背景

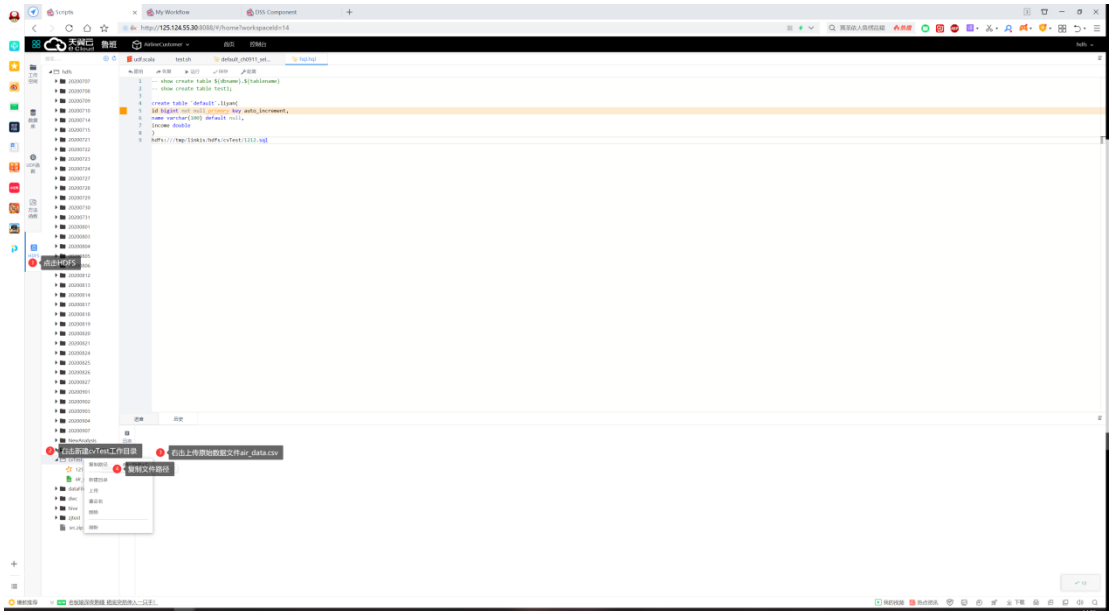
模型	L [单位：月]	R [单位：月]	F [单位：次]	M 位	C
航空公司 LRFMC 模型	会员入会时间距观测窗口结束的月数	客户最近一次乘坐公司飞机距观测窗口结束的月数	客户在观测窗口内乘坐公司飞机的次数	客户在观测窗口内累计的飞行里程	客户在观测窗口内乘坐舱位所对应的折扣系数的平均值

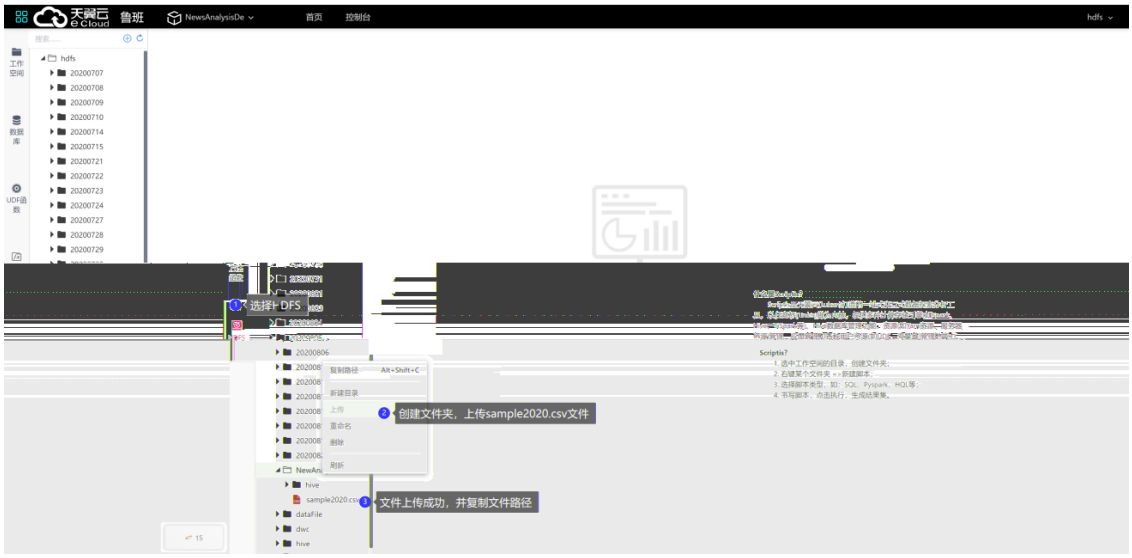
字段中文	字段英文
会	MEMBER_NO
会	FFP_DATE
一	FIRST_FLIGHT_DATE
	GENDER
会	FFP_TIER
作	WORK_CITY
作 份	WORK_PROVINCE
作	WORK_COUNTRY
	age
	LOAD_TIME
	FLIGHT_COUNT
	BP_SUM
一	EP_SUM_YR_1
二	EP_SUM_YR_2
一 价	SUM_YR_1
二 价	SUM_YR_2
	SEG_KM_SUM
位	WEIGHTED_SEG_KM

	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

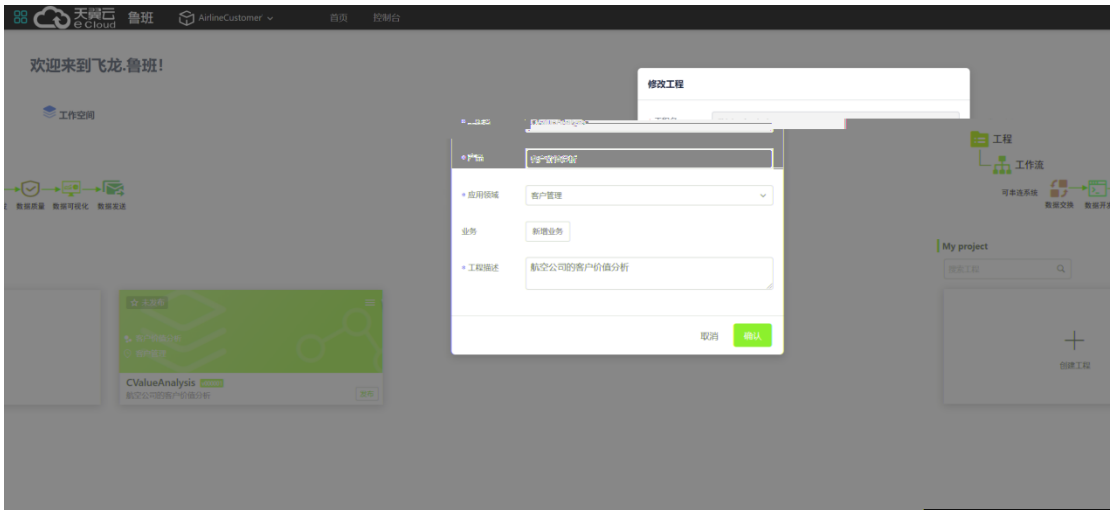
操作步骤

Step 1

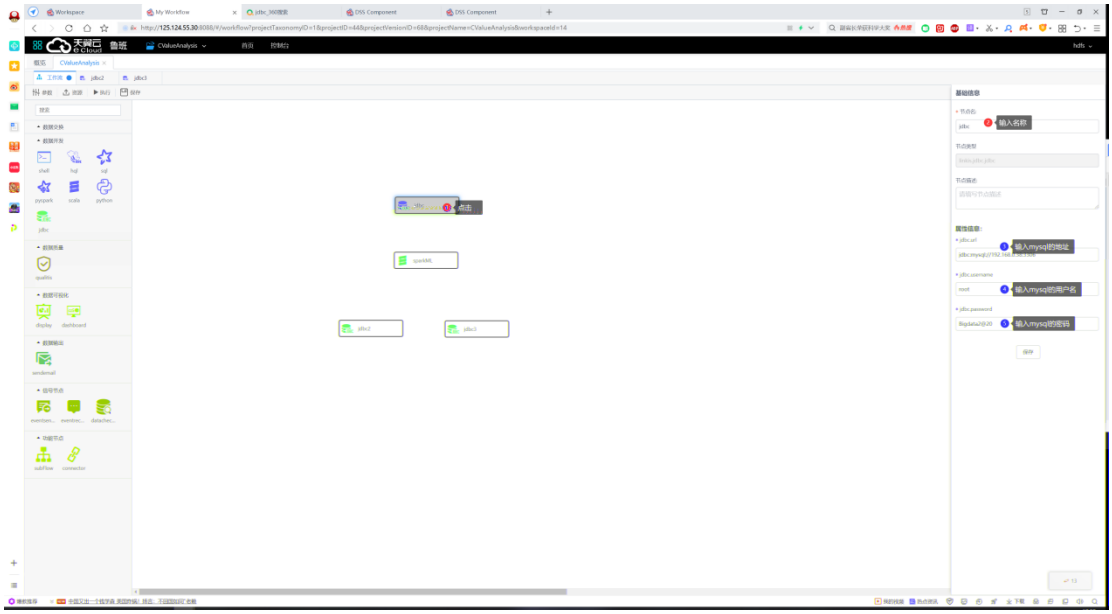


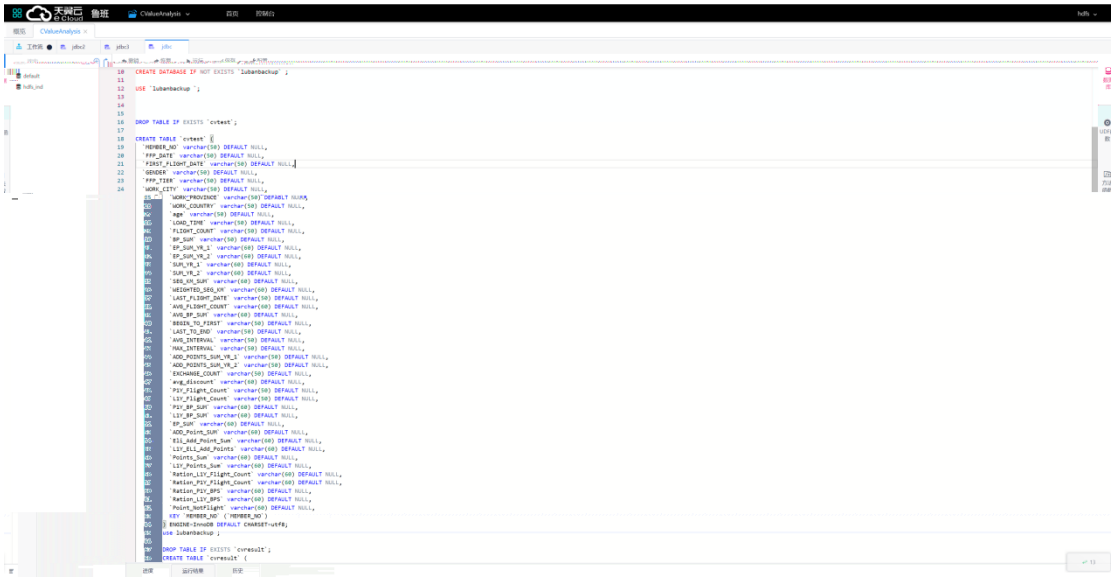


Step 2

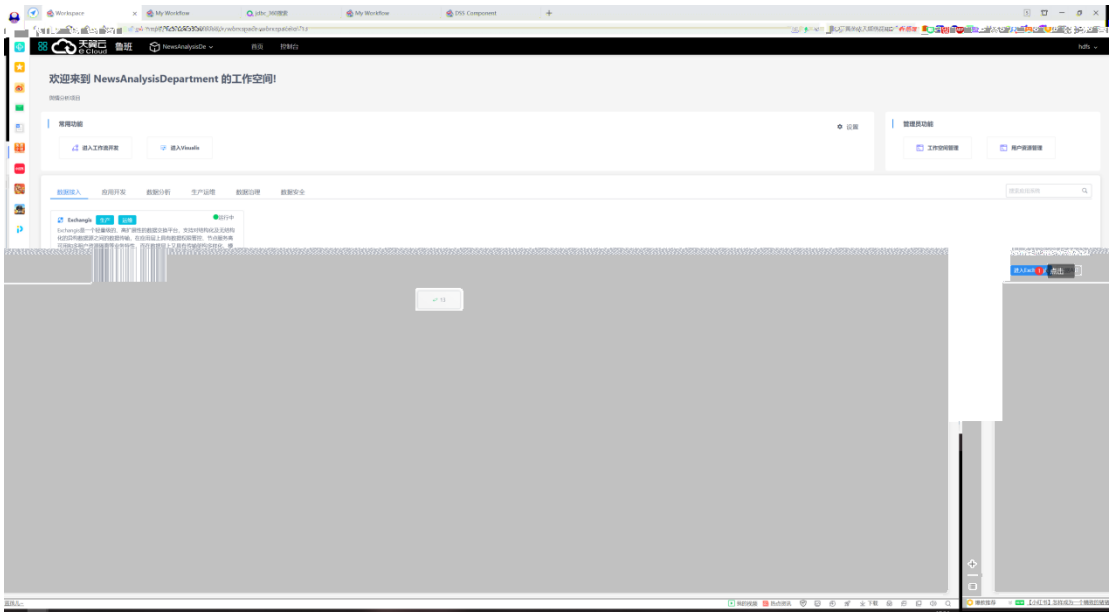

欢迎来到飞龙,鲁班!
工作空间
修改工程
应用领域
客户管理
业务
新增业务
工程描述
航空公司的客户价值分析
取消
确认
工程
工作流
My project
CValueAnalysis

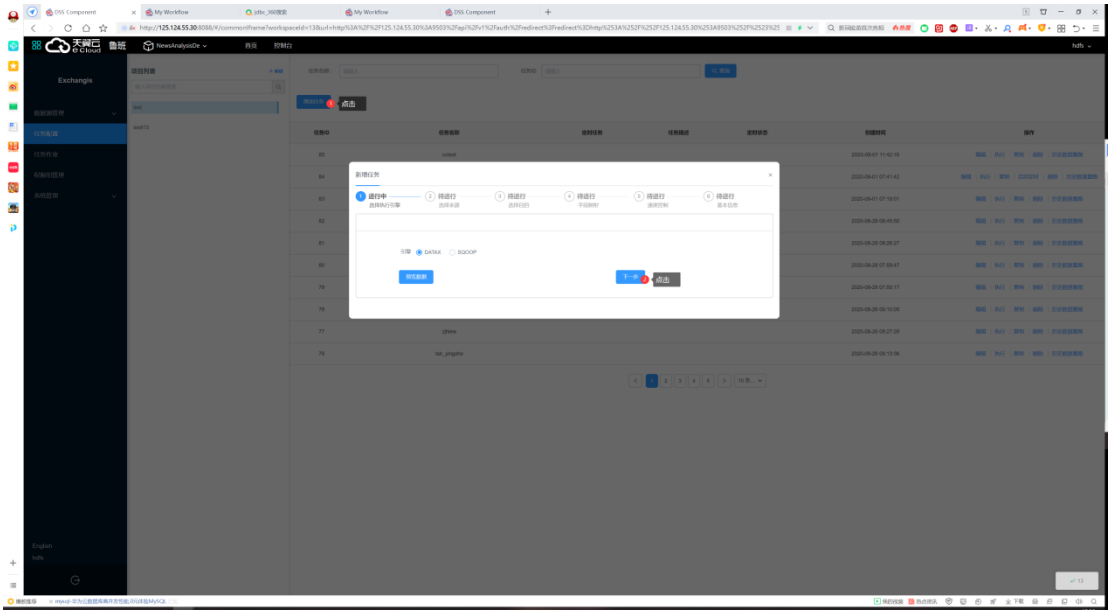
Step 3

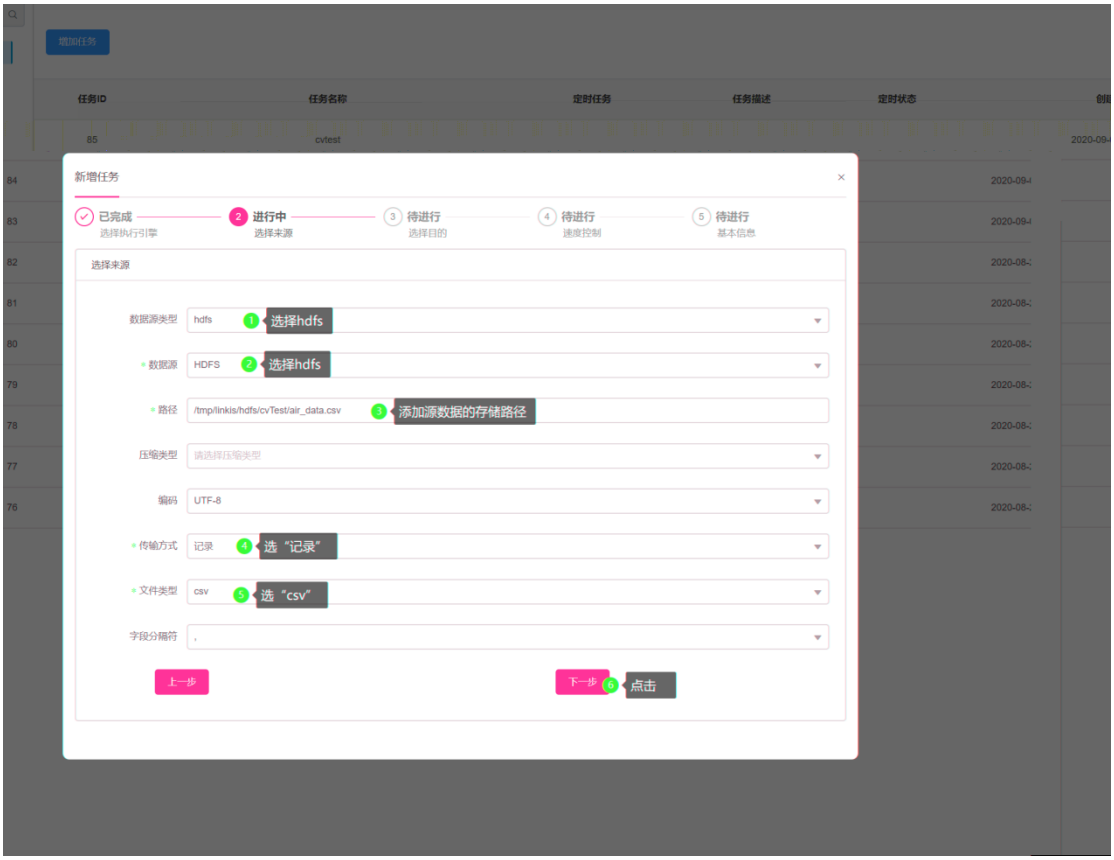

输入名称
输入mysql的地址
输入mysql的用户名
输入mysql的密码

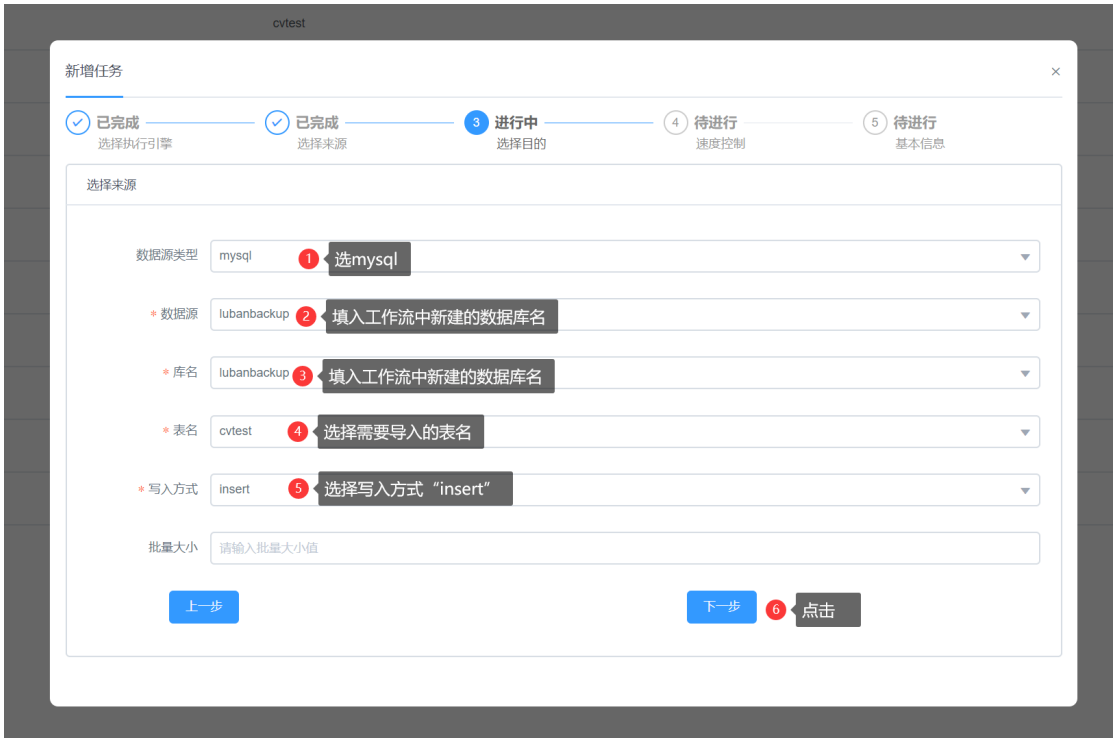


Step 4

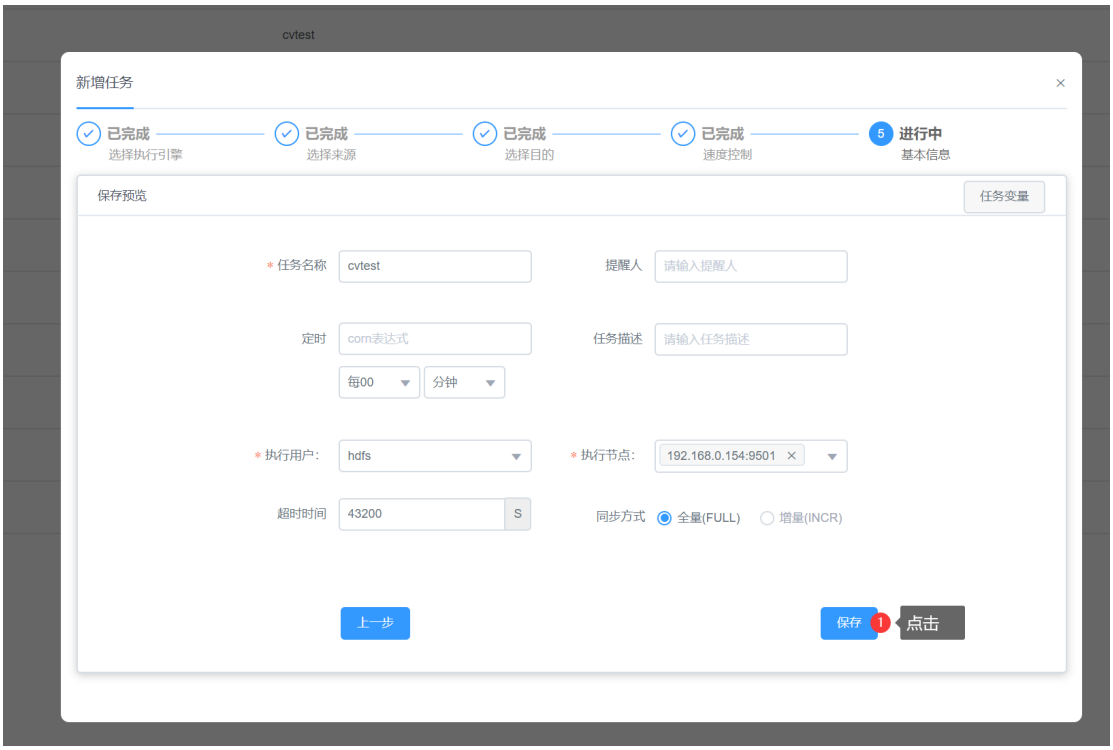


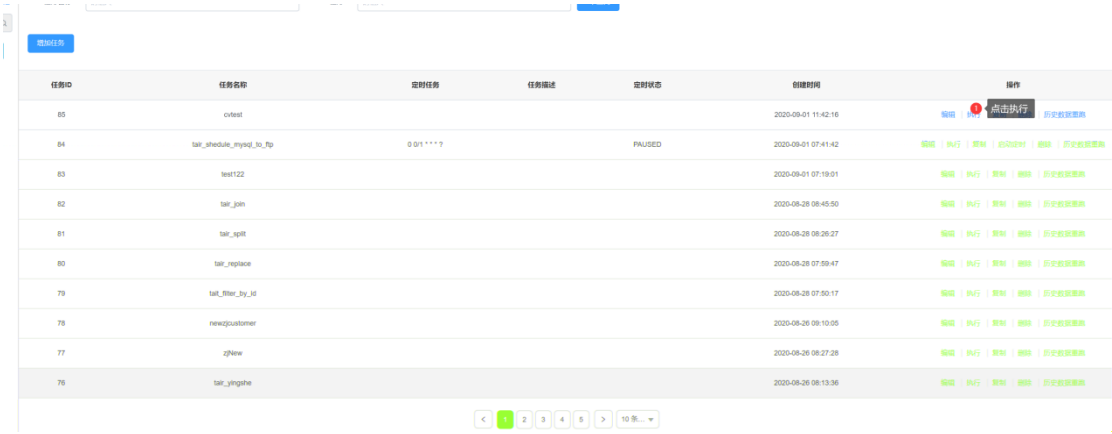

点击


增加任务
新增任务
已完成
选择执行引擎
进行中
选择来源
待进行
选择目的
待进行
速度控制
待进行
基本信息
选择来源
数据源类型
hdfs
选择hdfs
数据源
HDFS
选择hdfs
路径
/tmp/linkis/hdfs/cvTest/air_data.csv
添加源数据的存储路径
压缩类型
请选择压缩类型
编码
UTF-8
传输方式
记录
选“记录”
文件类型
csv
选“csv”
字段分隔符
,
上一步
下一步
点击

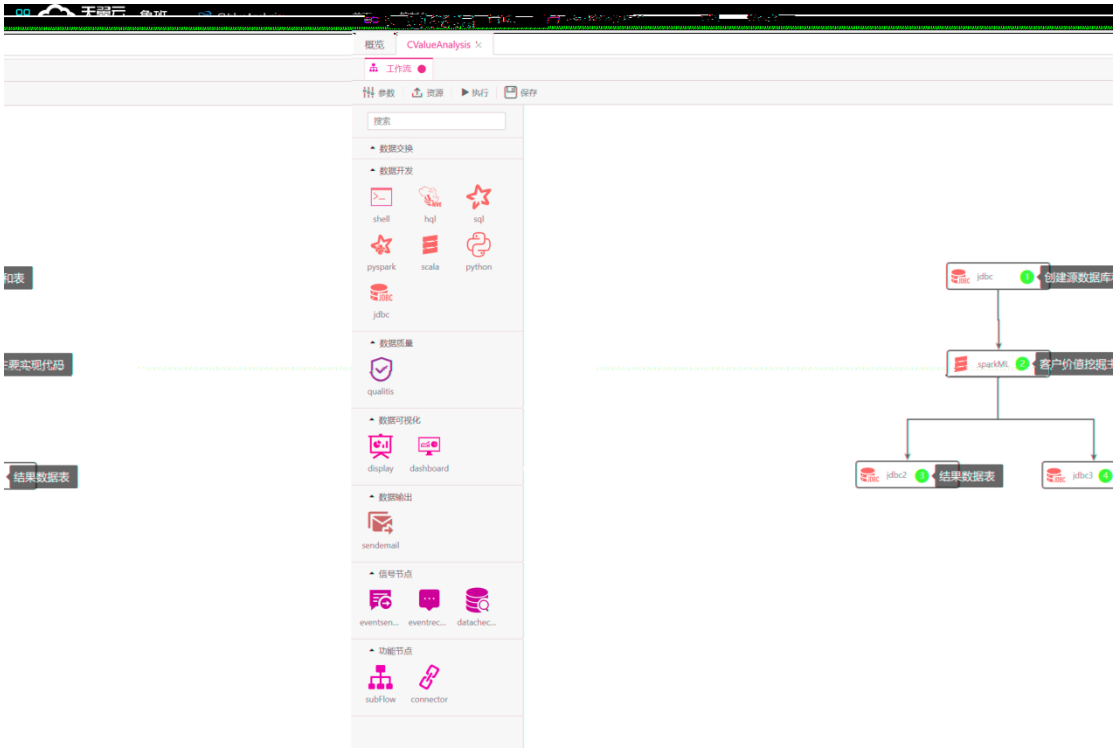

cvtest
新增任务
已完成
选择执行引擎
已完成
选择来源
3
进行中
选择目的
4
待进行
速度控制
5
待进行
基本信息
选择来源
数据源类型
mysql
1
选mysql
* 数据源
lubanbackup
2
填入工作流中新建的数据库名
* 库名
lubanbackup
3
填入工作流中新建的数据库名
* 表名
cvtest
4
选择需要导入的表名
* 写入方式
insert
5
选择写入方式“insert”
批量大小
请输入批量大小值
上一步
下一步
6
点击

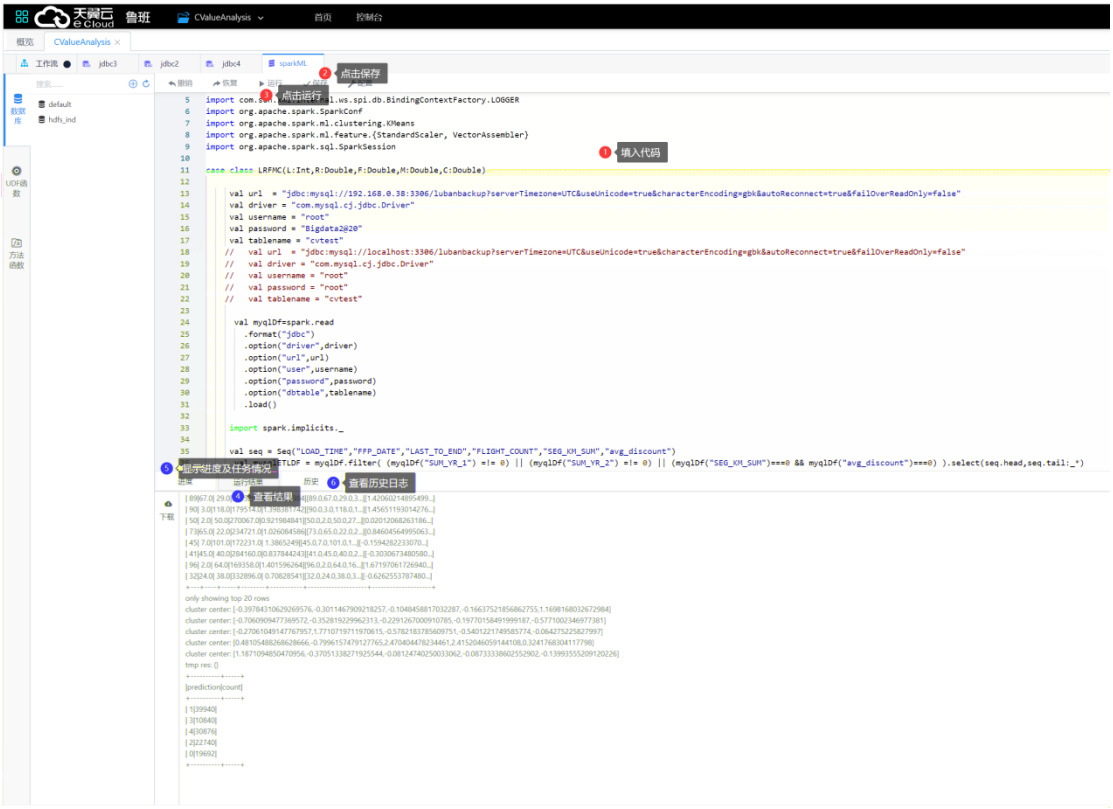

cvtest
新增任务
已完成
选择执行引擎
已完成
选择来源
已完成
选择目的
4
进行中
速度控制
5
待进行
基本信息
速度控制
* 作业速率限制
10
Mb/s
* 作业记录数限制
10000
条/s
* 错误记录数超过
1
条，任务失败
高级
上一条
下一步
1
点击


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
corn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

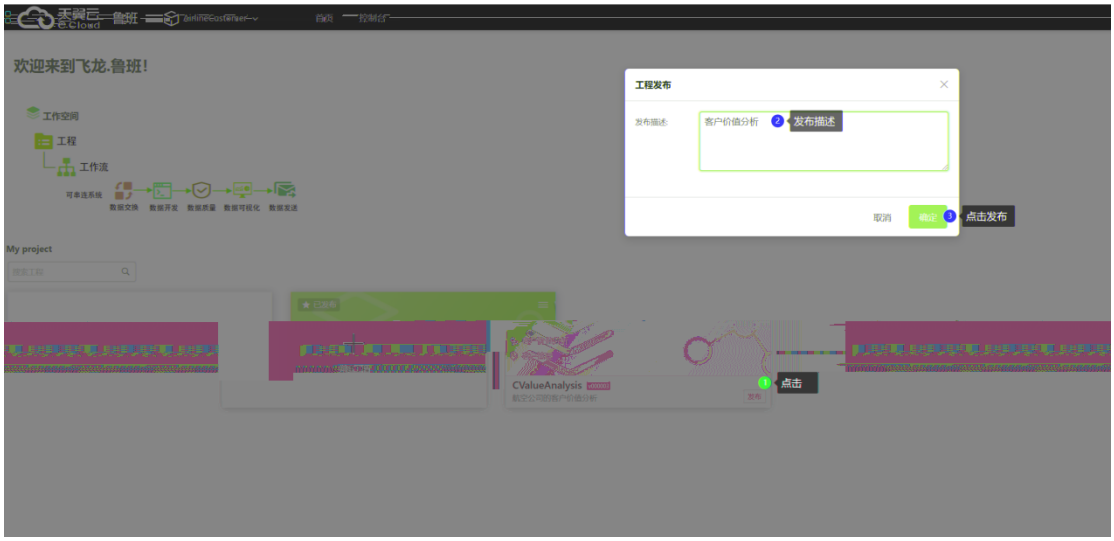

增加任务
任务ID
任务名称
定时任务
任务描述
定时状态
创建时间
操作
85
cvtest
2020-09-01 11:42:16
1
点击执行
84
tair_shedule_mysql_to_ftp
0 0/1 * * * ?
PAUSED
2020-09-01 07:41:42
83
test122
2020-09-01 07:19:01
82
tair_join
2020-08-28 08:45:50
81
tair_split
2020-08-28 08:26:27
80
tair_replace
2020-08-28 07:59:47
79
tait_filter_by_id
2020-08-28 07:50:17
78
newzjcustomer
2020-08-26 09:10:05
77
zjNew
2020-08-26 08:27:28
76
tair_yingshe
2020-08-26 08:13:36

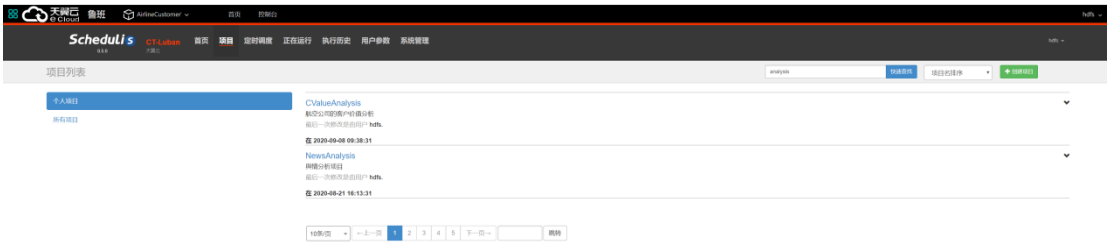
Step 5

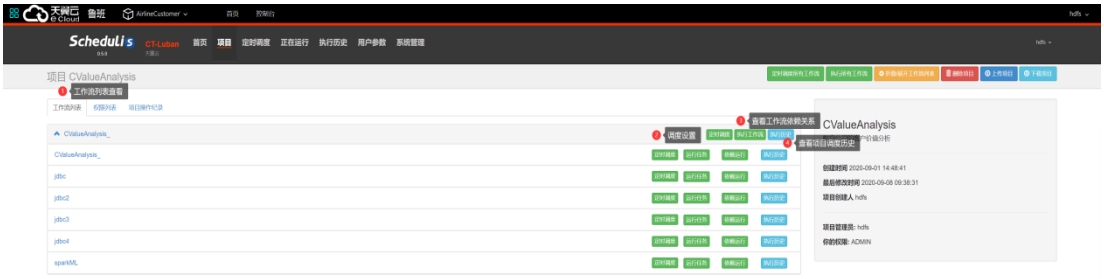

概览
CValueAnalysis
工作流
参数
资源
执行
保存
搜索
数据交换
数据开发
shell
hql
sql
pyspark
scala
python
jdbc
数据质量
qualitis
数据可视化
display
dashboard
数据输出
sendemail
信号节点
eventsen...
eventrec...
datachec...
功能节点
subFlow
connector
和表
要实现代码
结果数据表
jdbc
创建源数据库
sparkML
客户价值挖掘
jdbc2
结果数据表
jdbc3


鲁班
CValueAnalysis
首页
控制台
概览
CValueAnalysis
工作流
jdbc3
jdbc2
jdbc4
sparkML
点击保存
点击运行
填入代码
查看结果
查看历史日志

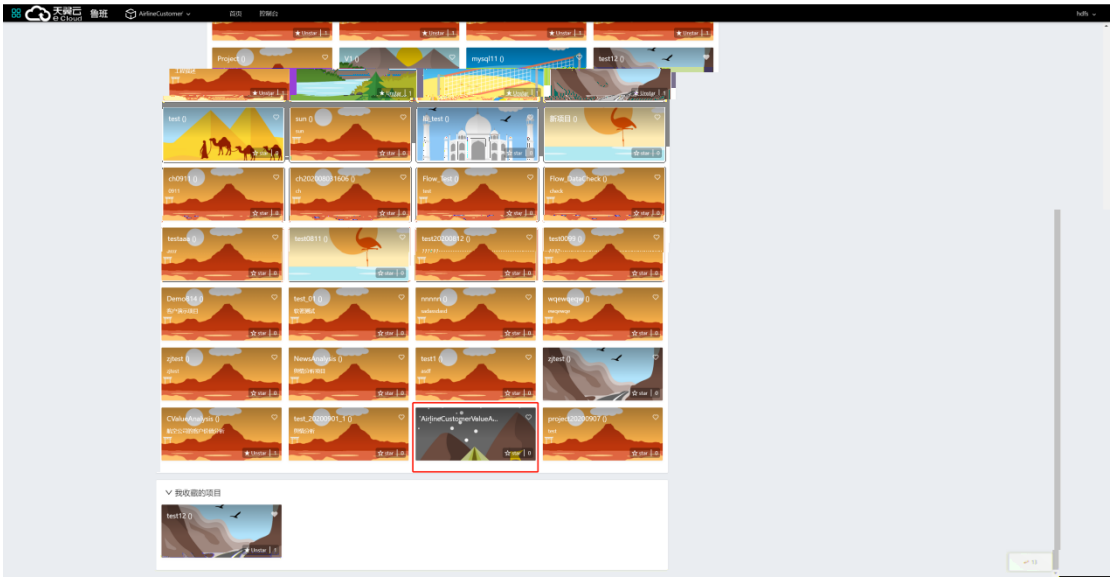
Step 6

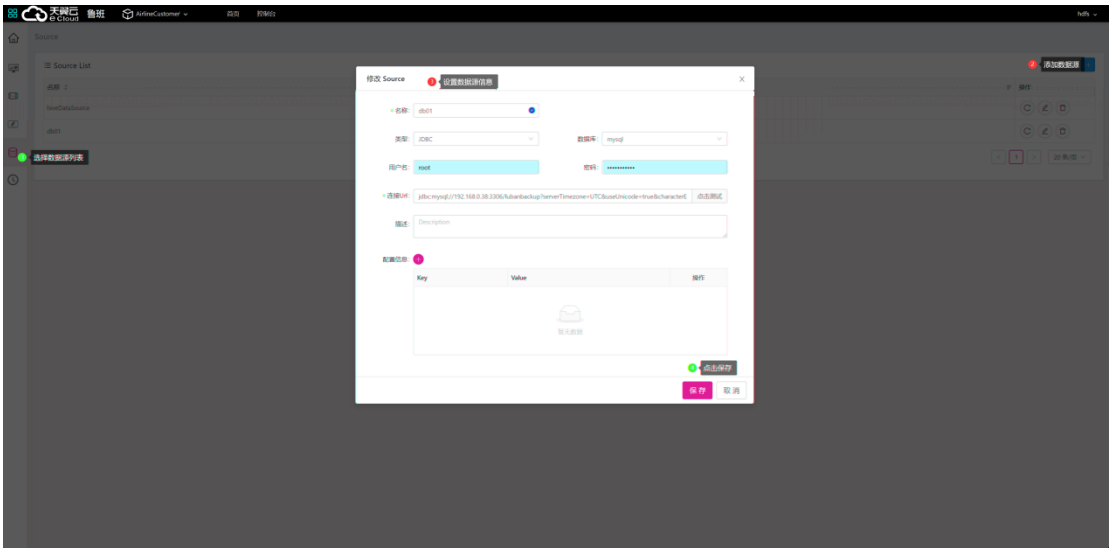

欢迎来到飞龙.鲁班!
工作空间
工程
工作流
工程发布
发布描述
客户价值分析
发布描述
取消
确定
点击发布
My project
CValueAnalysis
点击

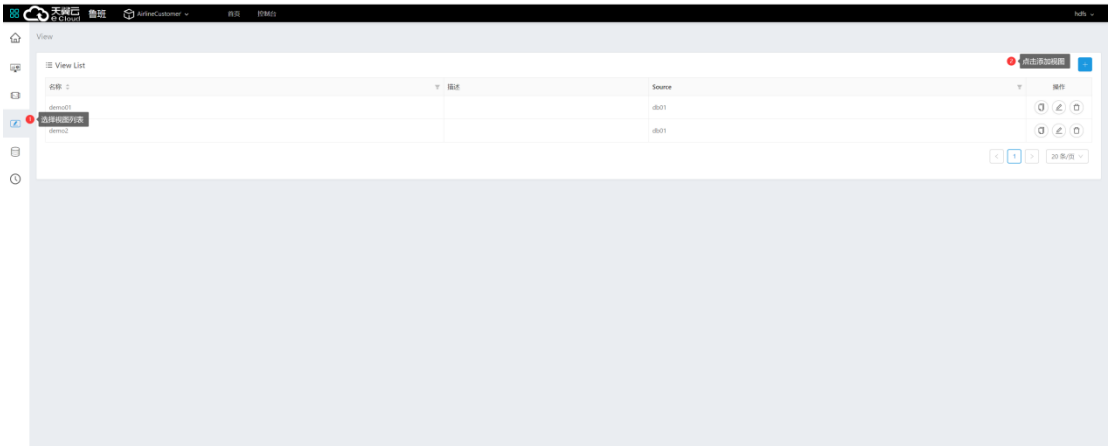


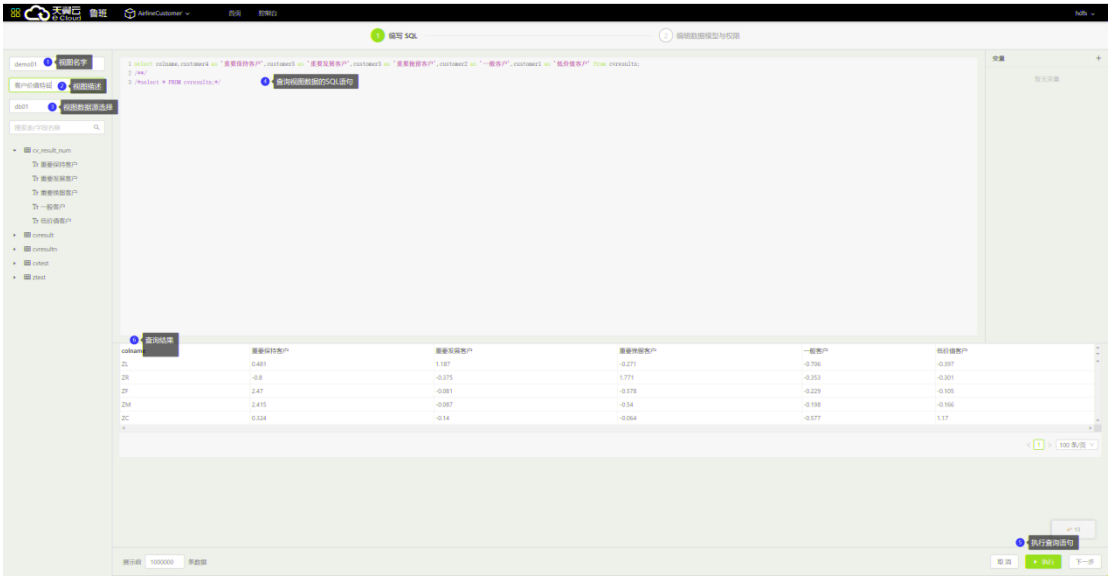


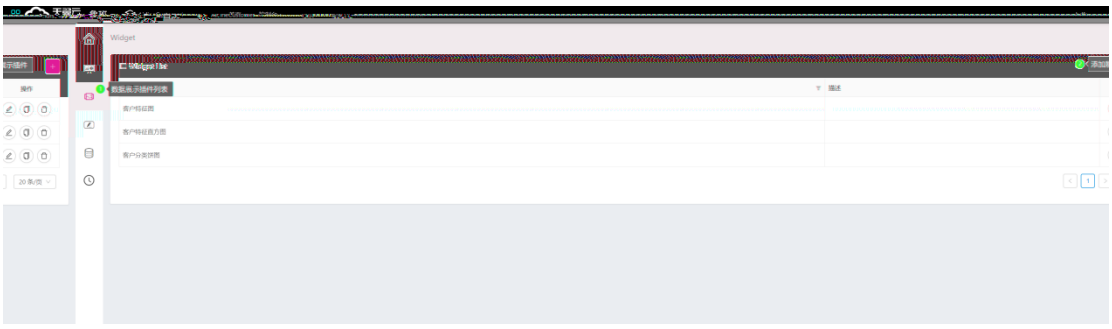
Step 5

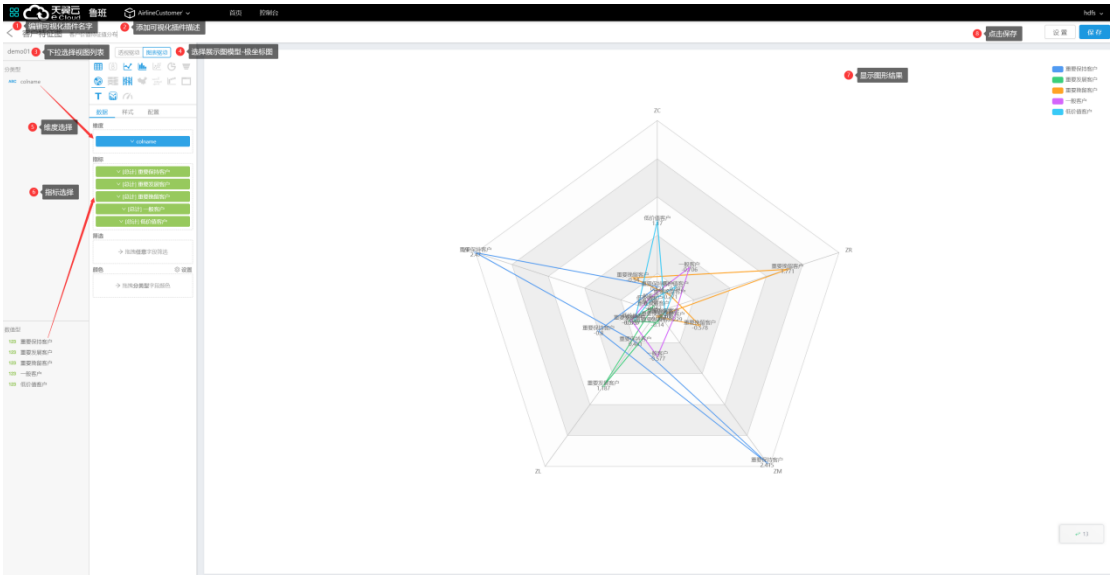


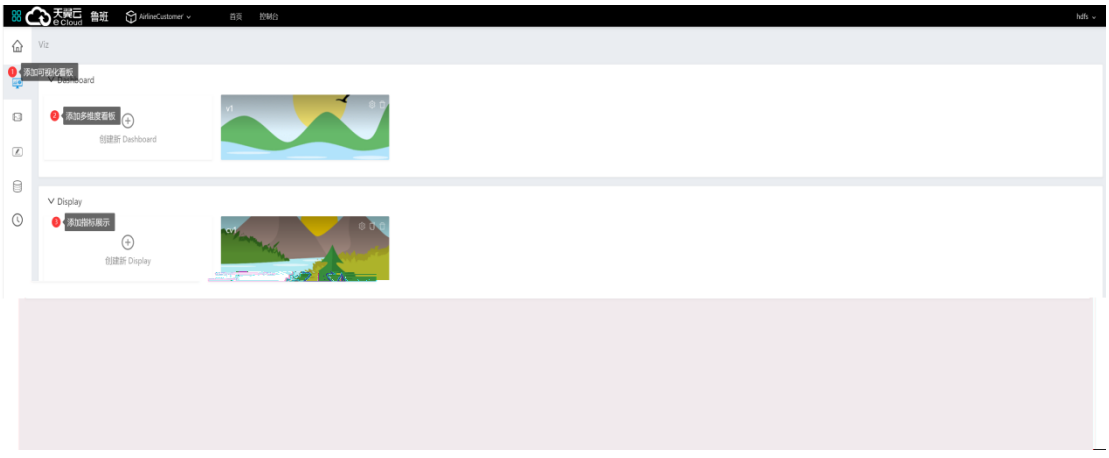

修改 Source
设置数据源信息
添加数据源
选择数据源列表
名称:
类型:
数据库:
用户名:
root
密码:
连接Url:
描述:
配置信息
Key
Value
操作
暂无数据
点击保存
保存
取消


View List
点击添加视图
选择视图列表
名称
描述
Source
操作
demo01
demo2
db01

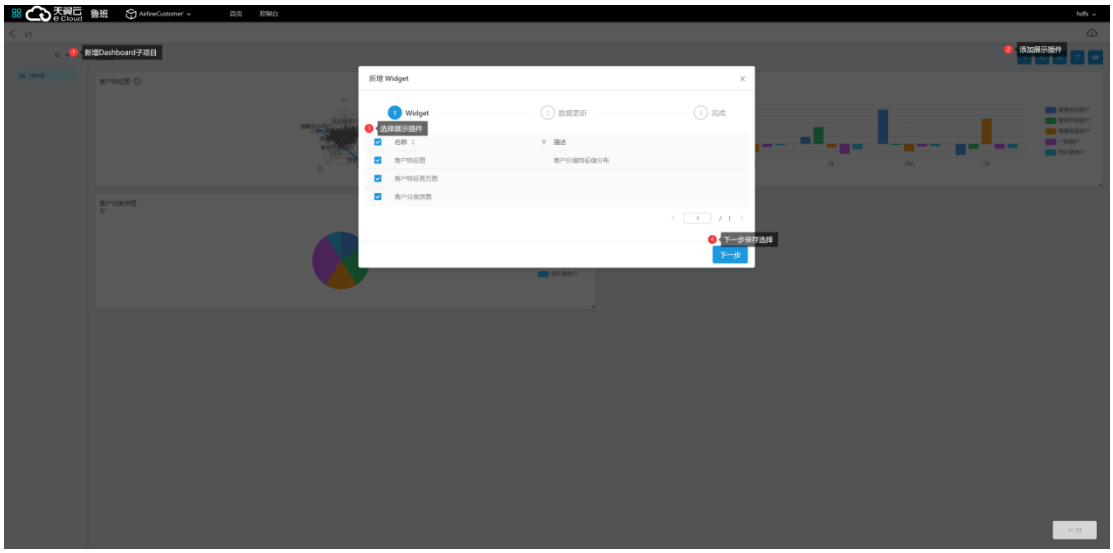

编写 SQL
视图名字
视图描述
视图数据源选择
查询视图数据的SQL语句
查询结果
执行查询语句
colnam
重要保持客户
重要发展客户
重要挽留客户
一般客户
低价值客户
ZL
ZR
ZF
ZM
ZC

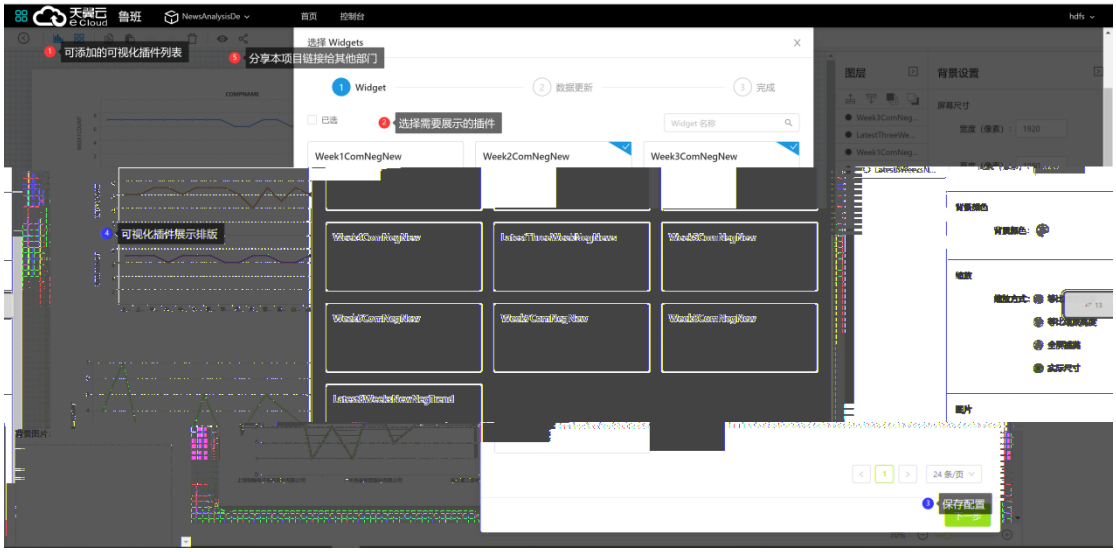


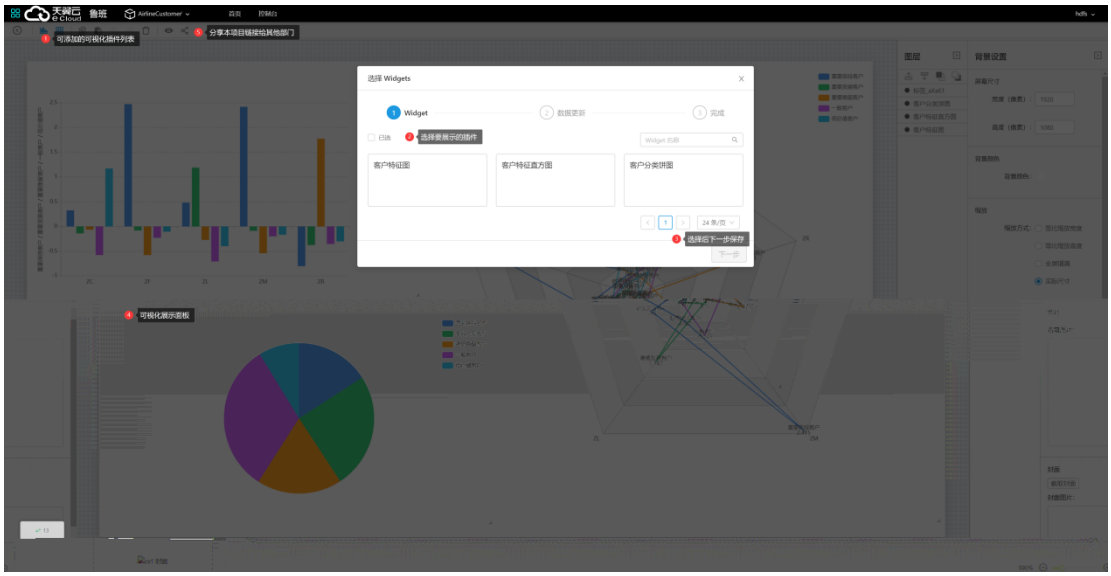




板


新增Dashboard子项目
添加展示插件
新增 Widget
Widget
数据更新
完成
选择展示插件
名称
描述
客户特征图
客户特征直方图
客户分类拼图
下一步保存选择
下一步


可添加的可视化插件列表
分享本项目链接给其他部门
选择 Widgets
Widget
数据更新
完成
选择需要展示的插件
Week1ComNegNew
Week2ComNegNew
Week3ComNegNew
可视化插件展示排版
图层
背景设置
屏幕尺寸
背景颜色
缩放
图片
保存配置


可添加的可视化插件列表
分享本项目链接给其他部门
选择 Widgets
Widget
数据更新
完成
选择要展示的插件
客户特征图
客户特征直方图
客户分类拼图
选择后下一步保存
可视化展示面板

终

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&characterEnc
oding=gbk&autoReconnect=true&failOverReadOnly=false"
     val driver = "com.mysql.cj.jdbc.Driver"
     val username = "root"
     val password = "Bigdata2@20"
```

```
    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 = { // 算出日期对
      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.toDo
uble, 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;
```