

## 45 | 数据清洗：如何使用SQL对数据进行清洗？

SQL 可以帮我们进行数据处理，总的来说可以分成 OLTP 和 OLAP 两种方式。

OLTP 称之为联机事务处理，我们之前讲解的对数据进行增删改查，SQL 查询优化，事务处理等就属于 OLTP 的范畴。它对实时性要求高，需要将用户的数据有效地存储到数据库中，同时有时候针对互联网应用的需求，我们还需要设置数据库的主从架构保证数据库的高并发和高可用性。

OLAP 称之为联机分析处理，它是对已经存储在数据库中的数据进行分析，帮我们得出报表，指导业务。它对数据的实时性要求不高，但数据量往往很大，存储在数据库（数据仓库）中的数据可能还存在数据质量的问题，比如数据重复、数据中有缺失值，或者单位不统一等，因此在进行数据分析之前，首要任务就是对收集的数据进行清洗，从而保证数据质量。

对于数据分析工作来说，好的数据质量才是至关重要的，它决定了后期数据分析和挖掘的结果上限。数据挖掘模型选择得再好，也只能最大化地将数据特征挖掘出来。

高质量的数据清洗，才有高质量的数据。今天我们就来看下，如何用 SQL 对数据进行清洗。

1. 想要进行数据清洗有怎样的准则呢？
2. 如何使用 SQL 对数据进行清洗？
3. 如何对清洗之后的数据进行可视化？

### 数据清洗的准则

我在《数据分析实战 45 讲》里专门讲到过数据清洗的原则，这里为了方便你理解，我用一个数据集实例讲一遍。

一般而言，数据集或多或少地会存在数据质量问题。这里我们使用泰坦尼克号乘客生存预测数据集，你可以从[GitHub](#)上下载这个数据集。

数据集格式为 csv，一共有两种文件：train.csv 是训练数据集，包含特征信息和存活与否的标签；test.csv 是测试数据集，只包含特征信息。

数据集中包括了以下字段，具体的含义如下：

字段	说明
PassengerId	乘客编号
Survived	是否幸存
Pclass	船票等级
Name	乘客姓名
Sex	乘客性别
SibSp	亲戚数量 (兄妹、配偶数)
Parch	亲戚数量 (父母、子女数)
Ticket	船票号码
Fare	船票价格
Cabin	船舱
Embarked	登陆港口

训练集给出了 891 名乘客幸存与否的结果，以及相关的乘客信息。通过训练集，我们可以对数据进行建模形成一个分类器，从而对测试集中的乘客生存情况进行预测。不过今天我们并不讲解数据分析的模型，而是来看下在数据分析之前，如何对数据进行清洗。

首先，我们可以通过 Navicat 将 CSV 文件导入到 MySQL 数据库中，然后浏览下数据集中的前几行，可以发现数据中存在缺失值的情况还是很明显的。

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	0	3	Braund, Mr. C male	male	22	1	0	A/5 21171	7.25	(Null)	S
2	1	1	Cumings, Mr. J female	female	38	1	0	PC 17599	71.2833	C85	C
3	1	3	Heikkinen, M female	female	26	0	0	STON/O2. 3101282	7.925	(Null)	S
4	1	1	Futrelle, Mrs. J female	female	35	1	0	113803	53.1	C123	S
5	0	3	Allen, Mr. Wil male	male	35	0	0	373450	8.05	(Null)	S
6	0	3	Moran, Mr. J male	male	(Null)	0	0	330877	8.4583	(Null)	Q
7	0	1	McCarthy, M male	male	54	0	0	17463	51.8625	E46	S
8	0	3	Palsson, M male	male	2	3	1	349909	21.075	(Null)	S
9	1	3	Johnson, Mrs. A female	female	27	0	2	347742	11.1333	(Null)	S
10	1	2	Nasser, Mrs. F female	female	14	1	0	237736	30.0708	(Null)	C

数据存在数据缺失值是非常常见的情况，此外我们还需要考虑数据集中某个字段是否存在单位标识不统一，数值是否合法，以及数据是否唯一等情况。要考虑的情况非常多，这里我将数据清洗中需要考虑的规则总结为 4 个关键点，统一起来称之为“完全合一”准则，你可以[点这里](#)看一下。

“完全合一”是个通用的准则，针对具体的数据集存在的问题，我们还需要对症下药，采取适合的解决办法，甚至为了后续分析方便，有时我们还需要将字符类型的字段替换成数值类型，比如我们想做一个 Steam 游戏用户的数据分析，统计数据存储在两张表上，一个是 user\_game 数据表，记录了用户购买的各种 Steam 游戏，其中数据表中的 game\_title 字段表示玩家购买的游戏名称，它们都采用英文字符的方式。另一个是 game 数据表，记录了游戏的 id、游戏名称等。因为这两张表存在关联关系，实际上在 user\_game 数据表中的 game\_title 对应了 game 数据表中的 name，这里我们就可以用 game 数据表中的 id 替换掉原有的 game\_title。替换之后，我们在进行数据清洗和质量评估的时候也会更清晰，比如如果还存在某个 game\_title 没有被替换的情况，就证明这款游戏在 game 数据表中缺少记录。

## 使用 SQL 对预测数据集进行清洗

了解了数据清洗的原则之后，下面我们就用 SQL 对泰坦尼克号数据集中的训练集进行数据清洗，也就是 train.csv 文件。我们先将这个文件导入到 titanic\_train 数据表中：

源表	目标表	新建表
train	titanic_train	<input checked="" type="checkbox"/>

### 检查完整性

在完整性这里，我们需要重点检查字段数值是否存在空值，在此之前，我们需要先统计每个字段空值的个数。在 SQL 中，我们可以分别统计每个字段的空值个数，比如针对 Age 字段进行空值个数的统计，使用下面的命令即可：

 复制代码

```
1 SELECT COUNT(*) as num FROM titanic_train WHERE Age IS NULL
```

运行结果为 177。

当然我们也可以同时对多个字段的非空值进行统计：

 复制代码

```
1 SELECT
2 SUM((CASE WHEN Age IS NULL THEN 1 ELSE 0 END)) AS age_null_num,
3 SUM((CASE WHEN Cabin IS NULL THEN 1 ELSE 0 END)) AS cabin_null_num
4 FROM titanic_train
```

运行结果：

age_null_num	cabin_null_num
177	687

不过这种方式适用于字段个数较少的情况，如果一个数据表存在几十个，甚至更多的字段，那么采用这种方式既麻烦又容易出错。这时我们可以采用存储过程的方式，用程序来进行字段的空值检查，代码如下：

 复制代码

```
1 CREATE PROCEDURE `check_column_null_num`(IN schema_name VARCHAR(100), IN table_name2 VA
2 BEGIN
3 -- 数据表 schema_name 中的列名称
4 DECLARE temp_column VARCHAR(100);
5 -- 创建结束标志变量
6 DECLARE done INT DEFAULT false;
7 -- 定义游标来操作每一个 COLUMN_NAME
8 DECLARE cursor_column CURSOR FOR
9 SELECT COLUMN_NAME FROM information_schema.COLUMNS WHERE table_schema = schema_name AND
10 -- 指定游标循环结束时的返回值
11 DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = true;
12 -- 打开游标
13 OPEN cursor_column;
14 read_loop:LOOP
15     FETCH cursor_column INTO temp_column;
16     -- 判断游标的循环是否结束
17     IF done THEN
18         LEAVE read_loop;
19     END IF;
20     -- 这里需要设置具体的 SQL 语句 temp_query
21     SET @temp_query=CONCAT('SELECT COUNT(*) as ', temp_column, '_null_num FROM '
22     -- 执行 SQL 语句
23     PREPARE stmt FROM @temp_query;
```

```
24     EXECUTE stmt;
25 END LOOP;
26 -- 关闭游标
27 CLOSE cursor_column;
28 END
```

我来说下这个存储过程的作用，首先我定义了两个输入的参数 schema\_name 和 table\_name2，用来接收想要检查的数据库的名称以及数据表名。

然后使用游标来操作读取出来的 column\_name，赋值给变量 temp\_column。对于列名，我们需要检查它是否为空，但是这个列名在 MySQL 中是动态的，我们无法使用 @temp\_column 来表示列名，对其进行判断，在这里我们需要使用 SQL 拼接的方式，这里我设置了 @temp\_query 表示想要进行查询的 SQL 语句，然后设置 COUNT(\*) 的别名为动态别名，也就是 temp\_column 加上 \_null\_num，同样在 WHERE 条件判断中，我们使用 temp\_column 进行动态列名的输出，以此来判断这个列数值是否为空。

然后我们执行这个 SQL 语句，提取相应的结果。

 复制代码

```
1 call check_column_null_num('wucai', 'titanic_train');
```

运行结果如下：

 复制代码

```
1 Age_null_num: 177
2 Cabin_null_num: 687
3 Embarked_null_num: 2
4 Fare_null_num: 0
5 Name_null_num: 0
6 Parch_null_num: 0
7 PassengerId_null_num: 0
8 Pclass_null_num: 0
9 Sex_null_num: 0
10 SibSp_null_num: 0
11 Survived_null_num: 0
12 Ticket_null_num: 0
```

为了浏览方便我调整了运行结果的格式，你能看到在 titanic\_train 数据表中，有 3 个字段是存在空值的，其中 Cabin 空值数最多为 687 个，Age 字段空值个数 177 个，Embarked 空值个数 2 个。

既然存在空值的情况，我们就需要对它进行处理。针对缺失值，我们有 3 种处理方式。

1. 删除：删除数据缺失的记录；
2. 均值：使用当前列的均值；
3. 高频：使用当前列出现频率最高的数据。

对于 Age 字段，这里我们采用均值的方式进行填充，但如果直接使用 SQL 语句可能会存在问题，比如下面这样。

 复制代码

```
1 UPDATE titanic_train SET age = (SELECT AVG(age) FROM titanic_train) WHERE age IS NULL
```

这时会报错：

 复制代码

```
1 1093 - You can't specify target table 'titanic_train' for update in FROM clause
```

也就是说同一条 SQL 语句不能先查询出来部分内容，再同时对当前表做修改。

这种情况下，最简单的方式就是复制一个临时表 titanic\_train2，数据和 titanic\_train 完全一样，然后再执行下面这条语句：

 复制代码

```
1 UPDATE titanic_train SET age = (SELECT ROUND(AVG(age),1) FROM titanic_train2) WHERE age
```

这里使用了 ROUND 函数，对 age 平均值 AVG(age) 进行四舍五入，只保留小数点后一位。

针对 Cabin 这个字段，我们了解到这个字段代表用户的船舱位置，我们先来看下 Cabin 字段的数值分布情况：

 复制代码

```
1 SELECT COUNT(cabin), COUNT(DISTINCT(cabin)) FROM titanic_train
```

运行结果：

COUNT(cabin)	COUNT(DISTINCT(cabin))
204	147

从结果中能看出 Cabin 字段的数值分布很广，而且根据常识，我们也可以知道船舱位置每个人的差异会很大，这里既不能删除掉记录航，又不能采用均值或者高频的方式填充空值，实际上这些空值即无法填充，也无法对后续分析结果产生影响，因此我们可以不处理这些空值，保留即可。

然后我们来看下 Embarked 字段，这里有 2 个空值，我们可以采用该字段中高价值作为填充，首先我们先了解字段的分布情况使用：

 复制代码

```
1 SELECT COUNT(*), embarked FROM titanic_train GROUP BY embarked
```

运行结果：

COUNT(*)	embarked
644	S
168	C
77	Q
2	

我们可以直接用 S 来对缺失值进行填充：

 复制代码

```
1 UPDATE titanic_train SET embarked = 'S' WHERE embarked IS NULL
```

至此，对于 titanic\_train 这张数据表中的缺失值我们就处理完了。

## 检查全面性

在这个过程中，我们需要观察每一列的数值情况，同时查看每个字段的类型。

名	类型	长度	小数点	不是 null	虚拟	键
PassengerId	varchar	255	0	<input type="checkbox"/>	<input type="checkbox"/>	
Survived	varchar	255	0	<input type="checkbox"/>	<input type="checkbox"/>	
Pclass	varchar	255	0	<input type="checkbox"/>	<input type="checkbox"/>	
Name	varchar	255	0	<input type="checkbox"/>	<input type="checkbox"/>	
Sex	varchar	255	0	<input type="checkbox"/>	<input type="checkbox"/>	
Age	varchar	255	0	<input type="checkbox"/>	<input type="checkbox"/>	
SibSp	varchar	255	0	<input type="checkbox"/>	<input type="checkbox"/>	
Parch	varchar	255	0	<input type="checkbox"/>	<input type="checkbox"/>	
Ticket	varchar	255	0	<input type="checkbox"/>	<input type="checkbox"/>	
Fare	varchar	255	0	<input type="checkbox"/>	<input type="checkbox"/>	
Cabin	varchar	255	0	<input type="checkbox"/>	<input type="checkbox"/>	
Embarked	varchar	255	0	<input type="checkbox"/>	<input type="checkbox"/>	

因为数据是直接从 CSV 文件中导进来的，所以每个字段默认都是 VARCHAR(255) 类型，但很明显 PassengerId、Survived、Pclass 和 Sibsp 应该设置为 INT 类型，Age 和 Fare 应该设置为 DECIMAL 类型，这样更方便后续的操作。使用下面的 SQL 命令即可：

 复制代码

```
1 ALTER TABLE titanic_train CHANGE PassengerId PassengerId INT(11) NOT NULL PRIMARY KEY;
2 ALTER TABLE titanic_train CHANGE Survived Survived INT(11) NOT NULL;
3 ALTER TABLE titanic_train CHANGE Pclass Pclass INT(11) NOT NULL;
4 ALTER TABLE titanic_train CHANGE Sibsp Sibsp INT(11) NOT NULL;
5 ALTER TABLE titanic_train CHANGE Age Age DECIMAL(5,2) NOT NULL;
6 ALTER TABLE titanic_train CHANGE Fare Fare DECIMAL(7,4) NOT NULL;
```

然后我们将其余的字段（除了 Cabin）都进行 NOT NULL，这样在后续进行数据插入或其他操作的时候，即使发现数据异常，也可以对字段进行约束规范。

在全面性这个检查阶段里，除了字段类型定义需要修改以外，我们没有发现其他问题。

**然后我们来检查下合法性及唯一性。** 合法性就是要检查数据内容、大小等是否合法，这里不存在数据合法性问题。

针对数据是否存在重复的情况，我们刚才对 PassengerId 字段类型进行更新的时候设置为了主键，并没有发现异常，证明数据是没有重复的。

## 对清洗之后的数据进行可视化

我们之前讲到过如何通过 Excel 来导入 MySQL 中的数据，以及如何使用 Excel 来进行数据透视表和数据透视图的呈现。

这里我们使用 MySQL For Excel 插件来进行操作，在操作之前有两个工具需要安装。

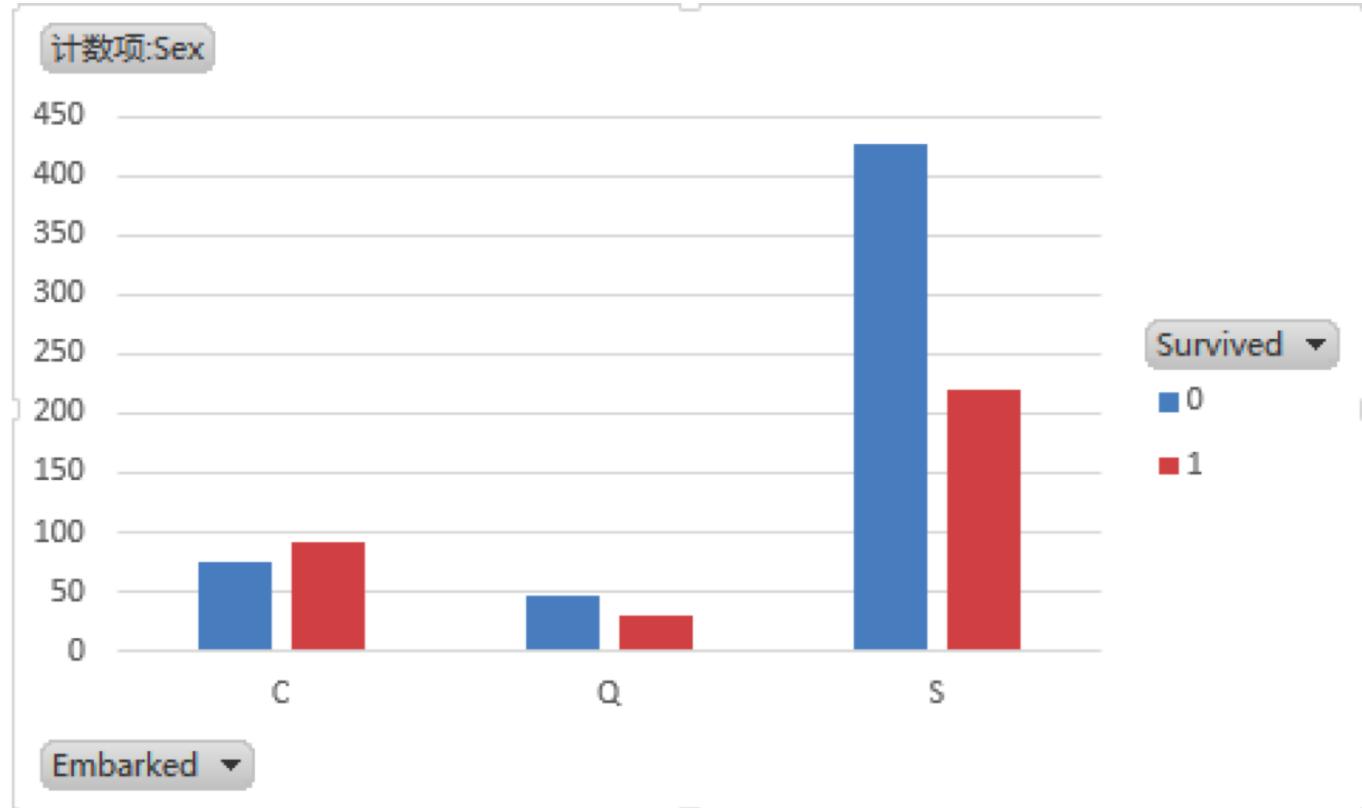
首先是 mysql-for-excel，点击[这里](#)进行下载；然后是 mysql-connector-odbc，点击[这里](#)进行下载。

安装好之后，我们新建一个空的 excel 文件，打开这个文件，在数据选项中可以找到“MySQL for Excel”按钮，点击进入，然后输入密码连接 MySQL 数据库。

然后选择我们的数据库以及数据表名称，在下面可以找到 Import MySQL Data 按钮，选中后将数据表导入到 Excel 文件中。

	A	B	C	D	E	F	G	H	I	J	K	L
1	PassengerId	Survived	Pclass	Name	Sex	Age	Sibsp	Parch	Ticket	Fare	Cabin	Embarked
2	1	0	3	Braund, Mr. Owen Harris	male	22	1	0	A/5 21171	7.25	S	
3	2	1	1	Cumings, Mrs. John Bradley (female)	female	38	1	0	PC 17599	71.2833 C85	C	
4	3	1	3	Heikkinen, Miss. Laina	female	26	0	0	STON/O2. 3101282	7.925	S	
5	4	1	1	Futrelle, Mrs. Jacques Heath	female	35	1	0	113803	53.1 C123	S	
6	5	0	3	Allen, Mr. William Henry	male	35	0	0	373450	8.05	S	
7	6	0	3	Moran, Mr. James	male	29.7	0	0	330877	8.4583	Q	
8	7	0	1	McCarthy, Mr. Timothy J	male	54	0	0	17463	51.8625 E46	S	
9	8	0	3	Paisson, Master. Gusta Leon	male	2	3	1	349909	21.075	S	
10	9	1	3	Johnson, Mrs. Oscar W (Elisafemale)	female	27	0	2	347742	11.1333	S	
11	10	1	2	Nasser, Mrs. Nicholas (Adele)female	female	14	1	0	237736	30.0708	C	
12	11	1	3	Sandstrom, Miss. Marguerite	female	4	1	1	PP 9549	16.7 G6	S	
13	12	1	1	Bonnel, Miss. Elizabeth	female	58	0	0	113783	26.55 C103	S	
14	13	0	3	Saundercock, Mr. William Hen	male	20	0	0	A/5. 2151	8.05	S	
15	14	0	3	Andersson, Mr. Anders Johan	male	39	1	5	347082	31.275	S	
16	15	0	3	Vestrom, Miss. Hulda Amanda	female	14	0	0	350406	7.8542	S	
17	16	1	2	Hewlett, Mrs. (Mary D King)c	female	55	0	0	248706	16	S	
18	17	0	3	Rice, Master. Eugene	male	2	4	1	382652	29.125	Q	
19	18	1	2	Williams, Mr. Charles Eugen	male	29.7	0	0	244373	13	S	
20	19	0	3	Vander Planke, Mrs. Julius	(female)	31	1	0	345763	18	S	
21	20	1	3	Masselmani, Mrs. Fatima	female	29.7	0	0	2649	7.225	C	
22	21	0	2	Fynney, Mr. Joseph J	male	35	0	0	239865	26	S	
23	22	1	2	Beesley, Mr. Lawrence	male	34	0	0	248698	13 D56	S	
24	23	1	3	McGowan, Miss. Anna "Annie"	female	15	0	0	330923	8.0292	Q	
25	24	1	1	Sloper, Mr. William Thompson	male	28	0	0	113788	35.5 A6	S	
26	25	0	3	Palsson, Miss. Torborg Danir	female	8	3	1	349909	21.075	S	
27	26	1	3	Asplund, Mrs. Carl Oscar (Sefemale)	female	38	1	5	347077	31.3875	S	
28	27	0	3	Emir, Mr. Farred Chehab	male	29.7	0	0	2631	7.225	C	
29	28	0	1	Fortune, Mr. Charles Alexander	male	19	3	2	19950	263 C23 C25 C27	S	
30	29	1	3	O'Diver, Miss. Ellen "Nellie"	female	29.7	0	0	330959	7.8792	Q	

在“插入”选项中找到“数据透视图”，这里我们选中 Survived、Sex 和 Embarked 字段，然后将 Survive 字段放到图例（系列）栏中，将 Sex 字段放到求和值栏中，可以看到呈现出如下的数据透视表：



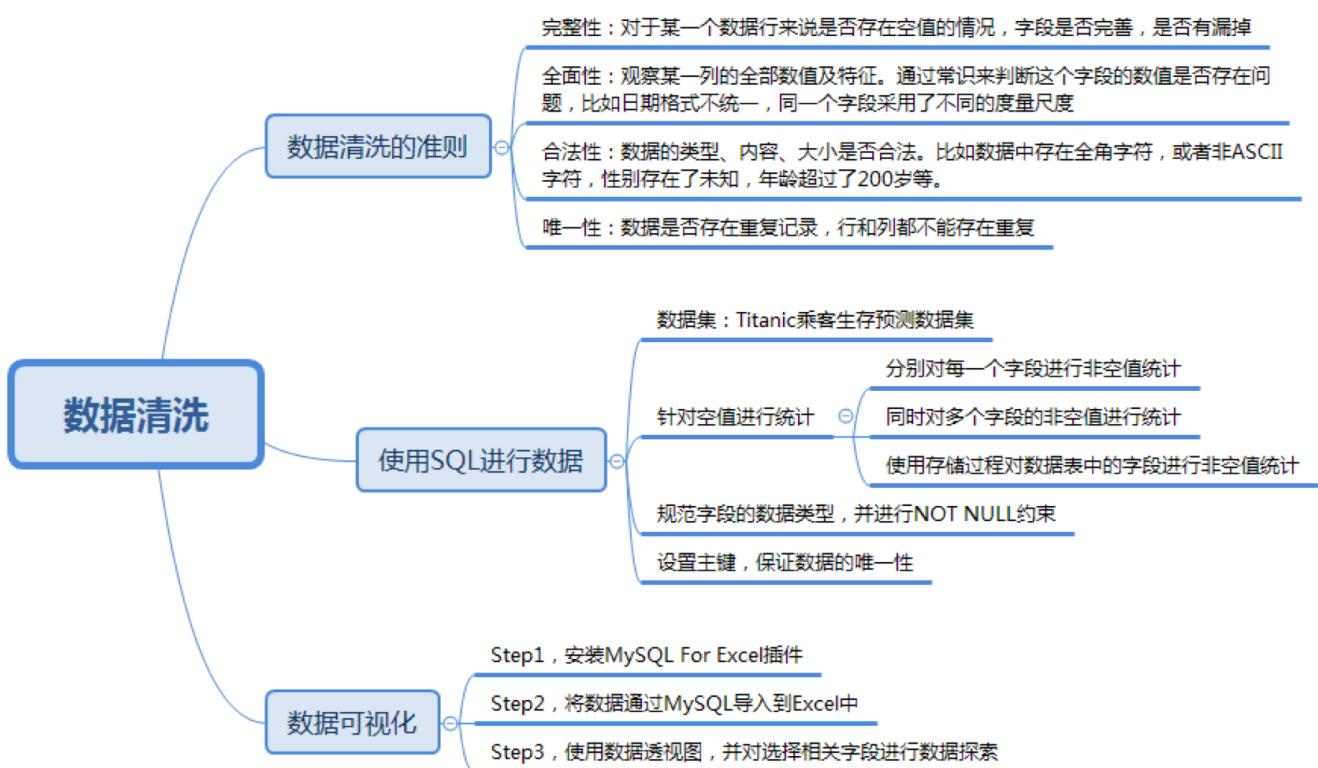
从这个透视表中你可以清晰地了解到用户生存情况 (Survived) 与 Embarked 字段的关系，当然你也可以通过数据透视图进行其他字段之间关系的探索。

为了让你能更好地理解操作的过程，我录制了一段操作视频。

## 总结

在数据清洗过程中，你能看到通过 SQL 来进行数据概览的查询还是很方便的，但是使用 SQL 做数据清洗，会有些繁琐，这时你可以采用存储过程对数据进行逐一处理，当然你也可以使用后端语言，比如使用 Python 来做具体的数据清洗。

在进行数据探索的过程中，我们可能也会使用到数据可视化，如果不采用 Python 进行可视化，你也可以选择使用 Excel 自带的数据透视图来进行可视化的呈现，它会让你对数据有个更直观的认识。



今天讲解的数据清洗的实例比较简单，实际上数据清洗是个反复的过程，有时候我们需要几天时间才能把数据完整清洗好。你在工作中，会使用哪些工具进行数据清洗呢？

另外，数据缺失问题在数据清洗中非常常见，我今天列举了三种填充数据缺失的方式，分别是删除、均值和高频的方式。实际上缺失值的处理方式不局限于这三种，你可以思考下，如果数据量非常大，某个字段的取值分布也很广，那么对这个字段中的缺失值该采用哪种方式来进行数据填充呢？