Big Data Analysis with IBM Cloud Database

By
K.Yuvaraj

PHASE 4: Development Part 2

Project Definition:

The project aims to harness the power of IBM Cloud Database services for handling and analyzing large datasets. IBM Cloud offers a range of database solutions, including IBM Db2, IBM Cloudant, and IBM Db2 on Cloud, which are ideal for managing Big Data. This project will involve collecting, storing, processing, and analyzing substantial amounts of data using these IBM Cloud Database services to derive meaningful insights, make data-driven decisions, and create valuable outcomes.

Abstract:

Big Data and Cloud Computing as two mainstream technologies, are at the center of concern in the IT feld. Every day a huge amount of data is produced from diferent sources. This data is so big in size that traditional processing tools are unable to deal with them. Besides being big, this data moves fast and has a lot of variety. Big Data is a concept that deals with storing, processing and analyzing large amounts of data. Cloud computing on the other hand is about ofering the infrastructure to enable such processes in a cost-efective and efcient manner. Many sectors, including among others businesses (small or large), healthcare, education, etc. are trying to leverage the power of Big Data. In healthcare, for example, Big Data is being used to reduce costs of treatment, predict outbreaks of pandemics, prevent diseases etc.

This paper, presents an overview of Big Data Analytics as a crucial process in many felds and sectors. We start by a brief introduction to the concept of Big Data, the amount of data that is generated on a daily bases, features and characteristics of Big Data. We then delve into Big Data Analytics were we discuss issues such as analytics cycle, analytics benefts and the movement from ETL to ELT paradigm as a result of Big Data analytics in Cloud. As a case study we analyze Google's BigQuery which is a fully-managed, serverless data warehouse that enables scalable analysis over petabytes of data. As a Platform as a Service (PaaS) supports querying using ANSI SQL. We use the tool to perform diferent experiments such as average read, average compute, average write, on diferent sizes of datasets

Our Project:

Requirements:

- SQL(Structured Query Language)
- PYTHON
- R
- JAVA
- NODE.JS
- IBM CLOUD

Designing:

➢ As initial step, We will set up our Development Environment by Installing Python.

Beginning the installation:

```
$ sudo apt-get install python3.8
```

To verify the installation enter the following commands in your Terminal.

```
Python3.8
```

Install Flask:

Use pip, Python's package manager, to install Flask by running **pip install flask** in your command line.

Setting up the structure(python):

```
from pyspark import SparkConf, SparkContext

import collections

import csv
from collections import namedtuple

from pyspark.sql import Row

from pyspark.sql import SQLContext

import os

from datetime import datetime,timedelta

import datetime

import webbrowser


from neo4jrestclient.client import GraphDatabase


registered_num = []


#ID0, CALLING_NUM1, CALLED NUMBER2, START TIME3, END TIME4, CALL TYPE5,
CHARGE6, CALL RESULT7

fields = ('userID', 'calling_num', 'called_num', 'startTime', 'endTime',
'callType','callResult','churn')

User = namedtuple('User', fields)
```

```
def preprocess(x):
    x = x.split(',')
    for i in range(len(x)):
        x[i] = str(x[i])
    #adding to userid
    if x[1] not in registered_num:
        registered_num.append(x[1])
    if x[2] not in registered_num:
        registered_num.append(x[2])
    #print len(registered_num)
    return


#Get graph  relation csv
def getGraphRelation(x):
    y = ()
    if(type(x)!=type(y)):
        x = x.split(',')
        for i in range(len(x)):
            x[i] = str(x[i])
    graph_startnode = x[2]          #calling NUM          This will also be its key
    graph_endnode = x[3] #called num
    key = str(graph_startnode)+'|'+str(graph_endnode)
    edgeWeight  = x[1]
    return key, edgeWeight
```

```
#get Pulse
def getPulse(startTime,endTime):
    t = startTime.split('T')[1].split(':') #2016-05-06T14:40:15.213+05:30
    hr = int(t[0])
    mn = int(t[1])
    sec = float(t[2].split('+')[0])
    t2 = endTime.split('T')[1].split(':')
    hr2 = int(t2[0])
    mn2 = int(t2[1])
    sec2 = float(t2[2].split('+')[0])
    hrdiff = hr2 - hr
    mndiff = mn2 - mn
    secdiff = sec2 - sec
    pul = (((hrdiff*60*60 + mndiff*60 + secdiff)%60)+1)
    d = startTime.split('T')[0]
    return d,pul

#divide into date key
def arrangeDateWiseFrames(x):
    x = x.split(',')
    for i in range(len(x)):
        x[i] = str(x[i])
    dat,pulse = getPulse(x[3],x[4])
    return dat,pulse,x[1],x[2],x[5],x[8]#date, pulse, calling, called, calltype, churn
```

```
#parsefile
def getUser(x):
    x = x.split(',')
    for i in range(len(x)):
        x[i] = str(x[i])
    user_row = User(x[0],x[1],x[2],x[3],x[4],x[5],x[6],x[7])
    return user_row


def getDate(dRange):
    datelist = dRange
    least = datelist[0]
    most = datelist[len(datelist)-1]
    least = least.split('-')
    l = ''
    l = l + least[2] + least[1] + least[0]
    most = most.split('-')
    m = ''
    m = m + most[2] + most[1] + most[0]
    l = datetime.datetime.strptime(l, "%d%m%Y")
    m = datetime.datetime.strptime(m, "%d%m%Y")
    return l, m


def getWeeks(d1, d2):
    monday1 = (d1 - timedelta(days=d1.weekday()))
    monday2 = (d2 - timedelta(days=d2.weekday()))
```

```
        return (monday2 - monday1).days / 7


def addSevenDays(oldDate):
        plusSeven = oldDate + datetime.timedelta(days=7)
        return plusSeven


def dateToString(d):
        s = d.strftime('%Y-%m-%d')
        return s


def stringToDate(s):
        s = s.split('-')
        l = ''
        l = l + s[2] + s[1] + s[0]
        d = datetime.datetime.strptime(l, "%d%m%Y")
        return d


def writeEdgecsv(edgeRdd, i):
        fedge = open('output/node4jinput/edgeWeight'+str(i)+'.csv', 'a')
        edgeList = edgeRdd.collect()
        for x in edgeList:
                weight = x[0]
                node = x[1].split('|')
                fedge.write(str(node[0])+','+str(node[1])+','+str(weight)+'\n')
        fedge.close()
```

```
    return


def findChurnInThisWeek(weekRdd, i):
    weekList = weekRdd.collect()
    nodeDict = {}
    for m in weekList:
        x = m[2]
        l = m[5]
        nodeDict[x] = l
    fweek = open('output/node4jinput/nodeLabel'+str(i)+'.csv', 'a')
    for x in nodeDict.keys():
        if(nodeDict[x] == 'false'):
            label = 'non-churner'
        else:
            label = 'churner'
        fweek.write(str(x)+','+str(label)+'\n')
    fweek.close()
    return


def graphParam():
    return


def main(sc):
    lines = sc.textFile("/Users/ankita_mehta/Desktop/BDA_Project/cdr_Dataset2.csv")
    #line_rdd = lines.map(getUser)
    parts = lines.map(lambda l: l.split(","))
    #Unique list of users....1. userid 2. userno
    user = parts.map(lambda p: (p[0], p[1].strip())).distinct()
    #Divide them in time frames
    dateKeyRdd = lines.map(arrangeDateWiseFrames)
    dateKeyRdd = dateKeyRdd.sortByKey()
    #dateKeyRdd.coalesce(1).saveAsTextFile("output/dateKeyRdd6")

    #get unique date ranges
    dateRange = dateKeyRdd.keys().distinct()
    l, m = getDate(sorted(dateRange.collect())) #returns min and max date range
    no_of_weeks = getWeeks(l,m)         #gets no of weeks of data we have



    #Now we split of data into weekly based Rdds
    weeksRdd = []
    edgeWeightRdd = []

    for i in range(0,no_of_weeks):
        weeksR = dateKeyRdd.filter(lambda (k,v1,v2,v3,v4,v5):
stringToDate(k).date()>=l.date() and stringToDate(k).date()<=addSevenDays(l).date())
        weeksRdd.append(weeksR)
        edgeWeight = weeksR.map(getGraphRelation).reduceByKey(lambda x, y: x + y)
        edgeWeight = edgeWeight.map(lambda (x,y): (y,x)).sortByKey()
        edgeWeightRdd.append(edgeWeight)
        l = addSevenDays(l)


#We find the relation between two nodes and its edge weight
#Write csv for node4j format
i = 0
for i in range(0,no_of_weeks):
        #write Edge
        filename = 'output/node4jinput/edgeWeight'+str(i)+'.csv'
        try:
                os.makedirs(os.path.dirname(filename))
        except:
                p = 0
        fedge = open(filename, 'w')
        fedge.write(':START_ID,:END_ID,:TYPE\n')
        fedge.close()
        writeEdgecsv(edgeWeightRdd[i], i)
        #Write Node
        filename = 'output/node4jinput/nodeLabel'+str(i)+'.csv'
        try:
                os.makedirs(os.path.dirname(filename))
        except:
                p = 0
        fweek = open(filename, 'w')
        fweek.write('personId:ID,:LABEL\n')
        fweek.close()
```

```
            findChurnInThisWeek(weeksRdd[i], i)
        #lets see if output comes here
        #weeksRdd[0].coalesce(1).saveAsTextFile("output/weekKeyRdd-4")
        #edgeWeightRdd[0].coalesce(1).saveAsTextFile("output/edgeWeightRdd-4")


        #We find the relation between two nodes and its edge weight
        edgeWeight = lines.map(getGraphRelation).reduceByKey(lambda x, y: x + y)
        edgeWeight = edgeWeight.map(lambda (x,y): (y,x)).sortByKey()
        #edgeWeight.coalesce(1).saveAsTextFile("output/edgeWeightRdd6")
        return


if __name__== "__main__":
        conf = SparkConf().setMaster("local").setAppName("ChurnPrediction")
        sc = SparkContext(conf = conf)
```

Setting up the structure(java):

```
import java.io.IOException;


import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.hbase.HBaseConfiguration;
import org.apache.hadoop.hbase.HTableDescriptor;
import org.apache.hadoop.hbase.client.HBaseAdmin;
import org.apache.hadoop.security.UserGroupInformation;
```

```
public class TestKrb5Login {
  public static void main(String[] args){
    try {
      UserGroupInformation.loginUserFromKeytab(args[0],
          "/root/demo/usera.keytab");
      UserGroupInformation ugi = UserGroupInformation.getCurrentUser();
      System.out.println("++++++" + ugi.getUserName() + "++++++");


      Configuration configuration = HBaseConfiguration.create();
      HBaseAdmin hBaseAdmin = new HBaseAdmin(configuration);


      HTableDescriptor[] htables = hBaseAdmin.listTables();
      for (HTableDescriptor descriptor : htables) {
        System.out.println(descriptor.getTableName().getNameAsString());
      }
    } catch (IOException e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
    }
  }
}
```

Anlysis the data using js (java script):

```
const conditionTemplate = `
```

```
<Row>
    <Col span="1" style="color:white; font-size:20px; text-align: right;line-
height:100px;margin-right:5px">
        时间：
    </Col>
    <Col span="5" >
        <Row>
            <Col span="6" style="color:white; font-size:15px; text-align: right;line-
height:50px;margin-right:15px">
                开始时间
            </Col>
            <Col span="6" style="text-align: center;line-height:50px">
                <DatePicker v-model="queryItem.startTime" size="large" type="month"
clearable placeholder="选择开始时间" style="width: 200px"></DatePicker>
            </Col>
        </Row>
        <Row>
            <Col span="6" style="color:white; font-size:15px; text-align: right;line-
height:50px;margin-right:15px">
                结束时间
            </Col>
            <Col span="6" style="text-align: center;line-height:50px">
                <DatePicker v-model="queryItem.endTime" size="large" type="month"
clearable placeholder="选择结束时间" style="width: 200px"></DatePicker>
```

```
            </Col>
        </Row>
    </Col>
    <Col span="1" style="color:white; font-size:20px; text-align: right;line-
height:100px;margin-right:50px">
        <Row  style="line-height:60px;">单位：</Row>
        <Row style="line-height:40px; font-size:17">
            <Checkbox id="checkBox" v-model="queryItem.zongdui" @on-
change="zongduiChange">总队</Checkbox>
        </Row>
    </Col>
    <Col span="8" >
        <Row>
            <Col span="2" style="color:white; font-size:15px; text-align: right;line-
height:50px;margin-right:15px">
                支队
            </Col>
            <Col span="5" style="text-align: center;line-height:50px">
                <Select placeholder="请选择支队" v-model="queryItem.zhidui" @on-
change="getDaduiSelect" multiple clearable :disabled="zongduiFlag"
style="width: 200px">
                    <Option v-for="item in zhiduiSelect" :value="item.id"
 :key="item.id">{{item.unitName}}</Option>
                </Select>
            </Col>
            <Col span="5" style="color:white; font-size:15px; text-align: right;line-height:50px;margin-right:15px">
                大队
            </Col>
            <Col span="5" style="text-align: center;line-height:50px">
                <Select placeholder="请选择大队" v-model="queryItem.dadui" @on-change="getStationSelect" multiple clearable :disabled="zongduiFlag" style="width: 200px">
                    <Option v-for="item in daduiSelect" :value="item.id" :key="item.id">{{item.unitName}}</Option>
                </Select>
            </Col>
        </Row>
        <Row>
            <Col span="2" style="color:white; font-size:15px; text-align: right;line-height:50px;margin-right:15px">
                消防站
            </Col>
            <Col span="5" style="text-align: center;line-height:50px">
                <Select placeholder="请选择消防站" v-model="queryItem.station" @on-change="stationChange" multiple clearable :disabled="zongduiFlag" style="width: 200px">
                    <Option v-for="item in stationSelect" :value="item.id" :key="item.id">{{item.unitName}}</Option>
                </Select>
            </Col>
            <Col span="5" style="color:white; font-size:15px; text-align: right;line-height:50px;margin-right:15px">
                个人
            </Col>
            <Col span="5" style="text-align: center;line-height:50px">
                <Select placeholder="请选择个人" v-model="queryItem.person" @on-change="personChange" multiple clearable :disabled="zongduiFlag" style="width: 200px">
                    <Option v-for="item in personSelect" :value="item.id" :key="item.id">{{item.name}}</Option>
                </Select>
            </Col>
        </Row>
    </Col>
    <Col span="1" style="color:white; font-size:20px; text-align: right;line-height:100px;margin-right:5px">
        指标:
    </Col>
    <Col span="5" >
        <Row>
            <Col span="6" style="color:white; font-size:15px; text-align: right;line-height:50px;margin-right:15px">
        一级指标
      </Col>
      <Col span="5" style="text-align:center; line-height:50px">
        <Select placeholder="请选择一级指标" v-model="queryItem.firstTarget" @on-change="getSecondTarget" clearable style="width: 200px">
          <Option v-for="item in firstTargetSelect" :value="item.id" :key="item.id">{{item.name}}</Option>
        </Select>
      </Col>
    </Row>
    <Row>
      <Col span="6" style="color:white; font-size:15px; text-align: right;line-height:50px;margin-right:15px">
        二级指标
      </Col>
      <Col span="5" style="text-align: center;line-height:50px">
        <Select placeholder="请选择二级指标" v-model="queryItem.secondTarget" clearable style="width: 200px">
          <Option v-for="item in secondTargetSelect" :value="item.id" :key="item.id">{{item.name}}</Option>
        </Select>
      </Col>
    </Row>
    </Col>
    <Col span="2" style="color:white;text-align: center;line-height:100px;">
        <Button type="primary" icon="ios-search" @click="search"
 style="width:80px; height:40px; font-size:15px">查询</Button>
    </Col>
</Row>
`

const condition = new Vue({
    el: '#condition',
    template: conditionTemplate,
    data(){
        return{
            queryItem:{
                startTime:'',
                endTime:'',
                zongdui:false,
                zhidui:[],
                dadui:[],
                station:[],
                person:[],
                firstTarget:'',
                secondTarget:''
            },
            oldStation:[],
            oldPerson:[],
            zhiduiSelect:[],
            daduiSelect:[],
            stationSelect:[],
            personSelect:[],
            zongduiFlag:false,


            firstTargetSelect:[],
            secondTargetSelect:[],
          }
        },
        methods: {
          check(){
            if(this.queryItem.startTime=='' || this.queryItem.endTime==''){
              this.$Message.error("月份不能为空！！")
              return false;
            }
            if(this.queryItem.startTime > this.queryItem.endTime){
              this.$Message.error("结束月份不能小于开始月份！！")
              return false;
        }
        if(this.queryItem.secondTarget == ''){
            this.$Message.error("二级指标未选择！！")
            return false;
        }

        return true;
    },
    search(){
        let flag = this.check()
        if(flag){
            initEcharts(this.queryItem)
        }
    },
    zongduiChange(){
        if(this.queryItem.zongdui){
            this.$Notice.info({
                title: '注意',
                desc: '选择总队，则支队、大队、消防站和个人都被置为不可选
！',
                duration: 8
            });
            this.zongduiFlag = true
        this.queryItem.zhidui = []
        this.queryItem.dadui  =  []
        this.queryItem.station = []
        this.queryItem.person = []
    }else{
        this.zongduiFlag = false
    }
},
getDaduiSelect(){
    let zhiduiData = this.queryItem.zhidui
    if(zhiduiData.length == 0){
        this.daduiSelect = []
    }else{
        // console.log(zhiduiData);
        $.ajax({
            type:'GET',
            url: 'http://localhost:8880/unit/getUnitByParentId',
            data:{
                parentId:  zhiduiData[zhiduiData.length-1]
            },
            success: function(response){
                condition.daduiSelect = response.extra.unitList
            },
                error: function(response){
                    console.log(response);
                }
            })
        }
    },
    getStationSelect(){
        let daduiData = this.queryItem.dadui
        if(daduiData.length == 0){
            this.stationSelect = []
        }else{
            $.ajax({
                type:'GET',
                url: 'http://localhost:8880/unit/getUnitByParentId',
                data:{
                    parentId: daduiData[daduiData.length-1]
                },
                success: function(response){
                    condition.stationSelect = response.extra.unitList
                },
                error: function(response){
                    console.log(response);
                }
                })
            }
        },
        stationChange(value){
            if((value.length!=0 && this.oldStation.length==0) || (value.length==0 &&
this.oldStation.length!=0)){
                this.queryItem.firstTarget=''
            }
            this.oldStation = [].concat(value)
        },
        personChange(value){
            if((value.length!=0 && this.oldPerson.length==0) || (value.length==0 &&
this.oldPerson.length!=0)){
                this.queryItem.firstTarget=''
            }
            this.oldPerson = [].concat(value)
        },
        getSecondTarget(){
            this.secondTargetSelect=[]
            if(this.queryItem.person!=''){
                if(this.queryItem.firstTarget == 1){
                    this.secondTargetSelect=[
                        {id:1, name:'十事联动参与次数'},
        {id:2, name:'三课一会参与次数'},
    ]
}
if(this.queryItem.firstTarget == 2){
    this.secondTargetSelect=[
        {id:3, name:'课程完成情况'},
        {id:4, name:'考试完成情况'},
        {id:5, name:'考试分数'},
    ]
}
if(this.queryItem.firstTarget == 3){
    this.secondTargetSelect=[
        {id:6, name:'躁狂'},
        {id:7, name:'抑郁'},
        {id:8, name:'焦虑'},
        {id:9, name:'敌对'},
        {id:10, name:'强迫'},
        {id:11, name:'其他'},
    ]
}
if(this.queryItem.firstTarget == 4){
    this.secondTargetSelect=[
          {id:12, name:'评价合格率'}
        ]
      }
      if(this.queryItem.firstTarget == 5){
        this.secondTargetSelect=[
          {id:13, name:'睡眠质量不良次数'},
          {id:14, name:'训练不合格次数'},
          {id:15, name:'违规驾驶次数'},
          {id:16, name:'手机违规使用次数'},
        ]
      }
    }else if(this.queryItem.station!='' && this.queryItem.person==''){
      if(this.queryItem.firstTarget == 1){
        this.secondTargetSelect=[
          {id:1, name:'十事联动参与率'},
          {id:2, name:'三课一会参与率'},
        ]
      }
      if(this.queryItem.firstTarget == 2){
        this.secondTargetSelect=[
          {id:3, name:'课程完成率'},
          {id:4, name:'考试完成率'},
        {id:5, name:'平均分'},
    ]
}
if(this.queryItem.firstTarget == 3){
    this.secondTargetSelect=[
        {id:6, name:'参评人数'},
        {id:7, name:'心理咨询师数量'},
        {id:8, name:'异常人数'},
        {id:9, name:'已干预人数'},
        {id:10, name:'正在干预人数'},
    ]
}
if(this.queryItem.firstTarget == 4){
    this.secondTargetSelect=[
        {id:11, name:'优秀人数'},
        {id:12, name:'称职人数'},
        {id:13, name:'基本称职人数'},
        {id:14, name:'不称职人数'},
    ]
}
if(this.queryItem.firstTarget == 5){
    this.secondTargetSelect=[
            {id:15, name:'睡眠质量不良人数'},
            {id:16, name:'训练不合格人数'},
            {id:17, name:'违规驾驶人数'},
            {id:18, name:'手机违规使用人数'},
          ]
      }
  }else{
      if(this.queryItem.firstTarget == 1){
          this.secondTargetSelect=[
            {id:1, name:'支部开展率'},
            {id:2, name:'党员参与率'},
            {id:3, name:'党委数'},
            {id:4, name:'支部数'},
            {id:5, name:'人员总数'},
          ]
      }
      if(this.queryItem.firstTarget == 2){
          this.secondTargetSelect=[
            {id:6, name:'总完成率'},
            {id:7, name:'平均分'}
          ]
      }
```

```
if(this.queryItem.firstTarget == 3){
   this.secondTargetSelect=[
       {id:8, name:'参评人数'},
       {id:9, name:'心理咨询师数量'},
       {id:10, name:'异常人数'},
       {id:11, name:'已干预人数'},
       {id:12, name:'正在干预人数'},
   ]
}
if(this.queryItem.firstTarget == 4){
   this.secondTargetSelect=[
       {id:13, name:'支部开展率'},
       {id:14, name:'人员参与率'},
       {id:15, name:'优秀人数'},
       {id:16, name:'称职人数'},
       {id:17, name:'基本称职人数'},
       {id:18, name:'不称职人数'},
   ]
}
if(this.queryItem.firstTarget == 5){
   this.secondTargetSelect=[
       {id:19, name:'睡眠质量不良人数'},
                    {id:20, name:'训练不合格人数'},
                    {id:21, name:'违规驾驶人数'},
                    {id:22, name:'手机违规使用人数'},
                  ]
                }
              }
            },
          }
        })

        //初始化折线图
        function initEcharts(queryItem) {
          // 实例化对象
          var myChart = echarts.init(document.querySelector("#result"));

          // 指定配置和数据
          option = {
            color: ['#FF6EB4', '#ffff00', '#7fff00', '#00f2f1', '#FD866A', '#9E87FF',
        '#58D5FF'],
            title:{
              show:true,
              text:'',
              textStyle:{
```

```
            color:'#ffffff',
            fontSize:20
        },
        left:"7%",
        top:"3%",
    },
    tooltip: {
        trigger: 'axis',
    },
    legend: {
        top: "5%",
        // 距离容器10%
        right: "10%",
        // 修饰图例文字的颜色
        textStyle: {
            color: "#FFFFFF",
            fontSize:16
        }
    },
    grid: {
        top: "20%",
        left: "5%",
    right: "5%",
    bottom: "10%",
    show: true,
    borderColor: "rgba(255,255,255,0.2)",
    containLabel: false
},
xAxis: {
    type: "category",
    boundaryGap: false,
    axisPointer: {
        type: 'shadow'
    },
    data: [],
    // 修饰刻度标签的颜色
    axisLine: {
        lineStyle: {
            color: "white"
        }
    },
    axisLabel: {
        interval: 0,
        fontSize: 16
    },
},
yAxis: {
    name: '平均分',
    type: "value",
    // 修饰刻度标签的颜色
    axisLine: {
        lineStyle: {
            color: "white"
        }
    },
    // 修改y轴分割线的颜色
    splitLine: {
        lineStyle: {
            color: "rgba(255,255,255,0.2)",
        }
    },
    axisLabel: {
        fontSize: 15
    },
  },

series: []
```

```
    };


    let xAxisDate = getMonthBetween(dateToString(queryItem.startTime),
dateToString(queryItem.endTime));
    let seriesDataClub = getseriesData(queryItem, xAxisDate)
    let seriesData = seriesDataClub.seriesData
    let unit = seriesDataClub.unit
    let yAxisName = getyAxisName(queryItem, unit)
    let title = getTitle(queryItem)


    console.log(seriesData);


    option.title.text = title + '---' + yAxisName
    option.xAxis.data = xAxisDate
    option.yAxis.name = yAxisName
    option.series = []
    for(let i=0; i<seriesData.length; i++){
        option.series.push(
            {
                name: seriesData[i].name,
                type: "line",
                lineStyle:{
                    width: 4
                },
                // 是否让线条圆滑显示
                smooth: true,
                data: seriesData[i].data
            }
        )
    }

    // 把配置给实例对象
    myChart.setOption(option, true);
    window.addEventListener("resize", function () {
        myChart.resize();
    });
}

function getseriesData(queryItem, xAxisDate){
    let info = getUnit(queryItem)
    let unitFlag = info.flag
    let unit = info.unit
    let seriesData = []
    console.log(unitFlag);
    console.log(unit);
    for(let i=0; i<unit.length;i++){
        let item = {name:'', data:[]}
        item.name = unitList[unitFlag-1][unit[i]-1].name
        for(let j=0; j<xAxisDate.length; j++){
            item.data.push(Math.round(Math.random()*51 + 49))
        }
        seriesData.push(item)
    }

    return {"seriesData": seriesData, "unit": unitFlag}
}

function getUnit(queryItem){
    if(queryItem.person != ''){
        return {"flag":5, "unit": queryItem.person}
    }else if(queryItem.station != ''){
        return {"flag":4, "unit": queryItem.station}
    }else if(queryItem.dadui != ''){
        return {"flag":3, "unit": queryItem.dadui}
    }else if(queryItem.zhidui != ''){
        return {"flag":2, "unit": queryItem.zhidui}
    }else if(queryItem.zongdui){
        return {"flag":1, "unit": [1]}
    }else{
        return {"flag":1, "unit": []}
    }
}

function getyAxisName(queryItem, unit){
    console.log(unit);
    let secondTarget = [];
    if(unit==5){
        secondTarget = [
            {id:1, name:'十事联动参与次数'},
            {id:2, name:'三课一会参与次数'},
            {id:3, name:'课程完成情况'},
            {id:4, name:'考试完成情况'},
            {id:5, name:'考试分数'},
            {id:6, name:'躁狂'},
            {id:7, name:'抑郁'},
            {id:8, name:'焦虑'},
            {id:9, name:'敌对'},
            {id:10, name:'强迫'},
            {id:11, name:'其他'},
            {id:12, name:'评价合格率'},
            {id:13, name:'睡眠质量不良次数'},
            {id:14, name:'训练不合格次数'},
            {id:15, name:'违规驾驶次数'},
            {id:16, name:'手机违规使用次数'},
        ]
    }else if(unit == 4){
        secondTarget = [
            {id:1, name:'十事联动参与率'},
            {id:2, name:'三课一会参与率'},
            {id:3, name:'课程完成率'},
            {id:4, name:'考试完成率'},
            {id:5, name:'平均分'},
            {id:6, name:'参评人数'},
            {id:7, name:'心理咨询师数量'},
            {id:8, name:'异常人数'},
            {id:9, name:'已干预人数'},
            {id:10, name:'正在干预人数'},
            {id:11, name:'优秀人数'},
            {id:12, name:'称职人数'},
            {id:13, name:'基本称职人数'},
            {id:14, name:'不称职人数'},
            {id:15, name:'睡眠质量不良人数'},
            {id:16, name:'训练不合格人数'},
            {id:17, name:'违规驾驶人数'},
            {id:18, name:'手机违规使用人数'},
        ]
    }else{
        secondTarget = [
            {id:1, name:'支部开展率'},
            {id:2, name:'党员参与率'},
            {id:3, name:'党委数'},
            {id:4, name:'支部数'},
            {id:5, name:'人员总数'},
            {id:6, name:'总完成率'},
            {id:7, name:'平均分'},
            {id:8, name:'参评人数'},
            {id:9, name:'心理咨询师数量'},
            {id:10, name:'异常人数'},
            {id:11, name:'已干预人数'},
            {id:12, name:'正在干预人数'},
            {id:13, name:'支部开展率'},
            {id:14, name:'人员参与率'},
            {id:15, name:'优秀人数'},
            {id:16, name:'称职人数'},
            {id:17, name:'基本称职人数'},
            {id:18, name:'不称职人数'},
            {id:19, name:'睡眠质量不良人数'},
            {id:20, name:'训练不合格人数'},
            {id:21, name:'违规驾驶人数'},
            {id:22, name:'手机违规使用人数'},
        ]
    }

    for(let i=0; i<secondTarget.length; i++){
        if(queryItem.secondTarget == secondTarget[i].id){
            return secondTarget[i].name
        }
    }
}

function getTitle(queryItem){
    let firstTarget = [
        {id:1, name:'智慧党建'},
        {id:2, name:'政治教育'},
        {id:3, name:'心理测询'},
        {id:4, name:'全员考核'},
        {id:5, name:'智慧营区'},
    ]
    for(let i=0; i<firstTarget.length; i++){
        if(queryItem.firstTarget == firstTarget[i].id){
            return firstTarget[i].name
        }
    }
}

function getMonthBetween(start, end){
    let result = []
    let s = start.split('-')
    let e = end.split('-')

    let min = new Date();
    let max = new Date();
    min.setFullYear(s[0], s[1])
    max.setFullYear(e[0], e[1])
    let curr = min
    let str = ""
    while(curr <= max){
        let month = curr.getMonth()
        if(month === 0){
            str = (curr.getFullYear()-1) + '-' + 12
        }else{
            str = curr.getFullYear() + '-' + (month<10 ? ('0'+month):month)
        }

        result.push(str)
        curr.setMonth(month+1)
    }
    return result;
}

function dateToString(date){
    var year = date.getFullYear()
    var month =(date.getMonth() + 1).toString()
    if (month.length == 1) {
        month = "0" + month;
    }
    var dateTime = year + "-" + month
    return dateTime;
}

let unitList = [
    [
        {id:1, name:'湖北总队'}
    ],
    [
        {id: 1, name: '武汉'},
        {id: 2, name: '鄂州'},
        {id: 3, name: '黄冈'},
        {id: 4, name: '襄阳'},
        {id: 5, name: '荆门'},
        {id: 6, name: '宜昌'},
        {id: 7, name: '孝感'},
        {id: 8, name: '荆州'},
    ],
    [
        {id: 1, name: '武昌'},
        {id: 2, name: '汉阳'}
    ],
    [
        {id: 1, name: '珞珈山'},
        {id: 2, name: '七里庙'}
    ],
    [
        {id: 1, name: '张三'},
        {id: 2, name: '李四'}
    ]
]

function getFirstTarget(){
    $.ajax({
        type:'GET',
        url: 'http://localhost:8880/statisticItem/getFirstItem',
        success: function(response){
            condition.firstTargetSelect = response.extra.firstItemList
        },
        error: function(response){
            console.log(response);
        }
    })
}
```

```
function getZhiduiSelect(){
    $.ajax({
        type:'GET',
        url: 'http://localhost:8880/unit/getZhidui',
        success: function(response){
            condition.zhiduiSelect = response.extra.zhiduiList
        },
        error: function(response){
            console.log(response);
        }
    })
}

window.onload=()=>{
    // getZhidui();
    getFirstTarget();
    getZhiduiSelect();


}
```

Now we are successfully created our big data analysis using python, java &node.js.

we are building the big data analyis solution by applying advanced analysis techniques and visualizing the results.

We are also applying more complex analysis techniques, such as machine learning algorithms, time series analysis, or sentiment analysis, depending on the dataset and objectives

Using tools like Matplotlib, Plotly, or IBM Watson Studio for creating graphs and charts. we create visualizations to showcase the analysis results

Summary

During the development phase of big data analysis with IBM Cloud Databases, the primary focus is on establishing a robust connection between the application and the database. This phase begins with configuring the necessary credentials and connection parameters to ensure secure and authorized access to the IBM Cloud Database service. Once the connection is established, SQL queries are formulated to extract relevant data from the database tables. These queries are tailored according to the specific analysis requirements, filtering and aggregating data as needed. The retrieved data is often transformed into a structured format, such as a pandas DataFrame in Python, enabling seamless integration with analytical tools. During this phase, developers also focus on error handling and data validation to ensure the accuracy and reliability of the analysis. Key statistical and analytical operations are performed on the retrieved data, providing valuable insights into patterns, trends, and relationships within the dataset. This phase is crucial for laying the foundation of the entire analysis process, ensuring that the subsequent stages, such as data cleaning, feature engineering, and modeling, can be executed effectively based on the insights gained from the initial database analysis. Attention to detail and precision in this phase is vital, as the accuracy of the subsequent analysis heavily depends on the integrity of the extracted and processed data.