

Report

Q1

(1)

经过检查，这里的数据是json格式，故采用json的写入方法。使用spark的to_timestamp方法修改数据类型。创建tempView后结果如下。

host	id.orig_h	id.orig_p	id.resp_h	id.resp_p	method	orig_filenames	orig_fuids	orig_mime_types	origin	proxied	referrer	request_body_len	resp_filenames	resp_fuids	resp_mime_type
192.168.202.78	192.168.204.70	46197	192.168.202.78	80	POST	NULL	[FPp8eT1d8gDc1zdRT7]	[text/plain]	NULL	NULL	http://192.168.20...	4490	NULL	NULL	NULL
192.168.205.253	192.168.202.107	1191	192.168.205.253	80	GET	NULL	NULL	NULL	NULL	NULL	NULL	0	NULL	[FleXV6ZoEw12BLo1]	[text/html]
127.0.0.1:80	192.168.202.102	1112	192.168.22.152	80	GET	NULL	NULL	NULL	NULL	NULL	NULL	0	NULL	[FV0d812QjtpIKvX1]	[text/html]
192.168.202.78	192.168.204.70	46198	192.168.202.78	80	GET	NULL	NULL	NULL	NULL	NULL	http://192.168.20...	0	NULL	[FImEg12hnmj8PAou1]	[text/html]
192.168.202.78	192.168.204.70	46200	192.168.202.78	80	GET	NULL	NULL	NULL	NULL	NULL	http://192.168.20...	0	NULL	NULL	NULL

only showing top 5 rows

AA	RA	RD	TC	TTLs	Z	answers	id.orig_h	id.orig_p	id.resp_h	id.resp_p	proto	qclass	qclass_name	qtype	qtype_name	query	rcode	rcode_name	rejected	rtt	trans_id	ts	uid
false	false	true	false	NULL	0	NULL	192.168.204.70	38795	192.168.207.4	53	udp	1	C_INTERNET	28	AAAA	creativecommons.org	3	NXDOMAIN	false	NULL	58106	2012-03-16 15:12:...	C1odva1G0162Wbq1
false	false	true	false	NULL	0	NULL	192.168.204.70	53918	192.168.207.4	53	udp	1	C_INTERNET	28	AAAA	www.dokwiki.org	3	NXDOMAIN	false	NULL	52251	2012-03-16 15:12:...	CQlw22e281Dp1c1k
false	false	true	false	NULL	0	NULL	192.168.204.70	45058	192.168.207.4	53	udp	1	C_INTERNET	28	AAAA	creativecommons.org	3	NXDOMAIN	false	NULL	61533	2012-03-16 15:12:...	CKalaf1pe1U03kntV8
false	false	true	false	NULL	0	NULL	192.168.204.70	56979	192.168.207.4	53	udp	1	C_INTERNET	28	AAAA	www.php.net	3	NXDOMAIN	false	NULL	37509	2012-03-16 15:12:...	CCEYy1YQ5ocDA5oV6
false	false	true	false	NULL	0	NULL	192.168.204.70	35954	192.168.207.4	53	udp	1	C_INTERNET	28	AAAA	www.dokwiki.org	3	NXDOMAIN	false	NULL	17572	2012-03-16 15:12:...	Cur1VC14uo1z7dPc

only showing top 5 rows

(2)

根据要求完成了相应的sql语句，结果如下。

SQL

```
SELECT uri, COUNT(*) AS access_count
FROM http_log
WHERE status_code = '200' AND method = 'GET'
GROUP BY uri
ORDER BY access_count DESC
```

[Stage 4:=====>		(1 + 1) / 2]
uri	access_count	
/	9475	
/admin/config.php...	556	
/main.php?logout=1	194	
/top.php?stuff=15...	191	
/top.php	179	
/main.php?stuff=1...	172	
/get_latest_id.php	159	
/admin/config.php...	138	
/cacti/index.php	129	
/en-US/api/messag...	118	
/index.php	105	

(3)

根据要求，先完成了join的sql语句，发现找不到uid相同的匹配结果，出现的两行uid是空行。但sql语句是没问题的。检查后，确认确实是没有相同的uid。

```
SELECT
  *
FROM http_log AS h
JOIN dns_log AS d
  ON h.uid = d.uid
```

```
host|id.orig_h|id.orig_p|id.resp_h|id.resp_p|method|orig_filenames|orig_fuids|orig_mime_types|origin|proxied|referrer|request_body_len|resp_filenames|resp_fuids|resp_mime_types|response_body_len|status_code|status_msg|tags|trans_depth
```

同样，这里的sql语句也是没问题的

```
SELECT
  uri, tcp_percentage
FROM (
  SELECT
    uri,
    COUNT(CASE WHEN dns_log.proto = 'TCP' THEN 1 END) * 100.0 / COUNT(*)
  AS tcp_percentage
  FROM http_log
  JOIN dns_log ON http_log.uid = dns_log.uid
  WHERE http_log.status_code = '200' AND http_log.method = 'GET'
  GROUP BY uri
) AS tcp_counts

WHERE tcp_percentage > 50
```

```
[Stage 11:=====> (1 + 1) / 2]
+---+-----+
|uri|tcp_percentage|
+---+-----+
+---+-----+
```

Q2

(1)

复现的代码比较长，详细注释见附录。这里主要讲解和原教程的不同之处。

首先是在加载数据的01部分，经检查，server中的spark似乎没有deltalake的jar包，查询后确认标准的发行版是不带有这个依赖的。故舍去相关format部分的代码并修改为能运行的版本。

此外，原文件是基于databricks的notebook完成的，许多依赖和库的使用和server中普通notebook的代码出现了冲突，故重写了例如dbutil, widget的相关内容。dashboard的代码也需要另外在notebook中添加依赖。

(2)

考虑到报告太长，这里放置在附件Survival Analysis中，便于另外查看。

(3)

在1的复现00部分，已经成功将数据写入到数据库内。如下。

customerID	gender	seniorCitizen	partner	dependents	tenure	phoneService	multipleLines	internetService	onlineSecurity	onlineBackup	deviceProtection	techSupport	streamingTV
7590-VHVEG	Female	0.0	Yes	No	1.0	No	No phone service	DSL	No	Yes	No	No	No
5575-GNVDE	Male	0.0	No	No	34.0	Yes	No	DSL	Yes	No	Yes	No	No
3668-QPYBK	Male	0.0	No	No	2.0	Yes	No	DSL	Yes	Yes	No	No	No
7795-CFOCH	Male	0.0	No	No	45.0	No	No phone service	DSL	Yes	No	Yes	Yes	No
9237-HQITU	Female	0.0	No	No	2.0	Yes	No	Fiber optic	No	No	No	No	No
9305-CDGKC	Female	0.0	No	No	8.0	Yes	Yes	Fiber optic	No	No	Yes	No	Yes
1452-KIOVK	Male	0.0	No	Yes	22.0	Yes	Yes	Fiber optic	No	Yes	No	No	Yes
6713-OKOMC	Female	0.0	No	No	10.0	No	No phone service	DSL	Yes	No	No	No	No
7892-POOKP	Female	0.0	Yes	No	28.0	Yes	Yes	Fiber optic	No	No	Yes	Yes	Yes
6388-TABGU	Male	0.0	No	Yes	62.0	Yes	No	DSL	Yes	Yes	No	No	No
9763-GRSKD	Male	0.0	Yes	Yes	13.0	Yes	No	DSL	Yes	No	No	No	No
7469-LKBCL	Male	0.0	No	No	16.0	Yes	No	No	No internet service	No internet service	No internet service	No internet service	No internet service
8091-TTVAX	Male	0.0	Yes	No	58.0	Yes	Yes	Fiber optic	No	No	Yes	No	Yes

customerID	gender	seniorCitizen	partner	dependents	tenure	phoneService	multipleLines	internetService	onlineSecurity	onlineBackup	deviceProtection	techSupport	streamingTV	streamingMovies	contract	paperlessBilling	payment	
7590-VHVEG	Female	0.0	Yes	No	1.0	No	No phone service	DSL	No	Yes	No	No	No	No	No	Month-to-month	Yes	Electronic
3668-QPYBK	Male	0.0	No	No	2.0	Yes	No	DSL	Yes	Yes	No	No	No	No	No	Month-to-month	Yes	Mailed
9237-HQITU	Female	0.0	No	No	2.0	Yes	No	Fiber optic	No	No	No	No	No	No	No	Month-to-month	Yes	Electronic
9305-CDGKC	Female	0.0	No	No	8.0	Yes	Yes	Fiber optic	No	No	Yes	No	Yes	Yes	Month-to-month	Yes	Electronic	
1452-KIOVK	Male	0.0	No	Yes	22.0	Yes	Yes	Fiber optic	No	Yes	No	No	Yes	No	Month-to-month	Yes	Credit card (a	
6713-OKOMC	Female	0.0	No	No	10.0	No	No phone service	DSL	Yes	No	No	No	No	No	No	Month-to-month	No	Mailed
7892-POOKP	Female	0.0	Yes	No	28.0	Yes	Yes	Fiber optic	No	No	Yes	Yes	Yes	Yes	Month-to-month	Yes	Electronic	
9763-GRSKD	Male	0.0	Yes	Yes	13.0	Yes	No	DSL	Yes	No	No	No	No	No	No	Month-to-month	Yes	Mailed
0280-XJGEX	Male	0.0	No	No	49.0	Yes	Yes	Fiber optic	No	Yes	Yes	No	Yes	Yes	Month-to-month	Yes	Bank transfer	
5129-JLPIS	Male	0.0	No	No	25.0	Yes	No	Fiber optic	Yes	No	Yes	Yes	Yes	Yes	Month-to-month	Yes	Electronic	
4190-MFLUM	Female	0.0	Yes	Yes	10.0	Yes	No	DSL	No	No	Yes	Yes	No	No	No	Month-to-month	No	Credit card (a
4492-PPGSR	Female	0.0	No	No	31.0	Yes	No	Fiber optic	No	Yes	Yes	No	No	Yes	Month-to-month	Yes	Electronic	

下面是对于LLM的text-to-sql分析

本次实验应用的prompt模板如下,是依据说明文档完成的，修改了对应的schema部分，问题部分每次询问时进行修改。

```
prompt = "" You are a smart agent responsible for generating the correct
SQL statements based on the following information: - Database structure
information: including table names, fields, relationships between tables
(such as foreign keys, etc.). - User questions: natural language queries or
questions. Your main tasks are: 1. Analyze database structure information: -
Based on the database structure information, understand the fields and
relationships of the table, and build the basic framework of the SQL
statement. 2. Generate SQL statements: - Based on user questions, query
requirements and conditions, tables involved, and auxiliary query
conditions, construct complete SQL statements. 3. Verification and
optimization: - Check whether the generated SQL statement is logical and
optimize it if necessary. ### Input: - Database structure information:
including table names, fields, relationships between tables (such as foreign
keys, etc.). - User questions: natural language queries or questions. ###
Output: - Return the result in json format, the format is {"sql": "SQL
statement that meets the user's question requirements"} ### Operation steps:
1. Analyze database structure information: build the basic framework of the
SQL statement. 2. Generate SQL statements: construct complete SQL
statements. 3. Verification and optimization: ensure the logical correctness
of the SQL statement and optimize it. ### Note: - Ensure that the SQL
statement accurately reflects the query requirements in the user questions.
- Reasonably construct query logic based on database structure. - When
generating SQL statements, consider all the information provided to ensure
the correctness and efficiency of the statements. - If the user question
involves complex query requirements, please consider all requirements to
generate SQL statements. ### given the following database schema : schema =
StructType([ StructField('customerID', StringType()), StructField('gender',
StringType()), StructField('seniorCitizen', DoubleType()),
StructField('partner', StringType()), StructField('dependents',
StringType()), StructField('tenure', DoubleType()),
StructField('phoneService', StringType()), StructField('multipleLines',
StringType()), StructField('internetService', StringType()),
StructField('onlineSecurity', StringType()), StructField('onlineBackup',
StringType()), StructField('deviceProtection', StringType()),
StructField('techSupport', StringType()), StructField('streamingTV',
StringType()), StructField('streamingMovies', StringType()),
StructField('contract', StringType()), StructField('paperlessBilling',
StringType()), StructField('paymentMethod', StringType()),
StructField('monthlyCharges', DoubleType()), StructField('totalCharges',
DoubleType()), StructField('churnString', StringType()) ]) Write a sql code
to answer the following question: How many customer are there ? ""
```

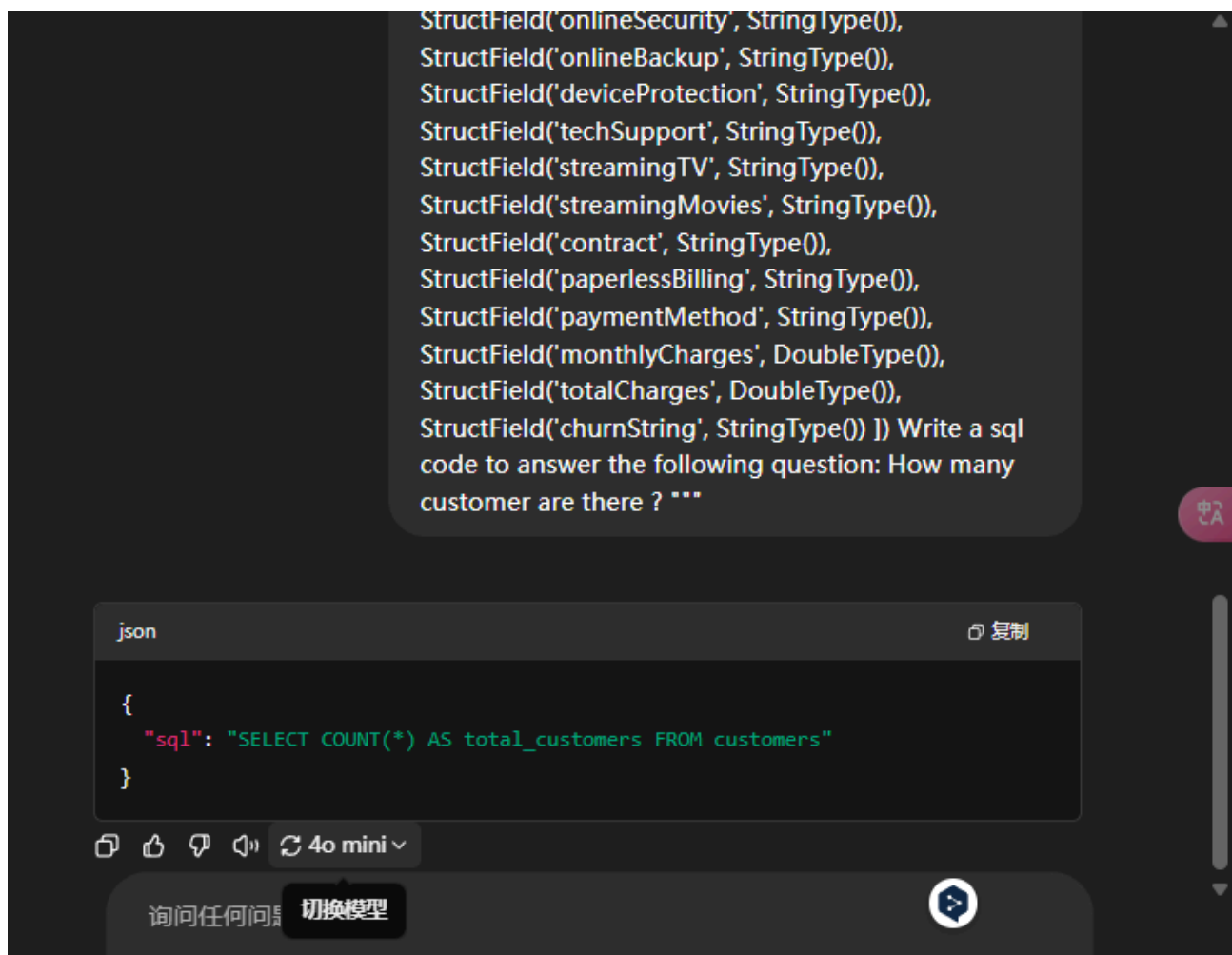
```
schema = StructType([ StructField('customerID', StringType()),
StructField('gender', StringType()), StructField('seniorCitizen',
DoubleType()), StructField('partner', StringType()),
StructField('dependents', StringType()), StructField('tenure',
DoubleType()), StructField('phoneService', StringType()),
StructField('multipleLines', StringType()), StructField('internetService',
StringType()), StructField('onlineSecurity', StringType()),
StructField('onlineBackup', StringType()), StructField('deviceProtection',
StringType()), StructField('techSupport', StringType()),
StructField('streamingTV', StringType()), StructField('streamingMovies',
StringType()), StructField('contract', StringType()),
StructField('paperlessBilling', StringType()), StructField('paymentMethod',
StringType()), StructField('monthlyCharges', DoubleType()),
StructField('totalCharges', DoubleType()), StructField('churnString',
StringType()) ])
```

Wrong Case 1:

Question: Write a sql code to answer the following question: How many customer are there ?

Answer by LLM:

```
{
  "sql": "SELECT COUNT(*) AS total_customers FROM customers"
}
```



True Answer:

```
SELECT COUNT(DISTINCT customerID) AS customer_count
FROM customers;
```

错误分析：LLM误认为这个数据表的每一行都是唯一的customer数据，但这个表并没有这样的Primary key限制，应当对customerID进行去重的计算得到结果。

Wrong Case 2:

Question: How much do a cunstomer probably pay?

Answer by LLM:

To answer the user's question, "How much do a customer probably pay?", we need to focus on the payment-related fields from the provided schema. Based on the schema, the most relevant fields for the payment would be `monthlyCharges` and `totalCharges`.

However, to understand how much a customer probably pays, it could be referring to the `monthlyCharges` (as it reflects ongoing charges). Therefore, the SQL query would need to fetch the average `monthlyCharges` for all customers, which will give an estimate of how much a customer probably pays.

Here's the SQL statement to answer the question:

```
`{  "sql": "SELECT AVG(monthlyCharges) AS averageMonthlyPayment FROM  
table_name" }`
```

Note: Replace `table_name` with the actual table name in your database. Since the schema doesn't specify the table name, it should be provided for the SQL to execute properly.

Explanation:

1. **AVG(monthlyCharges)**: Calculates the average monthly charge across all customers, which answers the question of how much a customer probably pays.
2. **FROM table_name**: Specifies the table from which the data will be fetched.

This SQL query assumes that the question is asking for an average monthly payment for customers. If the interpretation is different, additional context or clarification of the question would be needed.

True answer:

```
SELECT AVG(avgTotalCharges) AS avg_totalCharges  
FROM (  
    SELECT customerID, AVG(totalCharges) AS avgTotalCharges  
    FROM customers  
    GROUP BY customerID  
) AS sub;
```

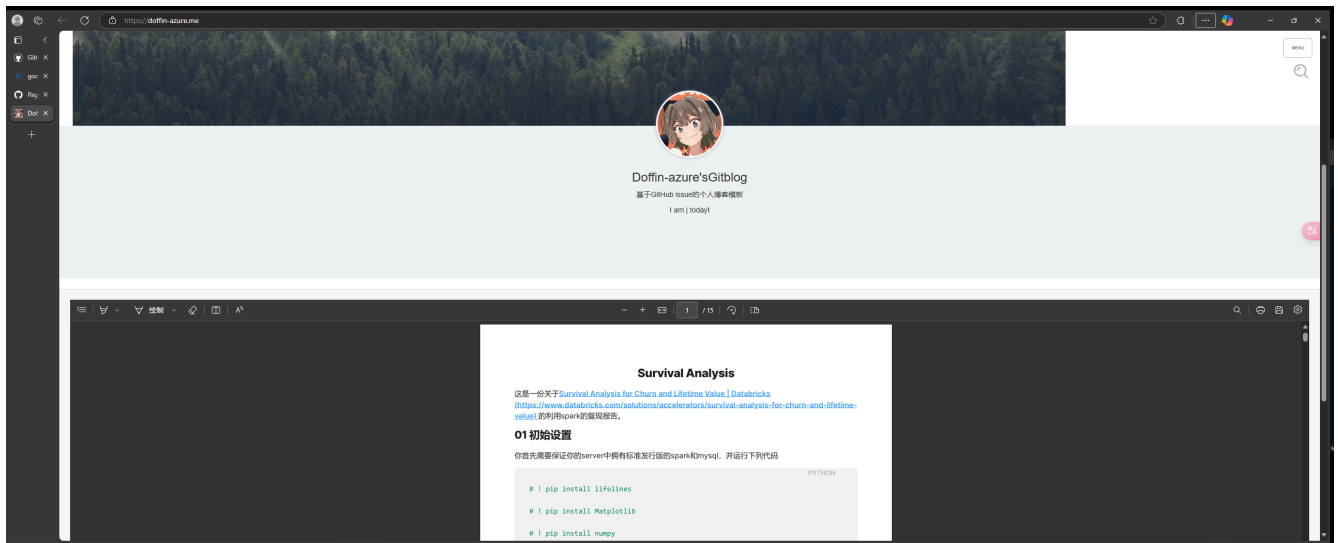
相比CASE1 这个问题取消了sql语句的限制, 故LLM生成了许多文字, 展示了其思考过程。此外, 该问题设置了陷阱, 故意模糊了是针对哪个charge列进行的询问。LLM发现了这点, 但误解了对象, 还是选择了monthlyCharges这个概念。另外, 其去重的需求依旧没有被发现。

总结:

- 问题的不同会确实影响大模型的生成文本，强调了sql to answer时，会只生成sql语句。
- LLM容易忽视数据的重复、清理等操作，也不能很好把握一个schema代表的数据对象。容易误解text中的指向，尤其在schema中存在复数相关的列时。

(4)

通过fork [imuncle/gitblog](https://github.com/imuncle/gitblog): 基于git issue的轻量级个人博客模板。
(<https://github.com/imuncle/gitblog>) 完成了个人网站Gitblog (<http://doffin-azure.me/>)



其中，关于域名的相关配置，参考了[Enforce HTTPS in GitHub Pages with Namecheap Domain](https://www.youtube.com/watch?v=FBtehan5DAo) (<https://www.youtube.com/watch?v=FBtehan5DAo>) 的视频，利用education pack中的namecheap网站的域名并使用了github的托管服务。