我接了一个任务，是将原本teradata的数据仓库迁移到redshift。为了校验迁移前后的数据是否一致，我写了一个工具脚本来比较迁移前后的数据的一致性。然后我现在写一个ppt来向同事介绍经验，请你帮我成日文，要求总结得适合ppt展示。

Ppt的第一部分，介绍データ検証目的：

确保迁移前后的数据与期待一致。

这里的期待一致有几点含义:  
1，迁移前后数据的类型和值完全一致

2，需要被迁移的数据全部被迁移，没有遗漏

3，不需要迁移的数据未被迁移到目标仓库(例如迁移后应当被删除的列)

4，意图改变的数据按照预期改变(如:迁移前为Int型,迁移后应为String型，列名在迁移前后改变)

5，由于数据仓库自身的原因，导致迁移前后不得不出现的差异，但是产生的差异是可以容忍的。

第二部分，介绍データ検証方法

最初客户想定的验证方法是通过

1. 获取迁移前后的数据条数，
2. 对迁移前后的表分别抽取一定数量的样本，输出为csv比较样本的值

但是在实施抽取样本到csv时，发现有以下问题:  
1，teradata的csv输出工具bteq.sh，只能抽出最多100列的数据，当表的列数超过100时，无法抽出完整的数据。

2，抽出为csv后，列的数据类型缺失,全部表现为字符串类型

1. Teradata的抽出的csv包含BOM和redshift的抽取命令输出的csv不包含BOM,由于BOM肉眼不可见，造成目视一致但使用diff命令时显示不一致的问题。
2. Teradata的固定长文字列迁移到redshift的可变长文字列时，teradata输出的csv包含填充固定长的空格，但redshift输出的值不包含填位的空格。
3. 目视列数庞大的表容易产生出错。
4. 手动执行sql容易出错

因此，本项目中我们利用数据分析库pandas开发了自动比较脚本。

第三部分,介绍比较脚本:  
1,由于数据分析库pandas是python的一个库，开发脚本时，客户已有teradata的连接client脚本，但未有python的redshift连接client脚本，因此本次开发主要由以下几部分构成:  
1，redhshift\_client.py：连接redshift数据库，获取指定表的数据的脚本

2，migration\_checker.py:

包含一个范用的DataMigrationBase的父类，负责:1,获取teradata和redshift的连接的设置，2，获取teradata和redshift的连接。3，将dataframe输出为csv的方法，4，处理迁移前后的列名的mapping以及删除不需要的列

一个实现DataMigrationBase的子类DataMigrationChecker,负责处理获取迁移前后的数据数量，抽样和比较样本数据。具体处理流程为:1,获取执行脚本时传入的参数，包括表名，迁移前后的数据库名，抽样比较时使用的key, 以及可选的参数:如列名的mapping文件，比较的日期或月份范围，指定的日期列名或月份列名，抽样数据的大小，2，比较指定条件下的数据条数，3，抽取数据样本，然后通过key或key的组合找出相应的迁移前后数据样本。4，处理teradata的样本dataframe，使得该dataframe的数据转变为符合redshift的样式后，与redshift抽取出来的dataframe作比较5,输出比较结果，包括输出不一致的详细信息为csv,输出teradata和redshift的抽样数据为csv.

3，na\_empty\_counter.py:由于在迁移过程中发现null迁移后变为空值的情况，另外准备了一个比较迁移前后表中各列的null和empty的数量的脚本，这个脚本的类继承自父类DataMigrationBase，获取表中各列是否允许null,是否是字符串类型，如果允许null，则执行统计null数据量的sql,如果是字符串，则执行统计空字符串数量的sql.