如果图片不显示，请 git clone到本地进行查看

git clone git@github.com:wanghuan2054/geektime.git

Hive作业

目录:

如果图片不显示，请 git clone到本地进行查看

Hive 建表

```
create table hive_sql_test1.t_user(
userid bigint,
sex String,
age int,
occupation String,
zipcode String
)
row format delimited fields terminated by ':';
load data local inpath '/home/hive/users.dat'  overwrite into  table
hive_sql_test1.t_user;


create table hive_sql_test1.t_movie
(
movieid bigint,
moviename string,
movietype string
)
row format delimited fields terminated by ':';

load data local inpath '/home/hive/movies.dat' overwrite into table
hive_sql_test1.t_movie;


create table hive_sql_test1.t_rating
(
userid bigint,
movieid bigint,
rate double,
times  string
)
row format delimited fields terminated by ':';
load data local inpath '/home/hive/ratings.dat' overwrite into table
hive_sql_test1.t_rating;
```

题目1

简单：展示电影ID为2116这部电影各年龄段的平均影评分
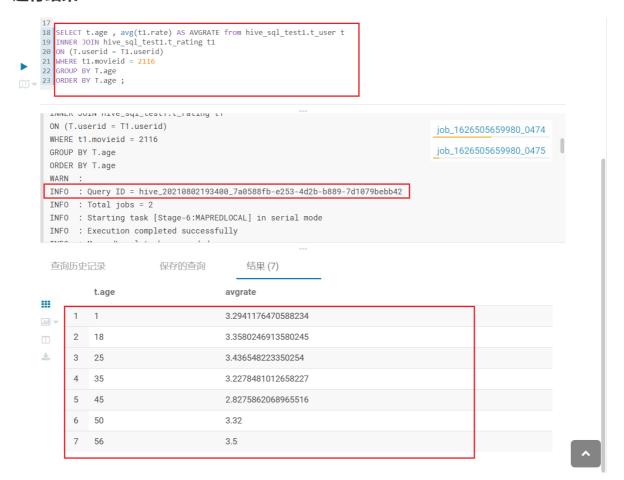
解法1 （不考虑数据倾斜）

不考虑数据倾斜问题

SQL

```
SELECT t.age , avg(t1.rate) AS AVGRATE from hive_sql_test1.t_user t
INNER JOIN hive_sql_test1.t_rating t1
ON (T.userid = T1.userid)
WHERE t1.movieid = 2116
GROUP BY T.age
ORDER BY T.age ;
```

运行结果


```sql
17
18  SELECT t.age , avg(t1.rate) AS AVGRATE from hive_sql_test1.t_user t
19  INNER JOIN hive_sql_test1.t_rating t1
20  ON (T.userid = T1.userid)
21  WHERE t1.movieid = 2116
22  GROUP BY T.age
23  ORDER BY T.age ;
```

```
INNER JOIN hive_sql_test1.t_rating t1
ON (T.userid = T1.userid)                                    job_1626505659980_0474
WHERE t1.movieid = 2116
GROUP BY T.age                                               job_1626505659980_0475
ORDER BY T.age
WARN  :
INFO  : Query ID = hive_20210802193400_7a0588fb-e253-4d2b-b889-7d1079bebb42
INFO  : Total jobs = 2
INFO  : Starting task [Stage-6:MAPREDLOCAL] in serial mode
INFO  : Execution completed successfully
```

查询历史记录 保存的查询 结果 (7)

	t.age	avgrate
1	1	3.2941176470588234
2	18	3.3580246913580245
3	25	3.436548223350254
4	35	3.2278481012658227
5	45	2.8275862068965516
6	50	3.32
7	56	3.5

解法2（考虑数据倾斜）

考虑数据倾斜问题，假设2116这部电影评分记录集中在28-30这个年龄段,其它年龄段寥寥无几

解决数据倾斜的方法很多，本文列出我的一条思路

1. 首先按照age和occupation 或者 age 和zipcode 进行分组， 对原先按照age分组所有age相同的数据会同时shuffle到一个reduce节点，按照新分组会将之前倾斜值的倾斜程度变为原来的1/N， N=（occupation 或者zipcode 值）%reducer 的值的不同个数

2. 将1的子查询结果（GROUP BY T.age ,t.zipcode 预聚合），之后再统一对age进行分组求avg，降低了之前一个节点处理所有数据量。

   核心思路，分治思想，对任务进行拆分，先多字段分组求和， 后将中间聚合结果统一求avg。

SQL

```sql
SELECT X.age as age , avg(X.sumrate) as avgrate FROM
(
SELECT t.age , t.zipcode ,  sum(t1.rate) AS sumrate from hive_sql_test1.t_user t
INNER JOIN hive_sql_test1.t_rating t1
ON (T.userid = T1.userid)
WHERE t1.movieid = 2116
GROUP BY T.age ,t.zipcode
) X
GROUP BY X.age
ORDER BY X.age ;
```

题目2

中等：找出男性评分最高且评分次数超过50次的10部电影，展示电影名，平均影评分和评分次数

SQL

```sql
SELECT t.sex AS sex,
       t2.moviename AS name,
       avg(t1.rate) AS avgrate,
       count(*) AS total
FROM hive_sql_test1.t_user t
INNER JOIN hive_sql_test1.t_rating t1 ON (T.userid = T1.userid)
INNER JOIN hive_sql_test1.t_movie t2 on(t1.movieid = t2.movieid)
WHERE t.sex = 'M'
GROUP BY T.sex,
         t2.moviename
HAVING count(*) > 50
ORDER BY avgrate DESC
LIMIT 10 ;
```

运行结果

	sex	name	avgrate	total
1	M	Sanjuro (1962)	4.639344262295082	61
2	M	Godfather, The (1972)	4.583333333333333	1740
3	M	Seven Samurai (The Magnificent Seven) (Shichinin no samurai) (1954)	4.576628352490421	522
4	M	Shawshank Redemption, The (1994)	4.560625	1600
5	M	Raiders of the Lost Ark (1981)	4.520597322348094	1942
6	M	Usual Suspects, The (1995)	4.518248175182482	1370
7	M	Star Wars: Episode IV - A New Hope (1977)	4.495307167235495	2344
8	M	Schindler's List (1993)	4.49141503848431	1689
9	M	Paths of Glory (1957)	4.485148514851486	202
10	M	Wrong Trousers, The (1993)	4.478260869565218	644

题目3(选做)

困难：找出影评次数最多的女士所给出最高分的10部电影的平均影评分，展示电影名和平均影评分（可使用多行SQL）

写法1

嵌套子查询

SQL

```
SELECT t3.moviename,
       avg(t2.rate) AS avgrate
FROM (SELECT a.movieid , a.rate
   FROM  (SELECT t.userid,
          count(*) AS total
   FROM hive_sql_test1.t_user t
   INNER JOIN hive_sql_test1.t_rating t1 ON (T.userid = T1.userid)
   WHERE t.sex = 'F'
   GROUP BY T.userid
   ORDER BY total DESC
   LIMIT 1) x
   inner join hive_sql_test1.t_rating a on (x.userid = a.userid)
   ORDER BY a.rate DESC
   LIMIT 10)  t1
INNER JOIN hive_sql_test1.t_rating t2 on(t1.movieid = t2.movieid)
INNER JOIN hive_sql_test1.t_movie t3 on(t2.movieid = t3.movieid)
GROUP BY t2.movieid , t3.moviename
ORDER BY avgrate DESC ;
```

运行结果

```
1  SELECT t3.moviename,
2        avg(t2.rate) AS avgrate
3  FROM (SELECT a.movieid , a.rate
4     FROM  (SELECT t.userid,
5           count(*) AS total
6     FROM hive_sql_test1.t_user t
7     INNER JOIN hive_sql_test1.t_rating t1 ON (T.userid = T1.userid)
8     WHERE t.sex = 'F'
9     GROUP BY T.userid
10    ORDER BY total DESC
11    LIMIT 1) x
12    inner join hive_sql_test1.t_rating a on (x.userid = a.userid)
13    ORDER BY a.rate DESC
14    LIMIT 10)  t1
15 INNER JOIN hive_sql_test1.t_rating t2 on(t1.movieid = t2.movieid)
16 INNER JOIN hive_sql_test1.t_movie t3 on(t2.movieid = t3.movieid)
17 GROUP BY t2.movieid , t3.moviename
18 ORDER BY avgrate DESC ;
```

```
WARN :
INFO  : Query ID = hive_20210803174349_f3dca635-db83-4a2f-b5b8-94ee55cd035b     job_1626505659980_1535
INFO  : Total jobs = 12
INFO  : Starting task [Stage-26:MAPREDLOCAL] in serial mode                      job_1626505659980_1545
```

查询历史记录   保存的查询   结果 (10)

		t3.moviename	avgrate
	1	Rear Window (1954)	4.476190476190476
	2	City Lights (1931)	4.387453874538745
	3	His Girl Friday (1940)	4.249370277078086
	4	Hands on a Hard Body (1996)	4.163043478260869
	5	Annie Hall (1977)	4.14167916041979
	6	Crumb (1994)	4.063136456211812
	7	Blazing Saddles (1974)	4.047363717605005

```
10    ORDER BY total DESC
11    LIMIT 1) x
12    inner join hive_sql_test1.t_rating a on (x.userid = a.userid)
13    ORDER BY a.rate DESC
14    LIMIT 10)  t1
15  INNER JOIN hive_sql_test1.t_rating t2 on(t1.movieid = t2.movieid)
16  INNER JOIN hive_sql_test1.t_movie t3 on(t2.movieid = t3.movieid)
17  GROUP BY t2.movieid , t3.moviename
18  ORDER BY avgrate DESC ;
```

```
INFO  : Query ID = hive_20210803174349_f3dca635-db83-4a2f-b5b8-94ee55cd035b
INFO  : Total jobs = 12
INFO  : Starting task [Stage-26:MAPREDLOCAL] in serial mode
```

job_1626505659980_1535

job_1626505659980_1545

查询历史记录　　　保存的查询　　　结果 (10)

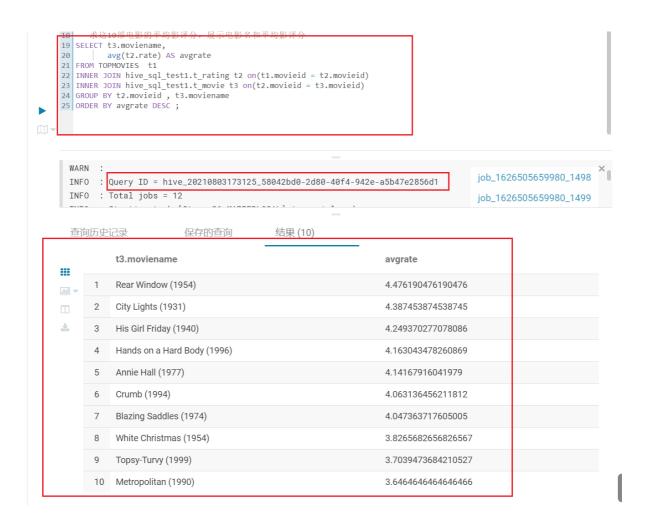|		t3.moviename	avgrate
	1	Rear Window (1954)	4.476190476190476
	2	City Lights (1931)	4.387453874538745
	3	His Girl Friday (1940)	4.249370277078086
	4	Hands on a Hard Body (1996)	4.163043478260869
	5	Annie Hall (1977)	4.14167916041979
	6	Crumb (1994)	4.063136456211812
	7	Blazing Saddles (1974)	4.047363717605005
	8	White Christmas (1954)	3.8265682656826567
	9	Topsy-Turvy (1999)	3.7039473684210527
	10	Metropolitan (1990)	3.6464646464646466

写法2

WITH ..... AS .....

SQL

```sql
-- 找出影评次数最多的女士
WITH Rating_CNT AS
  (SELECT t.userid,
          count(*) AS total
   FROM hive_sql_test1.t_user t
   INNER JOIN hive_sql_test1.t_rating t1 ON (T.userid = T1.userid)
   WHERE t.sex = 'F'
   GROUP BY T.userid
   ORDER BY total DESC
   LIMIT 1),
-- 找出该女士评分最高的10部电影
TOPMOVIES AS
  (SELECT a.movieid , a.rate
   FROM  Rating_CNT x
   inner join hive_sql_test1.t_rating a on (x.userid = a.userid)
   ORDER BY a.rate DESC
   LIMIT 10)

-- select t2.* from TOPMOVIES t2;

-- 求这10部电影的平均影评分，展示电影名和平均影评分
SELECT t3.moviename,
       avg(t2.rate) AS avgrate
FROM TOPMOVIES  t1
```

```sql
    INNER JOIN hive_sql_test1.t_rating t2 on(t1.movieid = t2.movieid)
    INNER JOIN hive_sql_test1.t_movie t3 on(t2.movieid = t3.movieid)
    GROUP BY t2.movieid , t3.moviename
    ORDER BY avgrate DESC ;
```

运行结果

```sql
1  -- 找出影评次数最多的女士
2  WITH Rating_CNT AS
3    (SELECT t.userid,
4          count(*) AS total
5     FROM hive_sql_test1.t_user t
6     INNER JOIN hive_sql_test1.t_rating t1 ON (T.userid = T1.userid)
7     WHERE t.sex = 'F'
8     GROUP BY T.userid
9     ORDER BY total DESC
10    LIMIT 1),
11 -- 找出该女士评分最高的10部电影
12 TOPMOVIES AS
13   (SELECT a.movieid , a.rate
14    FROM  Rating_CNT x
15    inner join hive_sql_test1.t_rating a on (x.userid = a.userid)
16    ORDER BY a.rate DESC
17    LIMIT 10)
18 -- 求这10部电影的平均影评分，展示电影名和平均影评分
19 SELECT t3.moviename,
20       avg(t2.rate) AS avgrate
21 FROM TOPMOVIES  t1
22 INNER JOIN hive_sql_test1.t_rating t2 on(t1.movieid = t2.movieid)
23 INNER JOIN hive_sql_test1.t_movie t3 on(t2.movieid = t3.movieid)
24 GROUP BY t2.movieid , t3.moviename
25 ORDER BY avgrate DESC ;
```

```
WARN  :
INFO  : Query ID = hive_20210803173125_58042bd0-2d80-40f4-942e-a5b47e2856d1      job_1626505659980_1498
INFO  : Total jobs = 12                                                          job_1626505659980_1499
```

查询历史记录　　　保存的查询　　　结果 (10)

	t3.moviename	avgrate
1	Rear Window (1954)	4.476190476190476
2	City Lights (1931)	4.387453874538745
3	His Girl Friday (1940)	4.249370277078086
4	Hands on a Hard Body (1996)	4.163043478260869

```
18    求这10部电影的平均影评分，展示电影名和平均影评分
19  SELECT t3.moviename,
20         avg(t2.rate) AS avgrate
21  FROM TOPMOVIES  t1
22  INNER JOIN hive_sql_test1.t_rating t2 on(t1.movieid = t2.movieid)
23  INNER JOIN hive_sql_test1.t_movie t3 on(t2.movieid = t3.movieid)
24  GROUP BY t2.movieid , t3.moviename
25  ORDER BY avgrate DESC ;
```

```
WARN  :
INFO  : Query ID = hive_20210803173125_58042bd0-2d80-40f4-942e-a5b47e2856d1      job_1626505659980_1498
INFO  : Total jobs = 12                                                          job_1626505659980_1499
```

查询历史记录   保存的查询   结果 (10)

		t3.moviename	avgrate
	1	Rear Window (1954)	4.476190476190476
	2	City Lights (1931)	4.387453874538745
	3	His Girl Friday (1940)	4.249370277078086
	4	Hands on a Hard Body (1996)	4.163043478260869
	5	Annie Hall (1977)	4.14167916041979
	6	Crumb (1994)	4.063136456211812
	7	Blazing Saddles (1974)	4.047363717605005
	8	White Christmas (1954)	3.8265682656826567
	9	Topsy-Turvy (1999)	3.7039473684210527
	10	Metropolitan (1990)	3.6464646464646466

GeekFileFormat实现

项目源码位置：https://github.com/wanghuan2054/geektime/tree/master/0801HiveSQL

POM文件

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>org.example</groupId>
    <artifactId>HiveCustomFormat</artifactId>
    <version>1.0</version>

    <properties>
        <maven.compiler.source>8</maven.compiler.source>
        <maven.compiler.target>8</maven.compiler.target>
        <java.version>1.8</java.version>
        <hadoop.version>3.1.3</hadoop.version>
        <hive.version>3.1.2</hive.version>
    </properties>
    <dependencies>
    <dependency>
        <groupId>org.apache.hadoop</groupId>
        <artifactId>hadoop-client</artifactId>
```

```
            <version>${hadoop.version}</version>
        </dependency>
        <dependency>
            <groupId>org.apache.hadoop</groupId>
            <artifactId>hadoop-common</artifactId>
            <version>${hadoop.version}</version>
        </dependency>
        <dependency>
            <groupId>org.apache.hive</groupId>
            <artifactId>hive-exec</artifactId>
            <version>${hive.version}</version>
        </dependency>
        </dependencies>
</project>
```

GeekTextInputFormat核心代码（解密）

```
@Override
    public boolean next(LongWritable key, Text value) throws IOException {
        while(reader.next(key, text)) {
            String strReplace =
text.toString().toLowerCase().replaceAll("ge{2,256}k", "");
            Text txtReplace = new Text();
            txtReplace.set(strReplace);
            value.set(txtReplace.getBytes(), 0, txtReplace.getLength());
            return true;
        }
            return false;
    }
```

GeekTextOutputFormat核心代码(加密)

```
    @Override
    public void write(Writable w) throws IOException {
        // Get input data
        Text tmpText = ((Text) w);
        StringBuilder sb = new StringBuilder() ;
        String str =  tmpText.toString() ;
        String[] words = str.toString().toLowerCase().split("\\s+") ;
        // 生成一个 min-max 之间的随机整数
        int min = 2  ;
        int max = 256 ;
        int randomNum = getRandomNum(min,max) ;
        int validWordCnt = 0 ;
        boolean flag = true ;
        for (String word: words) {
            // 如果是有效单词， 统计数+1 ， 方便后面生成gee...K ， 生成中间e的个数使用
            if(checkValidWord(word)){
                validWordCnt++ ;
            }
            // 随机数个单词标记
            randomNum -- ;
            // 行首单词不加空格
            if (flag){
```

```java
                flag =  false ;
                sb.append(word) ;
            }
            else  {
                // 非行首单词输出空格+单词
                sb.append(" " + word) ;
            }
            // 输出指定生成随机数个单词后，进行插入操作
            if (0 == randomNum){
                // 生成指定min-max区间的随机整数
                randomNum = getRandomNum(min,max) ;
                // 利用之前产生的有效单词数，生成gee...K
                String insertWord =  generateGeek(validWordCnt);
                sb.append(" " + insertWord);
                // 插入单词之后 ，有效单词总数从新开始计数
                validWordCnt = 0  ;
            }
        }
        String strReplace =  sb.toString();
        Text output = new Text();
        output.set(strReplace);
        text.set(output.getBytes(), 0, output.getLength());
        writer.write(text);
    }
    public static int getRandomNum(int min , int max){
        return (int) (Math.random()*(max-min)+min) ;
    }
    // 产生中间连续N个e
    public static String generateGeek(int eCnt){
        String prefix = "g" ;
        String loopStr = "e" ;
        String suffix = "k" ;
        StringBuilder sb = new StringBuilder(prefix) ;
        for (int i = 0; i < eCnt; i++) {
            sb.append(loopStr) ;
        }
        return sb.append(suffix).toString();
    }


    // 检测是否为有效字符 ，非gee...k单词
    public static boolean checkValidWord(String word){
        boolean res = true;

        String tmp = word;
        tmp = tmp.replaceAll("\\p{P}", "");
        if (word.length() != tmp.length()) {
            res = false;
        }
        // 定义模式匹配
        Pattern pattern = Pattern.compile("ge{2,256}k");
        Matcher matcher = pattern.matcher(word) ;
        // matches 进行全局匹配 与给定的字符串进行全匹配
        // 测试word 中包含geek ，返回false
        res = !matcher.matches();
        return res;
    }
```

上传数据和jar到hdfs

登录到jikehadoop01节点

```
#上传jar
[student@jikehadoop01 ~]$ hadoop fs -put
/home/student/wanghuan/HiveCustomFormat-1.0.jar /user/student/wanghuan/

[student@jikehadoop01 ~]$ hadoop fs -ls /user/student/wanghuan
Found 3 items
-rw-r--r--   3 student student      18479 2021-08-05 00:36
/user/student/wanghuan/HiveCustomFormat-1.0.jar
-rw-r--r--   3 student student       2110 2021-07-11 22:54
/user/student/wanghuan/phone_data.txt
-rw-r--r--   3 student student       8870 2021-07-11 22:58
/user/student/wanghuan/wanghuanmr-1.0-SNAPSHOT.jar

#上传数据文件
[student@jikehadoop01 ~]$ hadoop fs -put /home/student/wanghuan/geek.txt
/user/student/wanghuan/
[student@jikehadoop01 ~]$ hadoop fs -ls /user/student/wanghuan/
Found 4 items
-rw-r--r--   3 student student      18479 2021-08-05 00:36
/user/student/wanghuan/HiveCustomFormat-1.0.jar
-rw-r--r--   3 student student        730 2021-08-05 00:41
/user/student/wanghuan/geek.txt
-rw-r--r--   3 student student       2110 2021-07-11 22:54
/user/student/wanghuan/phone_data.txt
-rw-r--r--   3 student student       8870 2021-07-11 22:58
/user/student/wanghuan/wanghuanmr-1.0-SNAPSHOT.jar
```

Hive CLI 操作

登录到 Alibaba Cloud Elastic Compute Service

创建Hive数据库

```
hive> create database wanghuan ;
OK
Time taken: 0.077 seconds
```

add jar

```
hive>add jar hdfs:///user/student/wanghuan/HiveCustomFormat-1.0.jar;
Added [/tmp/e80feb11-9d66-4d37-a662-2c90e106a810_resources/HiveCustomFormat-
1.0.jar] to class path
Added resources: [hdfs:///user/student/wanghuan/HiveCustomFormat-1.0.jar]
```

hive支持本地模式

```
# hive的Job交给yarn执行太慢，hive支持本地模式
set hive.exec.mode.local.auto=true;
```

验证解密是否成功

创建Hive表

```
# 切换数据库
hive> use wanghuan ;
OK
Time taken: 0.029 seconds
# 创建表
create table IF NOT EXISTS geektime_input (
    content string
)
ROW FORMAT DELIMITED FIELDS TERMINATED BY '\n' LINES TERMINATED BY '\n'
stored as
INPUTFORMAT 'hive.inoutformat.GeekTextInputFormat'
OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat' ;

hive> create table IF NOT EXISTS geektime_input (
    >     content string
    > )
    > ROW FORMAT DELIMITED FIELDS TERMINATED BY '\n' LINES TERMINATED BY '\n'
    > stored as
    > INPUTFORMAT 'hive.inoutformat.GeekTextInputFormat'
    > OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
;
OK
Time taken: 0.24 seconds
```

Load data

```
# 本地加载
load data local inpath '/home/student/wanghuan/geek.txt' overwrite into table
geektime_input;
# hdfs 加载
load data inpath '/user/student/wanghuan/geek.txt' overwrite into table
geektime_input;
```

查看表（验证GeekTextInputFormat解密）

```
例如：    This notebook can be geeeek used to geek install gek on all geeeek
worker nodes, run data generation, and create the TPCDS geeeeeeeeek database.
解密为：This notebook can be used to install gek on all worker nodes, run data
generation, and create the TPCDS database.

hive> show tables ;
OK
geektime_input
Time taken: 0.084 seconds, Fetched: 1 row(s)
hive> select * from geektime_input ;
OK
this notebook can be  used to  install gek on all  worker nodes, run data
generation, and create the tpcds  database.
this notebook can be  used to  install gek on all  worker nodes, run data
generation, and create the tpcds  database.
```

```
this notebook can be  used to  install gek on all  worker nodes, run data
generation, and create the tpcds  database.
this notebook can be  used to  install gek on all  worker nodes, run data
generation, and create the tpcds  database.
this notebook can be  used to  install gek on all  worker nodes, run data
generation, and create the tpcds  database.
Time taken: 0.233 seconds, Fetched: 5 row(s)
```

Insert data

```
hive> insert into geektime_input (content) values("This notebook can be geeeek
used to geek install gek on all geeeek worker nodes, run data generation, and
create the TPCDS geeeeeeeeek database.");
Query ID = student_20210805004916_1da574e6-8bd7-4652-ad9f-b10f6b9e6307

# 查看刚才插入的数据，被成功解密显示
hive> select * from geektime_input ;
OK
this notebook can be  used to  install gek on all  worker nodes, run data
generation, and create the tpcds  database.
wang huan can be  kafka to  flink gek on all  hudi nodes, spark data generation,
and hadoop the nihao  database.
wang huan can be  kafka to  flink gek on all  hudi nodes, spark data generation,
and hadoop the nihao  database.
this notebook can be  used to  install gek on all  worker nodes, run data
generation, and create the tpcds  database.
this notebook can be  used to  install gek on all  worker nodes, run data
generation, and create the tpcds  database.
this notebook can be  used to  install gek on all  worker nodes, run data
generation, and create the tpcds  database.
this notebook can be  used to  install gek on all  worker nodes, run data
generation, and create the tpcds  database.
this notebook can be  used to  install gek on all  worker nodes, run data
generation, and create the tpcds  database.
this notebook can be  used to  install gek on all  worker nodes, run data
generation, and create the tpcds  database.
Time taken: 0.042 seconds, Fetched: 9 row(s)
```

验证加密是否成功

创建Hive表

```
# 切换数据库
hive> use wanghuan ;
OK
Time taken: 0.029 seconds
# 创建表
create table IF NOT EXISTS geektime_output (
   content string
)
ROW FORMAT DELIMITED
  FIELDS TERMINATED BY '\n'
  LINES TERMINATED BY '\n'
stored as
INPUTFORMAT 'org.apache.hadoop.mapred.TextInputFormat'
```

```
OUTPUTFORMAT 'hive.inoutformat.GeekTextOutputFormat' ;

hive> create table IF NOT EXISTS geektime_output (
    >     content string
    > )
    > ROW FORMAT DELIMITED
    >   FIELDS TERMINATED BY '\n'
    >   LINES TERMINATED BY '\n'
    > stored as
    > INPUTFORMAT 'org.apache.hadoop.mapred.TextInputFormat'
    > OUTPUTFORMAT 'hive.inoutformat.GeekTextOutputFormat' ;
OK
Time taken: 0.104 seconds
```

Load data

```
# 本地加载
load data local inpath '/home/student/wanghuan/geek.txt' overwrite into table
geektime_output;
# hdfs 加载
load data inpath '/user/student/wanghuan/geek.txt' overwrite into table
geektime_output;
```

查看表 (验证GeekTextInputFormat加密)

```
例如：    This notebook can be used to install gek on all worker nodes, run data
generation, and create the TPCDS database.
加密为：This notebook can be geeeek used to geek install gek on all geeeek worker
nodes, run data generation, and create the TPCDS geeeeeeeeek database.


hive> show tables;
OK
geektime_input
geektime_output
Time taken: 0.037 seconds, Fetched: 2 row(s)

# 旧数据没有加密
hive> select * from geektime_output ;
OK
This notebook can be geeeek used to geek install gek on all geeeek worker nodes,
run data generation, and create the TPCDS geeeeeeeeek database.
This notebook can be geeeek used to geek install gek on all geeeek worker nodes,
run data generation, and create the TPCDS geeeeeeeeek database.
This notebook can be geeeek used to geek install gek on all geeeek worker nodes,
run data generation, and create the TPCDS geeeeeeeeek database.
This notebook can be geeeek used to geek install gek on all geeeek worker nodes,
run data generation, and create the TPCDS geeeeeeeeek database.
This notebook can be geeeek used to geek install gek on all geeeek worker nodes,
run data generation, and create the TPCDS geeeeeeeeek database.
Time taken: 0.046 seconds, Fetched: 5 row(s)

# 插入新数据
hive> insert into geektime_output (content) values("This notebook can be used to
install gek on all worker nodes, run data generation, and create the TPCDS
database.");
```

```
# 查看新数据是否加密
hive> select * from geektime_output ;
OK
this notebook geek can be geek used to install gek geeeek on all geek worker
nodes, run geeek data generation, and create the geeeeek tpcds database.
This notebook can be geeeek used to geek install gek on all geeeek worker nodes,
run data generation, and create the TPCDS geeeeeeeeek database.
This notebook can be geeeek used to geek install gek on all geeeek worker nodes,
run data generation, and create the TPCDS geeeeeeeeek database.
This notebook can be geeeek used to geek install gek on all geeeek worker nodes,
run data generation, and create the TPCDS geeeeeeeeek database.
This notebook can be geeeek used to geek install gek on all geeeek worker nodes,
run data generation, and create the TPCDS geeeeeeeeek database.
This notebook can be geeeek used to geek install gek on all geeeek worker nodes,
run data generation, and create the TPCDS geeeeeeeeek database.
```

同时验证加密解密是否成功

创建Hive表

```
# 切换数据库
hive> use wanghuan ;
OK
Time taken: 0.029 seconds
# 创建表
create table IF NOT EXISTS geektime_inoutput (
    content string
)
ROW FORMAT DELIMITED
  FIELDS TERMINATED BY '\n'
  LINES TERMINATED BY '\n'
stored as
INPUTFORMAT 'hive.inoutformat.GeekTextInputFormat'
OUTPUTFORMAT 'hive.inoutformat.GeekTextOutputFormat' ;

hive> create table IF NOT EXISTS geektime_inoutput (
    >     content string
    > )
    > ROW FORMAT DELIMITED
    >   FIELDS TERMINATED BY '\n'
    >   LINES TERMINATED BY '\n'
    > stored as
    > INPUTFORMAT 'hive.inoutformat.GeekTextInputFormat'
    > OUTPUTFORMAT 'hive.inoutformat.GeekTextOutputFormat' ;
OK
Time taken: 0.094 seconds
```

Load data

```
# 本地加载
load data local inpath '/home/student/wanghuan/geek.txt' overwrite into table
geektime_inoutput;
# hdfs 加载
load data inpath '/user/student/wanghuan/geek.txt' overwrite into table
geektime_inoutput;
```

查看表（验证加密解密）

```
hive> show tables;
OK
geektime_inoutput
geektime_input
geektime_output
Time taken: 0.029 seconds, Fetched: 3 row(s)


# 数据已经解密
hive> select * from geektime_inoutput;
OK
this notebook can be  used to  install gek on all  worker nodes, run data
generation, and create the tpcds  database.
this notebook can be  used to  install gek on all  worker nodes, run data
generation, and create the tpcds  database.
this notebook can be  used to  install gek on all  worker nodes, run data
generation, and create the tpcds  database.
this notebook can be  used to  install gek on all  worker nodes, run data
generation, and create the tpcds  database.
this notebook can be  used to  install gek on all  worker nodes, run data
generation, and create the tpcds  database.


# 插入新数据
hive> insert into geektime_inoutput (content) values("This notebook can be used
to install gek on all worker nodes, run data generation, and create the TPCDS
database.");

# 查看新数据已经是解密过后的数据
hive> select * from geektime_inoutput ;
OK
this notebook can  be used to  install gek  on all worker nodes,  run data
generation, and  create the tpcds  database.
this notebook can  be used to install gek  on all worker nodes, run  data
generation,  and create the tpcds  database.
this notebook can be  used to  install gek on all  worker nodes, run data
generation, and create the tpcds  database.
this notebook can be  used to  install gek on all  worker nodes, run data
generation, and create the tpcds  database.
this notebook can be  used to  install gek on all  worker nodes, run data
generation, and create the tpcds  database.
this notebook can be  used to  install gek on all  worker nodes, run data
generation, and create the tpcds  database.
this notebook can be  used to  install gek on all  worker nodes, run data
generation, and create the tpcds  database.
this notebook can be  used to  install gek on all  worker nodes, run data
generation, and create the tpcds  database.
```

表Schema

geektime_input

InputFormat: hive.inoutformat.GeekTextInputFormat
OutputFormat: org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat

```
hive> desc formatted geektime_input ;
OK
# col_name                data_type                comment

content                   string

# Detailed Table Information
Database:                 wanghuan
OwnerType:                USER
Owner:                    student
CreateTime:               Thu Aug 05 00:39:46 CST 2021
LastAccessTime:           UNKNOWN
Retention:                0
Location:
hdfs://jikehadoop01:8020/user/hive/warehouse/wanghuan.db/geektime_input
Table Type:               MANAGED_TABLE
Table Parameters:
        numFiles             6
        numRows              0
        rawDataSize          0
        totalSize            1445
        transient_lastDdlTime   1628096654

# Storage Information
SerDe Library:            org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe
InputFormat:              hive.inoutformat.GeekTextInputFormat
OutputFormat:
org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat
Compressed:               NO
Num Buckets:              -1
Bucket Columns:           []
Sort Columns:             []
Storage Desc Params:
        field.delim          \n
        line.delim           \n
        serialization.format   \n
Time taken: 0.064 seconds, Fetched: 32 row(s)
```

geektime_output

InputFormat: org.apache.hadoop.mapred.TextInputFormat
OutputFormat: hive.inoutformat.GeekTextOutputFormat

```
hive> desc formatted geektime_output ;
OK
# col_name                data_type                comment

content                   string
```

```
# Detailed Table Information
Database:               wanghuan
OwnerType:              USER
Owner:                  student
CreateTime:             Thu Aug 05 00:53:27 CST 2021
LastAccessTime:         UNKNOWN
Retention:              0
Location:
hdfs://jikehadoop01:8020/user/hive/warehouse/wanghuan.db/geektime_output
Table Type:             MANAGED_TABLE
Table Parameters:
        numFiles                4
        numRows                 0
        rawDataSize             0
        totalSize               1200
        transient_lastDdlTime   1628097015

# Storage Information
SerDe Library:          org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe
InputFormat:            org.apache.hadoop.mapred.TextInputFormat
OutputFormat:           hive.inoutformat.GeekTextOutputFormat
Compressed:             NO
Num Buckets:            -1
Bucket Columns:         []
Sort Columns:           []
Storage Desc Params:
        field.delim             \n
        line.delim              \n
        serialization.format    \n
Time taken: 0.047 seconds, Fetched: 32 row(s)
```

geektime_inoutput

InputFormat: hive.inoutformat.GeekTextInputFormat
OutputFormat: hive.inoutformat.GeekTextOutputFormat

```
hive> desc formatted geektime_inoutput ;
OK
# col_name              data_type               comment

content                 string

# Detailed Table Information
Database:               wanghuan
OwnerType:              USER
Owner:                  student
CreateTime:             Thu Aug 05 01:06:45 CST 2021
LastAccessTime:         UNKNOWN
Retention:              0
Location:
hdfs://jikehadoop01:8020/user/hive/warehouse/wanghuan.db/geektime_inoutput
Table Type:             MANAGED_TABLE
Table Parameters:
        numFiles                3
        numRows                 0
        rawDataSize             0
        totalSize               1029
```

```
        transient_lastDdlTime    1628097087

# Storage Information
SerDe Library:          org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe
InputFormat:            hive.inoutformat.GeekTextInputFormat
OutputFormat:           hive.inoutformat.GeekTextOutputFormat
Compressed:             No
Num Buckets:            -1
Bucket Columns:         []
Sort Columns:           []
Storage Desc Params:
        field.delim             \n
        line.delim              \n
        serialization.format    \n
Time taken: 0.049 seconds, Fetched: 32 row(s)
```