TheBetterIndia - Assignment

Problem statement :

To create an ETL pipeline with following capabilities:

1. Load a .CSV file in mysql.
2. Load another related .CSV file in postgres.
3. Combine data from both of the above tables and load into another table.

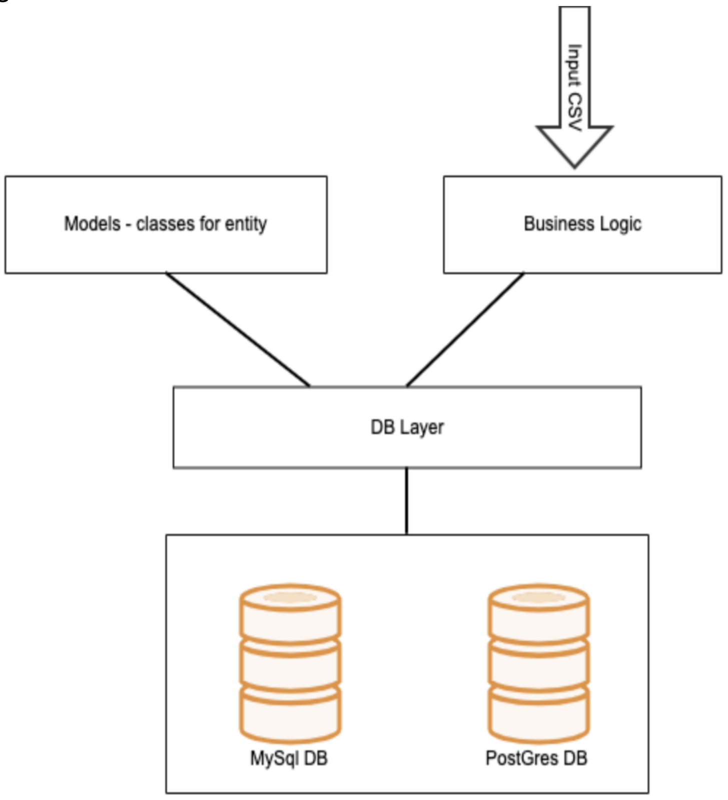
Solution :

Data Source : [https://www.kaggle.com/jirakst/bookcrossing (https://www.kaggle.com/jirakst/bookcrossing)](https://www.kaggle.com/jirakst/bookcrossing)

Data Description: The above folder contains following three files:

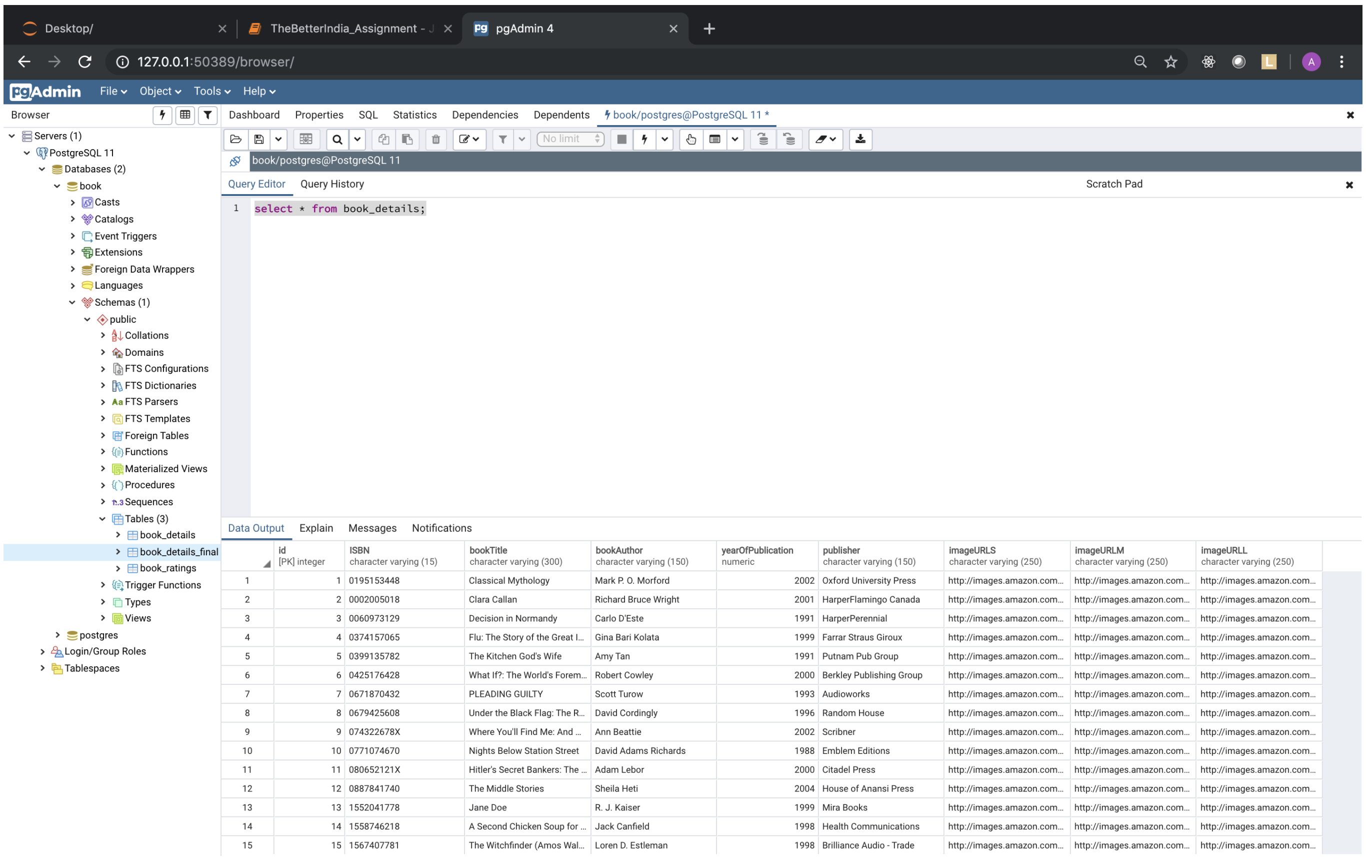
1. BX-Books.CSV : This file contains information related to books. I will load this file into postgres table.
2. BX-Books-Ratings.CSV : This file contains ratings provided by different users to different books. The range of rating is [0,10]. It is a huge file and i will load this file into mysql DB.
3. BX-Users.CSV : This file contain information related to users. I will ignore this file.

High Level Architecture Diagram:



Output after the execution of the code:

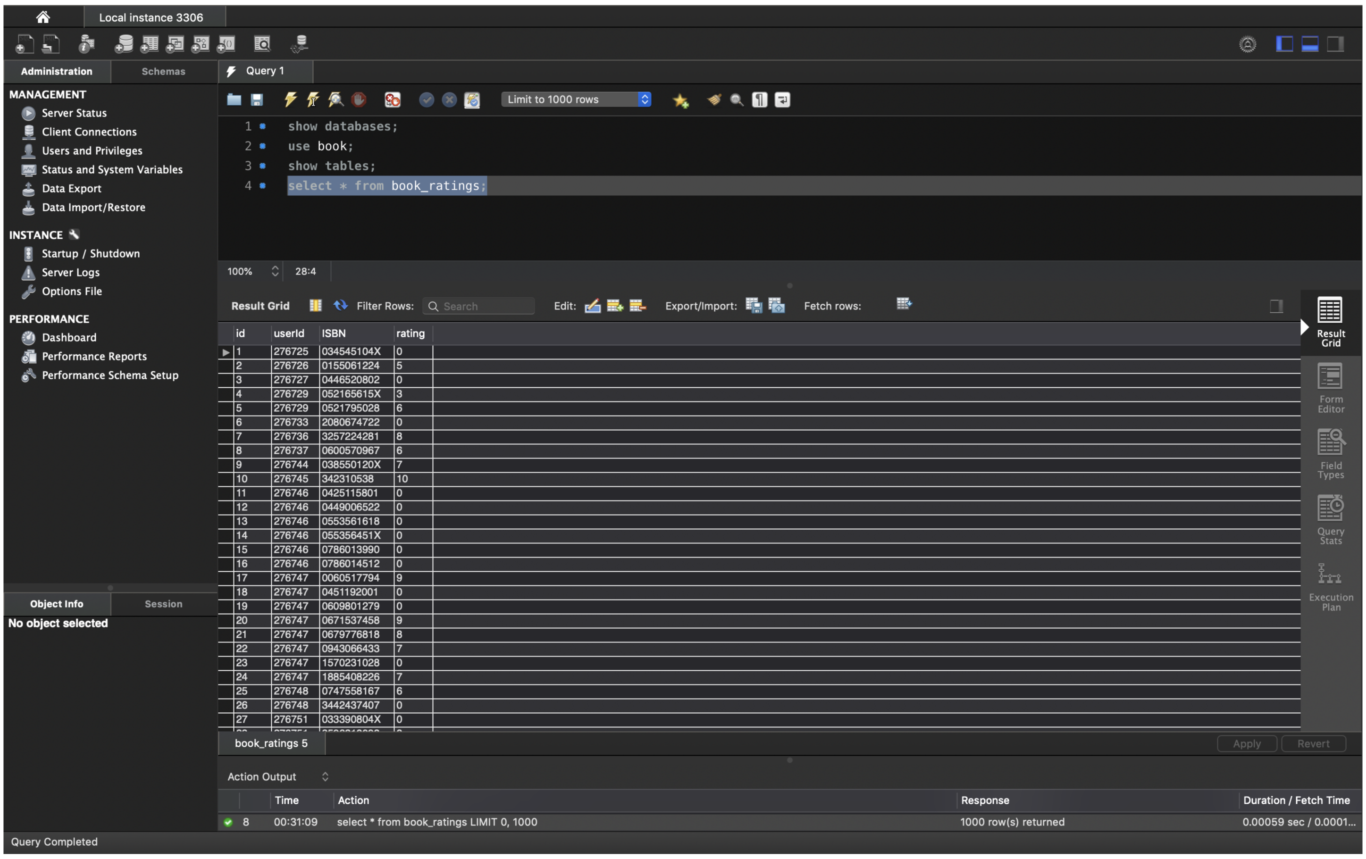
Output from book_details table:



```
select * from book_details;
```

	id [PK] integer	ISBN character varying (15)	bookTitle character varying (300)	bookAuthor character varying (150)	yearOfPublication numeric	publisher character varying (150)	imageURLS character varying (250)	imageURLM character varying (250)	imageURLL character varying (250)
1	1	0195153448	Classical Mythology	Mark P. O. Morford	2002	Oxford University Press	http://images.amazon.com...	http://images.amazon.com...	http://images.amazon.com...
2	2	0002005018	Clara Callan	Richard Bruce Wright	2001	HarperFlamingo Canada	http://images.amazon.com...	http://images.amazon.com...	http://images.amazon.com...
3	3	0060973129	Decision in Normandy	Carlo D'Este	1991	HarperPerennial	http://images.amazon.com...	http://images.amazon.com...	http://images.amazon.com...
4	4	0374157065	Flu: The Story of the Great I...	Gina Bari Kolata	1999	Farrar Straus Giroux	http://images.amazon.com...	http://images.amazon.com...	http://images.amazon.com...
5	5	0399135782	The Kitchen God's Wife	Amy Tan	1991	Putnam Pub Group	http://images.amazon.com...	http://images.amazon.com...	http://images.amazon.com...
6	6	0425176428	What If?: The World's Forem...	Robert Cowley	2000	Berkley Publishing Group	http://images.amazon.com...	http://images.amazon.com...	http://images.amazon.com...
7	7	0671870432	PLEADING GUILTY	Scott Turow	1993	Audioworks	http://images.amazon.com...	http://images.amazon.com...	http://images.amazon.com...
8	8	0679425608	Under the Black Flag: The R...	David Cordingly	1996	Random House	http://images.amazon.com...	http://images.amazon.com...	http://images.amazon.com...
9	9	074322678X	Where You'll Find Me: And ...	Ann Beattie	2002	Scribner	http://images.amazon.com...	http://images.amazon.com...	http://images.amazon.com...
10	10	0771074670	Nights Below Station Street	David Adams Richards	1988	Emblem Editions	http://images.amazon.com...	http://images.amazon.com...	http://images.amazon.com...
11	11	080652121X	Hitler's Secret Bankers: The ...	Adam Lebor	2000	Citadel Press	http://images.amazon.com...	http://images.amazon.com...	http://images.amazon.com...
12	12	0887841740	The Middle Stories	Sheila Heti	2004	House of Anansi Press	http://images.amazon.com...	http://images.amazon.com...	http://images.amazon.com...
13	13	1552041778	Jane Doe	R. J. Kaiser	1999	Mira Books	http://images.amazon.com...	http://images.amazon.com...	http://images.amazon.com...
14	14	1558746218	A Second Chicken Soup for ...	Jack Canfield	1998	Health Communications	http://images.amazon.com...	http://images.amazon.com...	http://images.amazon.com...
15	15	1567407781	The Witchfinder (Amos Wal...	Loren D. Estleman	1998	Brilliance Audio - Trade	http://images.amazon.com...	http://images.amazon.com...	http://images.amazon.com...

Output from book_ratings table:



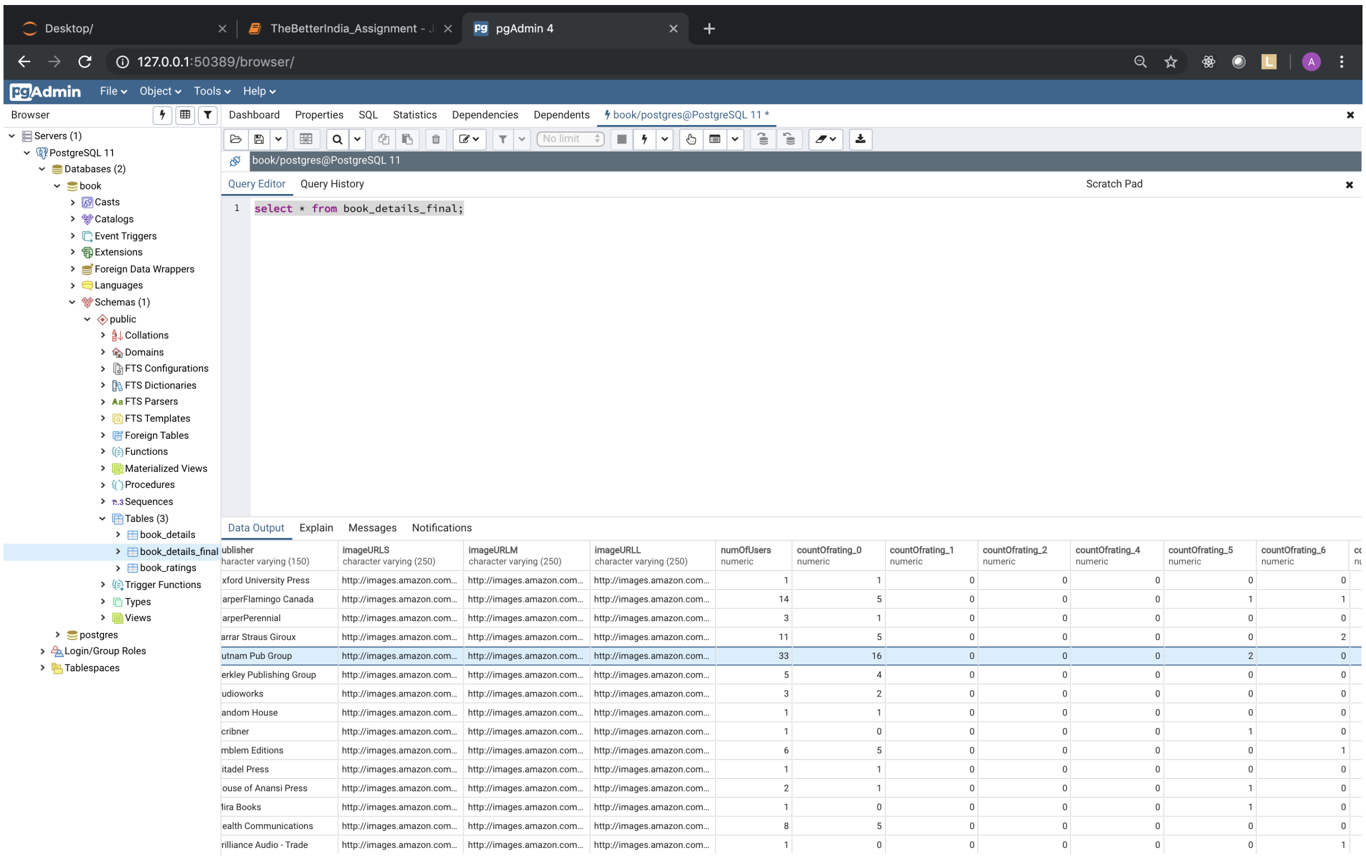
```
show databases;
use book;
show tables;
select * from book_ratings;
```

id	userId	ISBN	rating
1	276725	034545104X	0
2	276726	0155061224	5
3	276727	0446520802	0
4	276729	052165615X	3
5	276729	0521795028	6
6	276733	2080674722	0
7	276736	3257224281	8
8	276737	0600570967	6
9	276744	038550120X	7
10	276745	342310538	10
11	276746	0425115801	0
12	276746	0449006522	0
13	276746	0553561618	0
14	276746	055356451X	0
15	276746	0786013990	0
16	276746	0786014512	0
17	276747	0060517794	9
18	276747	0451192001	0
19	276747	0609801279	0
20	276747	0671537458	9
21	276747	0679776818	8
22	276747	0943066433	7
23	276747	1570231028	0
24	276747	1885408226	7
25	276748	0747558167	6
26	276748	3442437407	0
27	276751	033390804X	0

```
     Time      Action                                    Response                 Duration / Fetch Time
✓ 8  00:31:09  select * from book_ratings LIMIT 0, 1000  1000 row(s) returned     0.00059 sec / 0.0001...
```

Query Completed

Output from book_detsils_final table:



```
select * from book_details_final;
```

ublisher haracter varying (150)	imageURLS character varying (250)	imageURLM character varying (250)	imageURLL character varying (250)	numOfUsers numeric	countOfrating_0 numeric	countOfrating_1 numeric	countOfrating_2 numeric	countOfrating_4 numeric	countOfrating_5 numeric	countOfrating_6 numeric
xford University Press	http://images.amazon.com...	http://images.amazon.com...	http://images.amazon.com...	1	1	0	0	0	0	0
arperFlamingo Canada	http://images.amazon.com...	http://images.amazon.com...	http://images.amazon.com...	14	5	0	0	0	1	1
arperPerennial	http://images.amazon.com...	http://images.amazon.com...	http://images.amazon.com...	3	1	0	0	0	0	0
arrar Straus Giroux	http://images.amazon.com...	http://images.amazon.com...	http://images.amazon.com...	11	5	0	0	0	0	2
utnam Pub Group	http://images.amazon.com...	http://images.amazon.com...	http://images.amazon.com...	33	16	0	0	0	2	0
erkley Publishing Group	http://images.amazon.com...	http://images.amazon.com...	http://images.amazon.com...	5	4	0	0	0	0	0
udioworks	http://images.amazon.com...	http://images.amazon.com...	http://images.amazon.com...	3	2	0	0	0	0	0
andom House	http://images.amazon.com...	http://images.amazon.com...	http://images.amazon.com...	1	1	0	0	0	0	0
cribner	http://images.amazon.com...	http://images.amazon.com...	http://images.amazon.com...	1	0	0	0	0	1	0
mblem Editions	http://images.amazon.com...	http://images.amazon.com...	http://images.amazon.com...	6	5	0	0	0	0	1
itadel Press	http://images.amazon.com...	http://images.amazon.com...	http://images.amazon.com...	1	1	0	0	0	0	0
ouse of Anansi Press	http://images.amazon.com...	http://images.amazon.com...	http://images.amazon.com...	2	1	0	0	0	1	0
lira Books	http://images.amazon.com...	http://images.amazon.com...	http://images.amazon.com...	1	0	0	0	0	1	0
ealth Communications	http://images.amazon.com...	http://images.amazon.com...	http://images.amazon.com...	8	5	0	0	0	0	0
rilliance Audio - Trade	http://images.amazon.com...	http://images.amazon.com...	http://images.amazon.com...	1	0	0	0	0	0	1

Code description:

1. DBLayer : This package have code for creating session using SQLAlchemy ORM.

2. Model: This package holds all the models for different entity like books , ratings and final book.

3. Main class: Holds all the business logic.

4. Input_files: Having input data file

5. bad_data_files: Having files which conatins data which is not in the format.

1. DBLayer

In [1]:

```
from sqlalchemy import create_engine
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import sessionmaker
import json


Base = declarative_base()


def make_db_uri(db_type):
        db_config_file = open('DB_config.json', 'r')
        if db_config_file.mode == 'r':
            db_config_data = json.load(db_config_file)
            if db_type == 'mysql':
                JsonObj = db_config_data['mysql']
                host = JsonObj['host']
                port = JsonObj['port']
                username = JsonObj['userName']
                password = JsonObj['password']
                dbName = JsonObj['dbName']
                db_uri = "mysql+pymysql://%s:%s@%s:%s/%s" % (username,password,h
ost,port,dbName)
                return db_uri
            elif db_type == 'postgres':
                JsonObj = db_config_data['postgres']
                host = JsonObj['host']
                port = JsonObj['port']
                username = JsonObj['userName']
                password = JsonObj['password']
                dbName = JsonObj['dbName']
                db_uri = "postgres+psycopg2://%s:%s@%s:%s/%s" % (username,passwo
rd,host,port,dbName)
                return db_uri
            else:
                raise ValueError(db_type)


def session_factory(db_type):
    if db_type == 'mysql':
        engine_mysql = create_engine(make_db_uri(db_type))
        _SessionFactory = sessionmaker(bind=engine_mysql)
        Base.metadata.create_all(engine_mysql)
        return _SessionFactory()
    elif db_type == 'postgres':
        engine_postgres = create_engine(make_db_uri(db_type))
        _SessionFactory = sessionmaker(bind=engine_postgres)
        Base.metadata.create_all(engine_postgres)
        return _SessionFactory()


#used for testing
#print(make_db_uri('postgres'))
#print(make_db_uri('mysql'))
#print(session_factory('postgres'))
#print(session_factory('mysql'))
```

2. Models:

In [21]:

```
# model for book

from sqlalchemy import Column, String, Integer, Numeric
from DBLayer.DBConnection import Base


class Book(Base):
    __tablename__ = 'book_details'
    id = Column(Integer, primary_key=True)
    ISBN = Column(String(15))
    bookTitle = Column(String(300))
    bookAuthor = Column(String(150))
    yearOfPublication = Column(Numeric)
    publisher = Column(String(150))
    imageURLS = Column(String(250))
    imageURLM = Column(String(250))
    imageURLL = Column(String(250))

    def __init__(self, ISBN, bookTitle, bookAuthor, yearOfPublication, publisher
, imageURLS, imageURLM, imageURLL):
        self.ISBN = ISBN
        self.bookTitle = bookTitle
        self.bookAuthor = bookAuthor
        self.yearOfPublication = yearOfPublication
        self.publisher = publisher
        self.imageURLS = imageURLS
        self.imageURLM = imageURLM
        self.imageURLL = imageURLL
```

In [22]:

```
#model for rating

from sqlalchemy import Column, String, Date, Integer, Numeric
from DBLayer.DBConnection import Base


class Rating(Base):
    __tablename__ = 'book_ratings'
    id = Column(Integer, primary_key=True)
    userId = Column(Numeric)
    ISBN = Column(String(15))
    rating = Column(Numeric)

    def __init__(self, userId, ISBN, rating):
        self.userId = userId
        self.ISBN = ISBN
        self.rating = rating
```

In [23]:

```
# Model for final - Combination of books with ratings

from sqlalchemy import Column, String, Date, Integer, Numeric
from DBLayer.DBConnection import Base


class BookFinal(Base):
    __tablename__ = 'book_details_final'

    id = Column(Integer, primary_key=True)
    ISBN = Column(String(15))
    bookTitle = Column(String(300))
    bookAuthor = Column(String(150))
    yearOfPublication = Column(Numeric)
    publisher = Column(String(150))
    imageURLS = Column(String(250))
    imageURLM = Column(String(250))
    imageURLL = Column(String(250))
    numOfUsers = Column(Numeric)
    countOfrating_0 = Column(Numeric)
    countOfrating_1 = Column(Numeric)
    countOfrating_2 = Column(Numeric)
    countOfrating_4 = Column(Numeric)
    countOfrating_5 = Column(Numeric)
    countOfrating_6 = Column(Numeric)
    countOfrating_7 = Column(Numeric)
    countOfrating_8 = Column(Numeric)
    countOfrating_9 = Column(Numeric)
    countOfrating_10 = Column(Numeric)


    def __init__(self, ISBN, bookTitle , bookAuthor , yearOfPublication, publish
er, imageURLS , imageURLM, imageURLL,
                 numOfUsers, countOfrating_0 ,countOfrating_1 ,countOfrating_2 ,
countOfrating_3 ,countOfrating_4 ,
                 countOfrating_5,countOfrating_6 ,countOfrating_7 ,countOfrating
_8 ,countOfrating_9 ,countOfrating_10):
        self.ISBN = ISBN
        self.bookTitle = bookTitle
        self.bookAuthor = bookAuthor
        self.yearOfPublication = yearOfPublication
        self.publisher = publisher
        self.imageURLS = imageURLS
        self.imageURLM = imageURLM
        self.imageURLL = imageURLL
        self.numOfUsers = numOfUsers
        self.countOfrating_0 = countOfrating_0
        self.countOfrating_1 = countOfrating_1
        self.countOfrating_2 = countOfrating_2
        self.countOfrating_3 = countOfrating_3
        self.countOfrating_4 = countOfrating_4
        self.countOfrating_5 = countOfrating_5
        self.countOfrating_6 = countOfrating_6
        self.countOfrating_7 = countOfrating_7
        self.countOfrating_8 = countOfrating_8
        self.countOfrating_9 = countOfrating_9
        self.countOfrating_10 = countOfrating_10
```

3. Main Class:

In [ ]:

```
# this one is main class

from sqlalchemy import func, case

from DBLayer.DBConnection import Base, session_factory
from model.book import Book
from model.rating import Rating
from model.book_final import BookFinal


# Below function load book data to postgres
def create_books(db_type, input_file, bad_data_file):
    session = session_factory(db_type)
    print(session)
    loop_var = 0
    if input_file.mode == 'r':
        for book_data in input_file:
            print(loop_var)
            # ignore header row
            if loop_var == 0:
                loop_var = 1
                continue
            book_data_list = book_data.rstrip().replace('\n', '').replace('"',''
).split(";")

            if len(book_data_list) == 8:
                try:
                    ISBN = book_data_list[0]
                    bookTitle = book_data_list[1]
                    bookAuthor = book_data_list[2]
                    yearOfPublication = int(book_data_list[3].replace('"',''))
                    publisher = book_data_list[4]
                    imageURLS = book_data_list[5]
                    imageURLM = book_data_list[6]
                    imageURLL = book_data_list[7]
                    bookObj = Book(ISBN, bookTitle, bookAuthor, yearOfPublicatio
n, publisher, imageURLS, imageURLM,
                                   imageURLL)
                    session.add(bookObj)
                    session.commit()
                except:
                    # put data into bad_data_file
                    bad_data_file.write(book_data)
            else:
                bad_data_file.write(book_data)
            loop_var = loop_var + 1 # used for checking the execution
        loop_var = 0
    session.commit()
    session.close()


# load rating csv file to mysql
def create_ratings(db_type, input_file, bad_data_file):
    session = session_factory(db_type)
    loop_var = 0
    if input_file.mode == 'r':
        for book_rating in input_file:
            print(loop_var)
            # ignoring header row
            if loop_var == 0:
                loop_var = 1
                continue
            book_rating_list = book_rating.rstrip().replace('\n', '').replace(
'"','').split(";")
            #print(len(book_rating_list))
            if len(book_rating_list) == 3:
                try:
                    #print("inside try")
                    userId = int(book_rating_list[0])
                    ISBN = book_rating_list[1]
                    rating = int(book_rating_list[2])
                    if len(ISBN) > 15:
                        raise ValueError('ISBN length is not 10')
                    ratingObj = Rating(userId, ISBN, rating)
                    session.add(ratingObj)
                    session.commit()
                except:
                    # put data into bad_data_file
                    bad_data_file.write(book_rating)
            else:
                bad_data_file.write(book_rating)
            loop_var = loop_var + 1
        loop_var = 0
    session.commit()
    session.close()


# get all books
def get_books(db_type):
    session = session_factory(db_type)
    books_query = session.query(Book)
    session.close()
    return books_query.all()


# get ratings of the book by isbn
def get_ratings_by_isbn(req_isbn, session):
    xpr0 = func.sum(case([(Rating.rating == 0, 1), ], else_=0)).label("countOfra
ting_0")
    xpr1 = func.sum(case([(Rating.rating == 1, 1), ], else_=0)).label("countOfra
ting_1")
    xpr2 = func.sum(case([(Rating.rating == 2, 1), ], else_=0)).label("countOfra
ting_2")
    xpr3 = func.sum(case([(Rating.rating == 3, 1), ], else_=0)).label("countOfra
ting_3")
    xpr4 = func.sum(case([(Rating.rating == 4, 1), ], else_=0)).label("countOfra
ting_4")
    xpr5 = func.sum(case([(Rating.rating == 5, 1), ], else_=0)).label("countOfra
ting_5")
    xpr6 = func.sum(case([(Rating.rating == 6, 1), ], else_=0)).label("countOfra
ting_6")
    xpr7 = func.sum(case([(Rating.rating == 7, 1), ], else_=0)).label("countOfra
ting_7")
    xpr8 = func.sum(case([(Rating.rating == 8, 1), ], else_=0)).label("countOfra
ting_8")
    xpr9 = func.sum(case([(Rating.rating == 9, 1), ], else_=0)).label("countOfra
ting_9")
    xpr10 = func.sum(case([(Rating.rating == 10, 1), ], else_=0)).label("countOf
rating_10")
```

```
    books_query = session.query(Rating.ISBN, xpr0, xpr1, xpr2, xpr3, xpr4, xpr5,
xpr6, xpr7, xpr8, xpr9, xpr10,
                                func.count(Rating.userId)).filter(Rating.ISBN ==
req_isbn).group_by(Rating.ISBN)
    return books_query.all()


# business logic to combine both the data
def combineBookAndRating(book, rating):
    ISBN = book.ISBN
    bookTitle = book.bookTitle
    bookAuthor = book.bookAuthor
    yearOfPublication = book.yearOfPublication
    publisher = book.publisher
    imageURLS = book.imageURLS
    imageURLM = book.imageURLM
    imageURLL = book.imageURLL
    numOfUsers = rating[12]
    countOfrating_0 = rating[1]
    countOfrating_1 = rating[2]
    countOfrating_2 = rating[3]
    countOfrating_3 = rating[4]
    countOfrating_4 = rating[5]
    countOfrating_5 = rating[6]
    countOfrating_6 = rating[7]
    countOfrating_7 = rating[8]
    countOfrating_8 = rating[9]
    countOfrating_9 = rating[10]
    countOfrating_10 = rating[11]
    book_final_obj = BookFinal(ISBN, bookTitle, bookAuthor, yearOfPublication, p
ublisher, imageURLS, imageURLM,
                               imageURLL,
                               numOfUsers, countOfrating_0, countOfrating_1, cou
ntOfrating_2, countOfrating_3,
                               countOfrating_4,
                               countOfrating_5, countOfrating_6, countOfrating_7
, countOfrating_8, countOfrating_9,
                               countOfrating_10)

    return book_final_obj


if __name__ == "__main__":

    db_type_for_book = 'postgres'
    book_file = open('input_files/BX-Books.csv', 'r', encoding='latin-1')
    bad_book_data_file = open('bad_data/bad_books_file.csv', 'w')
    create_books(db_type_for_book, book_file, bad_book_data_file)
    book_file.close();
    bad_book_data_file.close()


    db_type_for_rating = 'mysql'
    ratings_file = open('input_files/BX-Book-Ratings.csv', 'r', encoding='latin-
1')
    bad_ratings_data_file = open('bad_data/bad_rating_file.csv', 'w')
    create_ratings(db_type_for_rating, ratings_file, bad_ratings_data_file)
    ratings_file.close();
    bad_ratings_data_file.close()

    books = get_books('postgres')
    session_postgres = session_factory('postgres')
    session_mysql = session_factory('mysql')
    loop_var = 0
    for book in books:
        print(loop_var)
        ratings = get_ratings_by_isbn(book.ISBN, session_mysql)
        for rating in ratings:
            final_book_obj = combineBookAndRating(book, rating)
            session_postgres.add(final_book_obj)
            session_postgres.commit()
        loop_var = loop_var+1
    session_mysql.close()
    session_postgres.close()
```

--End of file--