openGauss AI特性创新实践课



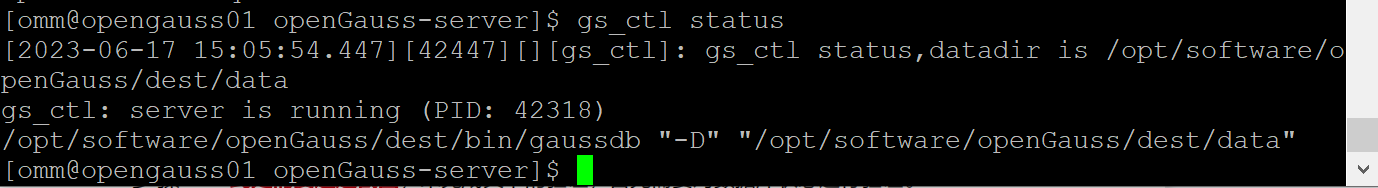
华为技术有限公司

# 关卡一、openGauss数据安装及基本操作

openGauss数据安装及基本操作, 作业提交任务如下：

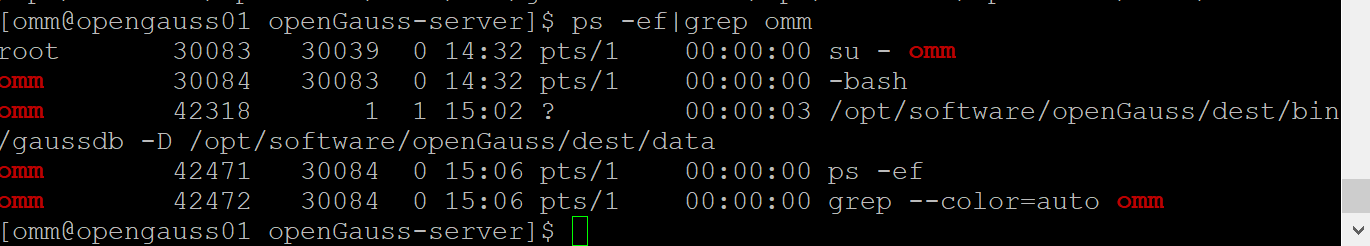
任务一：数据库状态验证

1. 查询数据库状态成功截图



任务二：数据库服务进程验证

1. 查看数据库服务进程截图（包含数据库服务器的主机名）



步骤 3 简单描述关卡一中，安装数据库所需要的步骤。

1. 创建omm用户
2. 下载第三方编译库openGauss-third\_party\_binarylibs\_openEuler\_arm.tar.gz，通过git下载openGauss源码
3. 安装依赖包，替换python版本
4. 将/opt/software路径的用户所属组替换为omm，权限设置为755
5. 设置环境变量，然后进入openGauss 源码下，生成配置文件
6. 编译、安装openGauss

实验思考题：为什么需要通过源码编译，安装数据库？

定制编译：源码编译允许对OpenGauss进行自定义编译，以满足特定需求。可以根据自己的环境、硬件要求和性能需求对编译参数进行配置，以获得最佳性能和功能。

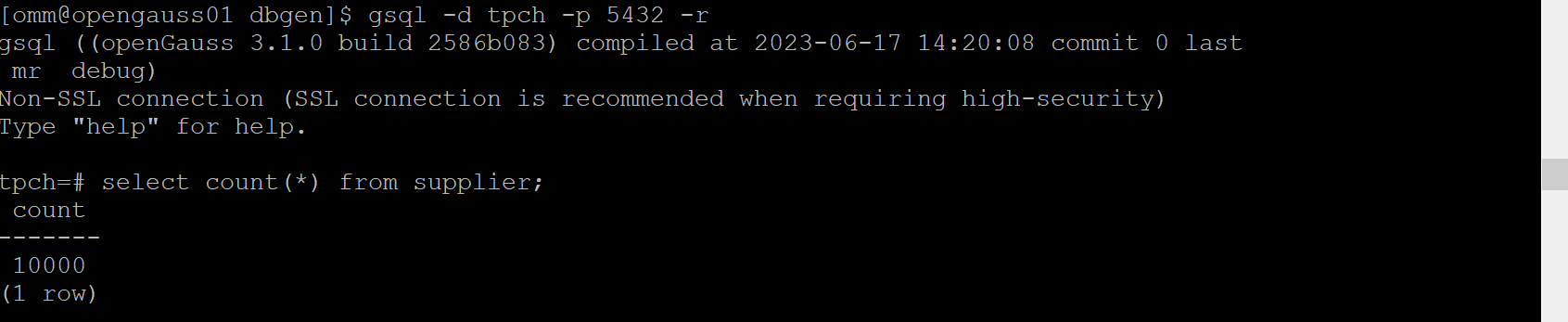
平台兼容性：OpenGauss的预编译二进制版本可能仅适用于特定操作系统或特定版本的操作系统。如果操作系统与可用的二进制版本不兼容，那么可能需要从源代码编译OpenGauss，以确保它能够在平台上正常运行。

# 关卡二、openGauss数据导入及基本操作

任务一：数据初始化验证

1. 查询supplier表的行数，并将结果进行图：

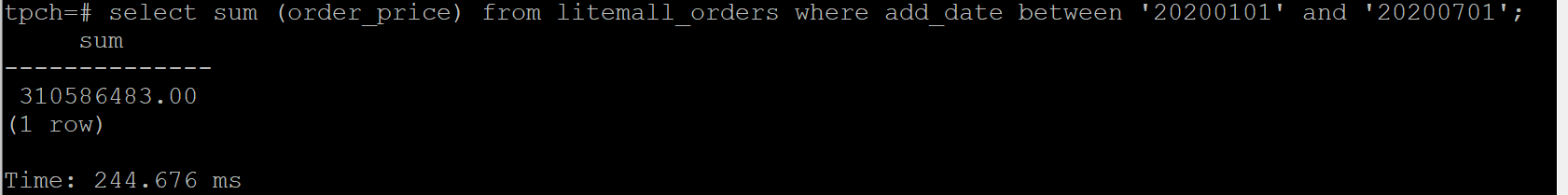
select count(\*) from supplier;;



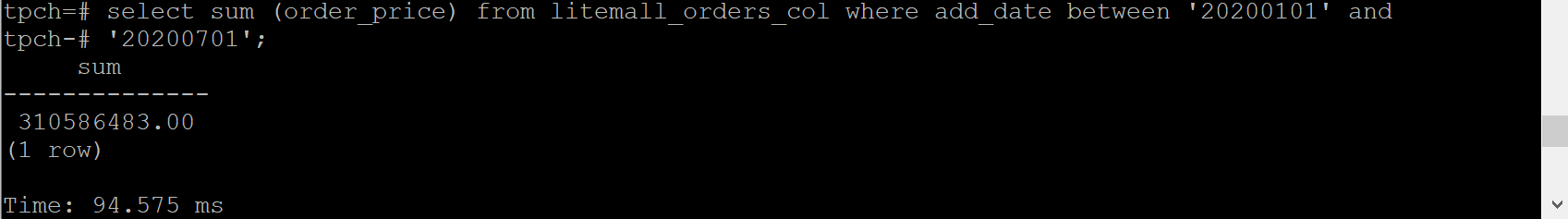
任务二：行存表与列存表执行效率对比

1. 2020年上半年litemall\_orders行存表与litemall\_orders\_col列存表中的order\_price的总和查询，并对比执行效率截图

select sum (order\_price) from litemall\_orders where add\_date between '20200101' and '20200701';

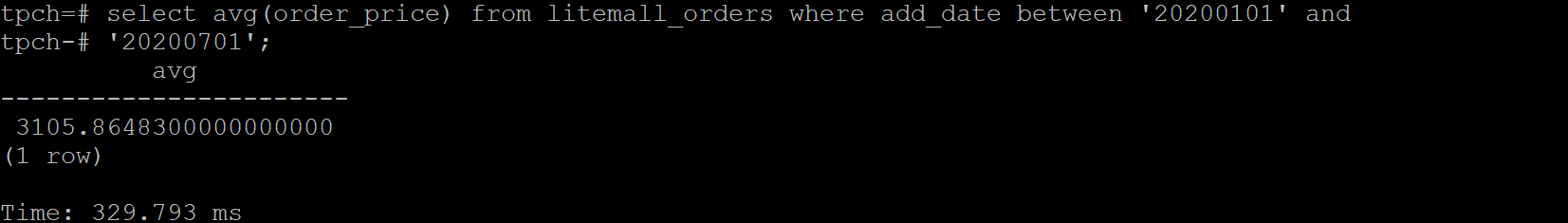


select sum (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

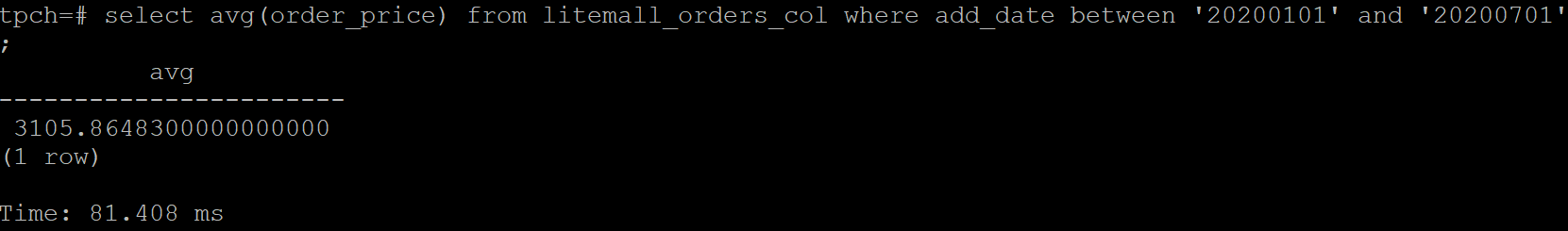


2. 2020年上半年litemall\_orders行存表与litemall\_orders\_col列存表中的order\_price的平均值查询，并对比执行效率截图

select avg (order\_price) from litemall\_orders where add\_date between '20200101' and '20200701';

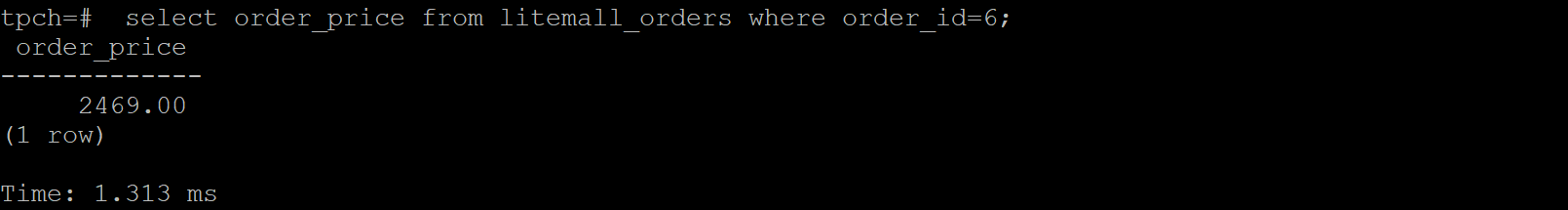


select avg (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

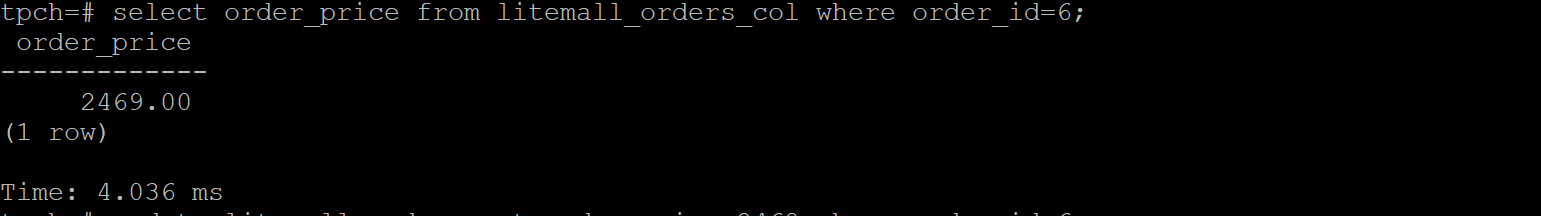


3. 查询litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price的值，并对比执行效率截图。

select order\_price from litemall\_orders where order\_id=6;

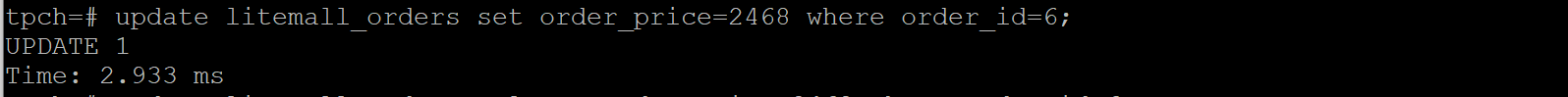


select order\_price from litemall\_orders\_col where order\_id=6;

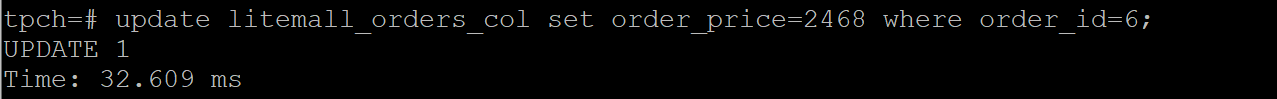


4. 将litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price修改为2468，并对比执行效率截图。

update litemall\_orders set order\_price=2468 where order\_id=6;



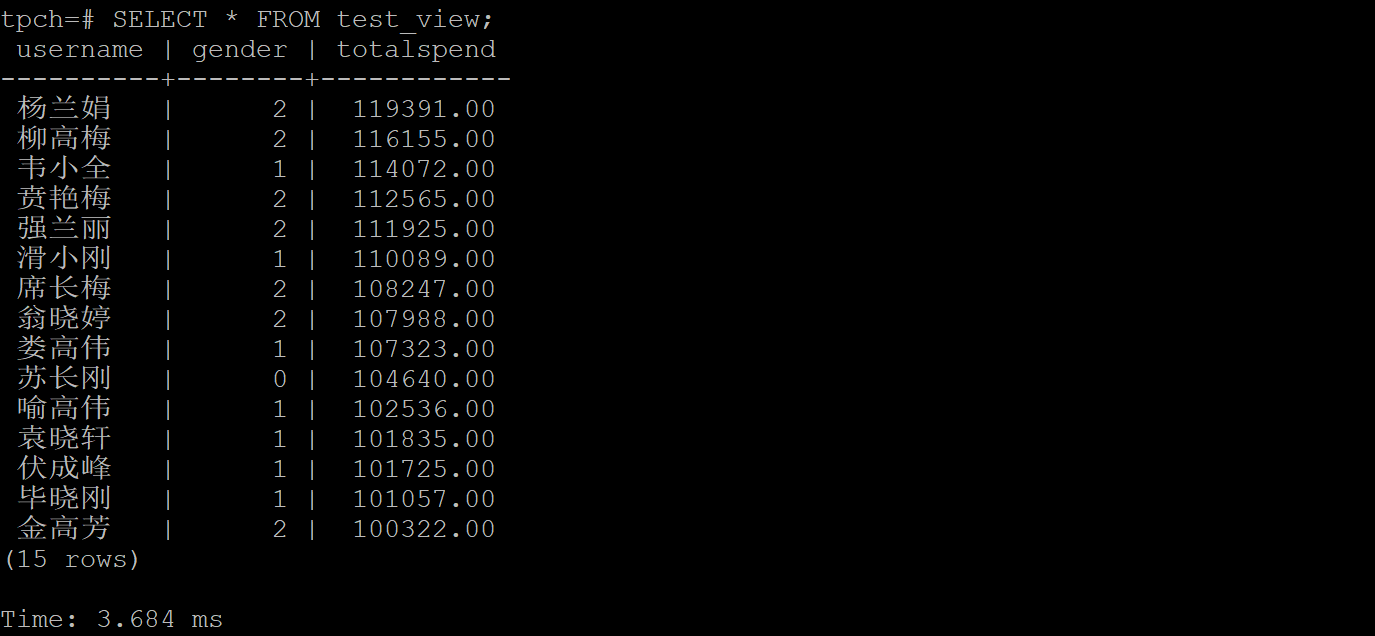
update litemall\_orders\_col set order\_price=2468 where order\_id=6;



任务三：物化视图的使用

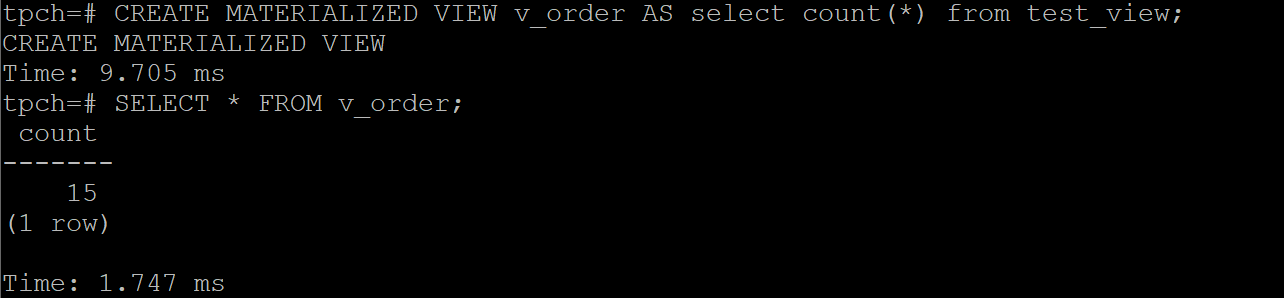
1. 创建物化视图所需要的表后，对表内容进行查询，对查询结果截图：

SELECT \* FROM test\_view;



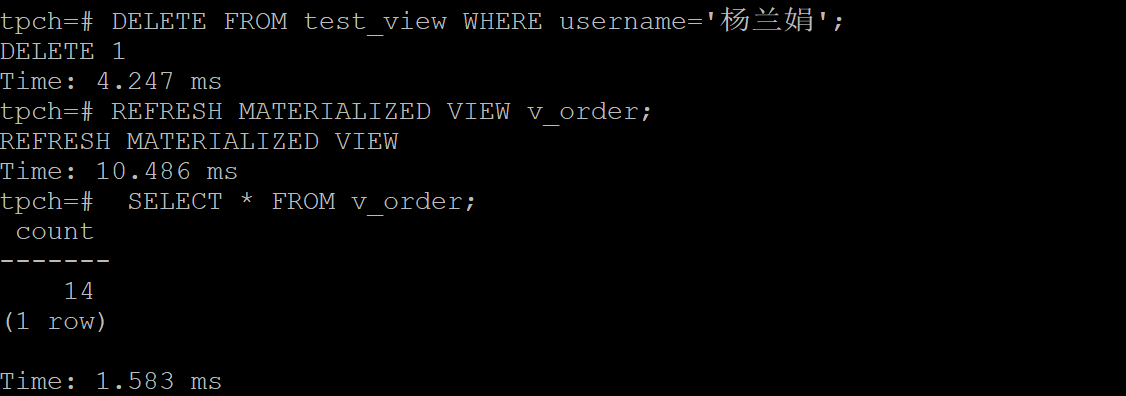
2. 使用物化视图统计人数，查询物化视图结果，将执行结果截图。

SELECT \* FROM v\_order;



3. 对表进行操作后，刷新物化视图，查询物化视图结果，将执行结果截图。

SELECT \* FROM v\_order;



4. 创建增量物化视图，查询物化视图结果，将执行结果截图。

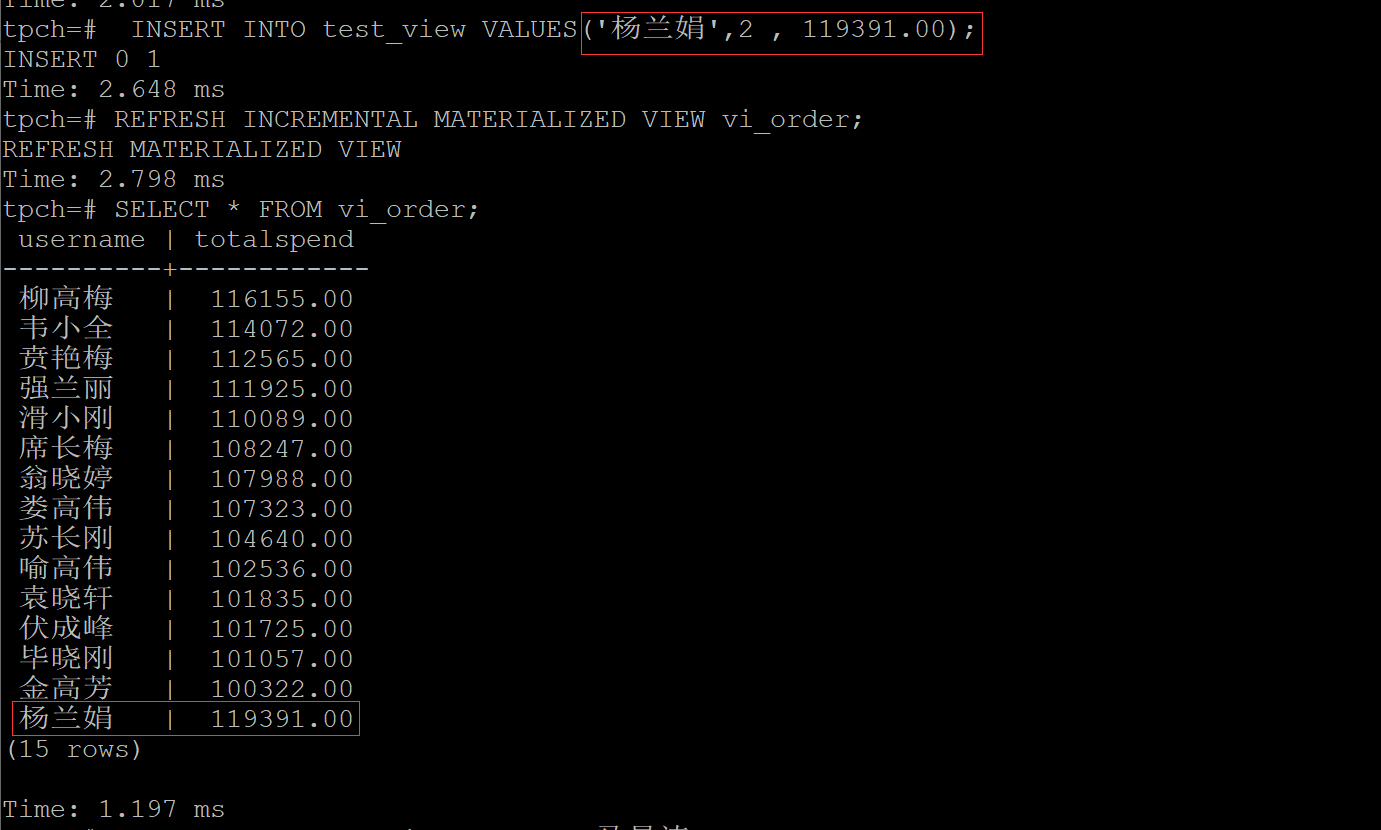
SELECT \* FROM vi\_order;



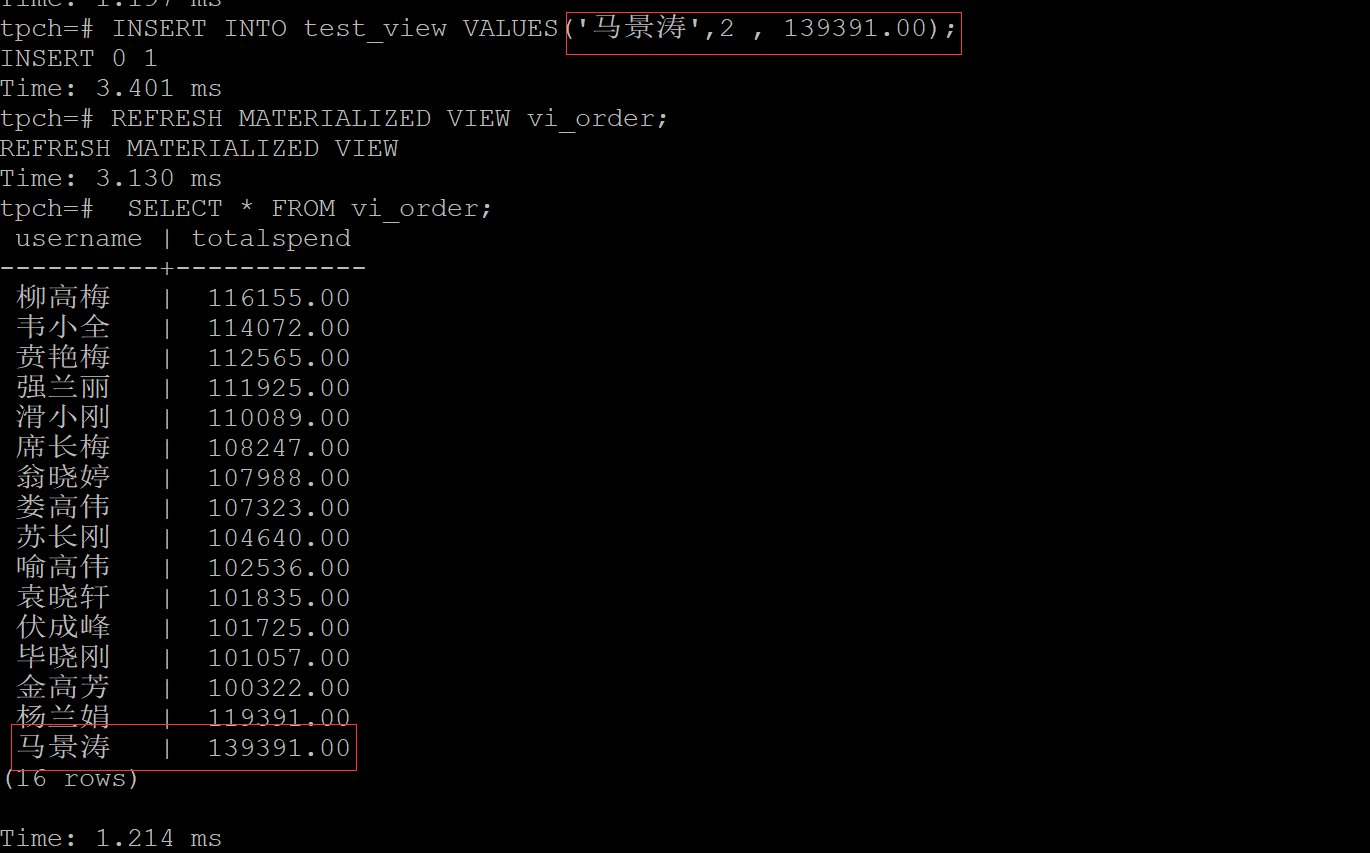
5. 对表进行操作后，刷新增量物化视图，查询物化视图结果，将执行结果截图。

SELECT \* FROM vi\_order;

插入杨兰娟



插入马景涛



实践思考题1：行存表与列存表在执行相同的SQL语句时，为何执行的时间不同？在执行哪些类型SQL时，行存表效率更高？在执行哪些类型SQL时，列存表效率更高？

行存储的写入是一次性完成，消耗的时间比列存储少。但行存储通常将一行数据完全读出，如果只需要其中几列数据，就会存在冗余列。而列存储每次读取的数据是集合中的一段或者全部。

由于列储存的数据是同质的，这种情况使数据解析变得容易。 行存储则复杂的多，因为在一行记录中保存了多种类型的数据，数据解析需要在多种数据类型之间频繁转换，这个操作很消耗cpu 所以列存储的解析过程中更有利于分析大数据。

行存表效率更高：随机的增删改查操作，需要在行中选取所有属性的查询操作，需要频繁插入或更新的操作，其操作与索引和行的大小更为相关。

列存表效率更高：所选择查询的目标即使只涉及少数几项属性，只访问查询涉及的列。

实践思考题2：全量物化视图与增量物化视图有哪些差别？

全量物化视图仅支持对创建好的物化视图做全量更新，而不支持做增量更新。创建全量物化视图语法和CREATE TABLE AS语法一致，不支持对全量物化视图指定NodeGroup创建。

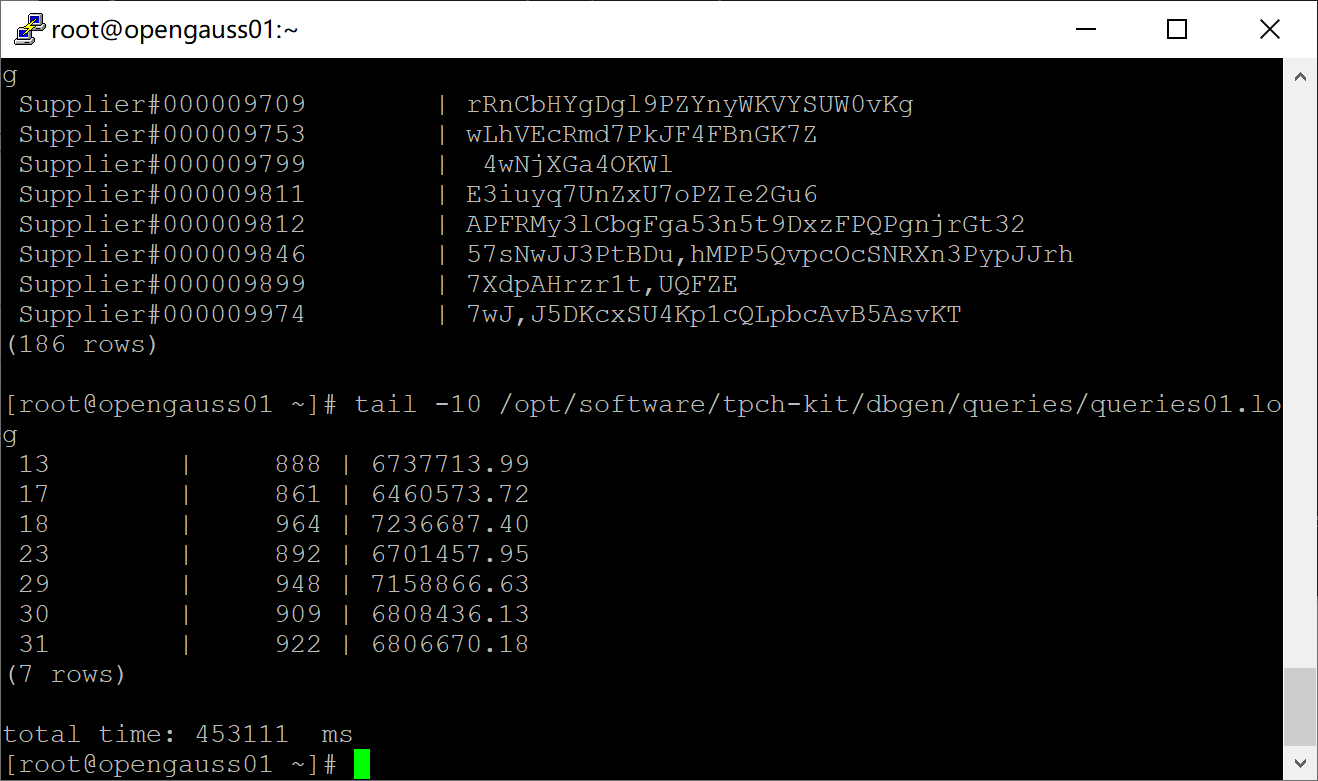
增量物化视图可以对物化视图增量刷新，需要用户手动执行语句完成对物化视图在一段时间内的增量数据刷新。与全量创建物化视图的不同在于目前增量物化视图所支持场景较小。目前物化视图创建语句仅支持基表扫描语句或者UNION ALL语句。

# 关卡三、openGauss的AI4DB特性应用

任务一：使用X-Tuner进行参数优化

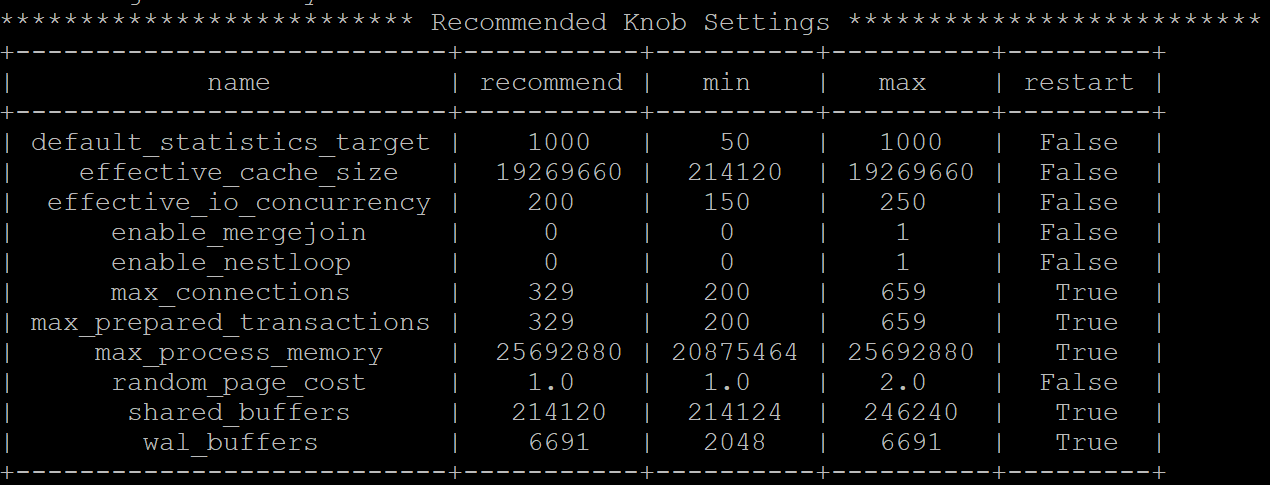
1. 执行TPCH脚本，获得测试时间，将执行结果截图：

gsql -d tpch -p 5432 -r -f /opt/software/tpch-kit/dbgen/queries/queries.sql > /opt/software/tpch-kit/dbgen/queries/queries01.log



2. 使用root用户，执行X-Tuner进行参数建议优化，将执行结果截图

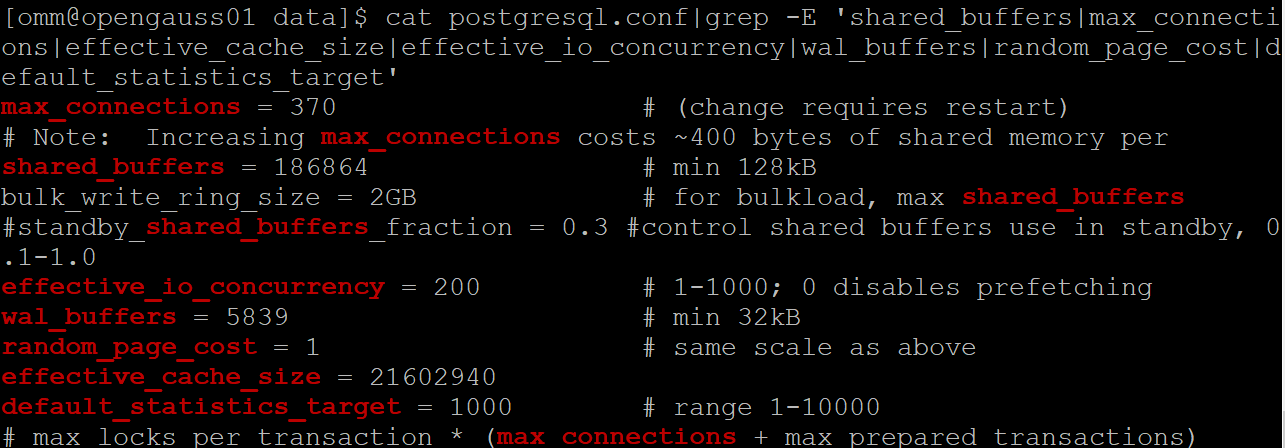
gs\_xtuner recommend --db-name tpch --db-user omm --port 5432 --host 127.0.0.1 --host-user omm



3.重启完成后，获取参数值：

cd /opt/software/openGauss/data

cat postgresql.conf|grep -E 'shared\_buffers|max\_connections|effective\_cache\_size|effective\_io\_concurrency|wal\_buffers|random\_page\_cost|default\_statistics\_target'



任务二：使用Index-advisor对select 查询语句进行优化，并通过对比执行计划，得到优化前后的不同。

1. 使用explain，对查询2020年3月订单表收入并进行排序的SQL加以分析，将结果截图。

EXPLAIN

SELECT ad.province AS province, SUM(o.actual\_price) AS GMV

FROM litemall\_orders o,

address\_dimension ad,

date\_dimension dd

WHERE o.address\_key = ad.address\_key

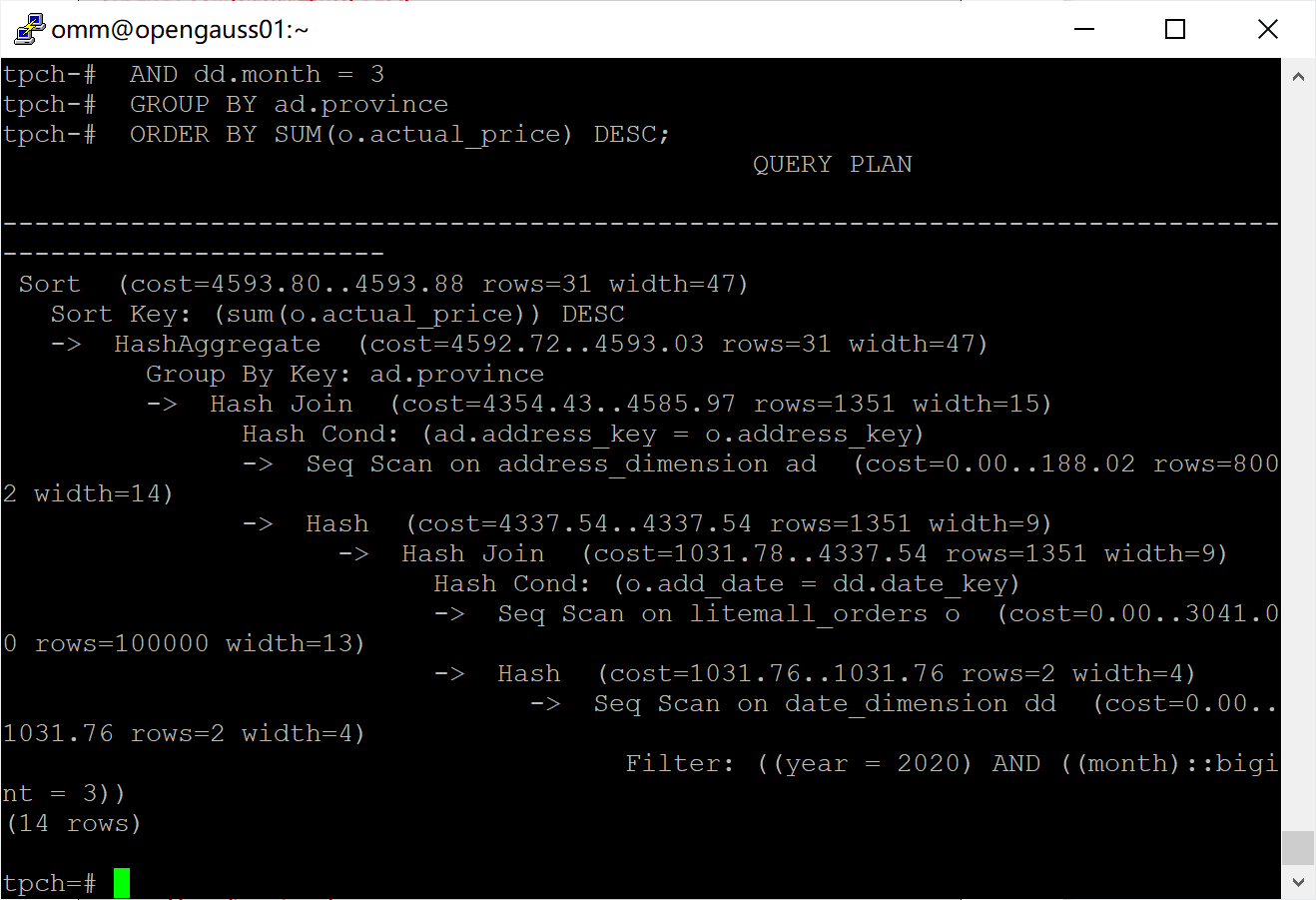
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

GROUP BY ad.province

ORDER BY SUM(o.actual\_price) DESC;



2. 使用索引推荐功能，对查询语句进行推荐，将执行结果截图。

select \* from gs\_index\_advise('

SELECT ad.province AS province, SUM(o.actual\_price) AS GMV

FROM litemall\_orders o,

address\_dimension ad,

date\_dimension dd

WHERE o.address\_key = ad.address\_key

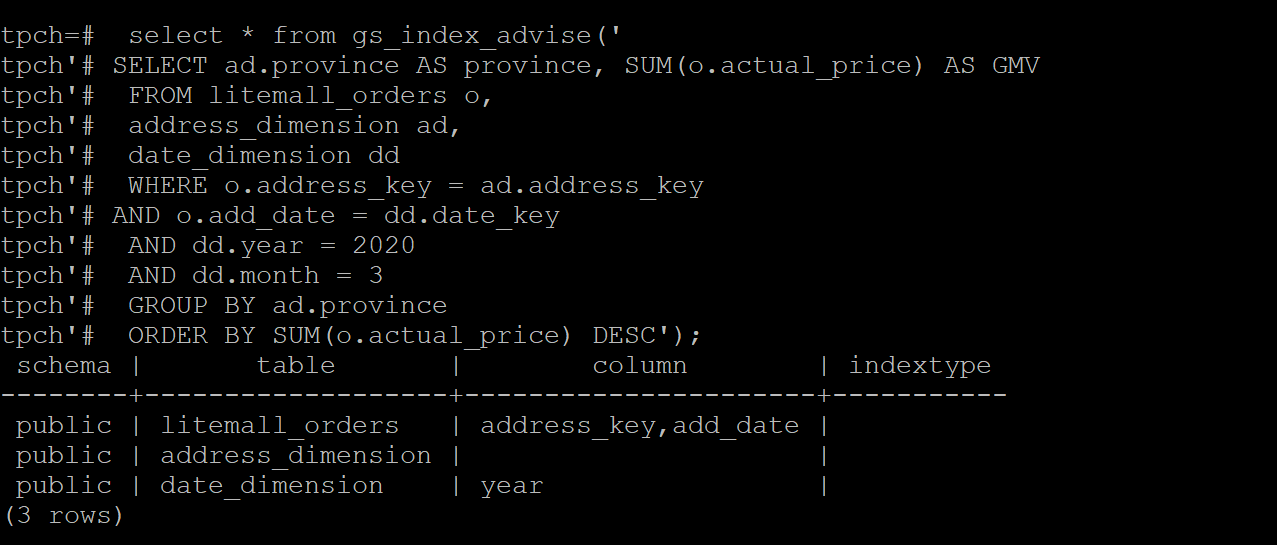
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

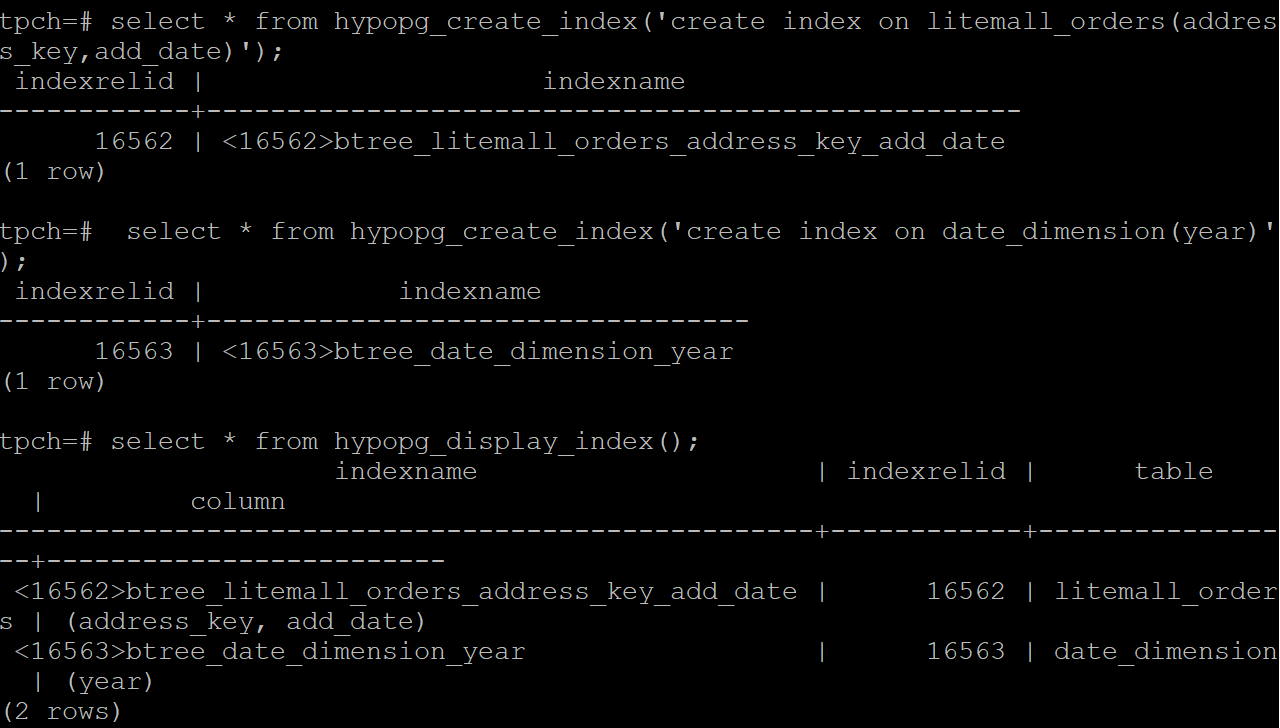
GROUP BY ad.province

ORDER BY SUM(o.actual\_price) DESC');



3. 查看创建的虚拟索引列，将执行结果截图。

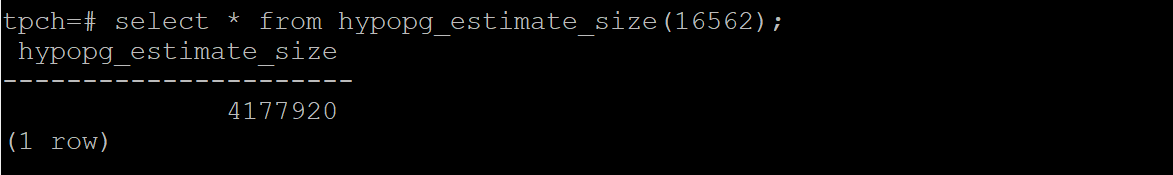
select \* from hypopg\_display\_index();

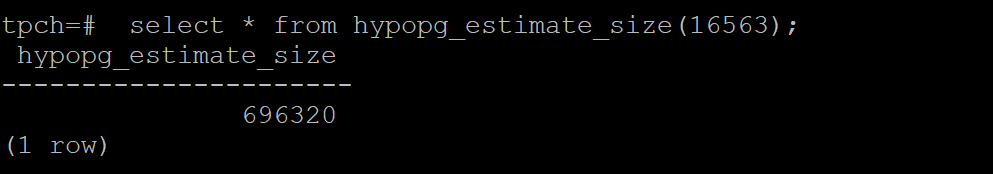


4. 获取索引虚拟列大小结果（单位为：字节），将执行结果截图。

select \* from hypopg\_estimate\_size(16715);

select \* from hypopg\_estimate\_size(16716);





5.再次使用explain，对该SQL加以分析，将执行结果截图。

EXPLAIN

SELECT ad.province AS province, SUM(o.actual\_price) AS GMV

FROM litemall\_orders o,

address\_dimension ad,

date\_dimension dd

WHERE o.address\_key = ad.address\_key

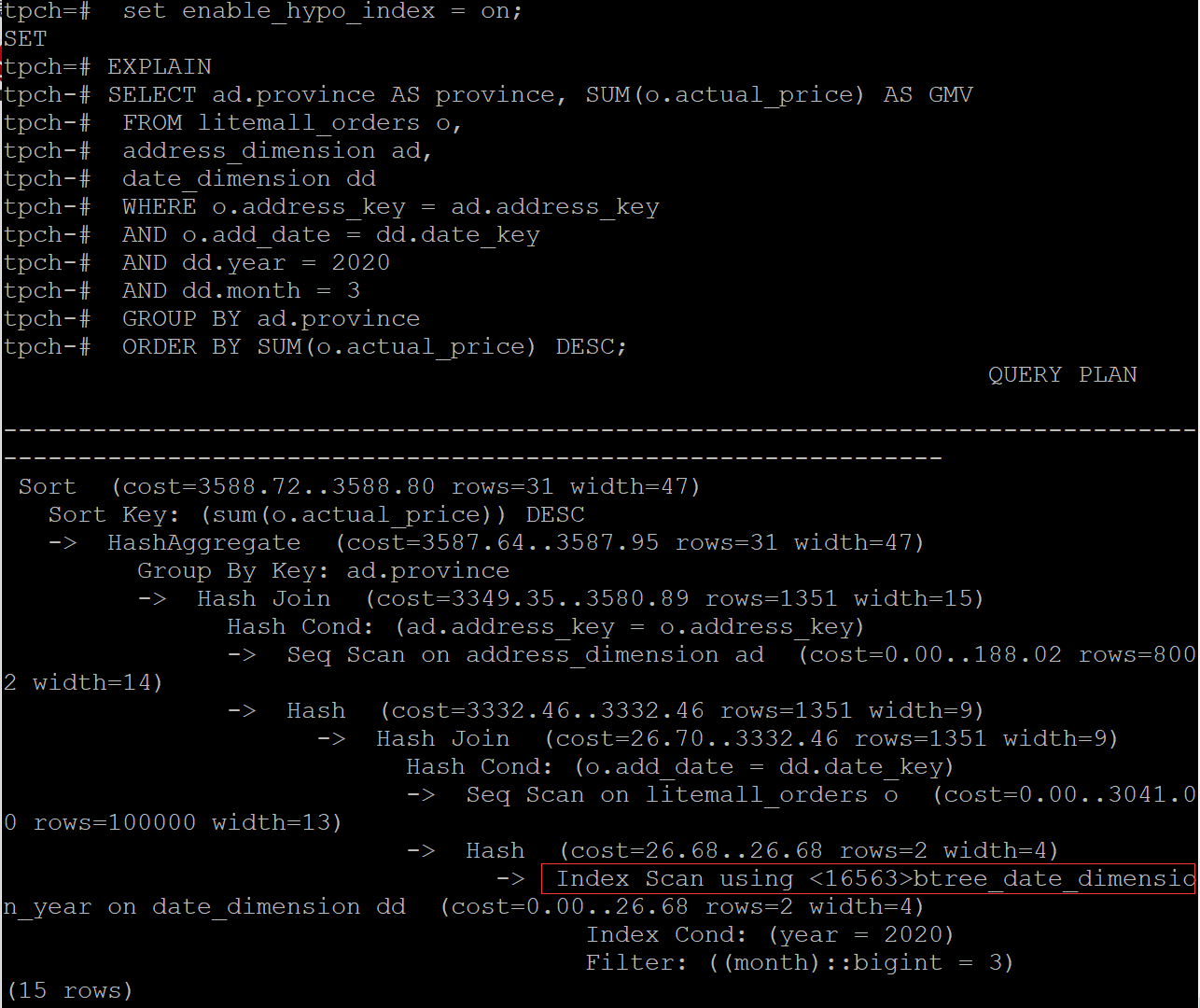
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

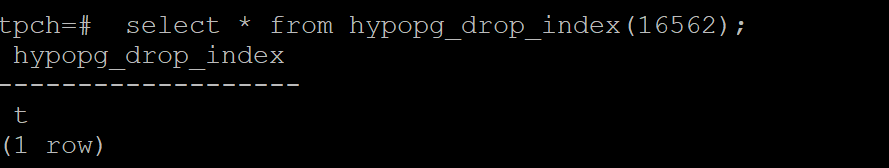
GROUP BY ad.province

ORDER BY SUM(o.actual\_price) DESC;



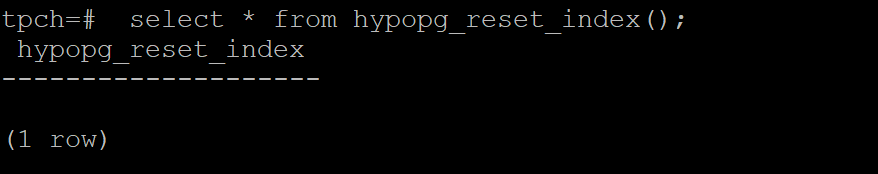
6. 删除某一个索引虚拟列，将执行结果截图。

select \* from hypopg\_drop\_index(16715);



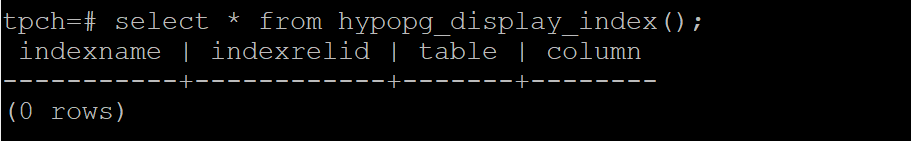
7. 删除所有（某一个）索引虚拟列，将执行结果截图。

select \* from hypopg\_reset\_index();



8. 查看索引虚拟列，将执行结果截图。

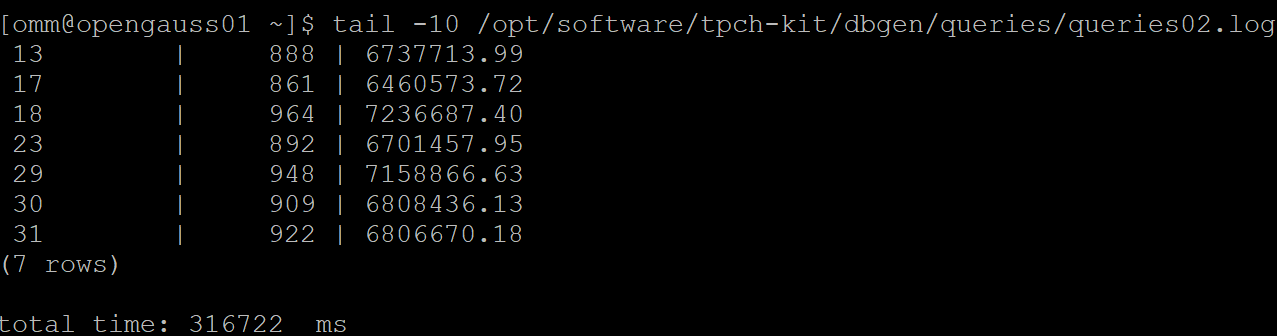
select \* from hypopg\_display\_index();



任务三：通过创建索引，对queries.sql中的SQL语句进行优化，并对比优化前后queries.sql执行的时间。

1. 重新执行queries.sql查询，将执行结果截图：

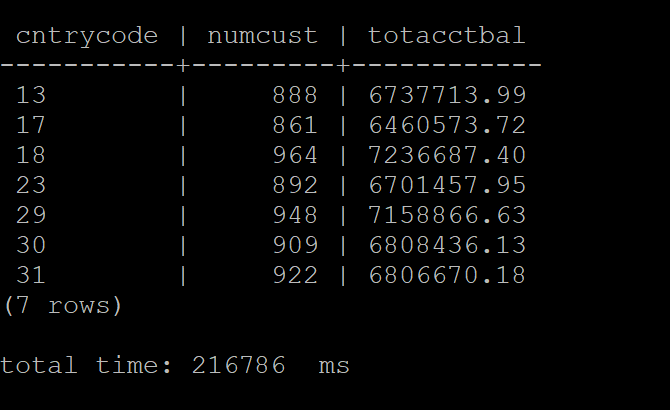
gsql -d tpch -p 5432 -r -f /opt/software/tpch-kit/dbgen/queries/queries.sql > /opt/software/tpch-kit/dbgen/queries/queries02.log



挑战一：进一步优化queries.sql中的查询语句，使得前后执行时间出现倍数级的提升。

1. 重新执行queries.sql查询，将执行结果截图：

gsql -d tpch -p 5432 -r -f /opt/software/tpch-kit/dbgen/queries/queries.sql > /opt/software/tpch-kit/dbgen/queries/queries03.log



实践思考题1：根据X-Tuner给出的参数优化，在哪些参数上进行了优化，为何要对这些参数进行优化？

优化的参数有：shared\_buffers ；max\_connections ；effective\_cache\_size ；effective\_io\_concurrency；wal\_buffers ；random\_page\_cost；default\_statistics\_target

原因：

shared\_buffers（共享缓冲区）：调整该参数的值可以影响内存缓存的大小，进而影响磁盘I/O的频率。合理的共享缓冲区设置可以提高数据库的性能，减少磁盘访问。

max\_connections（最大连接数）：该参数限制了同时与数据库建立的最大连接数。根据系统负载和可用资源，适当设置最大连接数可以避免资源竞争和性能下降。

effective\_cache\_size（有效缓存大小）：这个参数用于指定整个数据库系统可用的缓存大小，对于查询优化器的成本估算很重要。设置适当的值可以提高查询性能，避免频繁的磁盘访问。

effective\_io\_concurrency（有效I/O并发性）：该参数控制数据库并发读取和写入的并发级别。通过调整并发级别，可以更好地利用磁盘I/O能力，提高数据库的并发性能。

wal\_buffers（WAL缓冲区）：WAL是用于数据持久性和故障恢复的机制。调整WAL缓冲区的大小可以影响日志写入性能和数据恢复速度。

实践思考题2：索引的使用，对于执行SQL有什么好处？除了使用索引和参数外，还有哪些方面可以对数据库进行优化？

索引的使用对执行SQL查询有以下好处：

加速数据检索：索引可以提供快速的数据查找，避免全表扫描，从而加速查询操作。

减少磁盘I/O：索引可以减少磁盘I/O操作，因为它们允许数据库引擎更快地定位和检索所需的数据。

优化查询计划：索引提供了统计信息，数据库优化器可以使用这些信息来生成更优化的查询计划，选择更高效的执行路径。

除了使用索引和参数进行优化之外，还可以在以下方面对数据库进行优化：

查询优化：编写高效的查询语句、避免不必要的连接和子查询、使用适当的JOIN类型、合理使用查询限制等。

数据库设计：设计良好的数据库模式和表结构，避免冗余和不必要的数据，规范化和优化数据模型。

缓存和缓存策略：使用适当的缓存机制，如应用程序级缓存或数据库级缓存，以减少对数据库的频繁访问。

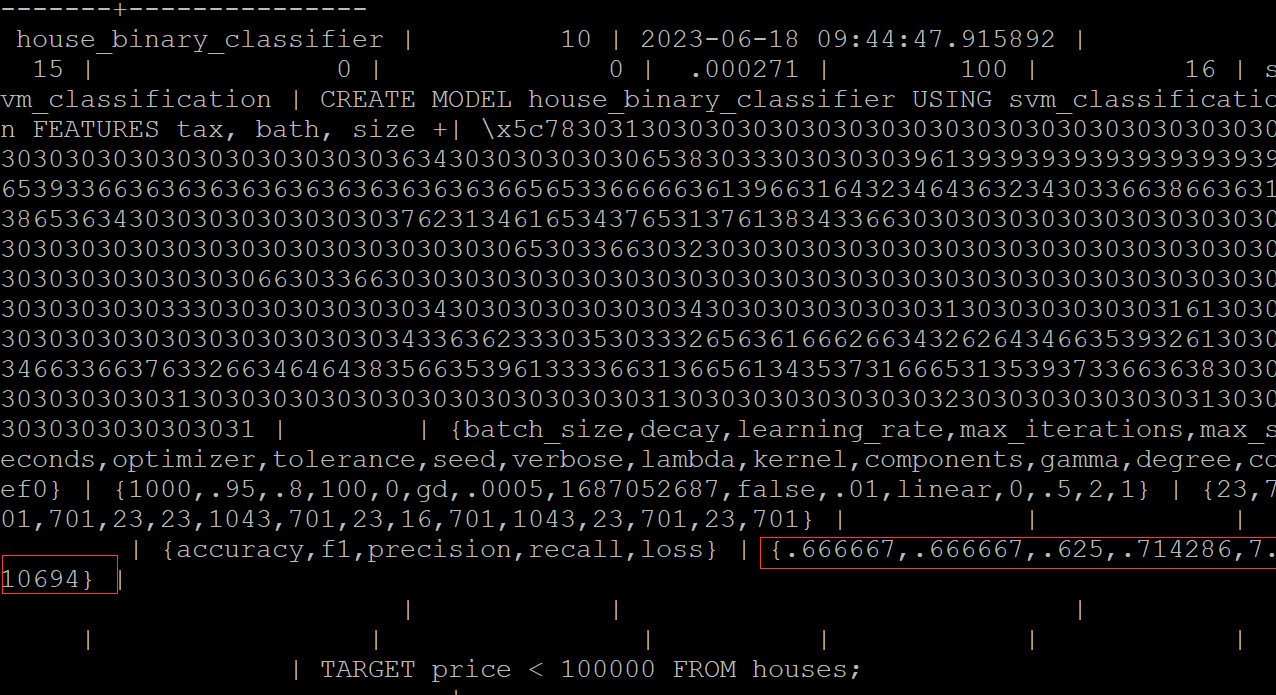
硬件升级：根据数据库工作负载的需求，考虑升级硬件组件，如CPU、内存和存储设备，以提供更好的性能和扩展性。

定期维护和监控：执行数据库的定期维护任务，如索引重建、统计信息更新、日志清理等，并使用监控工具跟踪数据库性能并进行调优。

# 关卡四、openGauss的DB4AI特性应用

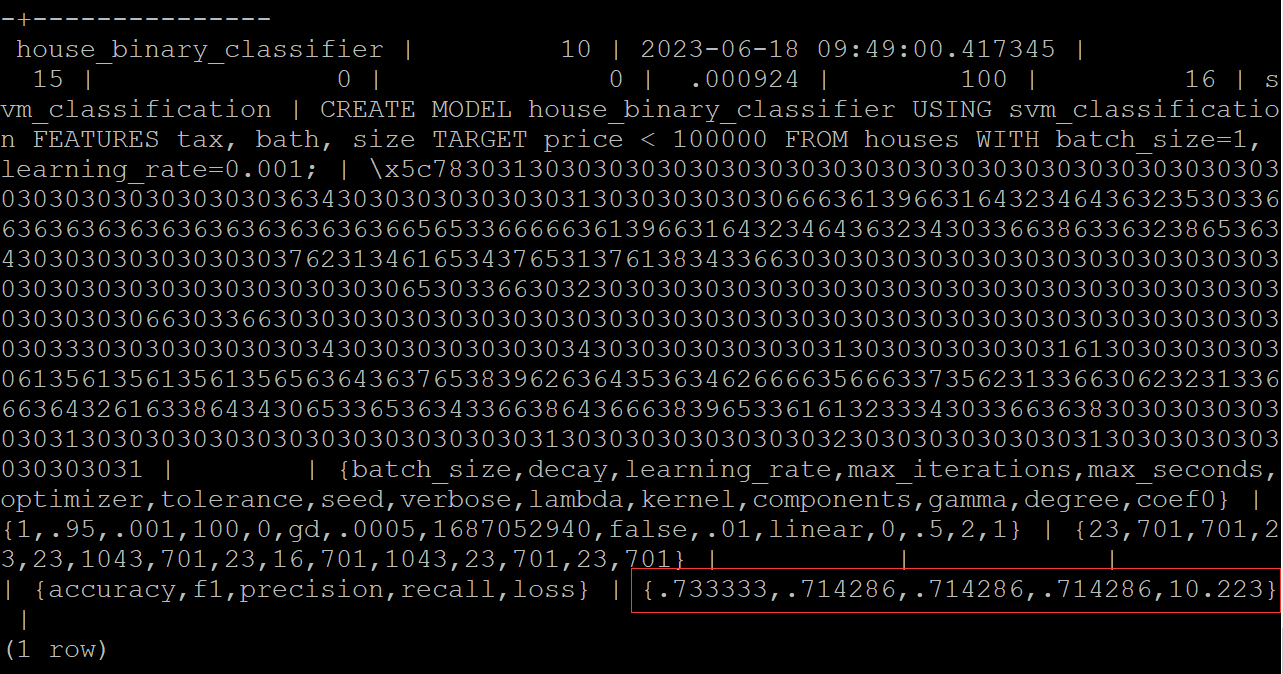
任务一：在gs\_model\_warehouse系统表中查看训练后的模型信息，将执行结果截图：

postgres=# SELECT \* FROM gs\_model\_warehouse WHERE modelname = 'house\_binary\_classifier';



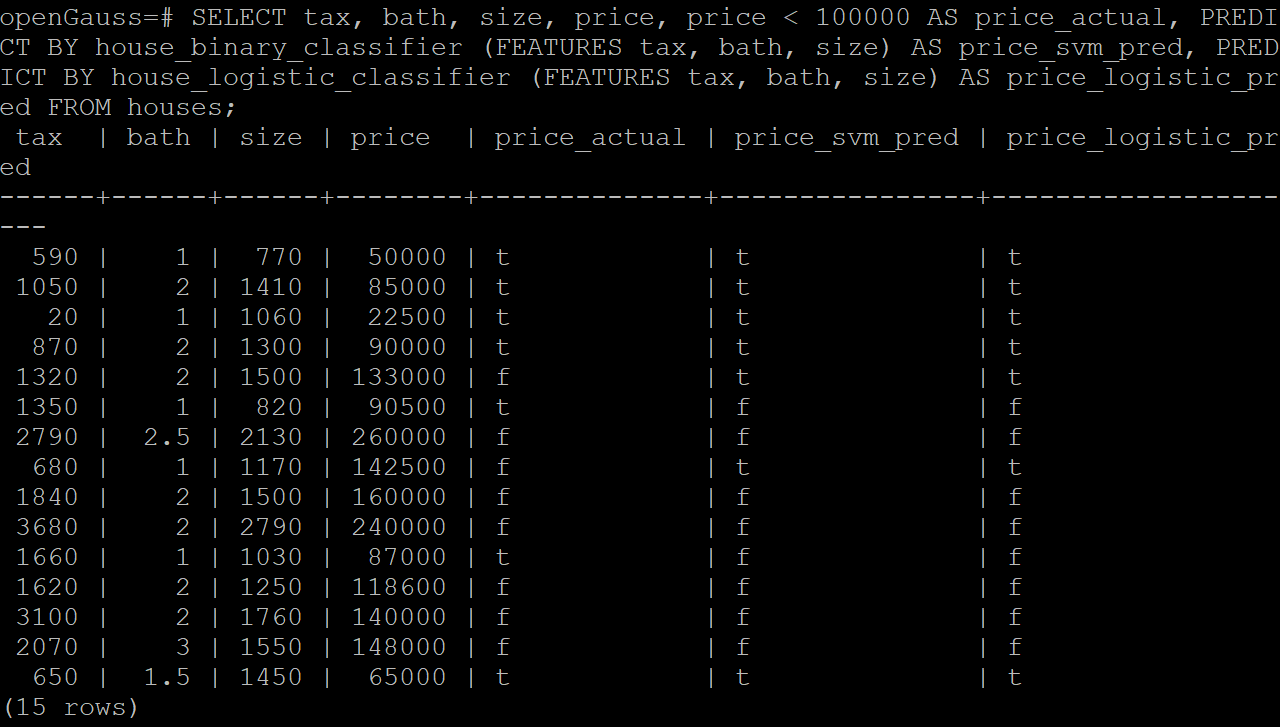
任务二：观察新模型的信息，将执行结果截图。

postgres=# SELECT \* FROM gs\_model\_warehouse WHERE modelname = 'house\_binary\_classifier';



任务三：利用训练好的逻辑回归模型预测数据，并与SVM算法进行比较，将执行结果截图。

postgres=# SELECT tax, bath, size, price, price < 100000 AS price\_actual, PREDICT BY house\_binary\_classifier (FEATURES tax, bath, size) AS price\_svm\_pred, PREDICT BY house\_logistic\_classifier (FEATURES tax, bath, size) AS price\_logistic\_pred FROM houses;



实践思考题1：分类模型与回归模型有何不同？

分类模型用于预测离散型目标变量，该目标变量可以是预定义的类别或标签。分类模型的目标是将输入数据映射到预定义的类别中的一个。常见的分类算法包括逻辑回归、决策树、支持向量机、随机森林和神经网络。分类模型的输出通常是离散的类别或类别概率。

回归模型则用于预测连续型目标变量。回归模型的目标是通过对输入数据进行建模，预测出一个连续的数值输出。回归模型通过学习输入特征与目标变量之间的关系来进行预测。常见的回归算法包括线性回归、岭回归、Lasso回归、决策树回归和支持向量回归。回归模型的输出是一个连续的数值。

实践思考题2：什么是SVM算法？

SVM，即支持向量机，是一种用于分类和回归分析的监督学习算法。

其基本思想是找到一个最优的超平面（在二维空间中是一条直线，而在高维空间中是一个超平面），将不同类别的样本分开。超平面的选择是通过最大化样本间的间隔来实现的，这个间隔被称为“最大间隔”。间隔是指从超平面到最靠近它的样本点的距离。

SVM算法的核心是构建一个最优的超平面。它将训练数据映射到高维空间，然后在该空间中找到一个能够最好地将不同类别样本分开的超平面。这个超平面被称为分隔超平面，而位于超平面上的样本点被称为支持向量。支持向量是决定超平面位置的关键样本点。

实践思考题3：分类问题有哪些评价指标，请分别说明他们的含义？

分类问题的评价指标有：准确率accuracy, F1 分数F1 Score, 精确率precision, 召回率recall,损失loss。

准确率：表示分类模型正确预测的样本数量与总样本数量之间的比例。准确率越高，模型的性能越好。但准确率在处理不平衡数据集时可能会受到误导。

F1 分数：综合考虑了精确率（Precision）和召回率（Recall）的指标。F1 分数是这两个指标的调和平均值，它对精确率和召回率的平衡给予了更高的权重。F1 分数适用于数据不平衡或关注误分类的情况。

精确率：表示模型预测为正类别的样本中真正为正类别的比例。它衡量了模型对正类别的分类准确程度。精确率高表示模型预测为正的样本很可能是真正的正样本，但它不考虑模型是否遗漏了一些真正的正样本。

召回率：也称为敏感度或真正例率，它表示正样本中被模型正确预测为正样本的比例。召回率衡量了模型对正类别的覆盖程度。较高的召回率意味着模型能够捕捉到更多的正样本，但可能会有更多的误分类。

损失：损失是一种衡量模型预测与真实值之间差异的指标。损失函数的选择取决于具体的问题和模型。在分类问题中，常见的损失函数包括交叉熵损失和对数损失等。通常，较低的损失值表示模型更好地拟合了训练数据。

实践思考题4：回归问题有哪些评价指标，请分别说明他们的含义？

回归问题的评价指标有：均方误差Mean Squared Error，均方根误差Root Mean Squared Error，平均绝对误差Mean Absolute Error，MAE，决定系数R² R-squared，相关系数Correlation Coefficient。

均方误差：计算预测值与真实值之间的平均平方差。均方误差越小表示模型的拟合效果越好。然而，均方误差对异常值敏感，因为它会放大异常值的影响。

均方根误差：均方根误差是均方误差的平方根。同样衡量了预测值与真实值之间的平均误差，但它的单位与目标变量的单位相同，更直观地反映了误差的大小。

平均绝对误差：平均绝对误差计算预测值与真实值之间的平均绝对差。MAE度量了预测值与真实值之间的平均误差大小，它对异常值不敏感。然而，MAE没有考虑误差的平方，因此可能无法捕捉到误差的实际波动情况。

决定系数R²：用于衡量模型对因变量变异性的解释程度。它表示因变量的变异性中有多少百分比可以由模型所解释，取值范围为0到1。R²越接近1表示模型对因变量的解释能力越好，越接近0表示模型的解释能力较弱。

相关系数：相关系数衡量了预测值与真实值之间的线性相关性。常用的相关系数是皮尔逊相关系数，取值范围为-1到1。相关系数为1表示完全正相关，为-1表示完全负相关，为0表示无相关性。