习题答案

第一章

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
B	C	A	D	B	B	A	B	A	D	B	C	D	D	A

第二章

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
C	B	B	D	A	D	B	A	B	A	D	A	C	C	A

第三章

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
A	C	C	A	C	A	A	C	A	A	A	D	A	D	A

第四章

实验案例答案

```
    object SparkSQL_Movie {
        def main(args: Array[String]): Unit = {
            val spark = SparkSession.builder()
              .appName(this.getClass.getSimpleName.stripSuffix("$"))
              .master("local[*]")
              // TODO: 设置 shuffle 时分区数目
              .config("spark.sql.shuffle.partitions", "4")
              .getOrCreate()
            val sc: SparkContext = spark.sparkContext
            sc.setLogLevel("WARN")
            import spark.implicits._

            // 1. 读取电影评分数据，从本地文件系统读取
            val rawRatingsDS: Dataset[String] =
spark.read.textFile("data/input/rating_100k.data")
```

```
        // 2. 转换数据
        val ratingsDF: DataFrame = rawRatingsDS
            // 过滤数据
            .filter(line => null != line &&
line.trim.split("\t").length == 4)
            // 提取转换数据
            .mapPartitions{iter =>
                iter.map{line =>
                    // 按照分割符分割，拆箱到变量中
                    val Array(userId, movieId, rating, timestamp) =
line.trim.split("\t")
                    // 返回四元组
                    (userId, movieId, rating.toDouble,
timestamp.toLong)
                }
            }
            // 指定列名添加 Schema
            .toDF("userId", "movieId", "rating", "timestamp")
               ratingsDF.printSchema()
               ratingsDF.show(4)

        // TODO： 基于 SQL 方式分析
        // 第一步、注册 DataFrame 为临时视图
        ratingsDF.createOrReplaceTempView("view_temp_ratings")

        // 第二步、编写 SQL
        val top10MovieDF: DataFrame = spark.sql(
            """
              |SELECT
```

```
                |  movieId, ROUND(AVG(rating), 2) AS avg_rating,
COUNT(movieId) AS cnt_rating
                |FROM
                |  view_temp_ratings
                |GROUP BY
                |  movieId
                |HAVING
                |  cnt_rating > 200
                |ORDER BY
                |  avg_rating DESC, cnt_rating DESC
                |LIMIT
                |  10
            """.stripMargin)
        //top10MovieDF.printSchema()
        top10MovieDF.show(10, truncate = false)


println("============================================================
===")

        // TODO: 基于 DSL=Domain Special Language（特定领域语言） 分
析
        import org.apache.spark.sql.functions._
        val resultDF: DataFrame = ratingsDF
            // 选取字段
            .select($"movieId", $"rating")
            // 分组：按照电影 ID，获取平均评分和评分次数
            .groupBy($"movieId")
            .agg(
                round(avg($"rating"), 2).as("avg_rating"),
                count($"movieId").as("cnt_rating")
```

```
            )
            // 过滤：评分次数大于 200
            .filter($"cnt_rating" > 200)
            // 排序：先按照评分降序，再按照次数降序
            .orderBy($"avg_rating".desc, $"cnt_rating".desc)
            // 获取前 10
            .limit(10)
        //resultDF.printSchema()
        resultDF.show(10)

        // TODO: 将分析的结果数据保存 MySQL 数据库

        // 保存 MySQL 数据库表汇总
        resultDF
            .coalesce(1)
            .write
            .mode("overwrite")
            .option("driver", "com.mysql.jdbc.Driver")
            .option("user", "root")
            .option("password", "root")
            .jdbc(

"jdbc:mysql://localhost:3306/hongya?characterEncoding=UTF-8",
                "movies",
                new Properties()
            )


        spark.stop()
    }
}
```

习题答案

1-4：B C B B

5. schema

6. show(N)

7. Column

8. import org.apache.spark.sql.SparkSession

```
val spark=SparkSession.builder().getOrCreate()
//使支持 RDDs 转换为 DataFrames 及后续 sql 操作
import spark.implicits._
val df =
spark.read.json("file:///usr/local/spark/examples/src/main/resources/people.json")
df.show()
```

9. df.filter(df("age") > 20).show()

10. df.sort(df("age").desc).show()

第五章

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
B	B	B	B	C	A	B	C	D	D	A	B	A	A	B

第六章

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
B	C	A	D	B	C	A	B	D	C	B	C	A	D	C

第七章

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
C	C	A	A	A	B	B	C	A	B	C	C	B	B	C