第15次实验顺序指南

1. 第一部分：数据准备（File01_DataPreprocess.py）

【前期准备】

1. 首先创建一个新项目，项目名称输入为`HenanFinal`。

2. 在此项目中创建`Dataset`、`Figure`、`result`目录。

3. 将数据集文件`books.csv`先移动到Windows的`~/Downloads`目录，然后复制到`Dataset`文件夹。请务必确认。

```
cd Dataset
cp ~/Downloads/books.csv
```

*시간 절약을 위하여 기존의 ~/PycharmProjects/Project15/Dataset에서 복사해서 사용

【数据预处理】

1. 现在为了在数据分析之前进行预处理，新建`File01_DataPreprocess.py`文件，并输入以下代码：

```
import pandas as pd
import numpy as np
dataFrame = pd.read_csv('Dataset/books.csv')
```

2. 运行上述代码会出现错误，这表明存在重复值错误。

3. 解决方法：添加`on_bad_lines='skip'`来跳过有错误属性的行。

```
dataFrame = pd.read_csv('Dataset/books.csv', on_bad_lines='skip')
```

（如果使用旧版本，请使用`error_bad_lines=False`）

4. 添加`print(dataFrame.head(10))`命令后运行，将输出前10行的信息。

5. 检查数据集中某些列名前后是否有空格，输入并运行以下代码：

```
# 去除列名前后的空格
dataFrame.columns = dataFrame.columns.str.strip()
# 查看数据集信息
print("₩nraw dataFrame:")
print(dataFrame.info())
```

6. 从结果可以看出，数据集共有11,123行，12个属性列。属性列包括1个float64类型、5个int64类型和6个object类型。

7. 接下来检查数据中是否有空值，如果有则删除空值，运行以下代码：

```
# 删除空值
dataFrame_remove_null = dataFrame.dropna()
print("₩ndataFrame_remove_null:")
print(dataFrame_remove_null.info())
dataFrame = dataFrame_remove_null
```

8. 运行结果显示，删除空值后的数据集信息与原始数据集信息相同，表明原始数据集中没有空值。

9. 接下来查找并删除重复值：

```
# 删除重复值
dataFrame_remove_dup = dataFrame.drop_duplicates(keep='first')
print("₩ndataFrame_remove_dup:")
print(dataFrame_remove_dup.info())
dataFrame = dataFrame_remove_dup
```

10. 输出结果显示，删除重复值后的数据集信息与原始数据集信息相同，表明原始数据集中没有重复值（即没有重复行）。

11. 使用以下代码将时间列的格式从月日年改为年月日格式，如果再次出现空值则删除：

```
# 格式化时间列，由9/16/2006变成2006-9-16
# 处理不规范的数据
def convert_date(date_str):
    try:
        # 尝试转换日期格式
        converted_date = pd.to_datetime(date_str, format='%m/%d/%Y')
        return converted_date.strftime('%Y-%m-%d')  # 成功转换后重新格式化
    except ValueError:
        # 转换失败则返回NaN
        return np.nan

dataFrame['publication_date'] = dataFrame['publication_date'].apply(convert_date)
```

```
# 删除空值
dataFrame_remove_null = dataFrame.dropna()
print("₩ndataFrame_remove_null:")
print(dataFrame_remove_null.info())
dataFrame = dataFrame_remove_null
```

12. 运行结果显示，与原始数据集相比，格式化后的数据集多了两个空值，在这里被删除。总共剩下11,121条数据。

13. 使用以下代码检查`language_code`数据是否有异常值：

```
# 查看language_code的数据有没有异常值
print(dataFrame['language_code'].unique())
```

14. 结果显示所有语言代码（例如英语为eng）。确认后没有发现异常值。

15. 现在将预处理后的数据以新文件名`books_cleaned.csv`保存，编写并运行以下代码：

```
# 将处理后的数据写入新的csv文件中
dataFrame.to_csv('Dataset/books_clean.csv', encoding='utf-8', index=False)
```

16. 运行后确认数据集文件已保存。

【将预处理后的文件上传到Hadoop HDFS的方法】

1. 首先在环境中打开一个新的终端。确认Hadoop正在运行。我们最初是以批处理文件的形式运行了所有必需的程序，因此当前运行正常。但需要再次确认。输入`jps`命令，确认输出Hadoop的DataNode和NameNode（顺序无关）。

2. 当前Hadoop的工作目录是`/user/Hadoop/input`。我们先查看该文件夹中有哪些文件。将目录切换到Hadoop的`cd /usr/local/hadoop`。注意：`usr`和`user`完全不同。`User`是Hadoop内部的，我们看不到。

```
hdfs dfs -ls /user/hadoop/input
```

3. 使用以下命令将文件复制到Hadoop：（如果大学或项目不同，需要相应修改）

```
(base) hadoop@hadoop01:/usr/local/hadoop$ ./bin/hdfs dfs -put
/home/hadoop/PycharmProjects/HenanFinal/Dataset/books_clean.csv input/books_clean.csv
```

4. 务必使用`hdfs dfs -ls /user/hadoop/input`命令确认文件已上传。

2. 第二部分：数据分析（File02_DataAnalysis.py）

1. 现在为了进行数据分析，创建一个新文件，文件名为`File02_DataAnalysis.py`。务必先鼠标单击`HenanFinal`项目文件夹，然后创建新文件。否则文件可能创建在其他位置导致问题。

2. 由于有重复性，首先按以下格式设置文档结构，然后进行详细内容工作：

```
from pyspark import SparkConf
from pyspark.sql import SparkSession
from pyspark.sql.functions import date_format, split, rank
from pyspark.sql.window import Window

def run_task_1(spark):
    print("## Top 10 text_reviews_count₩n")

def run_task_2(spark):
    print("## Top 10 num_pages₩n")

def run_task_3(spark):
    print("## Publisher books num₩n")

def run_task_4(spark):
    print("##------ 4.不同语言的书籍数量")

def main():
    # 创建Spark会话
    spark = SparkSession.builder.config(conf=SparkConf()).getOrCreate()

    # 读取数据并进行预处理
    books_df = spark.read.csv("input/books_clean.csv", header=True, inferSchema=True)
    books_df = books_df.repartition(1)
    books_df.createOrReplaceTempView("books")

    while True:
        print("=" * 50)
        print("选择要执行的任务:")
        print("1. 前10本最受关注的书籍")
        print("2. 前10个最长篇幅的书籍")
        print("3. 不同出版社出版的书籍数量")
```

```
            print("4. 不同语言的书籍数量")
            print("0. 退出程序")
            print("=" * 50)

            choice = input("请输入选择 (0-4): ").strip()

            if choice == '0':
                print("程序退出")
                break
            elif choice == '1':
                run_task_1(spark)
            elif choice == '2':
                run_task_2(spark)
            elif choice == '3':
                run_task_3(spark)
            elif choice == '4':
                run_task_4(spark)
            else:
                print("无效的选择，请重新输入!")

            # 询问是否继续执行
            continue_choice = input("是否继续执行其他任务? (y/n): ").strip().lower()
            if continue_choice != 'y':
                print("程序退出")
                break

    # 关闭Spark会话
    spark.stop()

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

3. 现在开始正式分析。第一个分析目标是找出评论数最多的前10本书（`text_reviews_count`）。

4. 实现此目标的代码：通过`spark.sql()`方法执行SQL查询，使用`ORDER BY`按`text_reviews_count`（书籍收到的总文本评论数）对SQL语句进行排序，并设置`DESC`进行降

序排列。然后使用`show`方法显示结果的前10行，即评论数最多的前10本书。`truncate=False`表示显示内容不会被截断，长字符串会完整显示。最后使用`DataFrame.write.csv()`方法将结果保存到HDFS的结果目录中，如果目标文件已存在，则设置`mode='overwrite'`覆盖原文件。

```
def run_task_1(spark):
    """前10本最受关注的书籍(text_reviews_count)"""
    print("##------ 1.前10本最受关注的书籍(text_reviews_count)")

    top_10_text = spark.sql("SELECT bookID,title,SUBSTRING_INDEX(authors, '/', 1) AS
first_author,average_rating,language_code,text_reviews_count,publication_date ₩
    FROM books ₩
    ORDER BY text_reviews_count DESC")

    top_10_text.show(n=10, truncate=False)
    print("## Top 10 text_reviews_count₩n")
    top_10_text.write.csv("result/task01_10_text.csv", mode='overwrite')
    print("任务1完成: 前10本最受关注的书籍₩n")
```

5. 这次我们找出书中评论最多的10本书，即**查找页数最多的前10本书（num_pages）**。代码中通过`spark.sql()`方法执行SQL查询，使用`ORDER BY`按`num_pages`（书籍的总页数）对SQL语句进行排序，并设置`DESC`进行降序排列。然后使用`show`方法显示结果的前10行，即页数最多的前10本书。最后使用`DataFrame.write.csv()`方法将结果保存到HDFS的结果目录中。运行结果将输出前10本书的信息。

```
def run_task_2(spark):
    """前10个最长篇幅的书籍（num_pages）"""
    print("##------ 2.前10个最长篇幅的书籍（num_pages）")

    top_10_numpages = spark.sql("SELECT bookID,title,SUBSTRING_INDEX(authors, '/', 1) AS
first_author,average_rating,language_code,num_pages,publication_date ₩
    FROM books ₩
    ORDER BY num_pages DESC")

    top_10_numpages.show(n=10, truncate=False)
    print("## Top 10 num_pages₩n")
    top_10_numpages.write.csv("result/task02_10_numpages.csv", mode='overwrite')
    print("任务2完成: 前10个最长篇幅的书籍₩n")
```

6. 这次我们按出版社统计出版书籍数量最多的前20个，即**各出版社出版的书籍数量**。通过`spark.sql()`方法执行SQL查询，使用`GROUP BY`按出版社进行分组，使用SQL聚合函数`COUNT`计算每组的书籍数量`books_num`，使用`ORDER BY`按书籍数量降序排列。运行结果：输出20个结果信息。

```
def run_task_3(spark):
    """不同出版社出版的书籍数量，统计前50个"""
    print("##------ 3.不同出版社出版的书籍数量，统计前50个")

    publisher_books_num = spark.sql("SELECT publisher,COUNT(*) as books_num ₩
    FROM books ₩
    GROUP BY publisher ₩
    ORDER BY books_num DESC")

    publisher_books_num.show(truncate=False)
    print("## Publisher books num₩n")
    publisher_books_num.write.csv("result/task03_books_num.csv", mode='overwrite')
    print("任务3完成: 不同出版社出版的书籍数量₩n")
```

7. 这次我们了解按语言出版的书籍数量，即**找出不同语言的书籍数量**。通过`spark.sql()`方法执行SQL查询，使用`GROUP BY`子句按`language_code`（语言代码）对视图书籍进行分组，使用SQL聚合函数`COUNT`计算每组的书籍数量`books_num`，使用`ORDER BY`进行降序排列。运行结果：输出20个结果信息。

```
def run_task_4(spark):
    """不同语言的书籍数量"""
    print("##------ 4.不同语言的书籍数量")

    language_books_num = spark.sql("SELECT language_code,COUNT(*) as books_num ₩
    FROM books ₩
    GROUP BY language_code ₩
    ORDER BY books_num DESC")

    language_books_num.show(truncate=False)
    language_books_num.write.csv("result/task04_language_books_num.csv", mode='overwrite')
    print("任务4完成: 不同语言的书籍数量₩n")
```

3. 第三部分：数据可视化（File03_DataView.py）

1. 为了进行数据可视化，需要将HDFS中的结果数据复制到本地。使用`hdfs dfs -get`命令进行复制，打开新终端进行操作。保存的文件按task01、02、03、04等区分。

```
hdfs dfs -get /user/hadoop/result/task* ~/PycharmProjects/HenanFinal/result/.
(base) hadoop@hadoop01:/usr/local/hadoop$ cd ~/PycharmProjects/HenanFinal/result/
```

2. 复制后，由于文件名太长，可以简单修改为`part01.csv`、`part02.csv`、`part03.csv`、`part04.csv`等。

3. 现在对4个数据进行可视化，通过设置编号进行选择并可视化。为此，最好预先设置好格式，如前所述的数据分析部分。

```
import matplotlib.pyplot as plt
import pandas as pd
import matplotlib.colors as mcolors
import os
# 基本路径设置
BASE_DIR = os.path.dirname(os.path.abspath(__file__))  # 当前脚本所在目录
RESULT_DIR = os.path.join(BASE_DIR, 'result')  # result文件夹路径
FIGURE_DIR = os.path.join(BASE_DIR, 'Figure')  # Figure文件夹路径

# 如果Figure文件夹不存在则创建
if not os.path.exists(FIGURE_DIR):
    os.makedirs(FIGURE_DIR)

def plot_most_popular_books():
    """1. 前10本最受欢迎的书籍（text_reviews_count)"""

def plot_longest_books():
    """2. 前10本最长的书籍（num_pages)"""

def plot_publisher_stats():
    """3. 各出版社出版的书籍数量（前50个)"""

def plot_language_stats():
    """4. 各语言的书籍数量"""

def main():
```

```
    """主程序"""
    # 输出当前路径信息
    print(f"当前工作目录: {BASE_DIR}")
    print(f"数据文件路径: {RESULT_DIR}")
    print(f"图片保存路径: {FIGURE_DIR}")

    # 检查目录是否存在
    if not os.path.exists(RESULT_DIR):
        print(f"警告: result目录不存在: {RESULT_DIR}")

    while True:
        print("₩n" + "=" * 50)
        print("图表选择菜单")
        print("=" * 50)
        print("1. 前10本最受欢迎的书籍")
        print("2. 前10本最长的书籍")
        print("3. 各出版社出版的书籍数量（前50个)")
        print("4. 各语言的书籍数量")
        print("0. 程序退出")
        print("=" * 50)

        try:
            choice = input("请输入所需的图表编号 (0-4): ").strip()

            if choice == '0':
                print("程序退出。")
                break
            elif choice == '1':
                plot_most_popular_books()
            elif choice == '2':
                plot_longest_books()
            elif choice == '3':
                plot_publisher_stats()
            elif choice == '4':
                plot_language_stats()
            else:
                print("输入错误，请输入0-4之间的数字。")
```

```
            except KeyboardInterrupt:
                print("₩n程序退出。")
                break
            except FileNotFoundError as e:
                print(f"找不到文件: {e}")
            except Exception as e:
                print(f"发生错误: {e}")

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

4. 首先对**被提及最多的前10本书进行可视化**。结果显示，被提及最多的书是《Twilight》。

```
def plot_most_popular_books():
    """1. 前10本最受欢迎的书籍（text_reviews_count)"""
    csv_filename = 'task01_10_text.csv/part01.csv'
    csv_filepath = os.path.join(RESULT_DIR, csv_filename)

    column_names = ['bookID', 'title', 'first_author', 'average_rating', 'language_code', 'text_reviews_count',
'publication_date']
    df = pd.read_csv(csv_filepath, header=None, names=column_names)

    data_to_plot = df.head(10)
    titles = data_to_plot['title']
    texts = data_to_plot['text_reviews_count']

    plt.figure(figsize=(10, 5))
    plt.barh(titles, texts, color='skyblue')

    for index, value in enumerate(texts):
        plt.text(value, index, f'{value}', va='center', ha='left')

    plt.ylabel('书籍标题', fontsize=14, fontweight='bold')
    plt.xlabel('文本评论数', fontsize=14, fontweight='bold')
    plt.title('前10本最受欢迎的书籍', fontsize=14, fontweight='bold')
    plt.tight_layout()
    # 图片保存路径
    save_path = os.path.join(FIGURE_DIR, 'Top_10_Most_Popular_Books.png')
    plt.savefig(save_path)
    plt.show()
```

5. 第二个图表是**拥有最多页数的前 10 本书（num_pages）**。输出结果显示，最长的书是《The Complete Aubrey/Maturin Novels》。

```
def plot_longest_books():
    """2. 前10本最长的书籍 (num_pages)"""
    csv_filename = 'task02_10_numpages.csv/part02.csv'
    csv_filepath = os.path.join(RESULT_DIR, csv_filename)

    column_names = ['bookID', 'title', 'first_author', 'average_rating', 'language_code', 'num_pages',
'publication_date']
    df = pd.read_csv(csv_filepath, header=None, names=column_names)

    data_to_plot = df.head(10)
    titles = data_to_plot['title']
    num_pages = data_to_plot['num_pages']

    plt.figure(figsize=(15, 5))
    plt.barh(titles, num_pages, color='skyblue')

    for index, value in enumerate(num_pages):
        plt.text(value, index, f'{value}', va='center', ha='left')

    plt.title('页数最多的前10本书', fontsize=14, fontweight='bold')
    plt.ylabel('书籍标题', fontsize=14, fontweight='bold')
    plt.xlabel('页数', fontsize=14, fontweight='bold')
    plt.tight_layout()

    # 图片保存路径
    save_path = os.path.join(FIGURE_DIR, 'The_top_10_longest_length_books.png')
    plt.savefig(save_path)
    plt.show()
```

6. 第三个图表是可视化出版书籍最多的公司。**显示前50名**，结果显示出版书籍最多的出版社是Vintage，总共出版了318本书。

```
def plot_publisher_stats():
    """3. 各出版社出版的书籍数量（前50个）"""
    csv_filename = 'task03_books_num.csv/part03.csv'
    csv_filepath = os.path.join(RESULT_DIR, csv_filename)

    column_names = ['publisher', 'books_num']
    df = pd.read_csv(csv_filepath, header=None, names=column_names)

    data_to_plot = df.head(50)
    publisher = data_to_plot['publisher']
    books_num = data_to_plot['books_num']

    plt.figure(figsize=(10, 5))
    bars = plt.bar(publisher, books_num, color='skyblue')

    for bar in bars:
        yval = bar.get_height()
        plt.text(bar.get_x() + bar.get_width() / 2.0, yval, int(yval), ha='center', rotation=45)

    plt.title('不同出版社出版的书籍数量', fontsize=14, fontweight='bold')
    plt.ylabel('书籍数量', fontsize=14, fontweight='bold')
    plt.xlabel('出版社', fontsize=14, fontweight='bold')
    plt.xticks(rotation=90)
    plt.tight_layout()

    # 图片保存路径
    save_path = os.path.join(FIGURE_DIR, 'The_number_of_books_published_by_different_publishers.png')
    plt.savefig(save_path)
    plt.show()
```

7. 最后的可视化图表显示按语言出版最多的语言顺序。输出结果显示大部分是eng（英语），总共8907本。

```
def plot_language_stats():
    """4. 各语言的书籍数量"""
    csv_filename = 'task04_language_books_num.csv/part04.csv'
    csv_filepath = os.path.join(RESULT_DIR, csv_filename)

    column_names = ['language_code', 'books_num']
    df = pd.read_csv(csv_filepath, header=None, names=column_names)

    data_to_plot = df.head(50)
    language_code = data_to_plot['language_code']
    books_num = data_to_plot['books_num']

    plt.figure(figsize=(10, 5))
    bars = plt.bar(language_code, books_num, color='skyblue')

    for bar in bars:
        yval = bar.get_height()
        plt.text(bar.get_x() + bar.get_width() / 2.0, yval, int(yval), ha='center')

    plt.title('不同语言的书籍数量', fontsize=14, fontweight='bold')
    plt.ylabel('书籍数量', fontsize=14, fontweight='bold')
    plt.xlabel('语言代码', fontsize=14, fontweight='bold')
    plt.tight_layout()

    # 图片保存路径
    save_path = os.path.join(FIGURE_DIR, 'Number_of_books_in_different_languages.png')
    plt.savefig(save_path)
    plt.show()
```