第6章 统计分析

学习目标

- 完成商品类目销售分析表
- 掌握echars图表插件的基本用法
- 使用echars完成商品类目销售统计饼图

1. 商品类目销售分析表

1.1 需求分析

商品类目销售分析是统计一段时间内各商品分类（一级分类）的销售数量与销售额以及所占总额的比重，如下图：

分类名称	销售数量	数量比例	销售金额	金额比例
彩妆	100	20%	¥10000.00	20%
食品	100	20%	¥10000.00	20%
家电	100	20%	¥10000.00	20%
厨具	100	20%	¥10000.00	20%
男装	100	20%	¥10000.00	20%

1.2 实现思路

（1）销售分析表数据来自订单表和订单明细表，我们需要对订单表和订单明细表进行统计。订单明细表中记录了商品的一级分类，我们按照一级分类聚合运算就可以统计出各分类的销售额。

（2）销售金额的统计要按照优惠后的金额（pay_money）进行统计。

（3）订单表和订单明细表数据量庞大，如果每次都对订单表和订单明细表实时统计，查询效率必然低下。所以我们对大数量的表进行统计，通常是每天定时统计上一天的数据，将统计结果存储在一张表中。当用户执行管理后台的统计功能时，再对这张表进行统计，给前端返回统计结果。

存储统计结果的表结构如下：

字段名称	字段含义	字段类型	字段长度	备注
category_id1	一级分类ID	INT		主键
category_id2	二级分类ID	INT		主键
category_id3	三级分类ID	INT		主键
count_date	统计日期	DATE		主键
num	数量	INT		
money	金额	INT		

1.3 代码实现

1.3.1 类目统计SQL语句

按照商品一级分类、二级分类、三级分类聚合统计销售数量和销售金额，查询条件为支付状态为已支付，未删除的数据并且支付日期为某日的。

```
SELECT category_id1 ,category_id2,category_id3
,DATE_FORMAT(o.`pay_time`,'%Y-%m-%d' ) count_date,SUM(oi.num)
num,SUM(oi.pay_money) money
FROM tb_order_item  oi, tb_order o
WHERE  oi.`order_id`=o.`id` AND o.`pay_status`='1' AND
DATE_FORMAT(o.`pay_time`,'%Y-%m-%d' ) ='2019-04-15'
GROUP  BY
`category_id1`,`category_id2`,`category_id3`,DATE_FORMAT(o.`pay_time`,'%Y
-%m-%d' )
```

DATE_FORMAT是mysql提供的日期转换函数

1.3.2 类目统计代码实现

（1）创建实体类

```
@Table(name="tb_category_report")
public class CategoryReport  implements Serializable {

    @Id
    private Integer categoryId1;//1级分类

    @Id
    private Integer categoryId2;//2级分类

    @Id
    private Integer categoryId3;//3级分类

    @Id
    private Date countDate;//统计日期

    private Integer num;//销售量

    private Integer money;//销售额

    public Integer getCategoryId1() {
        return categoryId1;
    }

    public void setCategoryId1(Integer categoryId1) {
        this.categoryId1 = categoryId1;
    }

    public Integer getCategoryId2() {
        return categoryId2;
    }

    public void setCategoryId2(Integer categoryId2) {
        this.categoryId2 = categoryId2;
    }

    public Integer getCategoryId3() {
        return categoryId3;
    }


    public void setCategoryId3(Integer categoryId3) {
```

```
        this.categoryId3 = categoryId3;
    }

    public Date getCountDate() {
        return countDate;
    }

    public void setCountDate(Date countDate) {
        this.countDate = countDate;
    }

    public Integer getNum() {
        return num;
    }

    public void setNum(Integer num) {
        this.num = num;
    }

    public Integer getMoney() {
        return money;
    }

    public void setMoney(Integer money) {
        this.money = money;
    }

}
```

（2）新建数据访问接口CategoryReportMapper

```
public interface CategoryReportMapper extends Mapper<CategoryReport> {

    @Select("SELECT category_id1 categoryId1,category_id2
categoryId2,category_id3 categoryId3,DATE_FORMAT(o.`pay_time`,'%Y-%m-%d'
) countDate,SUM(oi.num) num,SUM(oi.pay_money) money " +
            "FROM tb_order_item  oi, tb_order o  " +
            "WHERE  oi.`order_id`=o.`id` AND o.`pay_status`='1' AND
DATE_FORMAT(o.`pay_time`,'%Y-%m-%d' ) =#{date} " +
            "GROUP  BY
`category_id1`,`category_id2`,`category_id3`,DATE_FORMAT(o.`pay_time`,'%Y
-%m-%d' ) ")
    public List<CategoryReport> categoryReport(@Param("date") LocalDate
date);


}
```

（3）新建业务接口CategoryReportService

```
/**
 * 报表服务层接口
 */
public interface CategoryReportService {

    /**
     * 商品类目按日期统计(订单表关联查询)
     * @param date
     * @return
     */
    public List<CategoryReport> categoryReport(LocalDate date);

}
```

（4）新建业务实现

```
@Service
public class CategoryReportServiceImpl implements CategoryReportService {

    @Autowired
    private CategoryReportMapper categoryReportMapper;

    @Override
    public List<CategoryReport> categoryReport(LocalDate date) {
        return categoryReportMapper.categoryReport(date);
    }
}
```

（5）新建Controller

```
@RestController
@RequestMapping("/categoryReport")
public class CategoryReportController {

    @Reference
    private CategoryReportService categoryReportService;

    /**
     * 昨天的数据统计
     * @return
     */
    @GetMapping("/yesterday")
    public List<CategoryReport> yesterday(){
        LocalDate localDate = LocalDate.now().minusDays(1);// 得到昨天的日期
        return categoryReportService.categoryReport(localDate);
    }

}
```

浏览器测试： http://localhost:9101/categoryReport/yesterday.do

1.3.3 定时任务-生成统计数据

创建定时任务，每天凌晨1点统计昨天的数据，插入到tb_category_report中。

（1）CategoryReportService新增方法定义

```
public void createData();
```

（2）CategoryReportServiceImpl实现方法

```
    @Transactional
    public void createData() {
        LocalDate localDate = LocalDate.now().minusDays(1);// 得到昨天的日
期
        List<CategoryReport> categoryReports =
categoryReportMapper.categoryReport(localDate);
        for(CategoryReport categoryReport:categoryReports){
            categoryReportMapper.insert(categoryReport);
        }
    }
```

CategoryReportServiceImpl的service注解添加属性

```
@Service(interfaceClass = CategoryReportService.class)
```

（3）OrderTask新增方法

```
    @Reference
    private CategoryReportService categoryReportService;

    @Scheduled(cron = "0 0 1 * * ?")
    public void createCategoryReportData(){
        System.out.println("createCategoryReportData");
        categoryReportService.createData();
    }
```

1.3.4 按日期统计一级分类数据

我们再次对tb_category_report进行聚合运算，得出一个时间段的统计数据

```
SELECT category_id1 categoryId1,SUM(num) num ,SUM(money) FROM
`tb_category_report` WHERE count_date>='2019-04-15' and
count_date<='2019-04-16' GROUP BY category_id1
```

（1）在CategoryReportMapper新增方法

```
 /**
     * 按时间段统计一级类目
     * @param date1
     * @param date2
     * @return
     */
    @Select("SELECT category_id1 categoryId1,SUM(num) num ,SUM(money)
FROM `tb_category_report` WHERE count_date>=#{date1} and count_date<=#
{date2}  GROUP BY category_id1")
    public List<Map> category1Count(@Param("date1") String date1 ,
@Param("date2") String date2 );
```

（2）CategoryReportService新增方法

```
    /**
     * 一级类目统计
     * @param date1
     * @param date2
     * @return
     */
    public List<Map> category1Count(String date1 , String date2 );
```

（3）CategoryReportServiceImpl实现方法

```
    @Override
    public List<Map> category1Count(String date1, String date2) {
        return categoryReportMapper.category1Count(date1,date2);
    }
```

（4）CategoryReportController新增方法

```
    /**
     * 统计一级类目
     * @param date1
     * @param date2
     * @return
     */
    @GetMapping("/category1Count")
    public List<Map> category1Count(String date1, String date2){
        return categoryReportService.category1Count(date1,date2);
    }
```

1.3.5 商品类目统计名称显示

当前的统计结果中，只有分类id，没有分类名称。而最终的结果是要显示分类名称的，如何实现呢？

（1）创建视图

分类名称所在的分类表在商品数据库中，而当前我们连接的是订单数据库，这种情况下我们需要在订单数据库中创建视图，让其得到一级分类列表。

```
CREATE VIEW v_category AS
SELECT id,NAME FROM qingcheng_goods.`tb_category` WHERE parent_id=0
```

（2）改造之前的sql语句

```
SELECT category_id1 categoryId1,c.name categoryName,SUM(num) num,
SUM(money) money
FROM tb_category_report r,v_category c
WHERE r.category_id1=c.id AND  count_date>='2019-04-15' AND
count_date<='2019-04-17'
GROUP BY category_id1,c.name
```

（3）修改CategoryReportMapper接口category1Count方法的sql语句

```
    @Select("SELECT category_id1 categoryId1,c.name categoryName,SUM(num)
num, SUM(money) money " +
            "FROM tb_category_report r,v_category1 c " +
            "WHERE r.category_id1=c.id AND count_date>=#{date1} AND
count_date<=#{date2} " +
            "GROUP BY category_id1,c.name")
    public List<Map>  category1Count( @Param("date1") String
date1,@Param("date2") String date2);
```

1.3.6 商品类目统计前端实现

（1）创建report/categoryReport.html页面，引入样式和js

```
 <!-- 引入样式 -->
    <link rel="stylesheet" href="https://unpkg.com/element-ui/lib/theme-
chalk/index.css">
    <link rel="stylesheet" href="../plugins/font-awesome/css/font-
awesome.min.css">
    <link rel="stylesheet" href="../css/style.css">
```

```
<script src="/js/util.js"></script>
<script src="/js/vue.js"></script>
<script src="/js/axios.js"></script>
<script src="/js/elementui.js"></script>
```

（2）创建表格

```
<div id="app">
<el-table :data="tableData" border style="width: 100%">
    <el-table-column prop="categoryName" label="一级分类" width="200">
</el-table-column>
    <el-table-column prop="num" label="数量" width="200"></el-table-
column>
    <el-table-column prop="money" label="销售额" width="200"></el-table-
column>
</el-table>
</div>
```

（3）创建日期范围控件

```
<el-date-picker
        v-model="dateRange"
        type="daterange"
        range-separator="至"
        start-placeholder="开始日期"
        end-placeholder="结束日期"
        @change="fetchData()">
</el-date-picker>
```

（4）编写脚本

```
<script>
    new Vue({
        el: '#app',
        data(){
            return {
                tableData: [],
                dateRange:[]
            }
        },
        methods:{
            fetchData (){
                //查询统计分析数据
                let date1=this.dateRange[0].Format("yyyy-MM-dd");
                let date2=this.dateRange[1].Format("yyyy-MM-dd");
                axios.get(`/categoryReport/category1Count.do?
date1=${date1}&date2=${date2}`).then(response => {
                    this.tableData = response.data;
                });
            }
        }
    })
</script>
```

1.3.7 比例计算

需求：统计表中显示销售数量比例与销售金额比例，销售金额单位转换为元。

思路：得到报表数据后，通过循环计算总销售数量和总销售额，在表格中通过模板列计算得出比例。

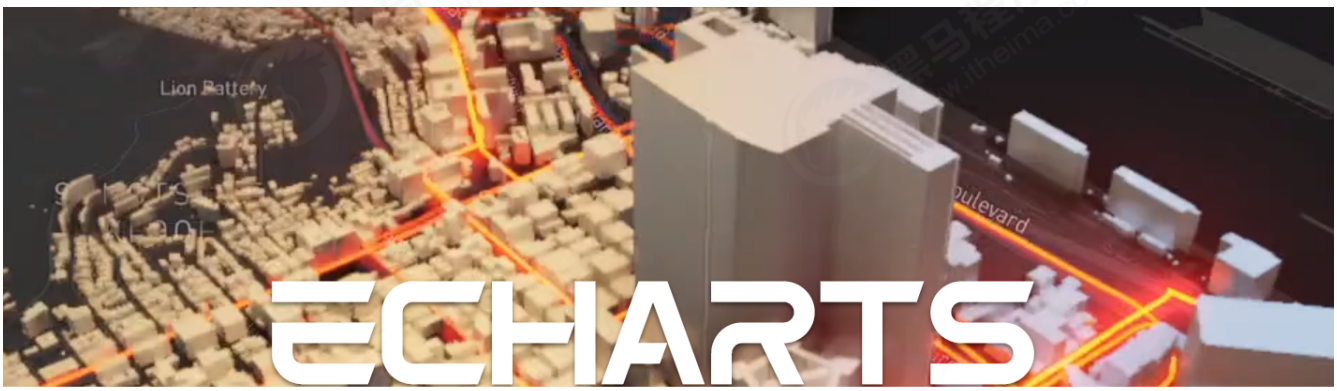
1.4 小结

- 分组聚合SQL语句的编写
- 跨库查询
- 视图的创建与使用
- 通用mapper自定义方法
- 定时任务
- 日期范围控件（前端）

2. EChars图表插件

2.1 EChars简介

ECharts，百度开源的图表插件。一个使用 JavaScript 实现的开源可视化库，可以流畅的运行在 PC 和移动设备上，兼容当前绝大部分浏览器（IE8/9/10/11，Chrome，Firefox，Safari等），底层依赖轻量级的矢量图形库 ZRender，提供直观，交互丰富，可高度个性化定制的数据可视化图表。



ECharts 提供了常规的折线图、柱状图、散点图、饼图、K线图，用于统计的盒形图，用于地理数据可视化的地图、热力图、线图，用于关系数据可视化的关系图、treemap、旭日图，多维数据可视化的平行坐标，还有用于 BI 的漏斗图，仪表盘，并且支持图与图之间的混搭。

详见EChars官网 https://echarts.baidu.com/

2.2 快速入门

2.2.1 简单柱状图

（1）新建页面，引入js

```
<script src="/js/echarts.common.min.js"></script>
```

（2）在页面放置 div用于显示报表

```
<div id="main" style="width: 600px;height:400px;"></div>
```

（3）编写js代码

```
    // 基于准备好的dom，初始化echarts实例
    var myChart = echarts.init(document.getElementById('main'));

    // 指定图表的配置项和数据
    var option = {
        title: {
            text: 'ECharts 入门示例'
        },
        tooltip: {},
        legend: {
            data:['销量']
        },
        xAxis: {
            data: ["衬衫","羊毛衫","雪纺衫","裤子","高跟鞋","袜子"]
        },
        yAxis: {},
        series: [{
            name: '销量',
            type: 'bar',
            data: [5, 20, 36, 10, 10, 20]
        }]
    };

    // 使用刚指定的配置项和数据显示图表。
    myChart.setOption(option);
```

2.2.2 多数据的柱状图

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>ECharts</title>
    <script src="echarts.common.min.js"></script>
</head>
<body>
    <!-- 为 ECharts 准备一个具备大小（宽高）的 DOM -->
    <div id="main" style="width: 600px;height:400px;"></div>

    <script type="text/javascript">
        // 基于准备好的dom，初始化echarts实例
        var myChart = echarts.init(document.getElementById('main'));

        // 指定图表的配置项和数据
        var option = {
            title: {
                text: 'ECharts 入门示例'
            },
            tooltip: {},
            legend: {
                data:['销量','销售额']
            },
            xAxis: {
                data: ["衬衫","羊毛衫","雪纺衫","裤子","高跟鞋","袜子"]
            },
            yAxis: {},
            series: [{
                name: '销量',
                type: 'bar',
                data: [5, 20, 36, 10, 10, 20]
            },{
                name: '销售额',
                type: 'bar',
                data: [30, 33, 56, 88, 23, 55]
            }]
        };


        // 使用刚指定的配置项和数据显示图表。
```

```
        myChart.setOption(option);
    </script>

</body>
</html>
```

2.2.3 简单饼图

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>ECharts</title>
    <script src="echarts.common.min.js"></script>
</head>
<body>
    <!-- 为 ECharts 准备一个具备大小（宽高）的 DOM -->
    <div id="main" style="width: 600px;height:400px;"></div>

    <script type="text/javascript">
        // 基于准备好的dom，初始化echarts实例
        var myChart = echarts.init(document.getElementById('main'));

        // 指定图表的配置项和数据
        option = {
            title : {
                text: '某站点用户访问来源',
                subtext: '纯属虚构',
                x:'center'
            },
            tooltip : {
                trigger: 'item',
                formatter: "{a} <br/>{b} : {c} ({d}%)"
            },
            legend: {
                orient: 'vertical',
                left: 'left',
                data: ['直接访问','邮件营销','联盟广告','视频广告','搜索引
擎']
            },
            series : [
                {
                    name: '访问来源',
                    type: 'pie',
                    radius : '55%',
                    center: ['50%', '60%'],
                    data:[

                        {value:335, name:'直接访问'},
```

```
                        {value:310, name:'邮件营销'},
                        {value:234, name:'联盟广告'},
                        {value:135, name:'视频广告'},
                        {value:1548, name:'搜索引擎'}
                    ],
                    itemStyle: {
                        emphasis: {
                            shadowBlur: 10,
                            shadowOffsetX: 0,
                            shadowColor: 'rgba(0, 0, 0, 0.5)'
                        }
                    }
                }
            ]
        };
        // 使用刚指定的配置项和数据显示图表。
        myChart.setOption(option);
    </script>

</body>
</html>
```

配置项详解：

1. title 标题组件

title.**text** 主标题文本

title.**subtext** 副标题文本

title.**x** （title.**textAlign**） 整体（包括 text 和 subtext）的水平对齐 可选值：`'auto'`、`'left'`、`'right'`、`'center'`。

2. tooltip 提示框组件

tooltip.**trigger** 触发类型

可选：

- `'item'`

  数据项图形触发，主要在散点图，饼图等无类目轴的图表中使用。

- `'axis'`

坐标轴触发，主要在柱状图，折线图等会使用类目轴的图表中使用。

- `'none'`

  什么都不触发。

tooltip.**formatter** 提示框浮层内容格式器，支持字符串模板和回调函数两种形式

模板变量有 `{a}`, `{b}`，`{c}`，`{d}`，分别表示系列名，数据名，数据值等。 不同图表类型下的 `{a}`，`{b}`，`{c}`，`{d}` 含义不一样。 其中变量`{a}`, `{b}`, `{c}`, `{d}`在不同图表类型下代表数据含义为：

- 折线（区域）图、柱状（条形）图、K线图 : `{a}`（系列名称），`{b}`（类目值），`{c}`（数值）, `{d}`（无）
- 散点图（气泡）图 : `{a}`（系列名称），`{b}`（数据名称），`{c}`（数值数组）, `{d}`（无）
- 地图 : `{a}`（系列名称），`{b}`（区域名称），`{c}`（合并数值）, `{d}`（无）
- 饼图、仪表盘、漏斗图: `{a}`（系列名称），`{b}`（数据项名称），`{c}`（数值）, `{d}`（百分比）

3. legend 图例组件

legend.**orient** 图例列表的布局朝向

可选：

- 'horizontal' 水平布局
- 'vertical' 垂直布局

legend.**left** 图例组件离容器左侧的距离。

`left` 的值可以是像 `20` 这样的具体像素值，可以是像 `'20%'` 这样相对于容器高宽的百分比，也可以是 `'left'`, `'center'`, `'right'`。

如果 `left` 的值为`'left'`, `'center'`, `'right'`，组件会根据相应的位置自动对齐。

legend.**data** 图例的数据数组。

4. series 系列列表

series[i]-pie.**radius** 饼图的半径。可以为如下类型：

- `number`：直接指定外半径值。
- `string`：例如，`'20%'`，表示外半径为可视区尺寸（容器高宽中较小一项）的 20%

长度。

- `Array.`：数组的第一项是内半径，第二项是外半径。每一项遵从上述 `number` `string` 的描述。

series[i]-pie.**center** 饼图的中心（圆心）坐标

数组的第一项是横坐标，第二项是纵坐标。

支持设置成百分比，设置成百分比时第一项是相对于容器宽度，第二项是相对于容器高度。

series[i]-pie.**itemStyle** 图形的样式

series[i]-pie.**emphasis** 高亮的扇区和标签样式。

series[i]-pie.emphasis.itemStyle.**shadowBlur** 图形阴影的模糊大小

series[i]-pie.emphasis.itemStyle.**shadowColor** 阴影颜色

series[i]-pie.emphasis.itemStyle.**shadowOffsetX** 阴影水平方向上的偏移距离

2.2.4 多数据的饼图

```
option = {
            title : {
                text: '商品类目销售分析',
                subtext: '这是个例子',
                x:'center'
            },
            tooltip : {
                trigger: 'item',
                formatter: "{a} <br/>{b} : {c} ({d}%)"
            },
            legend: {
                orient: 'vertical',
                left: 'left',
                data: ['水果','蔬菜','家电','数码','服装']
            },
            series : [
                {
                    name: '销售量',
                    type: 'pie',
                    radius : '35%',
                    center: ['30%', '50%'],
                    data:[
                        {value:335, name:'水果'},
                        {value:310, name:'蔬菜'},
                        {value:234, name:'家电'},
                        {value:135, name:'数码'},
                        {value:1548, name:'服装'}
                    ],
                    itemStyle: {
                        emphasis: {
                            shadowBlur: 10,
                            shadowOffsetX: 0,
                            shadowColor: 'rgba(0, 0, 0, 0.5)'
                        }
                    }
                },
                {
                    name: '销售额',
                    type: 'pie',

                    radius : '35%',
```

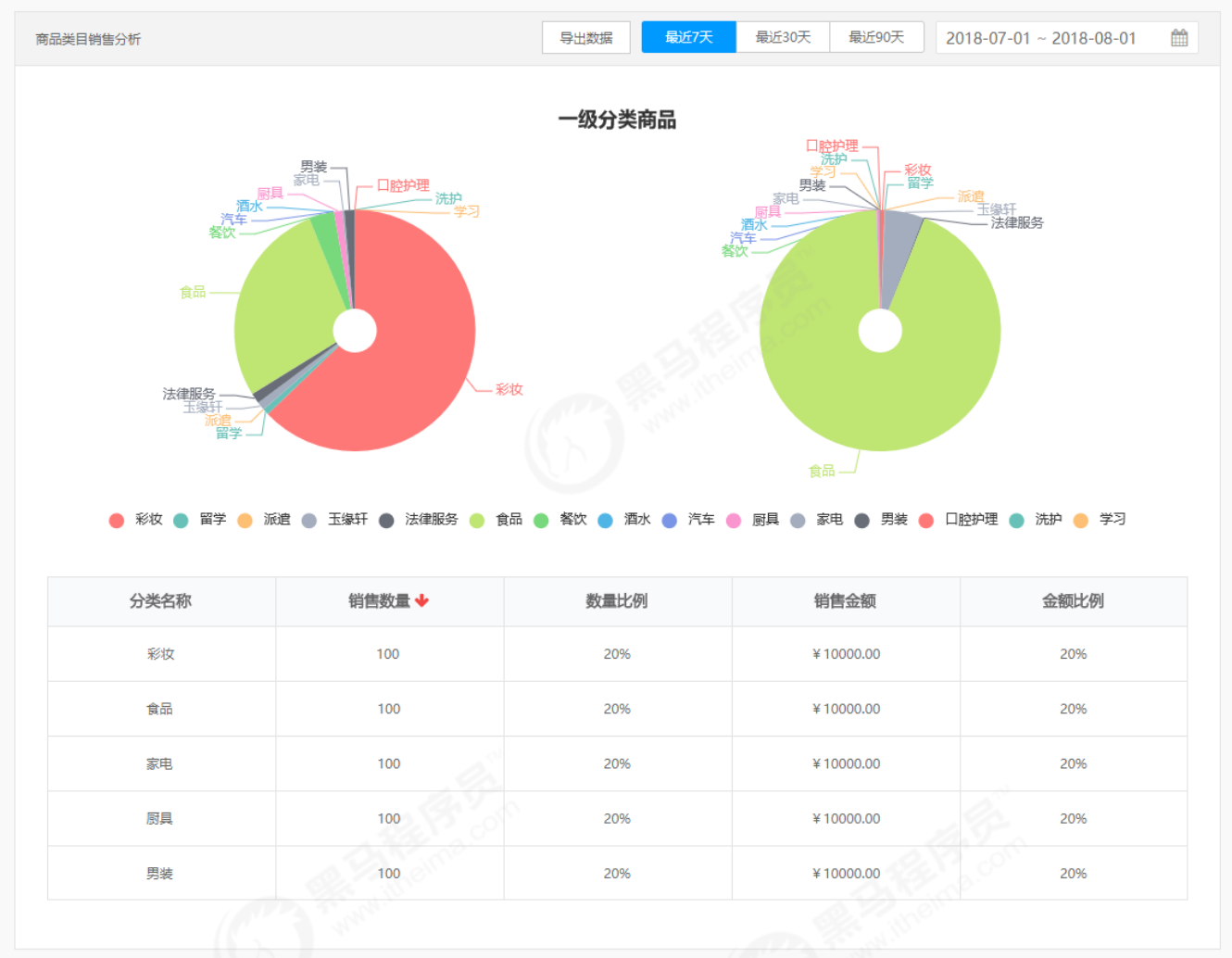
```
                center: ['80%', '50%'],
                data:[
                    {value:2003, name:'水果'},
                    {value:2310, name:'蔬菜'},
                    {value:4434, name:'家电'},
                    {value:1125, name:'数码'},
                    {value:1448, name:'服装'}
                ],
                itemStyle: {
                    emphasis: {
                        shadowBlur: 10,
                        shadowOffsetX: 0,
                        shadowColor: 'rgba(0, 0, 0, 0.5)'
                    }
                }
            }
        ]
    };
```

注意：这两个饼图的圆点坐标不要重复。

3. 商品类目销售统计（饼图）

3.1 需求分析

根据商品类目销售分析表生成饼图，如下图。左侧的饼图为销售量统计分析图，右侧的饼图为销售额统计分析图。



分类名称	销售数量	数量比例	销售金额	金额比例
彩妆	100	20%	¥10000.00	20%
食品	100	20%	¥10000.00	20%
家电	100	20%	¥10000.00	20%
厨具	100	20%	¥10000.00	20%
男装	100	20%	¥10000.00	20%

3.2 实现思路

（1）EChars有自己的数据格式，我们可以将后端返回的商品类目销售分析表的数据进行整理，得到EChars需要的数据格式。

（2）使用EChars的饼图控件。

3.3 代码实现

（1）页面引入js

```
<script src="/js/echarts.common.min.js"></script>
```

（2）在页面放置 div用于显示报表

```
<div id="main" style="width: 600px;height:400px;"></div>
```

（3）修改fetchData方法，在回调处添加以下代码

```
let legendData=[];//图例
let numData=[];//销售量数据
let moneyData=[];//销售额数据
for(let i=0;i<this.tableData.length;i++){
    legendData.push( this.tableData[i].categoryName  );
    numData.push( { name:this.tableData[i].categoryName
,value:this.tableData[i].num  } );
    moneyData.push( { name:this.tableData[i].categoryName
,value:this.tableData[i].money  } );
}

//创建饼图
let myChart = echarts.init(document.getElementById('main'));
let option = {
    title : {
        text: '商品类目销售分析',
        subtext: '',
        x:'center'
    },
    tooltip : {
        trigger: 'item',
        formatter: "{a} <br/>{b} : {c} ({d}%)"
    },
    legend: {
        orient: 'vertical',
        left: 'left',
        data: legendData
    },
    series : [
        {
            name: '销售量',
            type: 'pie',
            radius : '35%',
            center: ['30%', '50%'],
            data:numData,
            itemStyle: {
                emphasis: {
                    shadowBlur: 10,
                    shadowOffsetX: 0,

                    shadowColor: 'rgba(0, 0, 0, 0.5)'
```

```
                }
            }
        },
        {
            name: '销售额',
            type: 'pie',
            radius : '35%',
            center: ['80%', '50%'],
            data:moneyData,
            itemStyle: {
                emphasis: {
                    shadowBlur: 10,
                    shadowOffsetX: 0,
                    shadowColor: 'rgba(0, 0, 0, 0.5)'
                }
            }
        }
    ]
};

// 使用刚指定的配置项和数据显示图表。
myChart.setOption(option);
```

4. 流量统计

4.1 相关术语

4.1.1 IP

IP(独立IP),即Internet Protocol,这里是指独立IP数，独立IP是指不同IP地址的计算机访问网站时被计的总次数。独立IP数是衡量网站流量的一个重要指标。一般一天内(00:00-24:00)相同IP地址的客户端访问网站页面只会被计为一次. 假设公司很多人在局域网中同时打开网站，此时只能算一个独立IP。

4.1.2 UV

UV（独立访客）即Unique Visitor,同一个客户端（PC或移动端）访问网站被计为一个访客。一天(00:00-24:00)内相同的客户端访问同一个网站只计一次UV。UV一般是以客户端Cookie等技术作为统计依据的，实际统计会有误差。

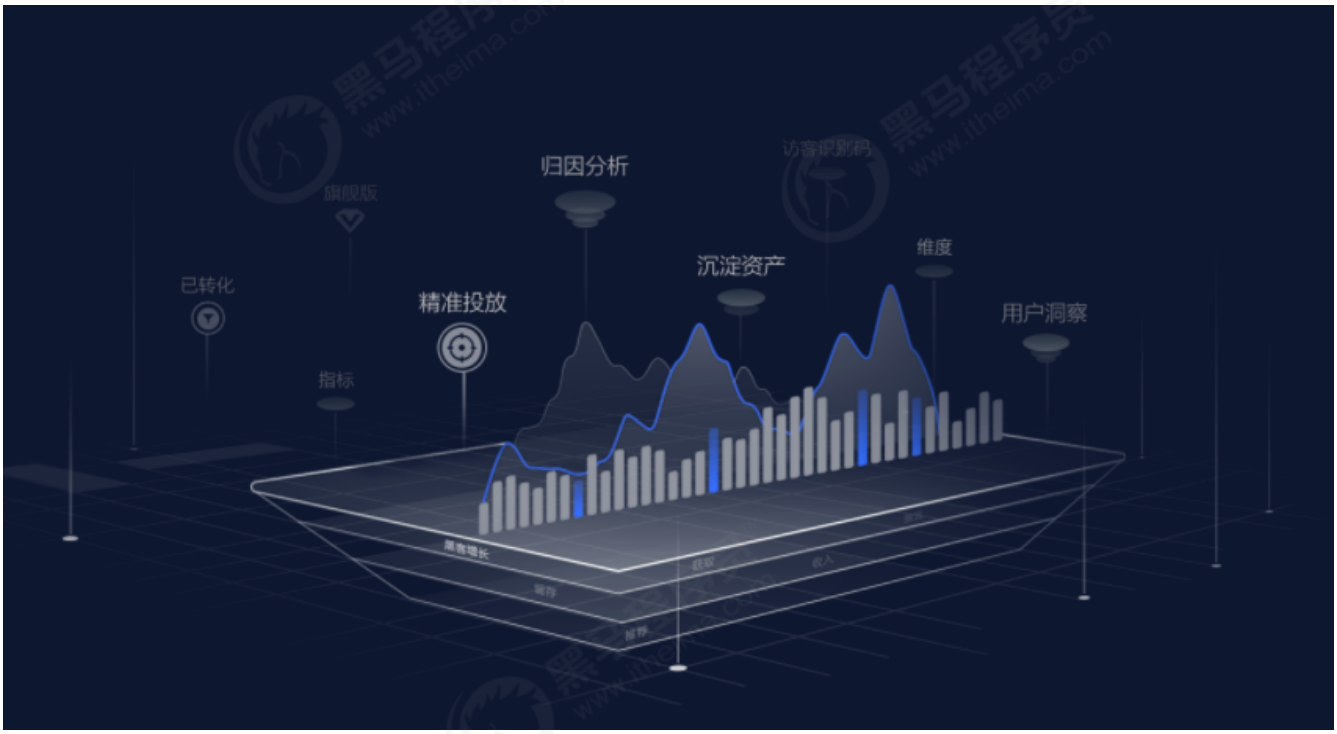
4.1.3 PV

PV(访问量）即Page View ,中文翻译为页面浏览，即页面浏览量或点击量，不管客户端是不是不同，也 不管IP是不是不同，用户只要访问网站页面就会被计算PV，一次计一个PV。

4.2 百度统计（了解）

全球最大的中文网站流量分析平台，帮助企业收集网站访问数据，提供流量趋势、来源分析、转化跟踪、页面热力图、访问流等多种统计分析服务，同时与百度搜索、百度推广、云服务无缝结合，为网站的精细化运营决策提供数据支持，进而有效提高企业的投资回报率。

基于百度强大的技术实力，百度统计提供了丰富的数据指标，系统稳定，功能强大但操作简易。登陆系统后按照系统说明完成代码添加，百度统计便可马上收集数据，为用户提高投资回报率提供决策依据。是提供给广大网站管理员免费使用的网站流量统计系统，帮助用户跟踪网站的真实流量，并优化网站的运营决策。



https://tongji.baidu.com/

5. 交易统计（作业）

5.1 交易统计表

5.1.1 需求分析

实现交易数据统计，以表格形式展现浏览人数 下单人数 订单数 下单件数 有效订单数 下单金额 退款金额 付款人数 付款订单数 付款件数 付款金额

浏览人数	下单人数	订单数	下单件数	有效订单数	下单金额
1888	80	144	643	130	￥1905871.71
退款金额	**付款人数**	**付款订单数**	**付款件数**	**付款金额**	**客单价**
￥1046.01	55	81	381	￥1967849.99	￥24294.44

5.1.2 实现思路

（1）创建交易统计表，用于存储每天的统计结果（表结构学员定义）

（2）编写代码，创建定时任务，在凌晨1点执行统计，将统计结果插入到交易统计表

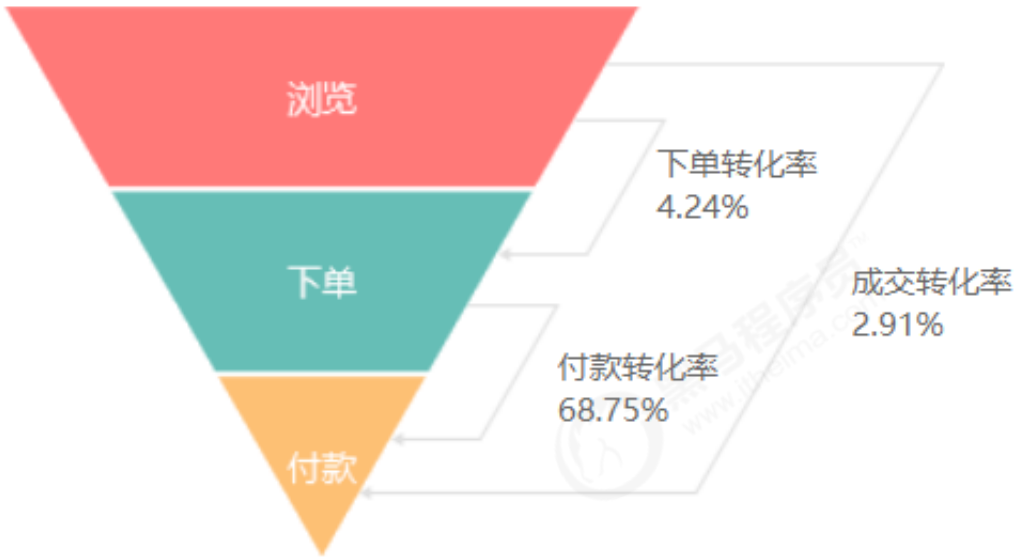
（3）编写代码，实现对交易统计表的聚合运算（按时间段查询）

（4）浏览人数可以通过调用“百度统计”API获取。

5.2 漏斗图

5.2.1 需求分析

根据浏览人数、下单人数和付款人数构建漏斗图。见静态原型。



5.2.2 实现思路

（1）使用EChars的漏斗图控件实现此功能，接受的数据是上一步统计出来的浏览人数、下单人数和付款人数

（2）下单转化率= 下单人数/浏览人数

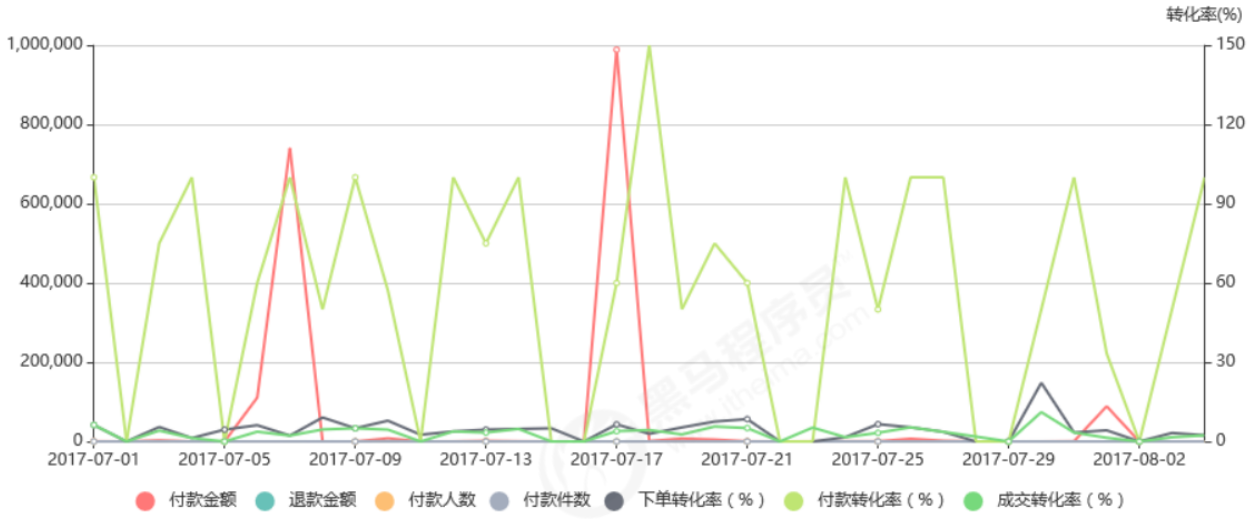
（3）付款转化率=付款人数/下单人数

（4）成交转化率=付款人数/浏览人数

5.3 折线图

5.3.1 需求分析

根据某一时间段的交易数据，展现折线图。交易数据包括付款金额、退款金额、付款人数、付款件数、下单转化率、付款转化率、成交转化率



5.3.2 实现思路

使用EChars的折线图实现此功能。