```
#Task 1

import pandas as pd

class BookLover:
    def __init__(self, name, email, fav_genre, num_books=0, book_list=None):
        self.name = name                  # Name of the person
        self.email = email                # Unique email identifier
        self.fav_genre = fav_genre        # Favorite book genre
        self.num_books = num_books        # Number of books read
        self.book_list = book_list if book_list is not None else pd.DataFrame({'book_name': [], 'book_rating': []})

    # Method 1: Add a book if it doesn't already exist in book_list
    def add_book(self, book_name, rating):
        if self.book_list['book_name'].eq(book_name).any():
            print(f"{book_name} is already in the book list.")
        else:
            new_book = pd.DataFrame({'book_name': [book_name], 'book_rating': [rating]})
            self.book_list = pd.concat([self.book_list, new_book], ignore_index=True)
            self.num_books += 1

    # Method 2: Check if the person has read a particular book
    def has_read(self, book_name):
        return (self.book_list['book_name'] == book_name).any()

    # Method 3: Return the total number of books read
    def num_books_read(self):
        return self.num_books

    # Method 4: Return a filtered list of favorite books with ratings > 3
    def fav_books(self):
        return self.book_list[self.book_list['book_rating'] > 3]

# Testing the class
if __name__ == '__main__':
    test_object = BookLover("Han Solo", "hsolo@millenniumfalcon.com", "scifi")
    test_object.add_book("War of the Worlds", 4)
    test_object.add_book("1984", 5)
    test_object.add_book("War of the Worlds", 4)  # Should not add again
    print("Books read:", test_object.num_books_read())
    print("Has read '1984':", test_object.has_read("1984"))
    print("Favorite books:0, test_object.fav_books())



#Task 2

import unittest
from booklover import BookLover
import pandas as pd

class BookLoverTestSuite(unittest.TestCase):

    def test_1_add_book(self):
        # Create a BookLover instance and add a book
        book_lover = BookLover("Tom", "tom@gmail.com", "fiction")
        book_lover.add_book("Hunger Games", 5)

        # Test if the book was added to book_list
        self.assertTrue("Hunger Games" in book_lover.book_list['book_name'].values)

    def test_2_add_book_twice(self):
        # Create a BookLover instance and add the same book twice
        book_lover = BookLover("Tom", "tom@gmail.com", "fiction")
        book_lover.add_book("Hunger Games", 5)
        book_lover.add_book("Hunger Games", 5)  # Attempt to add the same book again

        # Test that the book is only in book_list once
        self.assertEqual(book_lover.book_list[’book_name’].value_counts().get("Hunger Games", 0), 1)

    def test_3_has_read(self):
        # Create a BookLover instance and add a book
        book_lover = BookLover("Tom", "tom@gmail.com", "fiction")
        book_lover.add_book("Hunger Games", 5)

        # Test if has_read returns True for the book
        self.assertTrue(book_lover.has_read("Hunger Games"))

    def test_4_has_not_read(self):
        # Create a BookLover instance without adding any books
        book_lover = BookLover("Tom", "tom@gmail.com", "fiction")

        # Test if has_read returns False for a book not in the list
        self.assertFalse(book_lover.has_read("The Great Gatsby"))

    def test_5_num_books_read(self):
        # Create a BookLover instance and add books
        book_lover = BookLover("Tom", "tom@gmail.com", "fiction")
        book_lover.add_book("Hunger Games", 5)
        book_lover.add_book("To Kill a Mockingbird", 4)
        book_lover.add_book("The Catcher in the Rye", 3)

        # Test if num_books matches the expected count
        self.assertEqual(book_lover.num_books_read(), 3)

    def test_6_fav_books(self):
        # Create a BookLover instance and add books with various ratings
        book_lover = BookLover("Tom", "tom@gmail.com", "fiction")
        book_lover.add_book("Hunger Games", 5)
        book_lover.add_book("To Kill a Mockingbird", 4)
        book_lover.add_book("The Catcher in the Rye", 3)
        book_lover.add_book("The Great Gatsby", 2)

        # Get favorite books with rating > 3
        fav_books = book_lover.fav_books()

        # Test if the favorite books have rating > 3
        self.assertTrue(all(fav_books[’book_rating’] > 3))

if __name__ == ’__main__’:
    unittest.main(verbosity=3)


#Task 3

test_1_add_book (__main__.BookLoverTestSuite) ... ok
test_2_add_book_twice (__main__.BookLoverTestSuite) ... ok
test_3_has_read (__main__.BookLoverTestSuite) ... ok
test_4_has_not_read (__main__.BookLoverTestSuite) ... ok
test_5_num_books_read (__main__.BookLoverTestSuite) ... ok
test_6_fav_books (__main__.BookLoverTestSuite) ... ok

----------------------------------------------------------------------
Ran 6 tests in 0.018s

OK
```