第5章 数据分析

学习目标

- 掌握流量分析，能够根据不同日期和时间维度分析网站用户活跃度。
- 掌握商品分析，能够分析网站中的畅销品和滞销品。
- 掌握设备分析，能够分析用户在不同时间段的设备偏好。
- 掌握推荐系统，能够基于协同过滤为用户推荐可能感兴趣的商品。
- 掌握地域分析，能够实时统计每个城市的销售情况。

数据分析是一种从数据中获取有价值信息的方法，它可以帮助我们理解数据背后的规律。随着信息技术的飞速进步，数据的规模和复杂度不断增加，大数据时代已经到来。在这个时代，数据分析在各个领域都发挥着重要的作用。无论是商业决策，还是科学研究，数据分析都提供了新的视角和解决方案。本章详细介绍如何使用 Spark 进行用户行为数据分析。

5.1 流量分析

流量分析是一种利用网站访问数据来评估和优化网站运营效果的方法，它对于提升网站的设计、内容和营销策略有着重要的作用。流量分析包含多种指标，如页面浏览量(PV)、跳出率、访问时长等，它们可以从不同的角度反映用户在网站上的行为和偏好。本项目重点分析流量分析中的页面浏览量指标。

页面浏览量是指用户在网站上访问页面的总次数，每当用户打开一个网站页面，页面浏览量就会增加一次，即使用户对同一页面进行了多次访问，也不会影响页面浏览量的计算。页面浏览量作为网站运营的重要参考数据，能够直观地反映出网站用户的活跃程度和网站内容的吸引力。

本项目将从不同的日期和时间维度对页面浏览量进行分析，包括月度、季度等，以全面了解网站用户活跃度的变化趋势。接下来演示如何利用 Spark，对 2023 年的历史用户行为数据进行页面浏览量分析，具体操作步骤如下。

1. 创建 Python 文件

在项目 spark_project 中创建名为 analyze 的目录，在该目录中创建名为 visit_counts 的 Python 文件。

2. 实现 Spark 程序

在 visit_counts.py 文件中实现页面浏览量分析的 Spark 程序,具体实现过程如下。

(1) 导入用于创建和配置 Spark 程序的类 SparkSession,以及页面浏览量分析所用到的函数 col、lit、count、from_unixtime 和 unix_timestamp。在 visit_counts.py 文件中添加如下代码。

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, lit, count, \
    from_unixtime, unix_timestamp
```

(2) 创建 SparkSession 对象的同时启用 Hive 支持,以便 Spark 程序与 Hive 进行交互。在 visit_counts.py 文件中添加如下代码。

```
spark = SparkSession.builder \
    .appName("visit_count") \
    .config("hive.metastore.uris", "thrift://spark01:9083") \
    .enableHiveSupport() \
    .getOrCreate()
```

(3) 获取表 user_behavior_detail、dim_date 和 dim_time 的数据。在 visit_counts.py 文件中添加如下代码。

```
user_behavior_detail = spark.read \
    .table("user_behavior_db.user_behavior_detail")
dim_date = spark.read.table("user_behavior_db.dim_date")
dim_time = spark.read.table("user_behavior_db.dim_time")
```

上述代码中,第 1、2 行代码生成了一个 DataFrame 对象 user_behavior_detail,该对象包含了表 user_behavior_detail 的数据。第 3 行代码生成了一个 DataFrame 对象 dim_date,该对象包含了表 dim_date 的数据。第 4 行代码生成了一个 DataFrame 对象 dim_time,该对象包含了表 dim_time 的数据。

(4) 分析每个月的页面浏览量。在 visit_counts.py 文件中添加如下代码。

```
month_info = user_behavior_detail.groupBy("monthinfo") \
    .agg(count("*").alias("visit_count")) \
    .select(
        col("monthinfo").alias("month_info"),
        lit("-1").alias("day_info"),
        lit("-1").alias("quarter_info"),
        lit("-1").alias("am_pm_info"),
        lit("-1").alias("week_info"),
        lit("0").alias("group_type"),
        "visit_count"
    )
```

上述代码，首先，使用 groupBy 算子按照字段 monthinfo 对表 user_behavior_detail 中的数据进行分组。然后，使用 agg 算子对每组数据进行聚合操作，统计每组数据的行数（即页面访问量），并将结果保存到字段 visit_count 中。最后，使用 select 算子选择并创建所需的字段，生成 DataFrame 对象 month_info。

关于 select 算子选择并创建字段的介绍如下。

① 选择字段 monthinfo 并重命名为 month_info，用于获取月份的值。

② 创建值为－1 的常量字段 day_info，用于表示分析结果与日期无关。

③ 创建值为－1 的常量字段 quarter_info，用于表示分析结果与季度无关。

④ 创建值为－1 的常量字段 am_pm_info，用于表示分析结果与上、下午无关。

⑤ 创建值为－1 的常量字段 week_info，用于表示分析结果与星期无关。

⑥ 创建值为 0 的常量字段 group_type，用于表示分析结果与月份有关。

⑦ 选择字段 visit_count，用于获取分析结果。

（5）分析每天的页面浏览量。在 visit_counts.py 文件中添加如下代码。

```
day_info = user_behavior_detail.groupBy("action_date_key") \
    .agg(count("*").alias("visit_count")) \
    .select(
        lit("-1").alias("month_info"),
        from_unixtime(
            unix_timestamp("action_date_key", "yyyyMMdd"),
            "yyyy-MM-dd")
        .alias("day_info"),
        lit("-1").alias("quarter_info"),
        lit("-1").alias("am_pm_info"),
        lit("-1").alias("week_info"),
        lit("1").alias("group_type"),
        "visit_count"
    )
```

上述代码，首先，使用 groupBy 算子按照字段 action_date_key 对表 user_behavior_detail 中的数据进行分组。然后，使用 agg 算子对每组数据进行聚合操作，统计每组数据的行数（即页面访问量），并将结果保存到字段 visit_count 中。最后，使用 select 算子选择并创建所需的字段，生成 DataFrame 对象 day_info。

关于 select 算子选择并创建字段的介绍如下。

① 创建值为－1 的常量字段 month_info，用于表示分析结果与月份无关。

② 选择字段 action_date_key，将字段的值格式化为 yyyy-MM-dd 格式的日期字符串之后，保存到字段 day_info，用于获取日期的值。

③ 创建值为－1 的常量字段 quarter_info，用于表示分析结果与季度无关。

④ 创建值为－1 的常量字段 am_pm_info，用于表示分析结果与上、下午无关。

⑤ 创建值为－1 的常量字段 week_info，用于表示分析结果与星期无关。

⑥ 创建值为 1 的常量字段 group_type,用于表示分析结果与日期有关。

⑦ 选择字段 visit_count,用于获取分析结果。

(6) 分析每个季度的页面浏览量。在 visit_counts.py 文件中添加如下代码。

```
dim_date_join = user_behavior_detail.join(
    dim_date,
    user_behavior_detail.action_date_key == dim_date.date_key
).cache()
quarter_info = dim_date_join.groupBy("quarter") \
    .agg(count("*").alias("visit_count")) \
    .select(
        lit("-1").alias("month_info"),
        lit("-1").alias("day_info"),
        col("quarter").alias("quarter_info"),
        lit("-1").alias("am_pm_info"),
        lit("-1").alias("week_info"),
        lit("2").alias("group_type"),
        "visit_count"
    )
```

上述代码中,第 1～4 行代码使用 join 算子对 DataFrame 对象 user_behavior_detail 和 dim_date 进行内连接,指定关联条件为 user_behavior_detail 中字段 action_date_key 的值等于 dim_date 中字段 date_key 的值。将内连接的结果保存到 DataFrame 对象 dim_date_join 并进行缓存,以便后续重复使用。

第 5～15 行代码,首先,使用 groupBy 算子按照字段 quarter 对 dim_date_join 中的数据进行分组。然后,使用 agg 算子对每组数据进行聚合操作,统计每组数据的行数(即页面访问量),并将结果保存到字段 visit_count 中。最后,使用 select 算子选择并创建所需的字段,生成 DataFrame 对象 quarter_info。

关于 select 算子选择并创建字段的介绍如下。

① 创建值为－1 的常量字段 month_info,用于表示分析结果与月份无关。

② 创建值为－1 的常量字段 day_info,用于表示分析结果与日期无关。

③ 选择字段 quarter 并重命名为 quarter_info,用于获取季度的值。

④ 创建值为－1 的常量字段 am_pm_info,用于表示分析结果与上、下午无关。

⑤ 创建值为－1 的常量字段 week_info,用于表示分析结果与星期无关。

⑥ 创建值为 2 的常量字段 group_type,用于表示分析结果与季度有关。

⑦ 选择字段 visit_count,用于获取分析结果。

(7) 分析不同星期的页面浏览量。在 visit_counts.py 文件中添加如下代码。

```
week_info = dim_date_join.groupBy("day_of_week") \
    .agg(count("*").alias("visit_count")) \
    .select(
```

```
        lit("-1").alias("month_info"),
        lit("-1").alias("day_info"),
        lit("-1").alias("quarter_info"),
        lit("-1").alias("am_pm_info"),
        col("day_of_week").alias("week_info"),
        lit("4").alias("group_type"),
        "visit_count"
    )
```

上述代码，首先，使用 groupBy 算子按照字段 day_of_week 对 dim_date_join 中的数据进行分组。然后，使用 agg 算子对每组数据进行聚合操作，统计每组数据的行数（即页面访问量），并将结果保存到字段 visit_count 中。最后，使用 select 算子选择并创建所需的字段，生成 DataFrame 对象 week_info。

关于 select 算子选择并创建字段的介绍如下。

① 创建值为－1 的常量字段 month_info，用于表示分析结果与月份无关。

② 创建值为－1 的常量字段 day_info，用于表示分析结果与日期无关。

③ 创建值为－1 的常量字段 quarter_info，用于表示分析结果与季度无关。

④ 创建值为－1 的常量字段 am_pm_info，用于表示分析结果与上、下午无关。

⑤ 选择字段 day_of_week 并重命名为 week_info，用于获取星期的值。

⑥ 创建值为 4 的常量字段 group_type，用于表示分析结果与星期有关。

⑦ 选择字段 visit_count，用于获取分析结果。

（8）分析上午和下午的页面浏览量。在 visit_counts.py 文件中添加如下代码。

```
dim_time_join = user_behavior_detail.join(
    dim_time,
    user_behavior_detail.action_time_key == dim_time.time_key
)
am_pm_info = dim_time_join.groupBy("am_pm") \
    .agg(count("*").alias("visit_count")) \
    .select(
        lit("-1").alias("month_info"),
        lit("-1").alias("day_info"),
        lit("-1").alias("quarter_info"),
        col("am_pm").alias("am_pm_info"),
        lit("-1").alias("week_info"),
        lit("3").alias("group_type"),
        "visit_count"
    )
```

上述代码中，第 1～4 行代码使用 join 算子对 DataFrame 对象 user_behavior_detail 和 dim_time 进行内连接，指定关联条件为 user_behavior_detail 中字段 action_time_key 的值等于 dim_time 中字段 time_key 的值。将内连接的结果保存到 DataFrame 对象

dim_time_join。

第 5～15 行代码,首先,使用 groupBy 算子按照字段 am_pm 对 dim_time_join 中的数据进行分组。然后,使用 agg 算子对每组数据进行聚合操作,统计每组数据的行数(即页面访问量),并将结果保存到字段 visit_count 中。最后,使用 select 算子选择并创建所需的字段,生成 DataFrame 对象 am_pm_info。

关于 select 算子选择并创建字段的介绍如下。

① 创建值为－1 的常量字段 month_info,用于表示分析结果与月份无关。

② 创建值为－1 的常量字段 day_info,用于表示分析结果与日期无关。

③ 创建值为－1 的常量字段 quarter_info,用于表示分析结果与季度无关。

④ 选择字段 am_pm 并重命名为 am_pm_info,用于获取上午或下午的值。

⑤ 创建值为－1 的常量字段 week_info,用于表示分析结果与星期无关。

⑥ 创建值为 3 的常量字段 group_type,用于表示分析结果与上午和下午有关。

⑦ 选择字段 visit_count,用于获取分析结果。

(9) 使用 union 算子合并 DataFrame 对象 month_info、day_info、quarter_info、am_pm_info 和 week_info。在 visit_counts.py 文件中添加如下代码。

```
result = month_info.union(day_info).union(quarter_info) \
         .union(am_pm_info).union(week_info)
```

上述代码生成了 DataFrame 对象 result,该对象中包含了不同日期和时间维度的页面访问量分析结果。

(10) 将 DataFrame 对象 result 中的数据以覆盖模式写入表 ads_visit_counts_2023。在 visit_counts.py 文件中添加如下代码。

```
#指定表 ads_visit_counts_2023 在 HDFS 存储数据的目录
table_location = '/user_behavior/ads/ads_visit_counts_2023'
result.write.format("hive").mode("overwrite") \
.option('path', table_location) \
.saveAsTable("user_behavior_db.ads_visit_counts_2023")
```

3. 运行 Spark 程序

为了在 YARN 集群上运行 Spark 程序,需要将 visit_counts.py 文件上传到虚拟机 Spark02 的/export/servers 目录中。确保 MetaStore 服务、HDFS 集群和 YARN 集群处于启动状态下,将 visit_counts.py 文件中实现的 Spark 程序提交到 YARN 集群运行。在虚拟机 Spark02 执行如下命令。

```
spark-submit \
--master yarn \
--deploy-mode cluster \
/export/servers/visit_counts.py
```

上述命令执行完成后可以通过访问 YARN Web UI 查看 Spark 程序的运行状态，若其状态为 FINISHED 并且最终状态为 SUCCEEDED 表示运行成功。

需要说明的是，为了优化资源利用，在运行流量分析的 Spark 程序时，可以选择仅启动集群环境中的 HDFS 集群、YARN 集群和 MetaStore 服务。

4. 查看表 ads_visit_counts_2023 的数据

由于表 ads_visit_counts_2023 存储了不同日期和时间维度的流量分析结果，为避免全表查询产生大量查询结果，将根据月份维度查询流量分析的结果。在 Hive 命令行界面执行如下命令。

```
hive> SELECT * FROM user_behavior_db.ads_visit_counts_2023
    > WHERE group_type = 0;
```

上述命令执行完成的效果如图 5-1 所示。

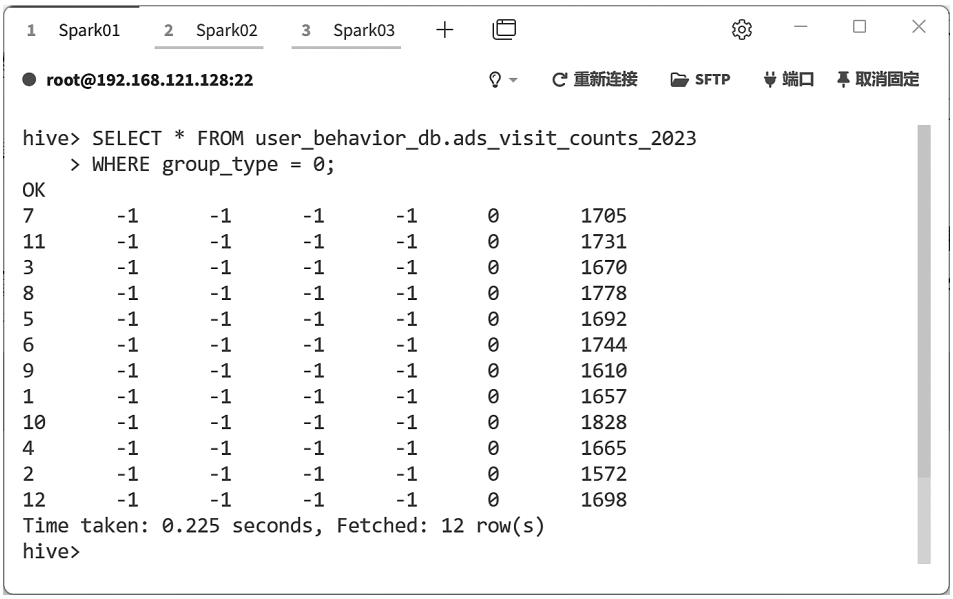

图 5-1　查看表 ads_visit_counts_2023 中的数据

图 5-1 展示了表 ads_visit_counts_2023 中关于月份维度的流量分析结果，这些数据反映了每个月用户访问网站的总次数。例如，2023 年 1 月份用户访问网站的总次数为 1657。需要说明的是，实际分析结果与读者采集的历史用户行为数据有关。

5.2　商品分析

商品分析是一种利用商品数据来了解商品销售情况、市场趋势和用户行为的方法。它对于优化商品、预测市场趋势和制定营销策略至关重要。例如，通过分析各商品的销量，可以识别畅销品和滞销品，为商品优化提供依据；通过分析历史销售数据，可以预测未来的销售趋势，为商品的进货、陈列、促销等决策提供依据。

本项目分析各商品的销量,将销量最高的 10 件商品定义为畅销品,销量最低的 10 件商品定义为滞销品,以全面了解网站中的畅销品和滞销品。接下来讲解如何利用 Spark 对 2023 年的历史用户行为数据进行商品分析,具体操作步骤如下。

1. 创建 Python 文件

在项目 spark_project 的 analyze 目录中创建名为 product_sale_counts 的 Python 文件。

2. 实现 Spark 程序

在 product_sale_counts.py 文件中实现分析各商品销量的 Spark 程序,具体实现过程如下。

(1) 导入用于创建和配置 Spark 程序的类 SparkSession。在 product_sale_counts.py 文件中添加如下代码。

```
from pyspark.sql import SparkSession
```

(2) 创建 SparkSession 对象的同时启用 Hive 支持,以便 Spark 程序与 Hive 进行交互。在 product_sale_counts.py 文件中添加如下代码。

```
spark = SparkSession.builder \
    .appName("sale_count") \
    .config("hive.metastore.uris", "thrift://spark01:9083") \
    .enableHiveSupport() \
    .getOrCreate()
```

(3) 从历史用户行为数据中获取被购买过的商品。在 product_sale_counts.py 文件中添加如下代码。

```
product_sales = spark.sql("""
    SELECT product_id
    FROM user_behavior_db.user_behavior_detail
    WHERE behavior_type = 'purchase'
    """)
```

上述代码通过执行 SQL 语句,从表 user_behavior_detail 中获取字段 behavior_type 值为 purchase 的数据,并从这些数据中提取字段 product_id 的值,将其存放在 DataFrame 对象 product_sales 中。

(4) 为了后续使用 SQL 语句查询 DataFrame 对象 product_sales 中的数据,这里将其注册为临时表 product_sales。在 product_sale_counts.py 文件中添加如下代码。

```
product_sales.createOrReplaceTempView("product_sales")
```

(5) 统计每件商品被购买的次数。在 product_sale_counts.py 文件中添加如下代码。

```
product_sale_counts = spark.sql("""
    SELECT product_id, COUNT(*) as sale_count
    FROM product_sales
    GROUP BY product_id
""")
```

上述代码通过执行 SQL 语句，对临时表 product_sales 中的数据按照字段 product_id 进行分组，并对每组数据进行聚合操作，统计每组数据的行数（即每个商品被购买的次数），将结果保存到字段 sale_count 中，最终生成一个 DataFrame 对象 product_sale_counts。

（6）为了后续使用 SQL 语句查询 DataFrame 对象 product_sale_counts 中的数据，这里将其注册为临时表 product_sale_counts。在 product_sale_counts.py 文件中添加如下代码。

```
product_sale_counts.createOrReplaceTempView("product_sale_counts")
```

（7）获取销量排名前 10 的商品。在 product_sale_counts.py 文件中添加如下代码。

```
top_10_products = spark.sql("""
    SELECT product_id,'0' AS sale_type,sale_count
    FROM product_sale_counts
    ORDER BY sale_count DESC
    LIMIT 10
""")
```

上述代码通过执行 SQL 语句，对临时表 product_sale_counts 中的数据按照字段 sale_count 进行降序排序，并获取排序结果的前 10 条数据，即销量最高的 10 件商品。最终生成一个 DataFrame 对象 top_10_products。

（8）获取销量排名后 10 的商品。在 product_sale_counts.py 文件中添加如下代码。

```
bottom_10_products = spark.sql("""
    SELECT product_id,'1' AS sale_type,sale_count
    FROM product_sale_counts
    ORDER BY sale_count ASC
    LIMIT 10
""")
```

上述代码通过执行 SQL 语句，对临时表 product_sale_counts 中的数据按照字段 sale_count 进行升序排序，并获取排序结果的前 10 条数据，即销量最低的 10 件商品。最终生成一个 DataFrame 对象 bottom_10_products。

（9）使用 union 算子合并 DataFrame 对象 top_10_products 和 bottom_10_products。在 visit_counts.py 文件中添加如下代码。

```
result = top_10_products.union(bottom_10_products))
```

上述代码生成了 DataFrame 对象 result,该对象中包含了网站中畅销品和滞销品的分析结果。

(10) 将 DataFrame 对象 result 中的数据以覆盖模式写入表 ads_sale_counts_2023。在 visit_counts.py 文件中添加如下代码。

```
#指定表 ads_sale_counts_2023 在 HDFS 存储数据的目录
table_location = '/user_behavior/ads/ads_sale_counts_2023'
result.write.format("hive").mode("overwrite") \
.option('path', table_location) \
.saveAsTable("user_behavior_db.ads_sale_counts_2023")
```

3. 运行 Spark 程序

为了在 YARN 集群上运行 Spark 程序,需要将 product_sale_counts.py 文件上传到虚拟机 Spark02 的/export/servers 目录中。确保 MetaStore 服务、HDFS 集群和 YARN 集群处于启动状态下,将 product_sale_counts.py 文件中实现的 Spark 程序提交到 YARN 集群运行。在虚拟机 Spark02 执行如下命令。

```
spark-submit \
--master yarn \
--deploy-mode cluster \
/export/servers/product_sale_counts.py
```

上述命令执行完成后可以通过访问 YARN Web UI 查看 Spark 程序的运行状态,若其状态为 FINISHED 并且最终状态为 SUCCEEDED 表示运行成功。

需要说明的是,为了优化资源利用,在运行商品分析的 Spark 程序时,可以选择仅启动集群环境中的 HDFS 集群、YARN 集群和 MetaStore 服务。

4. 查看表 ads_sale_counts_2023 的数据

通过查询表 ads_sale_counts_2023 的全部数据,获取电商网站中畅销品和滞销品的信息。在 Hive 命令行界面执行如下命令。

```
hive> SELECT * FROM user_behavior_db.ads_sale_counts_2023;
```

上述命令执行完成的效果如图 5-2 所示。

在图 5-2 中,表 ads_sale_counts_2023 中的每行数据依次记录了商品的唯一标识、销售类型的标识(0 表示畅销品,1 表示滞销品)和商品的销量。例如,第一行数据显示唯一标识为 269 的商品为畅销品,销量为 37。需要说明的是,实际分析结果与读者采集的历史用户行为数据有关。