common start hour
    df['hour'] = df['Start Time'].dt.hour
    most_hour = df['hour'].mode()[0]
    print(f'The most common hour is {most_hour}')

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


def station_stats(df):
    """Displays statistics on the most popular stations and trip."""

    print('\nCalculating The Most Popular Stations and Trip...\n')
    start_time = time.time()

    # display most commonly used start station
    most_station = df['Start Station'].mode()[0]
    print(f'The most common start station is {most_station}')

    # display most commonly used end station
    most_end_station = df['End Station'].mode()[0]
    print(f'The most common end station is {most_end_station}')
```

```
    # display most frequent combination of start station and end station trip
    most_start_end_stations = df.groupby(['End Station','Start Station']).size().ic
    print(f'The most common start-end stations are {most_start_end_stations[0]} and

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


def trip_duration_stats(df):
    """Displays statistics on the total and average trip duration."""

    print('\nCalculating Trip Duration...\n')
    start_time = time.time()

    # display total travel time
    total_trip = df['Trip Duration'].sum()
    print(f'The total travel time is: {total_trip}')
    # display mean travel time
    avg_trip = df['Trip Duration'].mean()
    print(f'The average travel time is: {avg_trip}')

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


def user_stats(df):
    """Displays statistics on bikeshare users."""

    print('\nCalculating User Stats...\n')
    start_time = time.time()

    # Display counts of user types
    user_count = df['User Type'].value_counts().to_string()
    print(f'The counts of user types is:\n {user_count}')

    # Display counts of gender
    # The washington.csv does not have a gender column
    if 'Gender' in df:
        gender_count = df['Gender'].value_counts().to_string()
        print(f'The counts of user genders is:\n {gender_count}')

    # Display earliest, most recent, and most common year of birth
    if 'Birth Year' in df.columns:
        early = int(df['Birth Year'].min())
        print(f'The earliest year of birth is: {early}')

        recent = int(df['Birth Year'].max())
        print(f'The recent year of birth is: {recent}')

        most = int(df['Birth Year'].mode()[0])
        print(f'The most frequent year of birth is: {most}')

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

def show_file(df):
    answer = input('Do you want to show data? ').lower().strip()
    if answer == 'yes':
        count = 0
        while True:
            answer = input('Do you wnat to show 5 rows of data or how many? ').lowe
            if answer == 'yes':
                count += 5
                print(df.iloc[count-5: count])
```

```
            try:
                count += int(answer)
                print(df.iloc[count-int(answer): count])
            except:
                if answer == 'no':
                    break
                else:
                    print('please enter a valid answer. ')


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

        time_stats(df)
        station_stats(df)
        trip_duration_stats(df)
        user_stats(df)
        show_file(df)

        restart = input('\nWould you like to restart? Enter yes or no.\n')
        if restart.lower() != 'yes':
            break

if __name__ == "__main__":
        main()
```

```
Hello! Let's explore some US bikeshare data!
enter the city from (chicago, new york city, washington): chicago
Enter the month's name (all, january, february, ... , june): jan
Enter the day's name (all, monday, tuesday, ... sunday): sun
----------------------------------------

Calculating The Most Frequent Times of Travel...

The most common month is january
The most common day is Sunday
The most common hour is 13

This took 0.01563096046447754 seconds.
----------------------------------------

Calculating The Most Popular Stations and Trip...

The most common start station is McClurg Ct & Illinois St
The most common end station is Wabash Ave & Roosevelt Rd
The most common start-end stations are Clark St & Armitage Ave and Wells St & Conc
ord Ln

This took 0.0 seconds.
----------------------------------------

Calculating Trip Duration...

The total travel time is: 1373037
The average travel time is: 732.2864

This took 0.0 seconds.
----------------------------------------

Calculating User Stats...

The counts of user types is:
 Subscriber    1643
Customer       232
The counts of user genders is:
 Male      1250
Female     393
The earliest year of birth is: 1945
The recent year of birth is: 2000
The most frequent year of birth is: 1989

This took 0.0 seconds.
----------------------------------------
Do you want to show data? yes
Do you wnat to show 5 rows of data or how many? 5
     Unnamed: 0          Start Time             End Time  Trip Duration  \
11        71678 2017-01-22 15:15:45  2017-01-22 15:31:02            917
12        19061 2017-01-08 16:03:00  2017-01-08 16:07:37            277
120        1647 2017-01-01 21:06:09  2017-01-01 21:10:37            268
143       18745 2017-01-08 13:23:19  2017-01-08 13:28:52            333
234        1188 2017-01-01 16:09:40  2017-01-01 16:16:08            388

                      Start Station                  End Station   User Type  \
11   Southport Ave & Wellington Ave       Clark St & Schiller St  Subscriber
12            Green St & Madison St     Ada St & Washington Blvd  Subscriber
120       Sedgwick St & Webster Ave  Halsted St & Wrightwood Ave  Subscriber
143          State St & Harrison St           Wells St & Polk St  Subscriber
234    Cornell Ave & Hyde Park Blvd      Greenwood Ave & 47th St  Subscriber

    Gender  Birth Year  month day_of_week  hour
```

```
11    Male      1964.0      1      Sunday    15
12    Male      1961.0      1      Sunday    16
120   Male      1984.0      1      Sunday    21
143   Male      1954.0      1      Sunday    13
234   Male      1966.0      1      Sunday    16
Do you wnat to show 5 rows of data or how many? 2
     Unnamed: 0          Start Time             End Time  Trip Duration  \
409       71771 2017-01-22 15:38:09  2017-01-22 15:42:32            263
415       39985 2017-01-15 12:22:38  2017-01-15 12:36:17            819

                  Start Station                 End Station   User Type  \
409    Racine Ave & Belmont Ave  Halsted St & Diversey Pkwy  Subscriber
415  Indiana Ave & Roosevelt Rd              Burnham Harbor  Subscriber

    Gender  Birth Year  month day_of_week  hour
409   Male      1986.0      1      Sunday    15
415   Male      1993.0      1      Sunday    12
```

In [ ]: