Task one

I use spark to process data leave the first part of url (In the example only show the top 20, whole data also store in this folder)

```
processed_data.show()
```

```
+---------+--------+--------------------+
|queryTime|  userId|             clickUrl|
+---------+--------+--------------------+
|  0:00:00|2.98E+15|  download.it.com.cn|
|  0:00:00|7.59E+15|       news.21cn.com|
|  0:00:00|5.23E+15|     www.greatoo.com|
|  0:00:00|6.14E+15|       www.jd-cd.com|
|  0:00:00|8.56E+15|       www.big38.net|
|  0:00:00|2.39E+16|  www.chinabaike.com|
|  0:00:00|1.80E+15|        www.6wei.net|
|  0:00:00|7.18E+14|    www.shanziba.com|
|  0:00:00|4.14E+16|       bbs.gouzai.cn|
|  0:00:00|9.98E+15|     ks.cn.yahoo.com|
|  0:00:00|2.16E+16|  www.fotolog.com.cn|
|  0:00:00|7.42E+15|     ks.cn.yahoo.com|
|  0:00:00|6.17E+14|    topic.bindou.com|
|  0:00:00|3.93E+15|     ks.cn.yahoo.com|
|  0:00:00|8.24E+15|shwamlys.blog.soh...|
|  0:00:00|8.25E+15|  download.it168.com|
|  0:00:00|6.24E+15|   www.songtaste.com|
|  0:00:00|6.55E+15|   product.it168.com|
|  0:00:00|2.35E+15|    pic.news.mop.com|
|  0:00:00|6.48E+15|       www.1000dy.cn|
+---------+--------+--------------------+
only showing top 20 rows
```

Task Two

```
+ 代码   + Markdown
```

```
def rank_tokens(data):
    tokenized_data = data.withColumn("tokens", explode(split(data.clickUrl, "\\.")))  # 使用转义符对`.`进行匹配
    token_counts = tokenized_data.groupBy("tokens").count().orderBy("count", ascending=False)

    top_ten_tokens = token_counts.limit(10)

    formatted_output = " ".join([f"({row.tokens}, {row['count']})" for row in top_ten_tokens.collect()])

    print(formatted_output)
```

```
rank_tokens(processed_data)
```

```
(com, 7935) (www, 4184) (cn, 2362) (baidu, 779) (news, 641) (net, 603) (zhidao, 530) (sina, 501) (bbs, 496) (sohu, 416)
```

Task Three

```python
def rank_time_periods(data):
    # Extract the minute portion from the queryTime column
    data = data.withColumn("minute", substring(data.queryTime, 4, 5))

    # Group the data by minute and count the number of queries for each minute
    query_counts = data.groupBy("minute").count()

    # Sort the results in descending order of the query count
    query_counts = query_counts.orderBy(query_counts["count"].desc())

    # Select the top ten time periods with the highest query count
    top_ten_periods = query_counts.limit(10)

    # Format the output as a string
    formatted_output = " ".join([f"({row.minute}, {row['count']})" for row in top_ten_periods.collect()])

    # Display the result
    print(formatted_output)
```

```python
# Execute the time period ranking task
rank_time_periods(processed_data)
```

```
(1:00, 31) (0:00, 29) (3:11, 28) (1:57, 28) (1:38, 28) (3:53, 28) (3:23, 27) (6:21, 27) (2:23, 27) (5:13, 27)
```