

背景

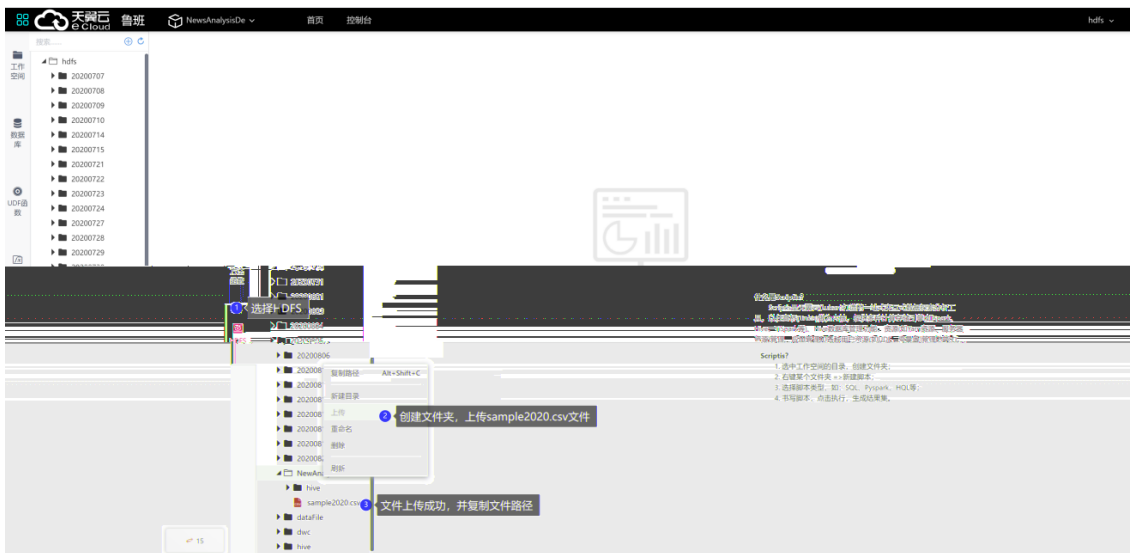
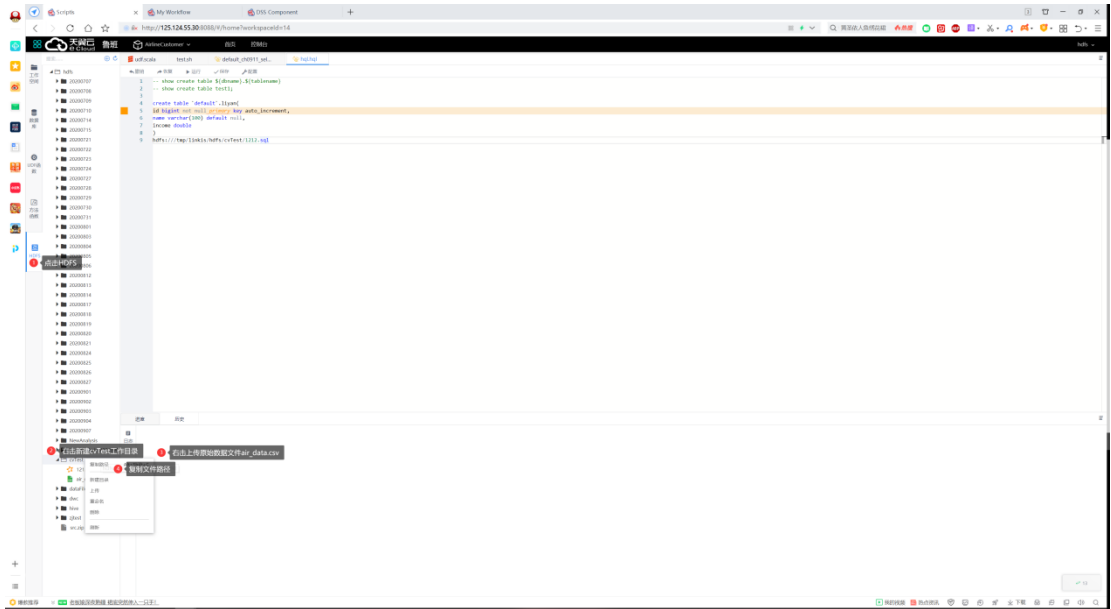
模型	L [单位：月]	R [单位：月]	F [单位：次]	M 位	C
航空公司 LRFMC 模型	会员入会时间 距观测窗口结 束的月数	客户最近一次乘 坐公司飞机距观 测窗口结束的月 数	客户在观测窗 口内乘坐公司 飞机的次数	客户在观测窗 口内累计的飞 行里程	客户在观测窗 口内乘坐舱位 所对应的折扣 系数的平均值

字段中文	字段英文
会	MEMBER_NO
会	FFP_DATE
—	FIRST_FLIGHT_DATE
	GENDER
会	FFP_TIER
作	WORK_CITY
作 份	WORK_PROVINCE
作	WORK_COUNTRY
	age
	LOAD_TIME
	FLIGHT_COUNT
	BP_SUM
—	EP_SUM_YR_1
二	EP_SUM_YR_2
— 价	SUM_YR_1
二 价	SUM_YR_2
	SEG_KM_SUM
位	WEIGHTED_SEG_KM

	LAST_FLIGHT_DATE
	AVG_FLIGHT_COUNT
	AVG_BP_SUM
一 乘 会	BEGIN_TO_FIRST
一 乘	LAST_TO_END
乘	AVG_INTERVAL
乘	MAX_INTERVAL
中 1 他 作伙伴、促 、	ADD_POINTS_SUM_YR_1
中 2 他 作伙伴、促 、	ADD_POINTS_SUM_YR_2
└	EXCHANGE_COUNT
	avg_discount
1 年乘机次数	P1Y_Flight_Count
2 乘	L1Y_Flight_Count
1 年里程积分	P1Y_BP_SUM
2	L1Y_BP_SUM
	EP_SUM
中 他 作伙伴、促 、	ADD_Point_SUM
乘	Eli_Add_Point_Sum
2 乘	L1Y_ELi_Add_Points
	Points_Sum
2 年观测窗口总累计积分	L1Y_Points_Sum
2 年的乘机次数比率	Ration_L1Y_Flight_Count
1 年的乘机次数比率	Ration_P1Y_Flight_Count
1 年里程积分占最近两年积分比例	Ration_P1Y_BPS
2 年里程积分占最近两年积分比例	Ration_L1Y_BPS
乘	Point_NotFlight

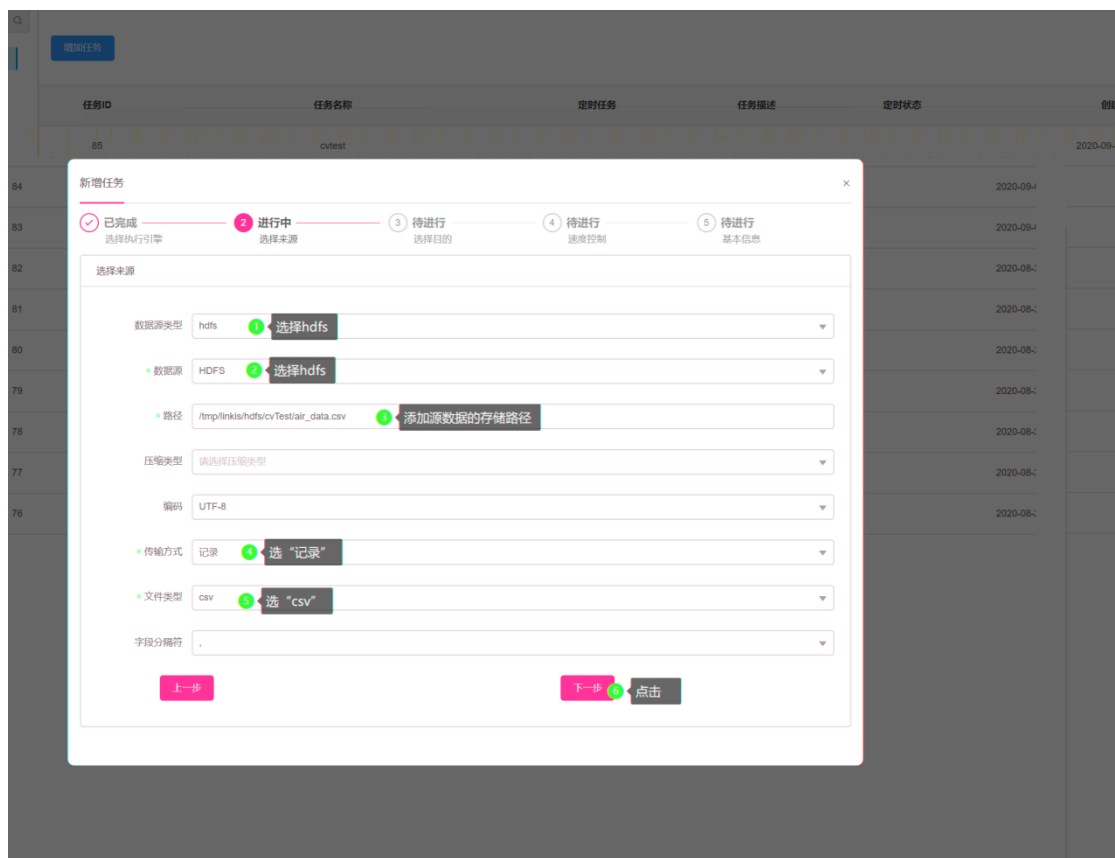
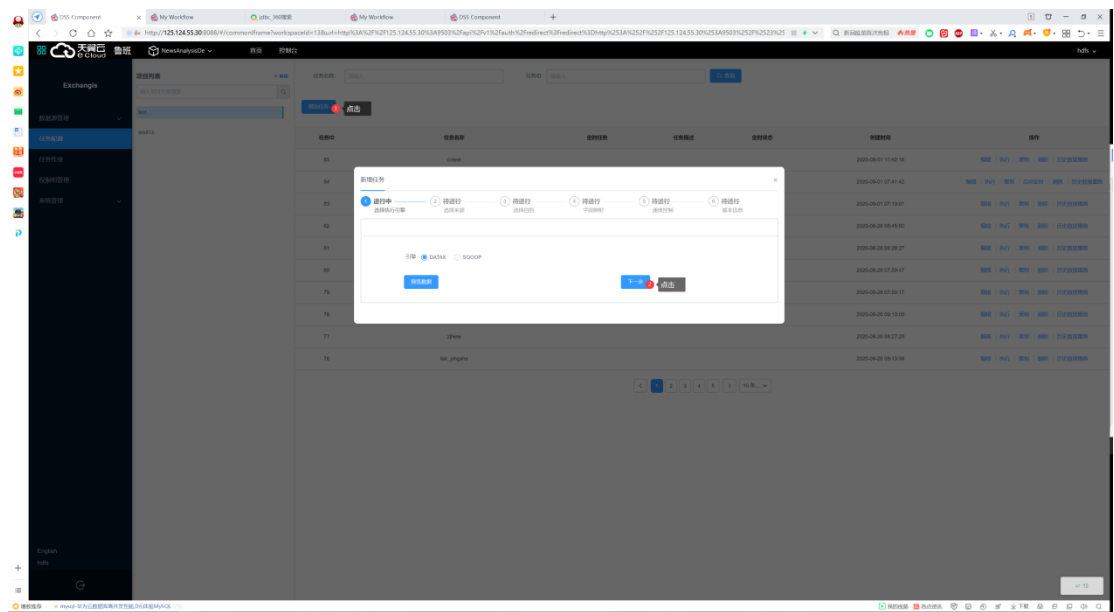
操作步骤

Step 1









cvtest

新增任务

1 已完成

2 已完成

3 进行中

4 待进行

5 待进行

选择执行引擎

选择来源

选择目的

速度控制

基本信息

选择来源

数据源类型

mysql

1 选mysql

\* 数据源

lubanbackup

2 填入工作流中新建的数据库名

\* 库名

lubanbackup

3 填入工作流中新建的数据库名

\* 表名

cvtest

4 选择需要导入的表名

\* 写入方式

insert

5 选择写入方式 "insert"

批量大小

请输入批量大小值

上一步

下一步

6 点击

cvtest

新增任务

1 已完成

2 已完成

3 已完成

4 进行中

5 待进行

选择执行引擎

选择来源

选择目的

速度控制

基本信息

速度控制

\* 作业速率限制

10

Mb/s

\* 作业记录数限制

10000

条/s

\* 错误记录数超过

1

条, 任务失败

高级

上一条

下一步

1 点击

cvtest

新增任务

×

✓ 已完成

选择执行引擎

✓ 已完成

选择来源

✓ 已完成

选择目的

✓ 已完成

速度控制

5 进行中

基本信息

保存预览

任务变量

\* 任务名称

cvtest

提醒人

请输入提醒人

定时

com表达式

任务描述

请输入任务描述

每00

分钟

\* 执行用户:

hdfs

\* 执行节点:

192.168.0.154:9501

×

▼

超时时间

43200

S

同步方式

☒ 全量(FULL)

☐ 增量(INCR)

上一步

保存

1

点击

新增任务

任务列表

任务ID	任务名称	定时任务	任务描述	定时状态	创建时间	操作
65	cvtest				2020-09-01 11:42:16	<a href="#">编辑</a> <a href="#">1 点击执行</a> <a href="#">历史数据查询</a>
64	task_schedule_mysql_to_redis	0 0/1 * * * ?		PAUSED	2020-09-01 07:41:42	<a href="#">编辑</a> <a href="#">执行</a> <a href="#">暂停</a> <a href="#">启动定时</a> <a href="#">删除</a> <a href="#">历史数据查询</a>
63	test122				2020-09-01 07:19:01	<a href="#">编辑</a> <a href="#">执行</a> <a href="#">暂停</a> <a href="#">删除</a> <a href="#">历史数据查询</a>
62	task_join				2020-09-28 08:45:50	<a href="#">编辑</a> <a href="#">执行</a> <a href="#">暂停</a> <a href="#">删除</a> <a href="#">历史数据查询</a>
61	task_split				2020-09-28 08:26:27	<a href="#">编辑</a> <a href="#">执行</a> <a href="#">暂停</a> <a href="#">删除</a> <a href="#">历史数据查询</a>
60	task_replace				2020-09-28 07:59:47	<a href="#">编辑</a> <a href="#">执行</a> <a href="#">暂停</a> <a href="#">删除</a> <a href="#">历史数据查询</a>
79	task_flow_by_id				2020-09-28 07:50:17	<a href="#">编辑</a> <a href="#">执行</a> <a href="#">暂停</a> <a href="#">删除</a> <a href="#">历史数据查询</a>
78	newcustomer				2020-09-26 09:10:05	<a href="#">编辑</a> <a href="#">执行</a> <a href="#">暂停</a> <a href="#">删除</a> <a href="#">历史数据查询</a>
77	zifnew				2020-09-26 09:27:28	<a href="#">编辑</a> <a href="#">执行</a> <a href="#">暂停</a> <a href="#">删除</a> <a href="#">历史数据查询</a>
76	task_yinghe				2020-09-26 09:13:36	<a href="#">编辑</a> <a href="#">执行</a> <a href="#">暂停</a> <a href="#">删除</a> <a href="#">历史数据查询</a>

<

1

2

3

4

5

>

10条...



Step 5

概览

CValueAnalysis

工作流

参数

资源

执行

保存

搜索

数据交换

数据开发

数据质量

数据可视化

数据输出

信号节点

功能节点

创建源数据库

客户价值挖掘

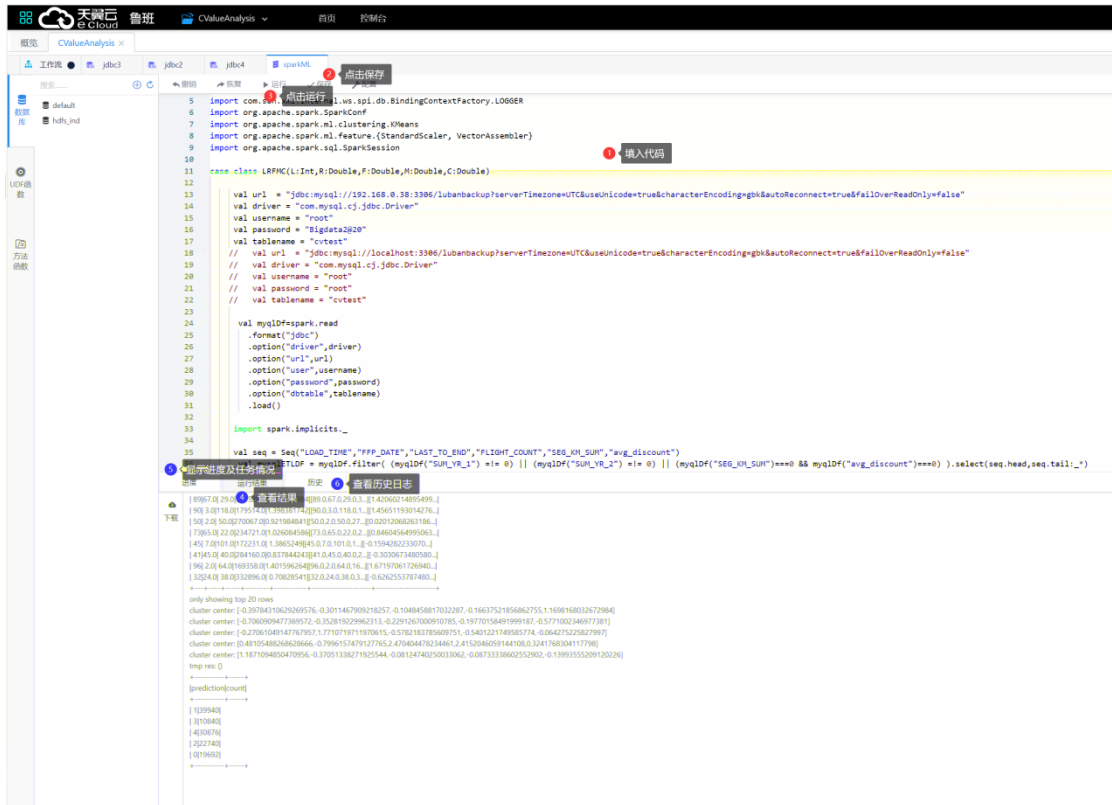
结果数据表

jdbc3

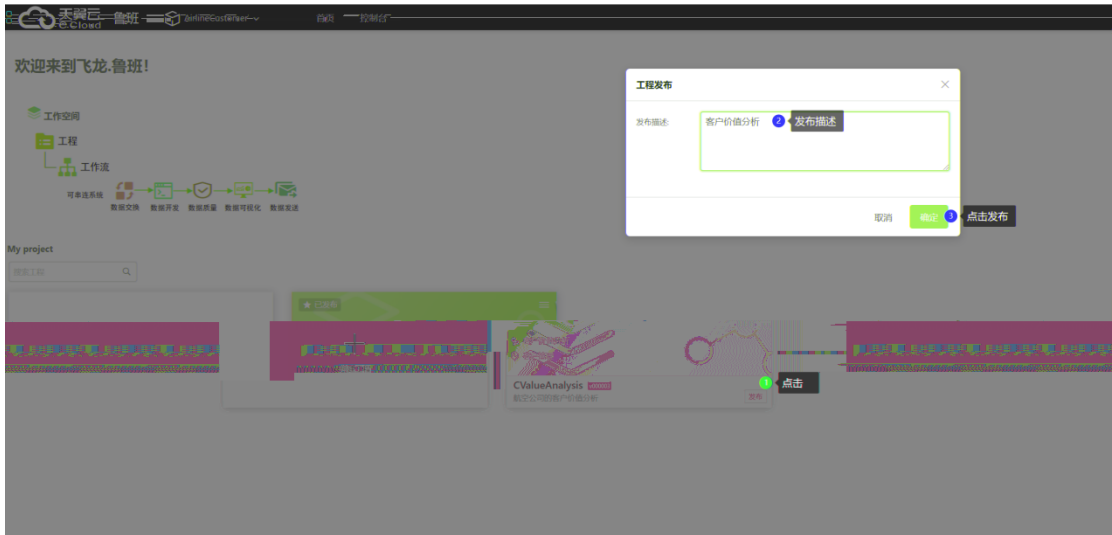
变量

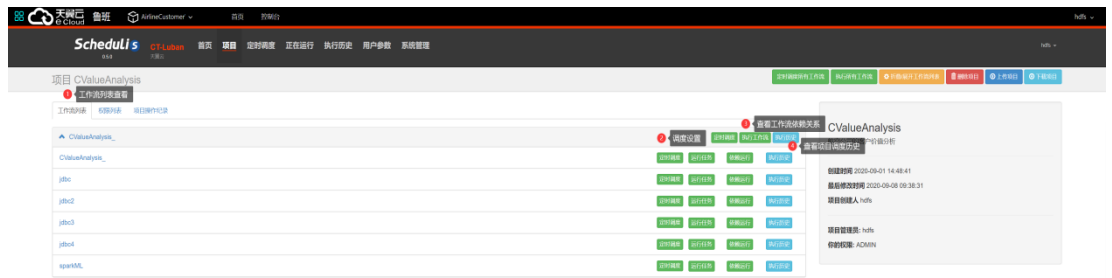
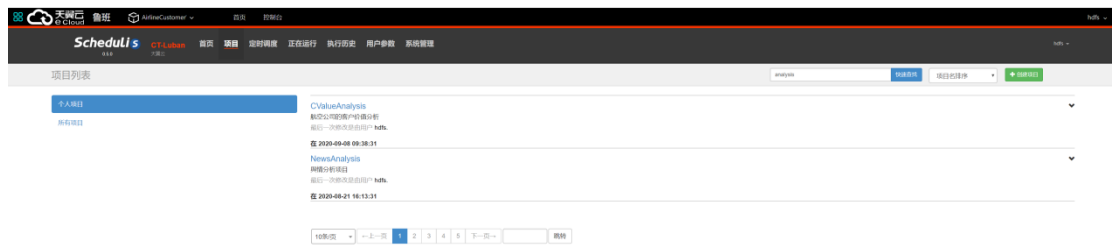
硬代码

结果数据表

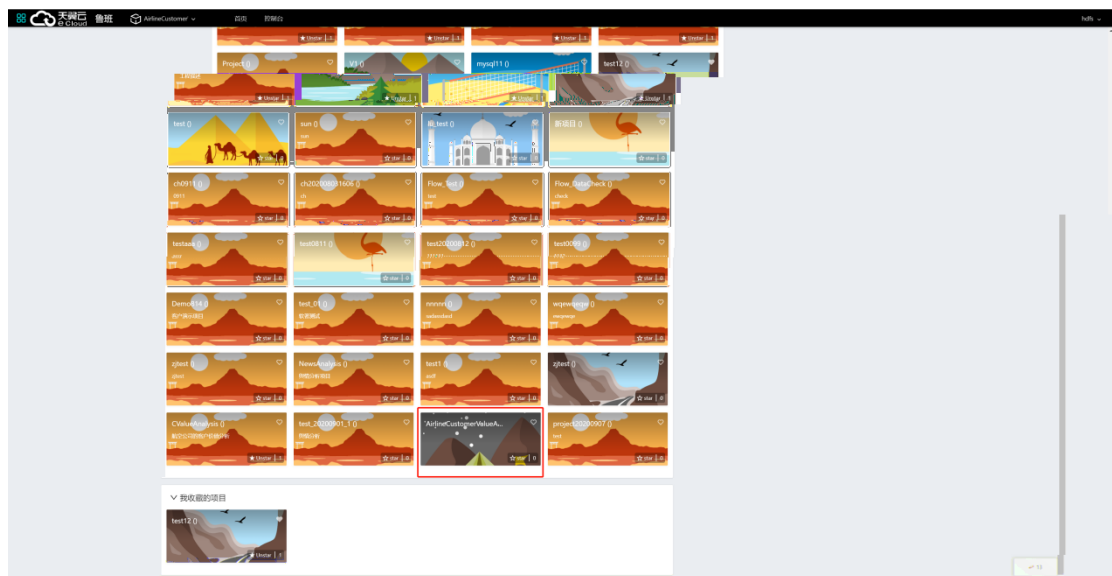


Step 6

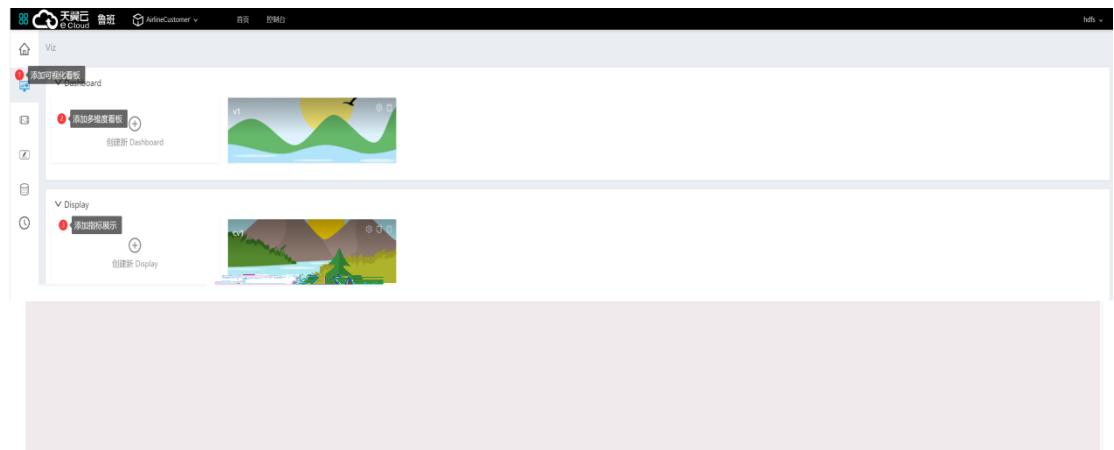
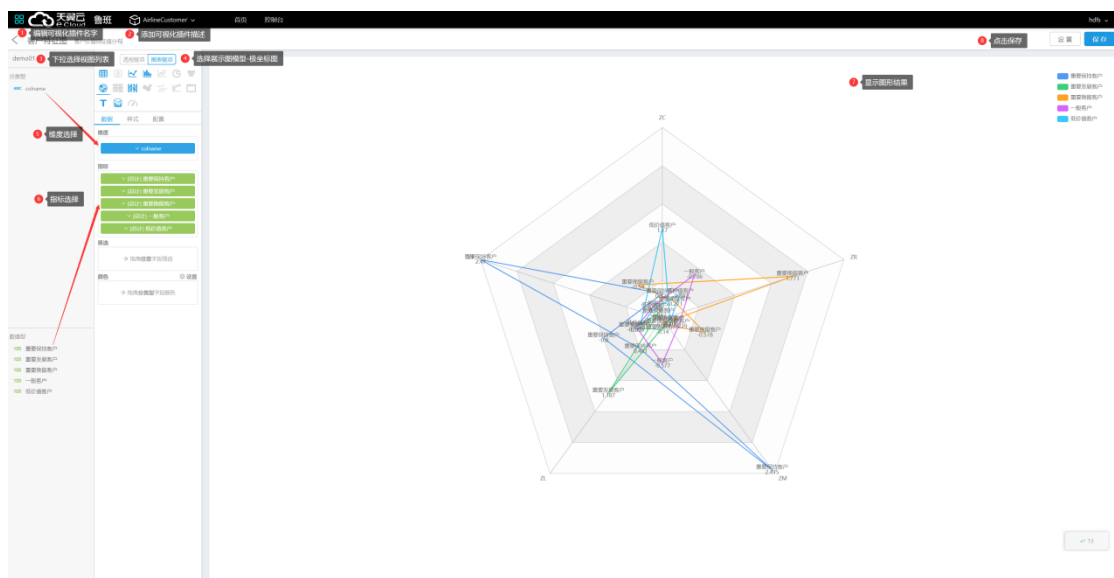
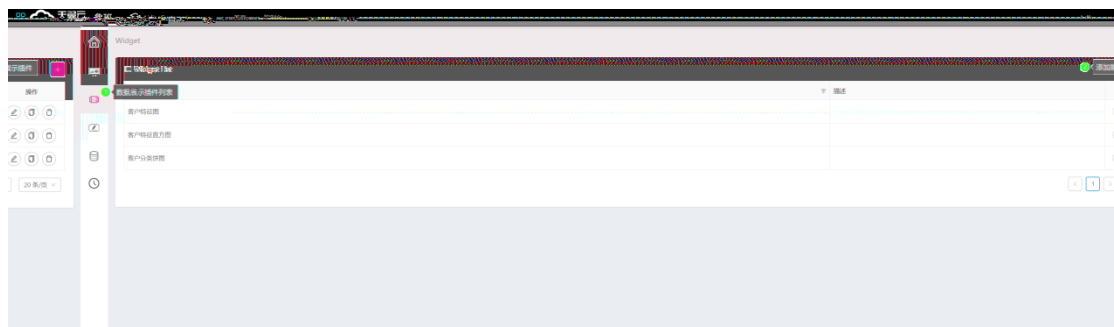




Step 5







板



# 附录：案例代码

## 1. JDBC1

```
CREATE DATABASE IF NOT EXISTS `lubanbackup` ;
USE `lubanbackup` ;
DROP TABLE IF EXISTS `cvtest`;
CREATE TABLE `cvtest` (
  `MEMBER_NO` varchar(50) DEFAULT NULL,
  `FFP_DATE` varchar(50) DEFAULT NULL,
  `FIRST_FLIGHT_DATE` varchar(50) DEFAULT NULL,
  `GENDER` varchar(50) DEFAULT NULL,
  `FFP_TIER` varchar(50) DEFAULT NULL,
  `WORK_CITY` varchar(50) DEFAULT NULL,
  `WORK_PROVINCE` varchar(50) DEFAULT NULL,
  `WORK_COUNTRY` varchar(50) DEFAULT NULL,
  `age` varchar(50) DEFAULT NULL,
  `LOAD_TIME` varchar(50) DEFAULT NULL,
  `FLIGHT_COUNT` varchar(50) DEFAULT NULL,
  `BP_SUM` varchar(50) DEFAULT NULL,
  `EP_SUM_YR_1` varchar(60) DEFAULT NULL,
  `EP_SUM_YR_2` varchar(60) DEFAULT NULL,
  `SUM_YR_1` varchar(60) DEFAULT NULL,
  `SUM_YR_2` varchar(60) DEFAULT NULL,
  `SEG_KM_SUM` varchar(60) DEFAULT NULL,
  `WEIGHTED_SEG_KM` varchar(60) DEFAULT NULL,
  `LAST_FLIGHT_DATE` varchar(50) DEFAULT NULL,
  `AVG_FLIGHT_COUNT` varchar(60) DEFAULT NULL,
  `AVG_BP_SUM` varchar(60) DEFAULT NULL,
  `BEGIN_TO_FIRST` varchar(50) DEFAULT NULL,
  `LAST_TO_END` varchar(50) DEFAULT NULL,
  `AVG_INTERVAL` varchar(50) DEFAULT NULL,
  `MAX_INTERVAL` varchar(50) DEFAULT NULL,
  `ADD_POINTS_SUM_YR_1` varchar(50) DEFAULT NULL,
  `ADD_POINTS_SUM_YR_2` varchar(50) DEFAULT NULL,
  `EXCHANGE_COUNT` varchar(50) DEFAULT NULL,
  `avg_discount` varchar(60) DEFAULT NULL,
  `P1Y_Flight_Count` varchar(50) DEFAULT NULL,
  `L1Y_Flight_Count` varchar(50) DEFAULT NULL,
  `P1Y_BP_SUM` varchar(60) DEFAULT NULL,
  `L1Y_BP_SUM` varchar(60) DEFAULT NULL,
  `EP_SUM` varchar(60) DEFAULT NULL,
```

```

`ADD_Point_SUM` varchar(60) DEFAULT NULL,
`Eli_Add_Point_Sum` varchar(60) DEFAULT NULL,
`L1Y_Eli_Add_Points` varchar(60) DEFAULT NULL,
`Points_Sum` varchar(60) DEFAULT NULL,
`L1Y_Points_Sum` varchar(60) DEFAULT NULL,
`Ration_L1Y_Flight_Count` varchar(60) DEFAULT NULL,
`Ration_P1Y_Flight_Count` varchar(60) DEFAULT NULL,
`Ration_P1Y_BPS` varchar(60) DEFAULT NULL,
`Ration_L1Y_BPS` varchar(60) DEFAULT NULL,
`Point_NotFlight` varchar(60) DEFAULT NULL,
KEY `MEMBER_NO` (`MEMBER_NO`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

use lubanbackup ;
DROP TABLE IF EXISTS `cvresult`;
CREATE TABLE `cvresult` (
  `ZL` double DEFAULT NULL,
  `ZR` double DEFAULT NULL,
  `ZF` double DEFAULT NULL,
  `ZM` double DEFAULT NULL,
  `ZC` double DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
show tables;

```

## 2.SPARKML

```

import java.sql.{Connection, DriverManager, PreparedStatement}
import java.text.{ParseException, SimpleDateFormat}
import java.util.{Calendar, Date}

import com.sun.xml.internal.ws.spi.db.BindingContextFactory.LOGGER
import org.apache.spark.SparkConf
import org.apache.spark.ml.clustering.KMeans
import org.apache.spark.ml.feature.{StandardScaler, VectorAssembler}
import org.apache.spark.sql.SparkSession

case class LRFMC(L:Int,R:Double,F:Double,M:Double,C:Double)

val url = "jdbc:mysql://192.168.0.38:3306/lubanbackup?serverTimezone=UTC&useUnicode=true&characterEncoding=gbk&autoReconnect=true&failOverReadOnly=false"
val driver = "com.mysql.cj.jdbc.Driver"
val username = "root"
val password = "Bigdata2020"

```



```

val tablename = "cvtest"

val myqldf=spark.read

    .format("jdbc")

    .option("driver",driver)

    .option("url",url)

    .option("user",username)

    .option("password",password)

    .option("dbtable",tablename)

    .load()

import spark.implicits._

val seq = Seq("LOAD_TIME", "FFP_DATE", "LAST_TO_END", "FLIGHT_COUNT", "SEG_KM_SUM", "avg_discount")

val mysqlETLDF = myqldf.filter( (myqldf("SUM_YR_1") != 0) || (myqldf("SUM_YR_2") != 0) || (myqldf("SEG_KM_SUM")===0 && myqldf("avg_discount")===0) ).select(seq.head,seq.tail:_*)

def monthDiff(bef: String, aft: String): Int = { // 折出日期对
    val sdf = new SimpleDateFormat("yyyy/MM/dd")
    var date1:Date= null
    var date2:Date = null
    try {
        date1 = sdf.parse(bef)
        date2 = sdf.parse(aft)
    } catch {
        case e: ParseException =>
            //e.printStackTrace();
            LOGGER.info("error mess :"+ e.getMessage + "," + e.toString + date1 + "," + date1)
            return -100
    }

    //使用 Calendar 对 来操作日期对
    val c1 = Calendar.getInstance
    val c2 = Calendar.getInstance
    c1.setTime(date1)
    c2.setTime(date2)

    //获取日期的毫秒数, 用于比 日期 先 后
    val time1 = date1.getTime
    val time2 = date2.getTime

    //年数相减
    val yearSubtract = c2.get(Calendar.YEAR) - c1.get(Calendar.YEAR)

    //月数相减
    var monthSubtract = 0

    //根据日期的先后应用不同的公式
    if (time1 < time2) monthSubtract = c2.get(Calendar.MONTH) - c1.get(Calendar.MONTH) + 1

```

```

        else monthSubtract = c2.get(Calendar.MONTH) - c1.get(Calendar.MONTH) - 1
        Math.abs(yearSubtract * 12 + monthSubtract).toInt
    }

    val schema = "L,R,F,M,C".split(",")
    val transDF = mysqlETLDF.map(line => {
        val L = monthDiff(line.get(0).toString, line.get(1).toString)
        if (L == -100) {
            null
        } else {
            LRFMC(L, line.get(2).toString.toDouble, line.get(3).toString.toDouble, line.get(4).toString.toDouble, line.get(5).toString.toDouble)
        }
    }).filter(line => line != null)

    val assembler = new VectorAssembler()
        .setInputCols(schema)
        .setOutputCol("feature_vec")

    //regular
    val scaler = new StandardScaler()
        .setInputCol("feature_vec")
        .setOutputCol("features")
        .setWithStd(true)
        .setWithMean(true)

    //df
    val df_vec = assembler.transform(transDF)
    val df_scaler = scaler.fit(df_vec).transform(df_vec)
    df_scaler.show()

    val kmeans = new KMeans().setFeaturesCol("features").setK(5).setMaxIter(20)
    val model = kmeans.fit(df_scaler)
    val df_Predict = model.transform(df_scaler)
    (df_Predict,model)

    def conn():Connection = {
        var conn: Connection = null
        try {
            Class.forName(driver)
            conn = DriverManager.getConnection(url,username,password)
        } catch {
            case e: Exception => println("Mysql Exception" + e.getMessage + "," + e.getStackTrace)
        }
    }

```

```

    }
    conn
  }

def write(conn:Connection,arr:Array[Double]): Unit = {
  var ps: PreparedStatement = null
  val sql = "insert into cvresult(ZL,ZR,ZF,ZM,ZC) values (?, ?, ?, ?, ?)"
  try {
    ps = conn.prepareStatement(sql)
    ps.setDouble(1,arr(0))
    ps.setDouble(2,arr(1))
    ps.setDouble(3,arr(2))
    ps.setDouble(4,arr(3))
    ps.setDouble(5,arr(4))
    ps.executeUpdate()
  } catch {
    case e: Exception => println("Mysql Exception"+ e.getMessage + "," + e.getStackTrace)
  } finally {
    ps.close()
  }
}

val res = model.clusterCenters.foreach(row => {
  println("cluster center: " + row)
  write(conn(),row.toArray)
  // ZLRFMC(row(0),row(1),row(2),row(3),row(4))
})
println("tmp res: " + res)

//show stat
df_Predict.groupBy("prediction").count().show()

```

### 3.JDBC2

```

USE `lubanbackup`;

/*Table structure for table `cvresultn` */

DROP TABLE IF EXISTS `cvresultn`;

CREATE TABLE `cvresultn` (
  `colname` varchar(20) DEFAULT NULL,

```

```

`customer1` double DEFAULT NULL,
`customer2` double DEFAULT NULL,
`customer3` double DEFAULT NULL,
`customer4` double DEFAULT NULL,
`customer5` double DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

/*Data for the table `cvresultn` */

insert into `cvresultn`(`colname`,`customer1`,`customer2`,`customer3`,`customer4`,`customer5`) values
('ZL',-0.397,-0.706,-0.271,0.481,1.187),
('ZR',-0.301,-0.353,1.771,-0.8,-0.375),
('ZF',-0.105,-0.229,-0.578,2.47,-0.081),
('ZM',-0.166,-0.198,-0.54,2.415,-0.087),
('ZC',1.17,-0.577,-0.064,0.324,-0.14);

```

#### 4.JDBC3

```

USE `lubanbackup`;

/*Table structure for table `cv_result_num` */

DROP TABLE IF EXISTS `cv_result_num`;

CREATE TABLE `cv_result_num` (
  `要保持客户` varchar(20) DEFAULT NULL,
  `要发展客户` varchar(20) DEFAULT NULL,
  `要挽留客户` varchar(20) DEFAULT NULL,
  `一般客户` varchar(20) DEFAULT NULL,
  `低价值客户` varchar(20) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

/*Data for the table `cv_result_num` */

insert into `cv_result_num`(`要保持客户`,`要发展客户`,`要挽留客户`,`一般客户`,`低价值客户`) values
('19692','30876','22740','39940','10840');

select * from cv_result_num;

```