一、 网站日志数据采集（代码及实现步骤）

a） 网站日志数据生成及按时间拆解日志数据（描述系统功能实现方法、运行方式、shell 脚本关键代码、结果截图）。（**要求截图和关键代码背景为白色。要求仅截图关键代码，每段关键代码不超过一页，必须有对应的文字说明**）（5 分）

1、 配置 nginx 的配置文件 nginx.conf

```
    log_format  main  '$remote_addr - $remote_user [$time_local] "$req
uest" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';

    access_log  logs/access.log  main;

    sendfile        on;
    #tcp_nopush     on;

    #keepalive_timeout  0;
    keepalive_timeout  65;

    #gzip  on;

    server {
        listen       80;
        server_name  localhost;

        charset koi8-r;

        access_log  logs/host.access.log  main;

        location / {
            root   html;
            index  index.html index.htm;
        }
```

初始的网站访问日志是由 nignx 生成的，在讲义 flume 里面有介绍

2、 在/opt/server/nginx/html 目录下新建 a.html、b.html 文件

```
hadoop@master:/opt/server/nginx/logs$ ls
access.log  error.log  nginx.pid
```

3、 在 /opt/server/nginx/scripts/project1 目录下新建一个文件 nginx_log.sh， 编写拆解日志文件脚本

```
hadoop@master:/opt/server/nginx$ cd scripts
hadoop@master:/opt/server/nginx/scripts$ touch nginx_log.sh
hadoop@master:/opt/server/nginx/scripts$ vim nginx_log.sh
```

4、 启动 crontab,进入 crontab 编辑指令，输入如下配置指令定时执行脚本 nginx_log.sh，按时间拆解日志数据，每 3 分钟执行一次脚本

```
hadoop@master:/opt/server/nginx/logs$ crontab -e
crontab: installing new crontab
hadoop@master:/opt/server/nginx/logs$ crontab -l
```

```
*/3 * * * * /opt/server/nginx/logs/nginx_log.sh
```

① 启动 nginx，打开浏览器，不断访问以下地址：

http://192.168.80.128:80/a.html；

http://192.168.80.128:80/b.html；

http://192.168.80.128:80/c.html；

http://192.168.80.128:80/d.html；

② 网站日志数据生成及按时间拆解日志数据结果如下

```
hadoop@master:/opt/server/nginx/logs$ ll
总用量 40
drwxr-xr-x  2 hadoop hadoop 4096 12月  14 14:03 ./
drwxrwxr-x 12 hadoop hadoop 4096 12月   9 00:33 ../
-rw-r--r--  1 hadoop hadoop 1194 12月  14 14:02 access_2024-12-13-14-03.log
-rw-r--r--  1 hadoop hadoop 1194 12月  14 14:04 access.log
-rwxr-xr-x  1 hadoop hadoop 4975 12月  14 12:14 error.log*
-rwxr-xr-x  1 hadoop hadoop    0 12月  13 12:15 filename*
-rwxr-xr-x  1 hadoop hadoop 2881 12月  14 09:48 host.access.log*
-rwxr-xr-x  1 hadoop hadoop  349 12月  14 11:57 nginx_cron.log*
-rwxr-xr-x  1 hadoop hadoop  995 12月  14 14:00 nginx_log.sh*
-rw-r--r--  1 hadoop hadoop    6 12月  14 14:00 nginx.pid
hadoop@master:/opt/server/nginx/logs$ cat access.log
192.168.80.133 - - [14/Dec/2024:14:04:22 +0800] "GET /b.html HTTP/1.1" 304
0 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
 like Gecko) Chrome/131.0.0.0 Safari/537.36" "-"
192.168.80.133 - - [14/Dec/2024:14:04:22 +0800] "GET /b.html HTTP/1.1" 304
0 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
 like Gecko) Chrome/131.0.0.0 Safari/537.36" "-"
192.168.80.133 - - [14/Dec/2024:14:04:22 +0800] "GET /b.html HTTP/1.1" 304
0 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
 like Gecko) Chrome/131.0.0.0 Safari/537.36" "-"
192.168.80.133 - - [14/Dec/2024:14:04:22 +0800] "GET /b.html HTTP/1.1" 304
0 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
 like Gecko) Chrome/131.0.0.0 Safari/537.36" "-"
192.168.80.133 - - [14/Dec/2024:14:04:23 +0800] "GET /b.html HTTP/1.1" 304
0 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
 like Gecko) Chrome/131.0.0.0 Safari/537.36" "-"
192.168.80.133 - - [14/Dec/2024:14:04:23 +0800] "GET /b.html HTTP/1.1" 304
0 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
 like Gecko) Chrome/131.0.0.0 Safari/537.36" "-"
```

```
hadoop@master:/opt/server/nginx/logs$ cat access_2024-12-17-20-51.log
192.168.80.133 - - [18/Dec/2024:20:48:15 +0800] "GET /c.html HTTP/1.1" 304 0 "-" "Mozilla/5.0 (
Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.0.0 Safari/537
.36" "-"
192.168.80.133 - - [18/Dec/2024:20:48:15 +0800] "GET /c.html HTTP/1.1" 304 0 "-" "Mozilla/5.0 (
Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.0.0 Safari/537
.36" "-"
192.168.80.133 - - [18/Dec/2024:20:48:15 +0800] "GET /c.html HTTP/1.1" 304 0 "-" "Mozilla/5.0 (
Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.0.0 Safari/537
.36" "-"
192.168.80.133 - - [18/Dec/2024:20:48:16 +0800] "GET /c.html HTTP/1.1" 304 0 "-" "Mozilla/5.0 (
Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.0.0 Safari/537
.36" "-"
192.168.80.133 - - [18/Dec/2024:20:48:17 +0800] "GET /a.html HTTP/1.1" 304 0 "-" "Mozilla/5.0 (
Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.0.0 Safari/537
.36" "-"
192.168.80.133 - - [18/Dec/2024:20:48:17 +0800] "GET /a.html HTTP/1.1" 304 0 "-" "Mozilla/5.0 (
Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.0.0 Safari/537
.36" "-"
```

b） 通过 flume 日志数据采集到 hdfs（描述系统功能实现方法、运行方式、flume 配置文件的关键代码、结果截图）。（**要求截图和关键代码背景为白色。要求仅截图关键代码，每段关键代码不超过一页，必须有对应**

的文字说明）（5 分）

1. 在/opt/server/flume/job 目录下新建一个 flume 配置文件 flume-file-hdfs.conf，编写配置并保存

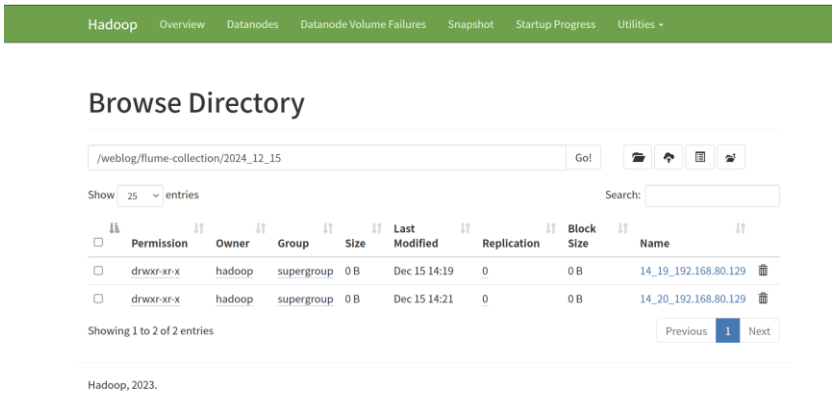
```
hadoop@master:/opt/server/flume$ cd job
hadoop@master:/opt/server/flume/job$ ls
flume-file-hdfs.conf
hadoop@master:/opt/server/flume/job$ vim flume-file-hdfs.conf
```

```
# Name the components on this agent
agent1.sources = source1
agent1.sinks = sink1
agent1.channels = channel1

# 监控日志文件的源配置
#监控一个目录下的多个文件新增的内容
agent1.sources.source1.type = TAILDIR
#通过 json 格式存下每个文件消费的偏移量，避免从头消费
agent1.sources.source1.positionfile = /opt/server/flume/offset/taildir_position.json
agent1.sources.source1.filegroups = f1
agent1.sources.source1.filegroups.f1 = /opt/server/nginx/logs/access_*.log
agent1.sources.source1.interceptors = i1
agent1.sources.source1.interceptors.i1.type = host
agent1.sources.source1.interceptors.i1.hostHeader = hostname

# HDFS Sink 配置
agent1.sinks.sink1.type = hdfs
agent1.sinks.sink1.hdfs.path = hdfs://master:9000/weblog/flume-collection/%Y_%m_%d/%H_%M_%{hostname}
#指定文件名前缀
agent1.sinks.sink1.hdfs.filePrefix = access_log
#指定每批下沉数据的记录条数
#是否按照时间滚动文件夹
agent1.sinks.sink1.hdfs.round = true
agent1.sinks.sink1.hdfs.batchsize = 100
agent1.sinks.sink1.hdfs.fileType = DataStream
agent1.sinks.sink1.hdfs.writeFormat = Text
#指定下沉文件按1MB大小滚动
agent1.sinks.sink1.hdfs.rollsize = 1048576
#指定下沉文件按1000000条数滚动
agent1.sinks.sink1.hdfs.rollcount = 1000000
#指定下沉文件按30分钟滚动
agent1.sinks.sink1.hdfs.rollInterval = 30
agent1.sinks.sink1.hdfs.useLocalTimeStamp = true

# 内存类型 Channel 配置
#使用memory类型channel
agent1.channels.channel1.type = memory
agent1.channels.channel1.capacity = 500000
agent1.channels.channel1.transactionCapacity = 600

# 绑定 Source 和 Sink 到 Channel
agent1.sources.source1.channels = channel1
agent1.sinks.sink1.channel = channel1

~
"flume-file-hdfs.conf" 45L, 1735B
```

2. 然后运行 flume 配置文件脚本，将数据采集到 hdfs,结果如下

```
bin/flume-ng agent --conf /opt/server/flume/conf --
conf-file /opt/server/flume/job/flume-file-hdfs.conf -
-name agent1 -Dflume.root.logger=INFO,console
```

```
hadoop@master:/opt/server/flume/bin$ flume-ng agent --conf /opt/server/flume/conf --conf-file /
opt/server/flume/job/flume-file-hdfs.conf --name agent1 -Dflume.root.logger=INFO,console
Info: Including Hadoop libraries found via (/usr/local/hadoop/bin/hadoop) for HDFS access
Info: Including HBASE libraries found via (/opt/server/hbase/bin/hbase) for HBASE access
Info: Including Hive libraries found via () for Hive access
+ exec /usr/lib/jvm/jdk/bin/java -Xmx20m -Dflume.root.logger=INFO,console -cp '/opt/server/flum
e/conf:/opt/server/flume/lib/*:/usr/local/hadoop/etc/hadoop:/usr/local/hadoop/share/hadoop/comm
on/lib/*:/usr/local/hadoop/share/hadoop/common/*:/usr/local/hadoop/share/hadoop/hdfs:/usr/local
/hadoop/share/hadoop/hdfs/lib/*:/usr/local/hadoop/share/hadoop/hdfs/*:/usr/local/hadoop/share/h
```



二、 数据预处理（代码及实现步骤）

a) 将数据从采集目录移动到预处理目录（描述系统功能实现方法、运行方式、关键代码、结果截图）。（**要求截图和关键代码背景为白色。要求仅截图关键代码，shell 脚本代码不超过一页，必须有对应的文字说明**）（5 分）

1. 在 /export/software/project1 目录下新建脚本文件 movework.sh，编写代码如下

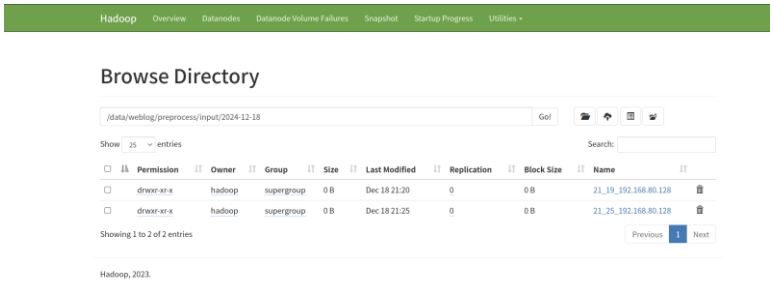
```
hadoop@master:/opt/scripts$ cd project1
hadoop@master:/opt/scripts/project1$ vim movework.sh
```

```
#!/bin/bash

# Flume 采集生成的日志文件存放的目录 (在 HDFS)
log_flume_dir="/weblog/flume-collection"

# 预处理程序的工作目录 (在 HDFS)
log_pre_input="/data/weblog/preprocess/input"

# 获取当前日期信息
day_01=$(date +%Y-%m-%d)
syear=$(date --date=$day_01 +%Y)
smonth=$(date --date=$day_01 +%m)
sday=$(date --date=$day_01 +%d)

# 读取日志文件的目录，判断是否有需要上传的文件
files=$(hadoop fs -ls $log_flume_dir | grep $day_01 | wc -l)

if [ $files -gt 0 ]; then
    # 移动文件到预处理目录
    hadoop fs -mv ${log_flume_dir}/${day_01} ${log_pre_input}
    echo "success moved ${log_flume_dir}/${day_01} to ${log_pre_input} ....."
else
    echo "No files to move for date $day_01."
fi

~
```

2. 执行 movework.sh 脚本，将数据从采集目录移动到预处理目录 /data/weblog/preprocess/input，在 hdfs 上查看结果

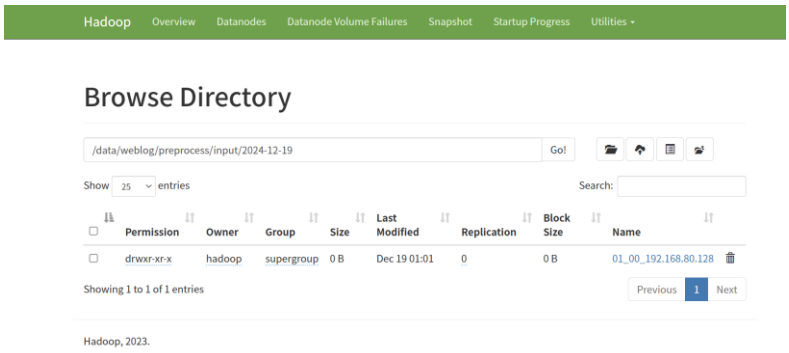
转移 2024 年 12 月 18 日的数据：

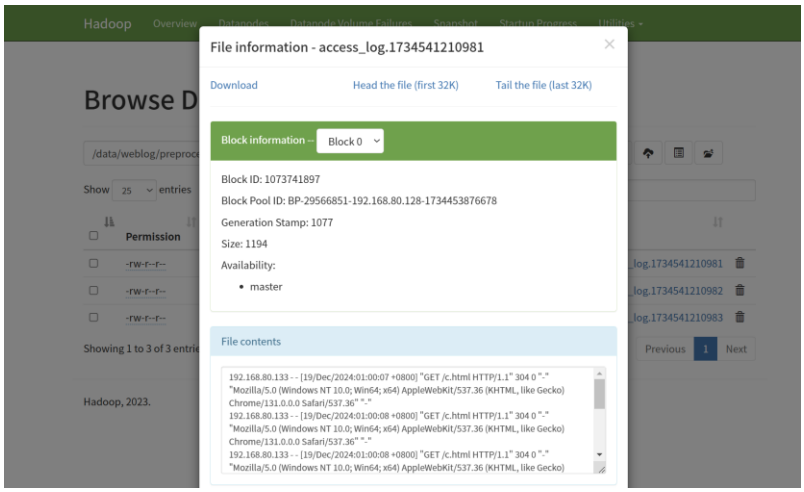
```
hadoop@master:/opt/scripts/project1$ sh movework.sh
success moved /weblog/flume-collection/2024-12-18 to /data/weblog/preprocess/input .....
```



转移 2024 年 12 月 19 日的数据：

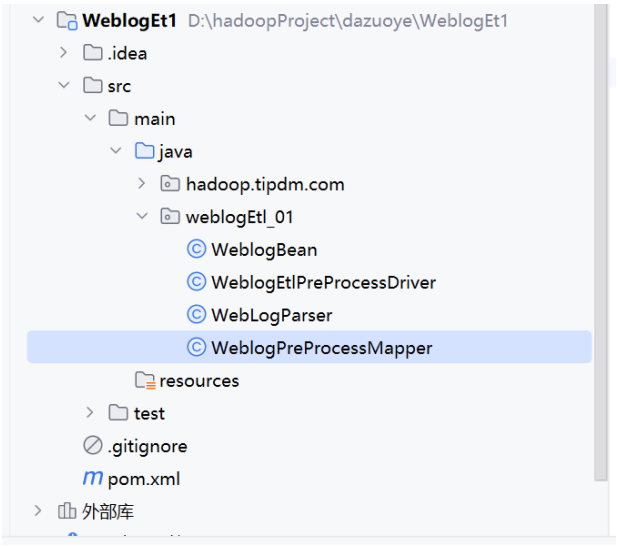
```
hadoop@master:/opt/scripts$ cd project1
hadoop@master:/opt/scripts/project1$ ls
movework.sh
hadoop@master:/opt/scripts/project1$ sh movework.sh
success moved /weblog/flume-collection/2024-12-19 to /data/weblog/preprocess/input .....
hadoop@master:/opt/scripts/project1$
```





b） 数据清洗（描述系统功能实现方法、运行方式、关键代码、结果截图）。（**要求截图和关键代码背景为白色。要求仅截图关键代码，关键代码不超过一页，必须有对应的文字说明**）（10 分）

1. 数据清洗，过滤“不合规”数据，清洗无意义的数据，在 IntelliJ IDEA 中新建一个项目 WeblogEtl



2. 编写 WeblogBean.java，部分代码如下：

   toString 方法，输出打印以\001 为分隔符作为数据拼接

```
    @Override
    public String toString() {
//  toString 方法将 WeblogBean 对象的各个字段拼接成一个字符串，并使用 \001 作为分隔符，这样可以方
        return this.valid +
                "\001" + this.getRemote_addr() +
                "\001" + this.getRemote_user() +
                "\001" + this.getTime_local() +
                "\001" + this.getRequests() +
                "\001" + this.getStatus() +
                "\001" + this.getBody_bytes_sent() +
                "\001" + this.getHttp_referer() +
                "\001" + this.getHttp_user_agent() +
                '}';
    }
```

```
    private boolean valid=true;//判断数据是否合法  3 个用法
    private  String remote_addr;//记录客户端的ip地址  5 个用法
    private String remote_user;//记录客户端的用户名称，忽略属性","  5 个用法
    private String time_local;//记录访问时间与地区  5 个用法
    private String requests;//请求的资源  5 个用法
    private String status;//HTTP状态码  5 个用法
    private String body_bytes_sent;//发生的字节数  5 个用法
    private String http_referer;//Referer  5 个用法
    private String http_user_agent;//用户代理  5 个用法
```

```java
@Override
//write 方法将 WeblogBean 对象的各个字段写入输出流（如 HDFS），在传输数据时保证序列化。
public void write(DataOutput out) throws IOException {
    out.writeBoolean(this.valid);//写入 valid 字段
    out.writeUTF(null==remote_addr?"":remote_addr);// 如果字段为 null，则写空字符串
    out.writeUTF(null==remote_user?"":remote_user);
    out.writeUTF(null==time_local?"":time_local);
    out.writeUTF(null==requests?"":requests);
    out.writeUTF(null==status?"":status);
    out.writeUTF(null==body_bytes_sent?"":body_bytes_sent);
    out.writeUTF(null==http_referer?"":http_referer);
    out.writeUTF(null==http_user_agent?"":http_user_agent);

}
```

```java
@Override
readFields 方法从输入流中读取数据，并还原为 WeblogBean 对象，反序列化过程。
public void readFields(DataInput in) throws IOException {
    this.valid=in.readBoolean();// 读取 valid 字段
    this.remote_addr=in.readUTF();// 读取各个字段
    this.remote_user=in.readUTF();
    this.time_local=in.readUTF();
    this.requests=in.readUTF();
    this.status=in.readUTF();
    this.body_bytes_sent=in.readUTF();
    this.http_referer=in.readUTF();
    this.http_user_agent=in.readUTF();
}
```

3. 编写 WebLogParser.java，部分代码如下：

```
7    public class WebLogParser {  2 个用法
35
36       // 过滤无效数据并解析日志
37       public static WeblogBean parser(String line) {  1 个用法
38           WeblogBean weblogBean = new WeblogBean();
39
40           // 通过空格分隔日志行
41           String[] arr = line.split( regex: " ");
42
43           // 如果字段少于 11 个，返回无效数据
44           if (arr.length < 11) {
45               weblogBean.setValid(false);
46               return weblogBean;
47           }
48
49           if (arr.length > 11) {
50               // 设置 WeblogBean 的字段值
51               weblogBean.setRemote_addr(arr[0]);  // 客户端 IP 地址
52               weblogBean.setRemote_user(arr[1]);  // 用户名，可能是 "-"，可以跳过
53
54               // 解析时间：将日志时间转换为目标格式
55               String time_local = arr[3].substring( beginIndex: 1); // 日期字段去掉前面的 "[" 符号
56               String formattedTime = convertDateFormat(time_local);
57               if (formattedTime != null) {
58                   weblogBean.setTime_local(formattedTime);
59               } else {
60                   weblogBean.setValid(false);
61               }
62
```

4. 编写 WeblogPreProcessMapper.java,部分代码如下
   设置路径，过滤掉非法数据

```
protected void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {
    String line = value.toString();
    // 使用日志解析器解析日志
    WeblogBean weblogBean = WebLogParser.parser(line);

    if (weblogBean != null && weblogBean.isValid()) {
        // 过滤静态资源，跳过 /a.html, /b.html, /c.html, /d.html
        WebLogParser.filtStaticResouce(weblogBean, pages);

        // 如果是有效数据，则输出
        if (weblogBean.isValid()) {
            k.set(weblogBean.getRemote_addr());  // 使用 IP 地址作为 key
            v = weblogBean;  // 使用 WeblogBean 对象作为 value
            context.write(k, v);
```

5. 编写 WeblogEtlPreProcessDriver,配置 hadoop 集群信息

```
public class WeblogEtlPreProcessDriver {
    public static void main(String[] args) throws IOException, InterruptedException, ClassNotFoundException {
        configuration.set("mapreduce.app-submission.cross-platform", "true");
        configuration.set("yarn.resourcemanager.hostname", "master");

        // 创建 Job 实例
        Job job = Job.getInstance(configuration,  jobName: "WeblogETLPreProcessJob");
        job.setJarByClass(WeblogEtlPreProcessDriver.class);

        // 设置 Mapper 和输出类型
        job.setMapperClass(WeblogPreProcessMapper.class);
        job.setMapOutputKeyClass(Text.class);  // 输出 key 类型
        job.setMapOutputValueClass(WeblogBean.class);  // 输出 value 类型
```

6. 把生成的 jar 包移动到 node1 虚拟机的/home/hadoop/下载目录下,运行该 jar

包，把清洗后的 2025 年 1 月 5 日的数据保存到 hdfs 的 /data/weblog/preprocess/output/2025-1-5 目录下，结果如图

```
root@master:/home/hadoop/下载# hadoop jar WeblogEt1-1.0-SNAPSHOT.jar weblogEtl_01.WeblogEtlPreProcessDr
iver /data/weblog/preprocess/input/2025-01-05/* /data/weblog/preprocess/output/2025-01-05
2025-01-05 21:53:33,555 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager a
t master/192.168.80.128:8032
2025-01-05 21:53:34,968 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/had
oop-yarn/staging/root/.staging/job_1736043599412_0012
2025-01-05 21:53:36,120 INFO input.FileInputFormat: Total input files to process : 2
```

File information - part-r-00000

Download　　Head the file (first 32K)　　Tail the file (last 32K)

Block information -- Block 0

Block ID: 1073742034

Block Pool ID: BP-29566851-192.168.80.128-1734453876678

Generation Stamp: 1214

Size: 25838

Availability:

- master
- slave2
- slave1

File contents

```
192.168.80.133 true☒192.168.80.133☒-☒2025-01-05 12:55:57☒/b.html☒304☒0☒"-
"☒"Mozilla/5.0(WindowsNT10.0;Win64;x64)AppleWebKit/537.36(KHTML,likeGecko)Chrome/131.0.0.0Safari/5
37.36""-"}
192.168.80.133 true☒192.168.80.133☒-☒2025-01-05 12:55:57☒/b.html☒304☒0☒"-
"☒"Mozilla/5.0(WindowsNT10.0;Win64;x64)AppleWebKit/537.36(KHTML,likeGecko)Chrome/131.0.0.0Safari/5
37.36""-"}
192.168.80.133 true☒192.168.80.133☒-☒2025-01-05 12:55:56☒/b.html☒304☒0☒"-
"☒"Mozilla/5.0(WindowsNT10.0;Win64;x64)AppleWebKit/537.36(KHTML,likeGecko)Chrome/131.0.0.0Safari/5
```

c） 数据点击流数据模型生成（描述系统功能实现方法、运行方式、关键代码、结果截图）。（**要求截图和关键代码背景为白色。要求仅截图关键代码，关键代码不超过一页，必须有对应的文字说明**）（10 分）

1. 按照分隔符\001 拆分数据,并把 ip 设置为 key， 这样相同的 ip 就会传给同一个 reduce 中

```java
public class ClickStreamPageView {

    static class ClickStreamMapper extends Mapper<LongWritable, Text, Text, WebLogBean> {  1 个用法

        Text k = new Text();  2 个用法
        WebLogBean v = new WebLogBean();  4 个用法

        @Override
        protected void map(LongWritable key, Text value, Context context) throws IOException, Interrupted

            String line = value.toString();

            String[] fields = line.split( regex: "\001");
            if (fields.length < 9) return;

            //将切分出来的各字段set到weblogbean中
            v.set("true".equals(fields[0]) ? true : false, fields[1], fields[2], fields[3], fields[4], fi

            //只有有效记录才进入后续处理 key ip value:剩余的状态为true的；一个对象
            //ip相同的为一组
            if (v.isValid()) {
                k.set(v.getRemote_addr());
                context.write(k, v);
            }
        }
    }
```

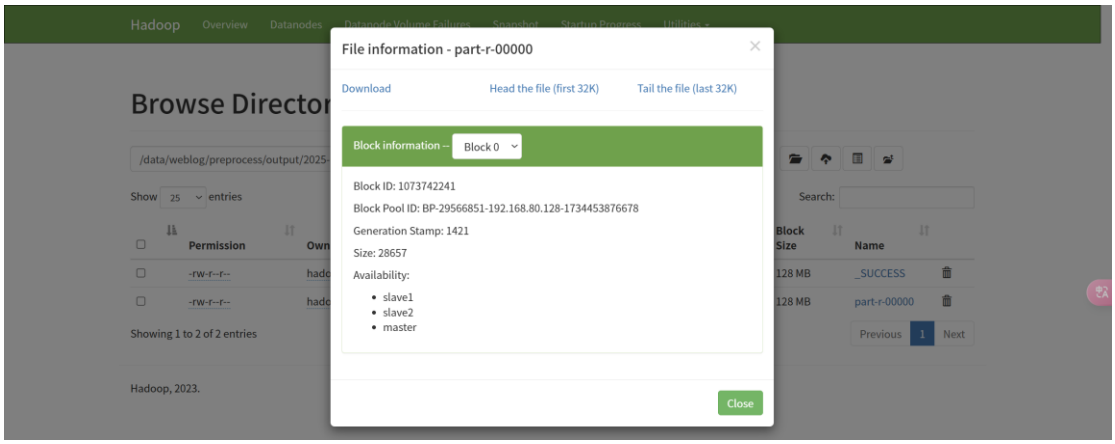
2. 在 reduce 中设置会话 id 若访问时间不超过 30 分钟，则为同一个会话

```java
static class ClickStreamReducer extends Reducer<Text, WebLogBean, NullWritable, Text> {  1 个用法

    Text v = new Text();  6 个用法

    @Override
    protected void reduce(Text key, Iterable<WebLogBean> values, Context context) throws IOException,
        ArrayList<WebLogBean> beans = new ArrayList<>();

        // 将所有 WebLogBean 加入到列表中
        try {
            for (WebLogBean bean : values) {
                WebLogBean webLogBean = new WebLogBean();
                try {
                    BeanUtils.copyProperties(webLogBean, bean);
                } catch (Exception e) {
                    e.printStackTrace();
                }
                beans.add(webLogBean);
            }

            // 按照时间排序
            Collections.sort(beans, new Comparator<WebLogBean>() {
                @Override
                public int compare(WebLogBean o1, WebLogBean o2) {
                    try {
                        Date d1 = toDate(o1.getTime_local());
                        Date d2 = toDate(o2.getTime_local());
                        if (d1 == null || d2 == null) return 0;
                        return d1.compareTo(d2);
                    } catch (Exception e) {
                        e.printStackTrace();
                        return 0;
                    }
                }
            });
```

3. 打包并在集群中运行

```
root@master:/home/hadoop/下载# hadoop jar git-1.0-SNAPSHOT.jar com.mazh.aura.clickstream.
eView /data/weblog/preprocess/log2/* /data/weblog/preprocess/log3
```

4. 在 hdfs 上查看运行结果



d) 数据点击访问页面数据模型生成（描述系统功能实现方法、运行方式、关键代码、结果截图）。（**要求截图和关键代码背景为白色。要求仅截图关键代码，每段关键代码不超过一页，必须有对应的文字说明**）（10 分）

a） 传入序列化集合 pvBean

```
PageViewsBean pvBean = new PageViewsBean();  3 个用法
Text k = new Text();  2 个用法
```

b） ：在 map 中设置会话 uid 为 key，pvBean 集合为 value 相同 uid 传入一个 rdeuce

```
protected void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {
    //一个session会话一组
    String line = value.toString();
    String[] fields = line.split( regex: "\001");
    int step = Integer.parseInt(fields[5]);
    //(String session, String remote_addr, String timestr, String request, int step, String staylong, String refe
    //299d6b78-9571-4fa9-bcc2-f2567c46df34SOH72.46.128.140SOH-SOH2013-09-18 07:58:50SOH/hadoop-zookeeper-intro/SO
    pvBean.set(fields[0], fields[1], fields[2], fields[3], fields[4], step, fields[6], fields[7], fields[8], fiel
    k.set(pvBean.getSession());
    context.write(k, pvBean);

}
```

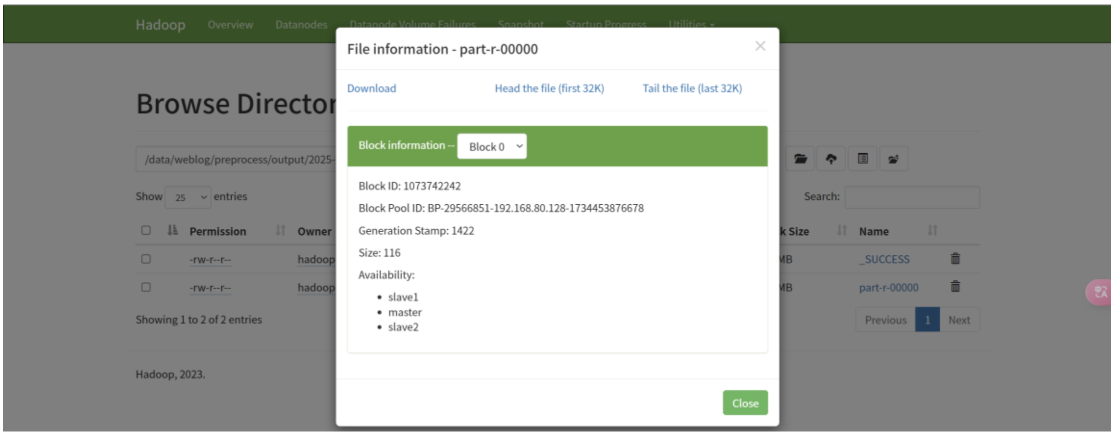
c） 在 reduce 中 通过 pvBean 集合实现排序等功能

```
protected void reduce(Text session, Iterable<PageViewsBean> pvBeans, Context context) throws IOException, InterruptedException {

    // 将pvBeans按照step排序
    ArrayList<PageViewsBean> pvBeansList = new ArrayList<~>();
    for (PageViewsBean pvBean : pvBeans) {
        PageViewsBean bean = new PageViewsBean();
        try {
            BeanUtils.copyProperties(bean, pvBean);
            pvBeansList.add(bean);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

d） 打包并运行

```
root@master:/home/hadoop/下载# hadoop jar git-1.0-SNAPSHOT.jar com.mazh.aura.clickstream.
it /data/weblog/preprocess/log4/* /data/weblog/preprocess/log5
2025-01-06 17:28:20,332 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to Res
t master/192.168.80.128:8032
2025-01-06 17:28:21,343 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for
oop-yarn/staging/root/.staging/job_1736043599412_0024
```

e） 在 hdfs 上查看结果



二、 将数据导入 hive（代码及实现步骤）

a) 将清洗后的数据导入 Hive 表（描述系统功能实现方法、运行方式、关键代码（创建表脚本及如何导入 hive）、结果截图）。（**要求截图和关键代码背景为白色。要求仅截图关键代码，每段关键代码不超过一页，必须有对应的文字说明**）（5 分）

导入数据到 ods_weblog_origin 表

```
hive> load data inpath '/data/weblog/preprocess/output/2025-01-05/log2/' overwrite into table ods_weblog_origin partition(datestr='2025-01-05');
Loading data to table dzy.ods_weblog_origin partition (datestr=2025-01-05)
OK
Time taken: 3.276 seconds
hive> select * from ods_weblog_origin;
OK
true    192.168.80.133  -       2025-01-05 12:55:59     /c.html 304     0       "-"     "Mozilla/5.0(WindowsNT10.0;Win64;x64)AppleWebKit/537.36(KHTM
L,likeGecko)Chrome/131.0.0.0Safari/537.36""-"   2025-01-05
true    192.168.80.133  -       2025-01-05 12:55:59     /c.html 304     0       "-"     "Mozilla/5.0(WindowsNT10.0;Win64;x64)AppleWebKit/537.36(KHTM
L,likeGecko)Chrome/131.0.0.0Safari/537.36""-"   2025-01-05
true    192.168.80.133  -       2025-01-05 12:55:59     /c.html 304     0       "-"     "Mozilla/5.0(WindowsNT10.0;Win64;x64)AppleWebKit/537.36(KHTM
L,likeGecko)Chrome/131.0.0.0Safari/537.36""-"   2025-01-05
true    192.168.80.133  -       2025-01-05 12:56:00     /c.html 304     0       "-"     "Mozilla/5.0(WindowsNT10.0;Win64;x64)AppleWebKit/537.36(KHTM
```

```
hive> load data inpath '/data/weblog/preprocess/output/2025-01-06/log2/' overwrite into table ods_weblog_origin partition(
datestr='2025-01-06');
Loading data to table dzy.ods_weblog_origin partition (datestr=2025-01-06)
OK
Time taken: 1.225 seconds
```

b) 将点击页面流数据导入 Hive 表（描述系统功能实现方法、运行方式、关键代码（创建表脚本及如何导入 hive）、结果截图）。（**要求截图和关键代码背景为白色。要求仅截图关键代码，每段关键代码不超过一页，必须有对应的文字说明**）（5 分）

```
hive> load data inpath '/data/weblog/preprocess/output/2025-01-05/log3/' overwrite into table ods_click_pageviews partitio
n(datestr='2025-01-05');
Loading data to table dzy.ods_click_pageviews partition (datestr=2025-01-05)
OK
Time taken: 1.101 seconds
hive> load data inpath '/data/weblog/preprocess/output/2025-01-06/log3/' overwrite into table ods_click_pageviews partitio
n(datestr='2025-01-06');
Loading data to table dzy.ods_click_pageviews partition (datestr=2025-01-06)
OK
Time taken: 1.12 seconds
hive> select * from ods_click_pageviews;
OK
a0375c52-c4dc-456f-94db-966b6368c087    192.168.80.133  -       2025-01-05 12:55:00     /c.html 1       0       "-"     "M
ozilla/5.0(WindowsNT10.0;Win64;x64)AppleWebKit/537.36(KHTML,likeGecko)Chrome/131.0.0.0Safari/537.36""-" 0       304     20
25-01-05
a0375c52-c4dc-456f-94db-966b6368c087    192.168.80.133  -       2025-01-05 12:55:00     /c.html 2       0       "-"     "M
ozilla/5.0(WindowsNT10.0;Win64;x64)AppleWebKit/537.36(KHTML,likeGecko)Chrome/131.0.0.0Safari/537.36""-" 0       304     20
25-01-05
a0375c52-c4dc-456f-94db-966b6368c087    192.168.80.133  -       2025-01-05 12:55:00     /c.html 3       1       "-"     "M
ozilla/5.0(WindowsNT10.0;Win64;x64)AppleWebKit/537.36(KHTML,likeGecko)Chrome/131.0.0.0Safari/537.36""-" 0       304     20
25-01-05
```

c) 将点击流访问表数据导入 Hive 表（描述系统功能实现方法、运行方式、关键代码（创建表脚本及如何导入 hive）、结果截图）。（**要求截图和关键代码背景为白色。要求仅截图关键代码，每段关键代码不超过一页，必须有对应的文字说明**）（5 分）

```
hive> load data inpath '/data/weblog/preprocess/output/2025-01-05/log4/' overwrite into table ods_click_stream_visit parti
tion(datestr='2025-01-05');
Loading data to table dzy.ods_click_stream_visit partition (datestr=2025-01-05)
OK
Time taken: 0.989 seconds
hive> load data inpath '/data/weblog/preprocess/output/2025-01-06/log4/' overwrite into table ods_click_stream_visit parti
tion(datestr='2025-01-06');
Loading data to table dzy.ods_click_stream_visit partition (datestr=2025-01-06)
OK
Time taken: 1.032 seconds
hive> select * from ods_click_stream_visit;
OK
a0375c52-c4dc-456f-94db-966b6368c087    192.168.80.133  2025-01-05 12:55:00     2025-01-05 12:56:58     /c.html /a.html "-
"       141     2025-01-05
5a01f18d-456b-4fd4-841a-5cf93ea3024a    192.168.80.133  2025-01-06 17:56:24     2025-01-06 17:59:58     /d.html /c.html "-
"       90      2025-01-06
Time taken: 0.346 seconds, Fetched: 2 row(s)
hive>
```

三、 产生统计指标表

a) 产生每天的 pvs 值，创建 dw_pvs_everyday 表及如何导入数据（描述系统功能实现方法、运行方式、关键代码（创建表脚本及如何导入数据）、结果截图）。（**要求截图和关键代码背景为白色。要求仅截图关键代码，每段关键代码不超过一页，必须有对应的文字说明**）（5 分）

1. 先创建一个临时表 ods_weblog_detail，将原始数据表 ods_weblog_origin 的数据经过拆分重命名后导入 ods_weblog_detail 表，为后面查询建表做准备

```
hive> create table ods_weblog_detail(
    > valid string,
    > remote_addr string,
    > remote_user string,
    > time_local string,
    > daystr string,
    > timestr string,
    > month string,
    > day string,
    > hour string,
    > request string,
    > status string,
    > body_bytes_sent string,
    > http_referer string,
    > ref_host string,
    > ref_path string,
    > ref_query string,
    > ref_query_id string,
    > http_user_agent string)
    > partitioned by(datestr string);
OK
Time taken: 0.897 seconds
```

```
hive> insert overwrite table ods_weblog_detail partition(datestr='2025-01-05')
    > select c.valid,c.remote_addr,c.remote_user,c.time_local,
    > substring(c.time_local,0,10)as daystr,
    > substring(c.time_local,12)as tmstr,
    > substring(c.time_local,6,2)as month,
    > substring(c.time_local,9,2)as day,
    > substring(c.time_local,11,3)as hour,
    > c.request,c.status,c.body_bytes_sent,c.http_referer,c.ref_host,c.ref_path,c.ref_query,c.ref_query id
,c.http_user_agent
    > from
    > (SELECT
    > a.valid,a.remote_addr,a.remote_user,a.time_local,a.request, a.status,a.body_bytes_sent,a.http_refere
r, a.http_user_agent,b.ref_host,b.ref_path,b.ref_query,b.ref_query_id
    > FROM ods_weblog_origin a LATERAL VIEW parse_url_tuple(regexp_replace(http_referer, "\"",""), 'HOST',
 'PATH','QUERY', 'QUERY:id') b as ref_host,ref_path,ref_query,ref_query_id) c;
Query ID = root_20250106221010_0948a235-1328-4f22-b718-478b9d395283
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1736043599412_0028, Tracking URL = http://master:8088/proxy/application_1736043599412_0
028/
Kill Command = /usr/local/hadoop/bin/mapred job  -kill job_1736043599412_0028
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2025-01-06 22:11:17,944 Stage-1 map = 0%,  reduce = 0%
2025-01-06 22:11:52,469 Stage-1 map = 100%,  reduce = 0%
2025-01-06 22:12:05,386 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 19.43 sec
MapReduce Total cumulative CPU time: 19 seconds 430 msec
Ended Job = job_1736043599412_0028
Stage-4 is selected by condition resolver.
```

2. 查看 ods_weblog_detail 表，已完成

```
hive> select * from ods_weblog_detail;
OK
true    192.168.80.133  -       2025-01-05 12:55:59     2025-01-05      12:55:59        01      05      1
2       /c.html 304     0       "-"     NULL    NULL    NULL    NULL    "Mozilla/5.0(WindowsNT10.0;Win64;x
64)AppleWebKit/537.36(KHTML,likeGecko)Chrome/131.0.0.0Safari/537.36""-" 2025-01-05
true    192.168.80.133  -       2025-01-05 12:55:59     2025-01-05      12:55:59        01      05      1
2       /c.html 304     0       "-"     NULL    NULL    NULL    NULL    "Mozilla/5.0(WindowsNT10.0;Win64;x
64)AppleWebKit/537.36(KHTML,likeGecko)Chrome/131.0.0.0Safari/537.36""-" 2025-01-05
true    192.168.80.133  -       2025-01-05 12:55:59     2025-01-05      12:55:59        01      05      1
2       /c.html 304     0       "-"     NULL    NULL    NULL    NULL    "Mozilla/5.0(WindowsNT10.0;Win64;x
64)AppleWebKit/537.36(KHTML,likeGecko)Chrome/131.0.0.0Safari/537.36""-" 2025-01-05
true    192.168.80.133  -       2025-01-05 12:56:00     2025-01-05      12:56:00        01      05      1
2       /c.html 304     0       "-"     NULL    NULL    NULL    NULL    "Mozilla/5.0(WindowsNT10.0;Win64;x
64)AppleWebKit/537.36(KHTML,likeGecko)Chrome/131.0.0.0Safari/537.36""-" 2025-01-05
true    192.168.80.133  -       2025-01-05 12:56:03     2025-01-05      12:56:03        01      05      1
2       /a.html 304     0       "-"     NULL    NULL    NULL    NULL    "Mozilla/5.0(WindowsNT10.0;Win64;x
64)AppleWebKit/537.36(KHTML,likeGecko)Chrome/131.0.0.0Safari/537.36""-" 2025-01-05
```

3. 聚合统计同一天的 pvs

```
hive> create table dw_pvs_everyday(pvs bigint,month string,day string);
OK
Time taken: 0.28 seconds
hive> insert into table dw_pvs_everyday
    > select count(*) as pvs,a.month as month,a.day as day from ods_weblog_detail a
    > group by a.month,a.day;
Query ID = root_20250106222319_384f40f6-c6d1-487b-8b83-3ff2eda0d1df
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
```

```
hive> select * from dw_pvs_everyday;
OK
142     01      05
91      01      06
```

b） 产生指定日期的每个小时的 pvs 值，创建 dw_pvs_everyhour_oneday 表及如何导入数据（描述系将数据从采集目录移动到预处理目录统功能实现方法、运行方式、关键代码（创建表脚本及如何导入数据）、结果截图）。（**要求截图和关键代码背景为白色。要求仅截图关键代码，每段关键代码不超过一页，必须有对应的文字说明**）（5 分）

```
hive>
    > create table  dw_pvs_everyhour_oneday(month string,day string,hour string,pvs bigint) partitioned by
 (datestr string);
OK
Time taken: 0.156 seconds
```

```
hive> insert into table dw_pvs_everyhour_oneday partition(datestr='2025-01-05')
    > select a.month as month,a.day as day,a.hour as hour,count(*) as pvs from ods_weblog_detail a
    > where a.datestr='2025-01-05' group by a.month,a.day,a.hour;
Query ID = root_20250106224349_2750b63d-da43-4901-a786-929b11801d92
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
```

```
hive> select * from dw_pvs_everyhour_oneday;
OK
01      05      12      142     2025-01-05
01      06      17      91      2025-01-05
Time taken: 0.363 seconds, Fetched: 2 row(s)
```

c） 产生每天的 page 的 pvs 值及 dw_pvs_request_everyday 表（描述系统功能实现方法、运行方式、关键代码（创建表脚本及如何导入数据）、结果截图）。（**要求截图和关键代码背景为白色。要求仅截图关键代码，每段关键代码不超过一页，必须有对应的文字说明**）（5 分）

```
hive> create table dw_pvs_request_everyday(pvs bigint,month string,day string,request string);
OK
Time taken: 0.228 seconds
hive>
```

```
hive> insert into table dw_pvs_request_everyday
    > select count(*) as pvs,a.month as month,a.day as day,a.request request from ods_weblog_detail a
    > group by a.month,a.day,a.request;
Query ID = root_20250106225048_87896f03-ff05-4a8e-8e4a-027221ae9efa
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1736043599412_0035, Tracking URL = http://master:8088/proxy/application_1736043599412_0
035/
```

```
hive> select * from dw_pvs_request_everyday;
OK
1       01      05      /
49      01      05      /a.html
30      01      05      /b.html
47      01      05      /c.html
15      01      05      /d.html
1       01      06      /
22      01      06      /a.html
26      01      06      /b.html
22      01      06      /c.html
20      01      06      /d.html
Time taken: 0.308 seconds, Fetched: 10 row(s)
```

四、 数据导入（代码及实现步骤）

a） 基于 sqoop 将数据导出 mysql（描述系统功能实现方法、运行方式、关键代码（sqoop 导出 mysql 数据库）、结果截图）。（**要求截图和关键代码背景为白色。要求仅截图关键代码，每段关键代码不超过一页，必须有对应的文字说明**）（10 分）

1. 这里使用 DataGrip 执行 MySQL 的 SQl 语句和查看表，通过 DataGrip软件远程连接位于node1虚拟机上的MySQL,在DataGrip 上编写 SQL 语句新建一个数据库 weblogs，在 weblogs 数据库中建立数据库表 dw_pvs_everyday、dw_pvs_everyhour_oneday、dw_pvs_request_page

   代码和结果如下：

   mysql 中创建数据库 数据表

```
mysql> CREATE TABLE dw_pvs_everyday (
    -> pvs varchar(32) DEFAULT NULL,
    -> month varchar(16) DEFAULT NULL,
    -> day varchar(16) DEFAULT NULL
    -> )ENGINE=InnoDB DEFAULT CHARSET=utf8;
Query OK, 0 rows affected, 1 warning (0.30 sec)
```

```
mysql> CREATE TABLE dw_pvs_everyhour_oneday (
    -> month varchar(32) DEFAULT NULL,
    -> day varchar(32) DEFAULT NULL,
    -> hour varchar(32) DEFAULT NULL,
    -> pvs varchar(32) DEFAULT NULL
    -> )ENGINE=InnoDB DEFAULT CHARSET=utf8;
Query OK, 0 rows affected, 1 warning (0.10 sec)
```

```
mysql> CREATE TABLE dw_pvs_request_page (
    -> pvs varchar(32) DEFAULT NULL,
    -> month varchar(16) DEFAULT NULL,
    -> day varchar(16) DEFAULT NULL,
    -> request varchar(16)DEFAULT NULL
    -> )ENGINE=InnoDB DEFAULT CHARSET=utf8;
Query OK, 0 rows affected, 1 warning (0.10 sec)
```

2. 通过 sqoop 工具将 hive 表数据导入 mysql 的数据库表

   代码和各 MySQL 表查询结果如下，均完成数据导入：

```
root@master:/opt/server/sqoop/bin# sqoop export --connect "jdbc:mysql://127.0.0.1:3306/weblogs?useSSl=fal
se&characterEnconding=UTF-8" --username root --password 123456 --m 1 --export-dir /user/hive/warehouse/ha
doop.db/dw_pvs_everyday --table dw_pvs_everyday  --inpute-fields-terminated-by '\001'
Warning: /opt/server/sqoop//../hcatalog does not exist! HCatalog jobs will fail.
Please set $HCAT_HOME to the root of your HCatalog installation.
```

```
root@master:/opt/server/sqoop/bin# sqoop export --connect "jdbc:mysql://127.0.0.1:3306/weblogs?useSSl=fal
se&characterEnconding=UTF-8" --username root --password 123456 --m 1 --export-dir /user/hive/warehouse/ha
doop.db/dw_pvs_everyhour_oneday/datastr=2025-01-05 --table dw_pvs_everyhour_oneday  --inpute-fields-termi
nated-by '\001'
```

```
root@master:/opt/server/sqoop/bin# sqoop export --connect "jdbc:mysql://127.0.0.1:3306/weblogs?useSSl=fal
se&characterEnconding=UTF-8" --username root --password 123456 --m 1 --export-dir /user/hive/warehouse/ha
doop.db/dw_pvs_request_page --table dw_pvs_request_page --inpute-fields-terminated-by '\001'
```

```
mysql> select * from dw_pvs_everyhour_oneday;

   142      01       05
   91       01       06
```

```
mysql> select * from dw_pvs_everyhour_oneday;
 01      05       12      142      2025-01-05
 01      06       17      91       2025-01-05
```

```
mysql> select * from dw_pvs_request_page;
1        01       05       /
49       01       05       /a.html
30       01       05       /b.html
47       01       05       /c.html
15       01       05       /d.html
1        01       06       /
22       01       06       /a.html
26       01       06       /b.html
22       01       06       /c.html
20       01       06       /d.html
```

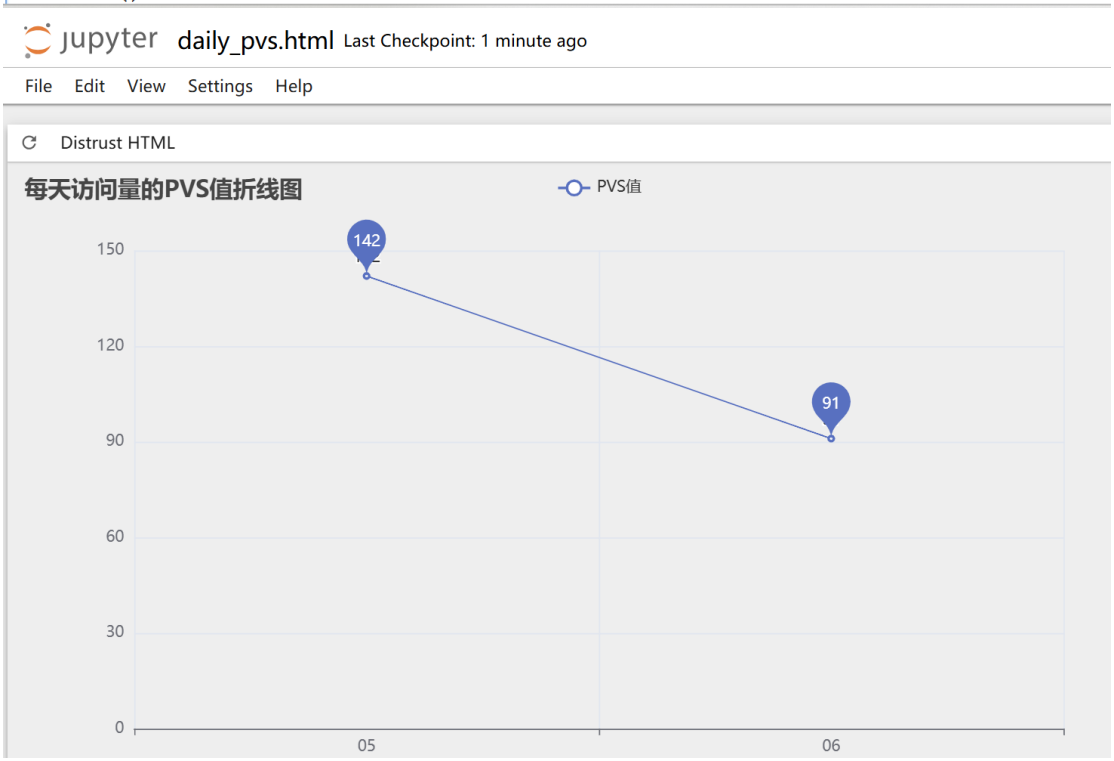
3.

五、 定制开发 web 图表（代码及实现步骤）

a) 每天访问量的 pvs 值（折线图）(描述系统功能实现方法、运行方式、关键代码、结果截图)。(**要求截图和关键代码背景为白色。要求仅截图关键代码，每段关键代码不超过一页，必须有对应的文字说明**)（5 分）

1. 连接数据库,传入数据表数据,数据可视化面

```python
import mysql.connector
from pyecharts import Line
conn = mysql.connector.connect(
    host="192.168.80.128:3306",
    user="root",
    password="123456",
    database="weblogs"
)
cursor = conn.cursor()
# 准备数据
days = [row[0] for row in data]
pvs_values = [int(row[1]) for row in data]
# 创建折线图
line = Line()
# 添加数据
line.add_xaxis(days)
line.add_yaxis(
    "PVS值",
    pvs_values,
    is_smooth=True  # 平滑曲线
)
line.set_series_opts(
    line_style_opts=opts.LineStyleOpts(width=2),  # 设置折线宽度
    markpoint_opts=opts.MarkPointOpts(
        data=[
            opts.MarkPointItem(type_="max", name="最大值"),
            opts.MarkPointItem(type_="min", name="最小值")
        ]
    ),
    label_opts=opts.LabelOpts(is_show=True)  # 显示标签
)
line.set_global_opts(title_opts=opts.TitleOpts(title="每天访问量的PVS值折线图"))# 设置全局选项
# 渲染为HTML文件
line.render("daily_pvs.html")
# 关闭数据库连接
cursor.close()
conn.close()
```

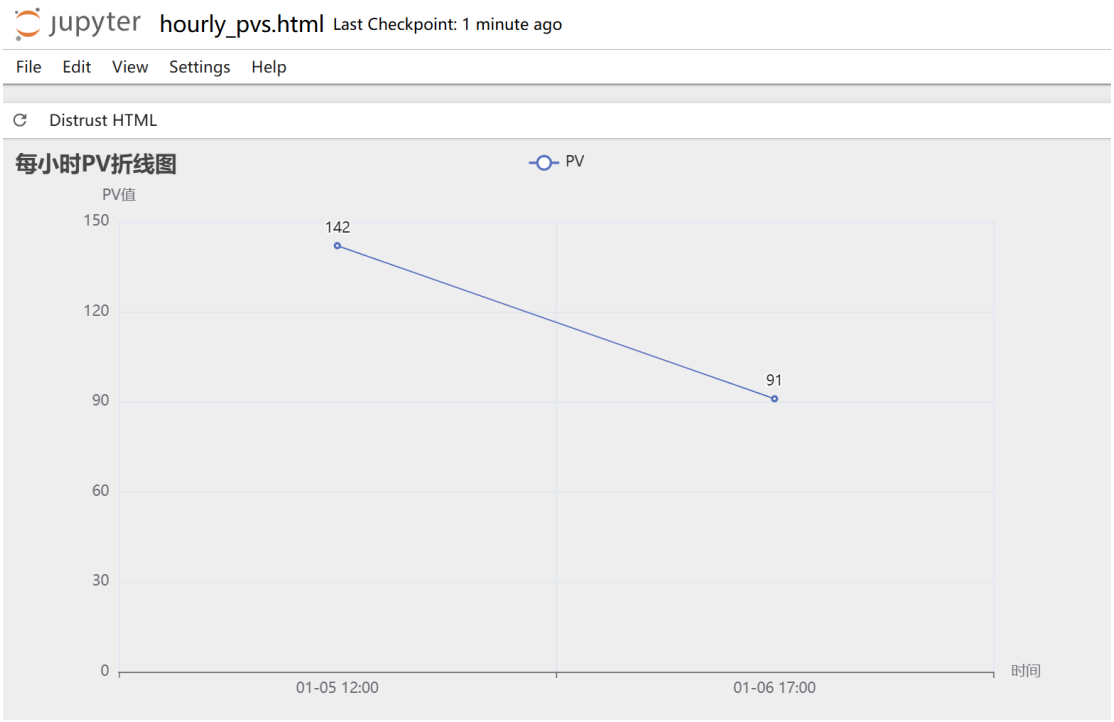


b) 指定某天的每小时的 pvs（折线图）（描述系统功能实现方法、运行方式、关键代码、结果截图）。（**要求截图和关键代码背景为白色。要求仅截图关**

键代码，每段关键代码不超过一页，必须有对应的文字说明）（5 分）

1. 连接数据库,传入数据表数据,数据可视化

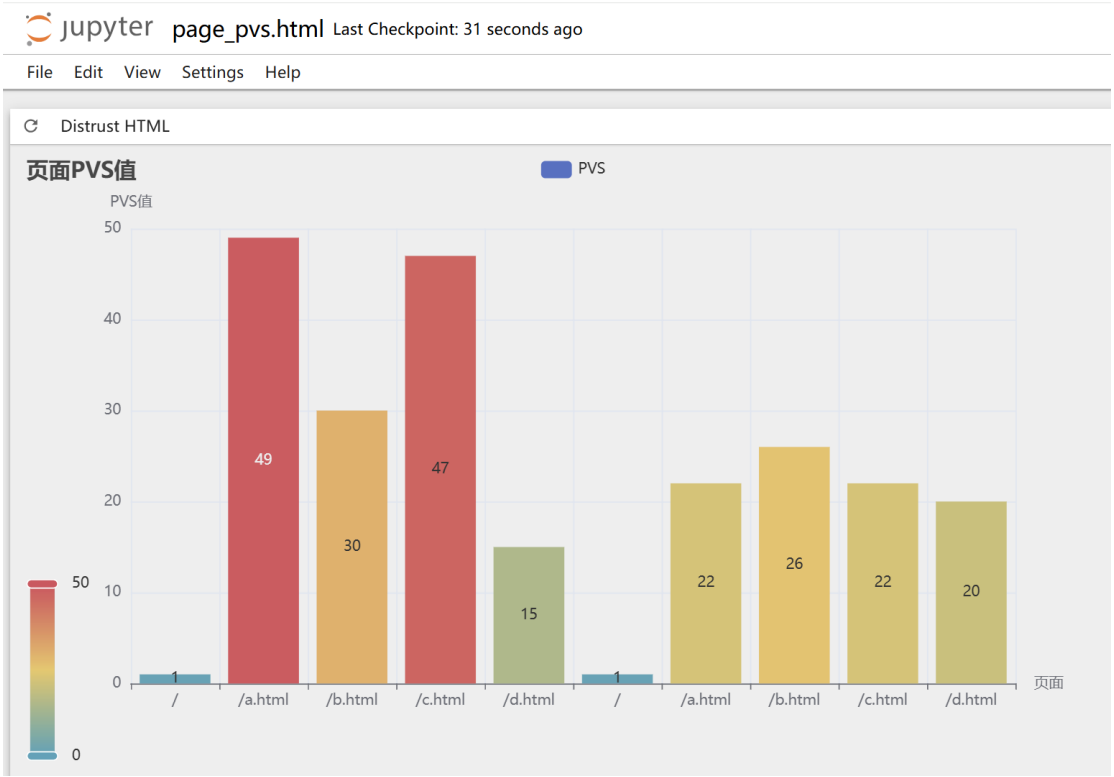
```
import mysql.connector
from pyecharts import Line

# 连接到MySQL数据库
conn = mysql.connector.connect(
    host="192.168.80.128:3306",
    user="root",
    password="123456",
    database="weblogs"
)
cursor = conn.cursor()
# 准备数据
hours = [row[0] for row in data]
pvs_values = [int(row[1]) for row in data]
# 提取时间和pvs值
times = [entry["date"] for entry in data]
pvs = [entry["pvs"] for entry in data]
# 创建折线图
line = Line()
line.add("PVS值", hours, pvs_values, is_smooth=True, line_width=2)
# 创建折线图
line = (
    Line()
    .add_xaxis(times)
    .add_yaxis("PV", pvs, is_smooth=True)
    .set_global_opts(
        title_opts=opts.TitleOpts(title="每小时PV折线图"),
        xaxis_opts=opts.AxisOpts(type_="category", name="时间"),
        yaxis_opts=opts.AxisOpts(name="PV值"),
        tooltip_opts=opts.TooltipOpts(trigger="axis"),
    )
)
# 渲染为HTML文件
line.render("hourly_pvs.html")
# 关闭数据库连接
cursor.close()
conn.close()
```



c) 统计页面 pvs 值（柱状图）(描述系统功能实现方法、运行方式、关键代码、结果截图)。(**要求截图和关键代码背景为白色。要求仅截图关键代码，每段关键代码不超过一页，必须有对应的文字说明**)（5 分）

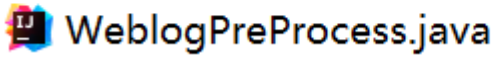
1. 连接数据库,传入数据表数据,数据可视化

```python
import mysql.connector
from pyecharts import Bar

# 连接到MySQL数据库
conn = mysql.connector.connect(
    host="192.168.80.128:3306",
    user="root",
    password="123456",
    database="weblogs"
)
cursor = conn.cursor()
# 执行SQL查询
cursor.execute("SELECT * FROM dw_pvs_request_page")
data = cursor.fetchall()
# 准备数据
days = [row[0] for row in data]
requests = [row[1] for row in data]
pvs_values = [int(row[2]) for row in data]
# 创建柱状图
bar = Bar()
# 添加数据
bar.add_xaxis(pages)  # 设置X轴数据为页面路径
bar.add_yaxis("PVS", pvs)  # 设置Y轴数据为PVS值
# 设置全局选项
bar.set_global_opts(
    title_opts=opts.TitleOpts(title="页面PVS值"),  # 设置标题
    xaxis_opts=opts.AxisOpts(name="页面"),  # 设置X轴名称
    yaxis_opts=opts.AxisOpts(name="PVS值"),  # 设置Y轴名称
    visualmap_opts=opts.VisualMapOpts(min_=0, max_=50),  # 为PVS值设置视觉映射（可以根据数据范围调整）
)
# 渲染为HTML文件
bar.render("page_pvs.html")
# 关闭数据库连接
cursor.close()
conn.close()
```



附录（源码）：

ClickStreamPageView.java	2025/1/6 17:22	IntelliJ IDEA	10 KB
ClickStreamVisit.java	2025/1/6 14:51	IntelliJ IDEA	7 KB



daily_pvs.html	2025/1/7 19:02
hourly_pvs.html	2025/1/7 19:33
page_pvs.html	2025/1/7 19:38

dw_pvs_everyday.sql	2025/1/7 19:59
dw_pvs_everyhour_oneday.sql	2025/1/7 19:59
dw_pvs_request_page.sql	2025/1/7 19:59