openGauss AI特性创新实践课



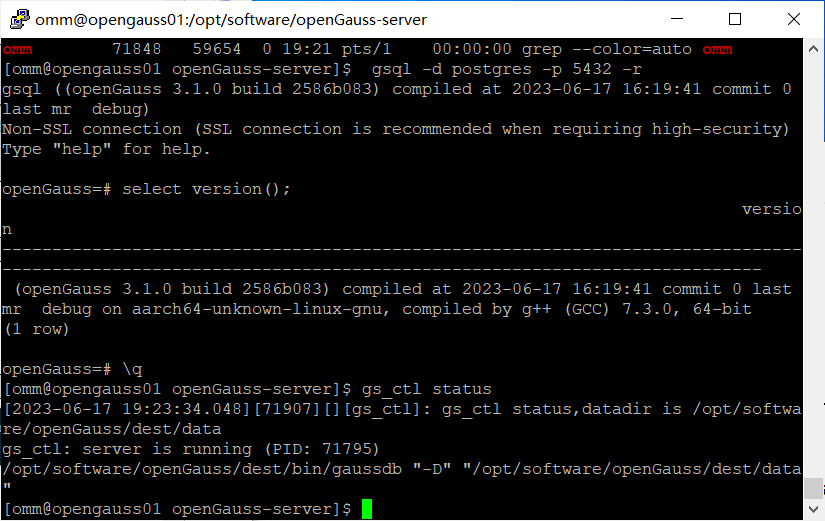
华为技术有限公司

# 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：

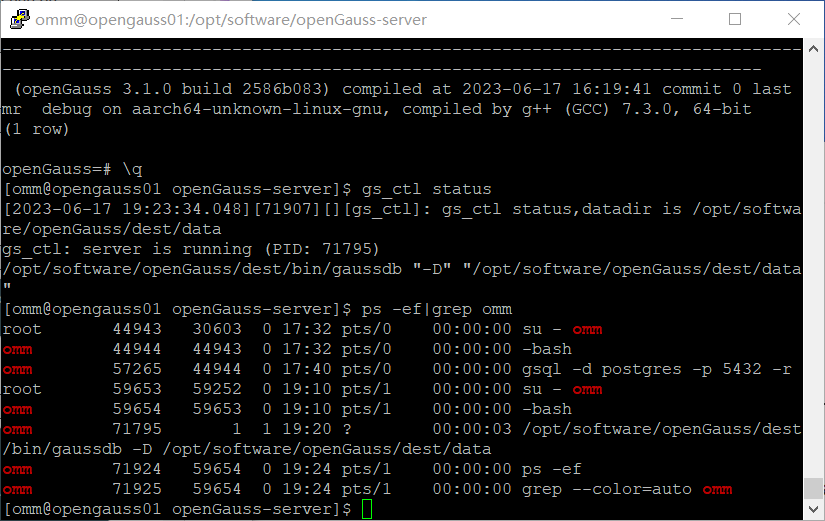
任务一：数据库状态验证

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）



实验思考题：为什么需要通过源码编译，安装数据库？

通过源码编译和安装数据库有以下几个主要原因：

1. 定制化需求：通过源码编译和安装数据库，可以根据自身的需求进行定制化配置和编译。你可以选择需要的功能和模块，并根据自己的系统环境进行优化和调整，以获得最佳性能和功能。

2. 最新特性和改进：源码编译和安装数据库可以确保你使用的是最新版本的数据库软件。这意味着你可以享受到最新的特性、改进和修复的 bug，以提高数据库的性能、安全性和功能。

3. 平台适配性：通过源码编译，你可以确保数据库软件能够与你的操作系统和硬件平台完全兼容。这是特别重要的，因为不同的操作系统和硬件架构可能存在一些差异，源码编译可以确保数据库在特定平台上运行稳定和高效。

4. 安全性和可靠性：源码编译和安装数据库可以提供更高的安全性和可靠性。你可以自行检查代码、配置和编译选项，以确保数据库在安全性方面的要求得到满足。此外，通过源码编译可以确保数据库的可靠性和稳定性，因为你可以根据自己的需求进行优化和调整，以解决可能存在的性能或稳定性问题。

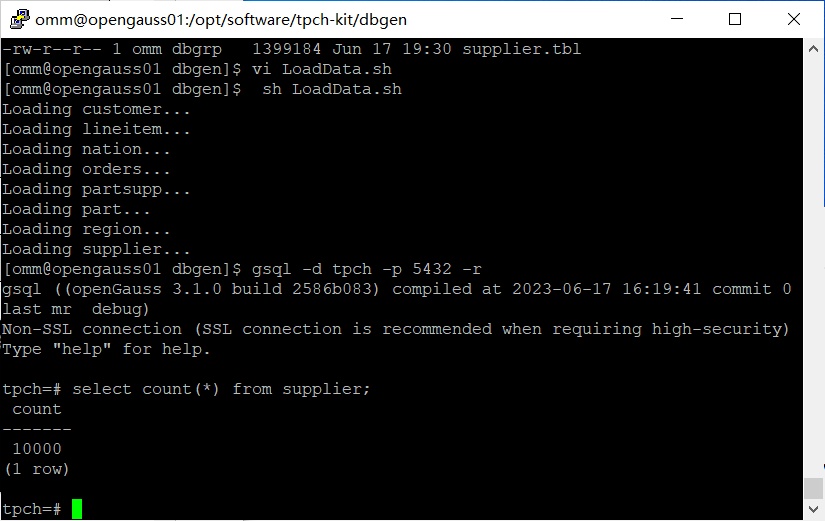
通过源码编译和安装数据库可以提供更大的灵活性、定制化能力、最新特性和改进、平台适配性、安全性和可靠性。这对于某些特定需求和高级用户来说是非常有价值的。然而，对于一般用户来说，预编译的数据库软件通常已经提供了足够的功能和性能，可以满足他们的需求。

# 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：

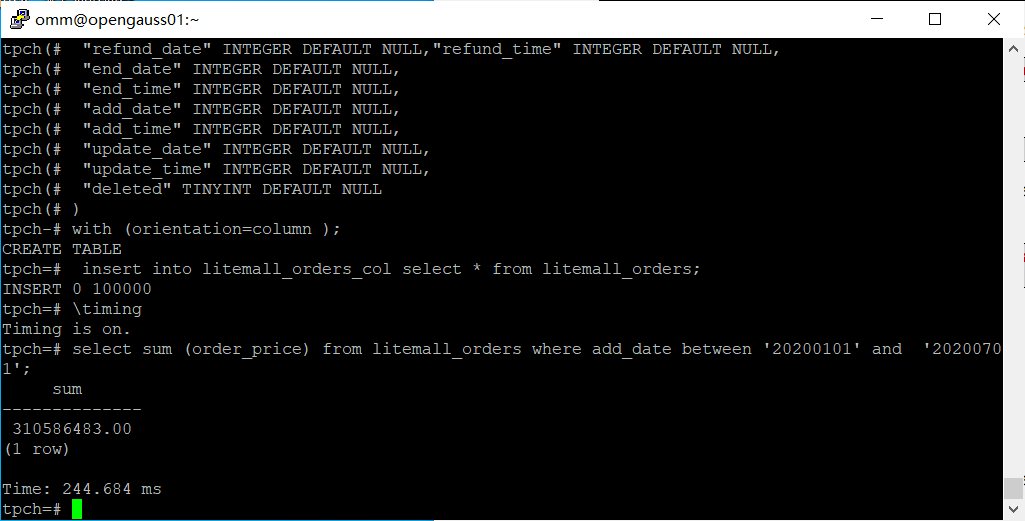
select count(\*) from supplier;;



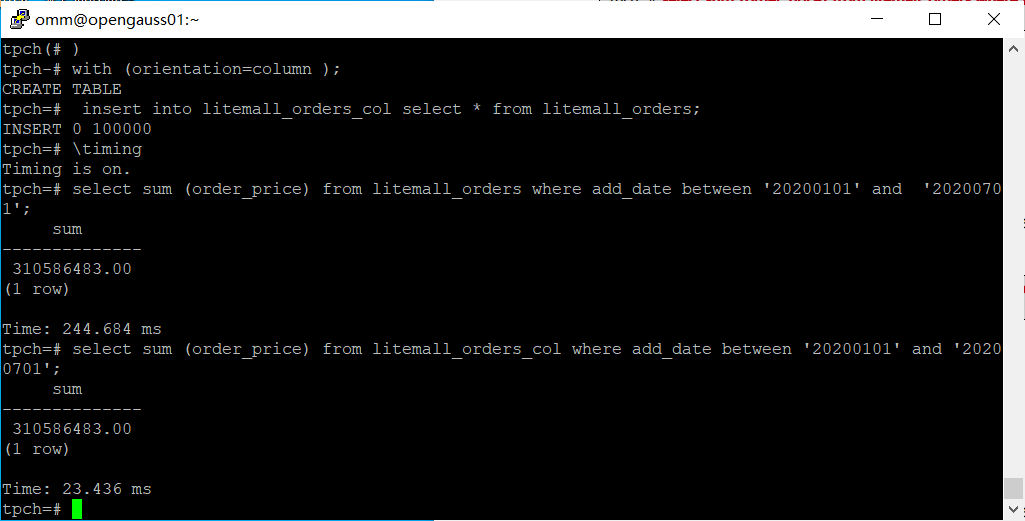
任务二：行存表与列存表执行效率对比

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';

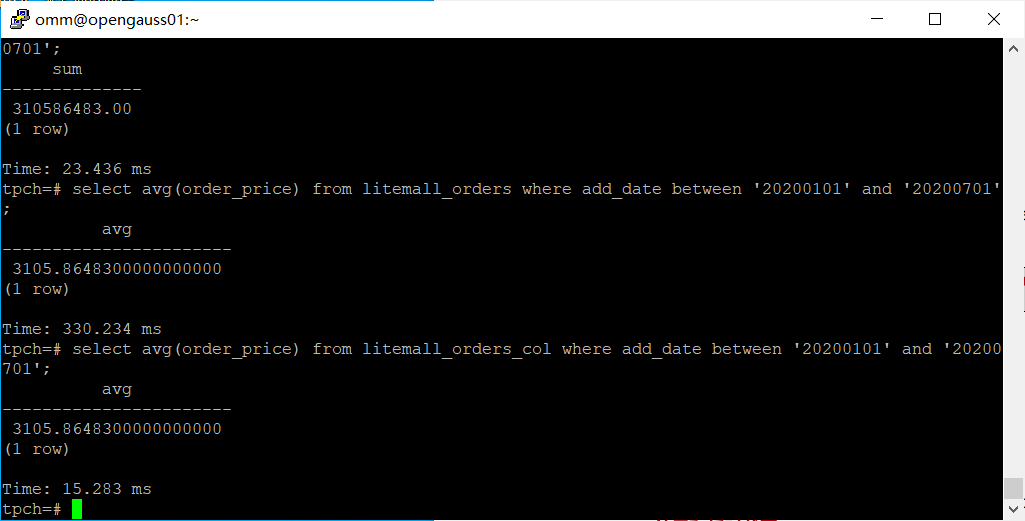


select sum (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

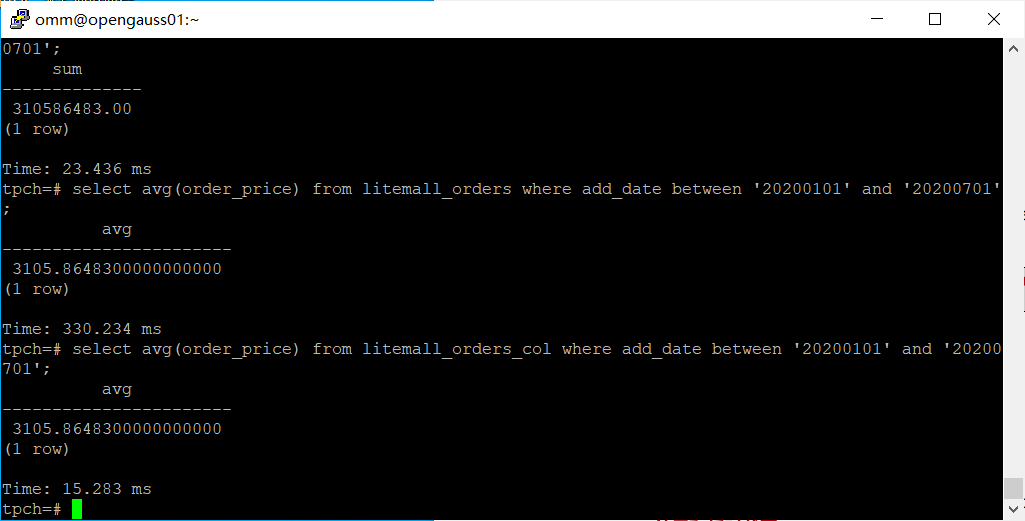


2. 2020年上半年litemall\_orders行存表与litemall\_orders\_col列存表中的order\_price的平均值查询，并对比执行效率截图

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

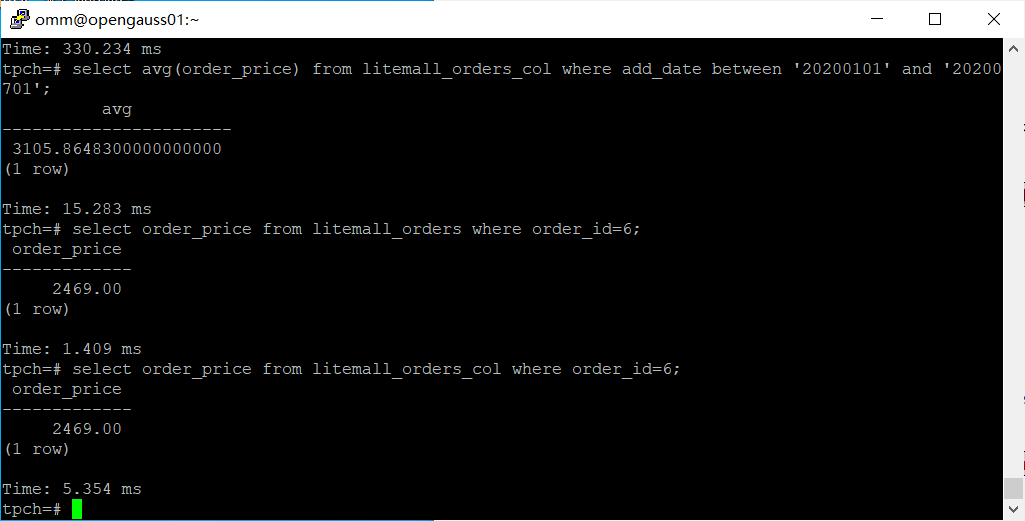


select avg (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

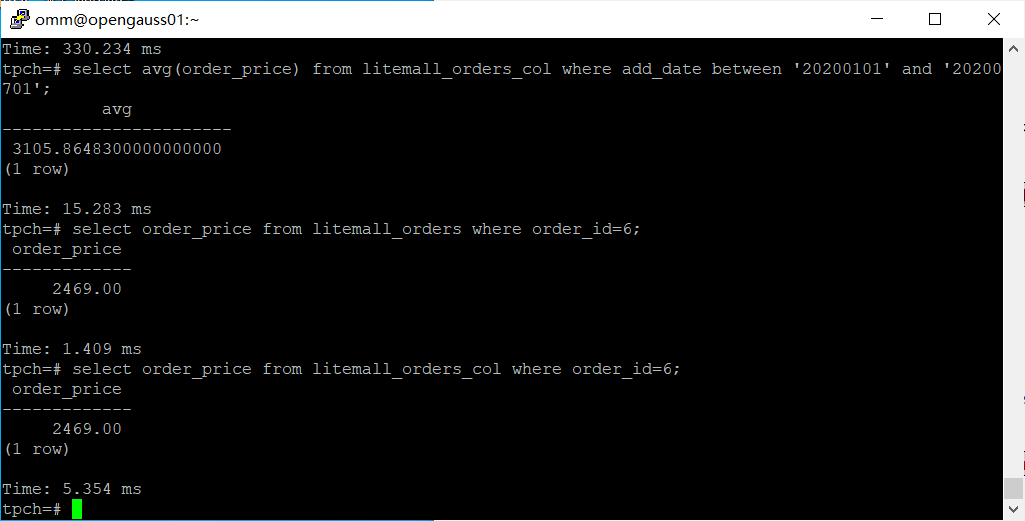


3. 查询litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price的值，并对比执行效率截图。

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

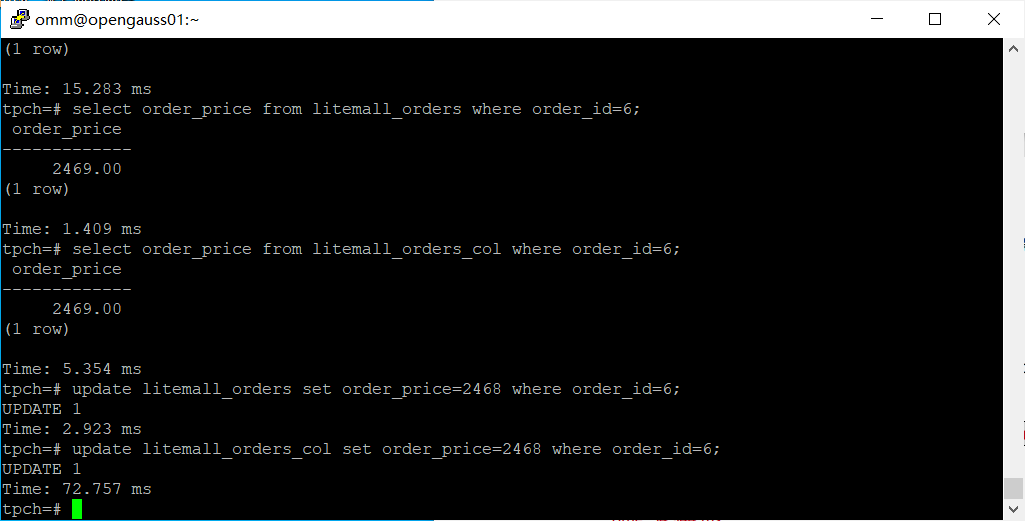


select order\_price from litemall\_orders\_col where order\_id=6;

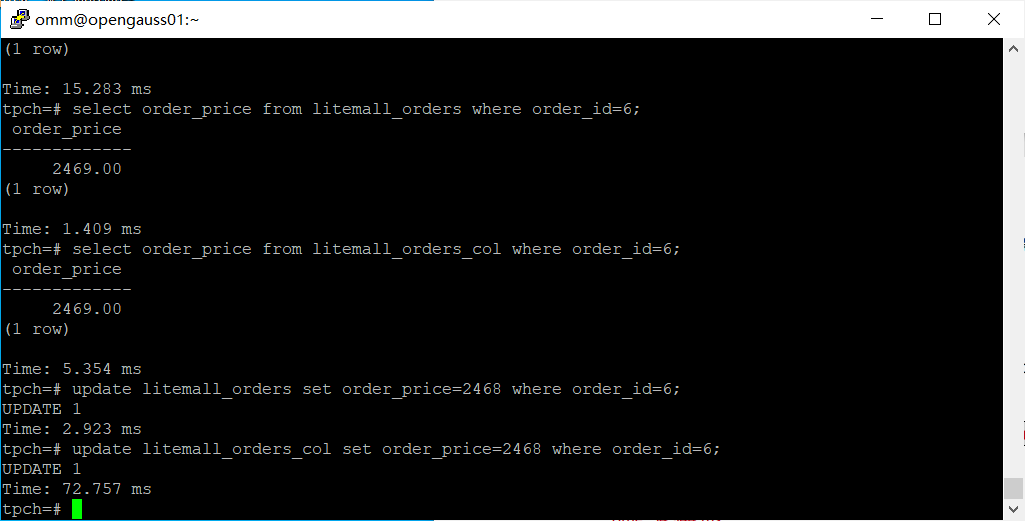


4. 将litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price修改为2468，并对比执行效率截图。

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



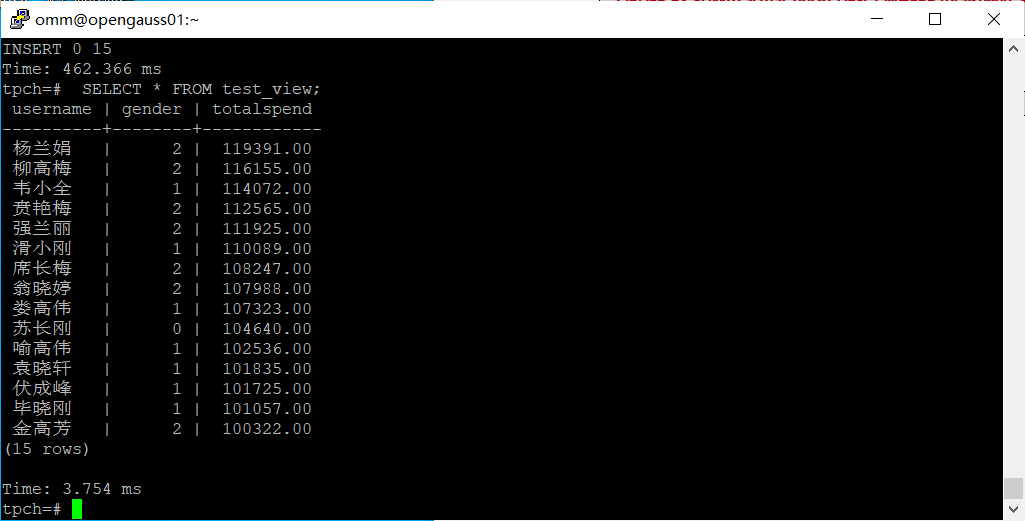
update litemall\_orders\_col set order\_price=2468 where order\_id=6;



任务三：物化视图的使用

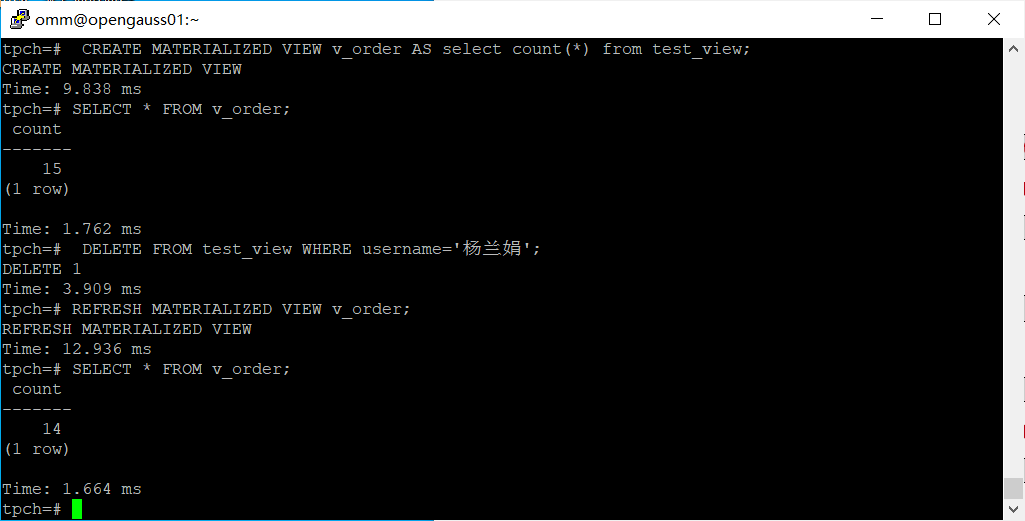
1. 创建物化视图所需要的表后，对表内容进行查询，对查询结果截图：

SELECT \* FROM test\_view;



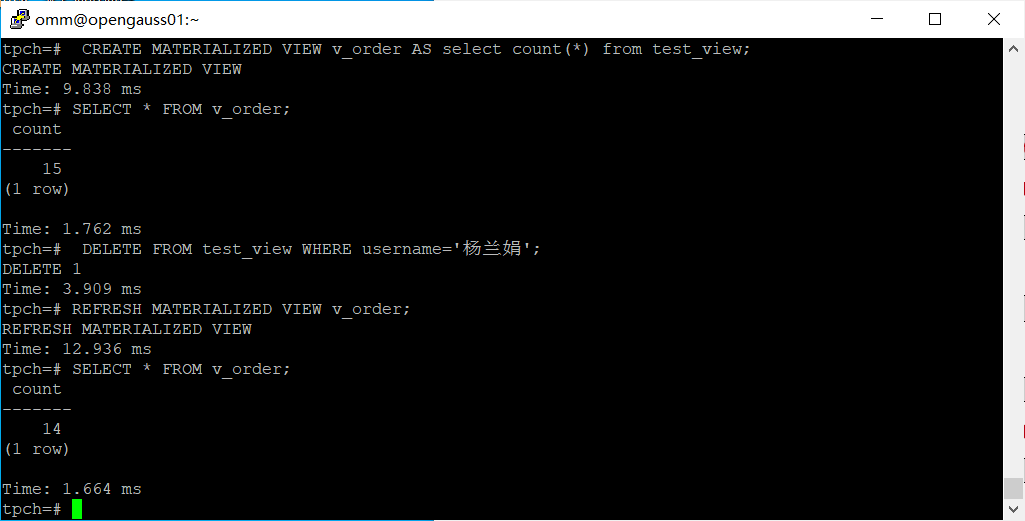
2. 使用物化视图统计人数，查询物化视图结果，将执行结果截图。

SELECT \* FROM v\_order;



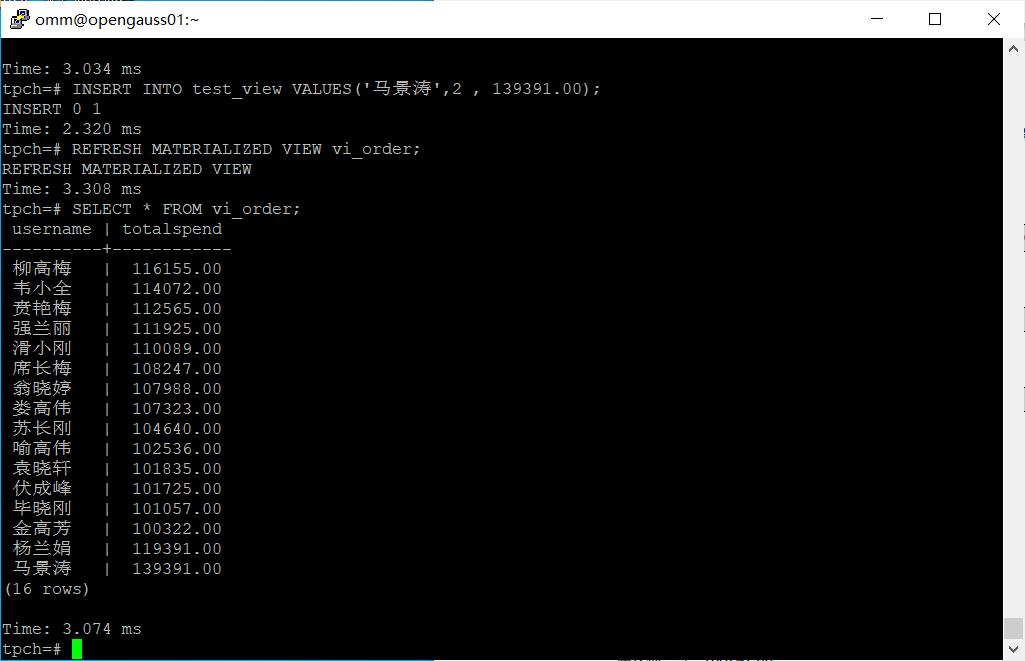
3. 对表进行操作后，刷新物化视图，查询物化视图结果，将执行结果截图。

SELECT \* FROM v\_order;



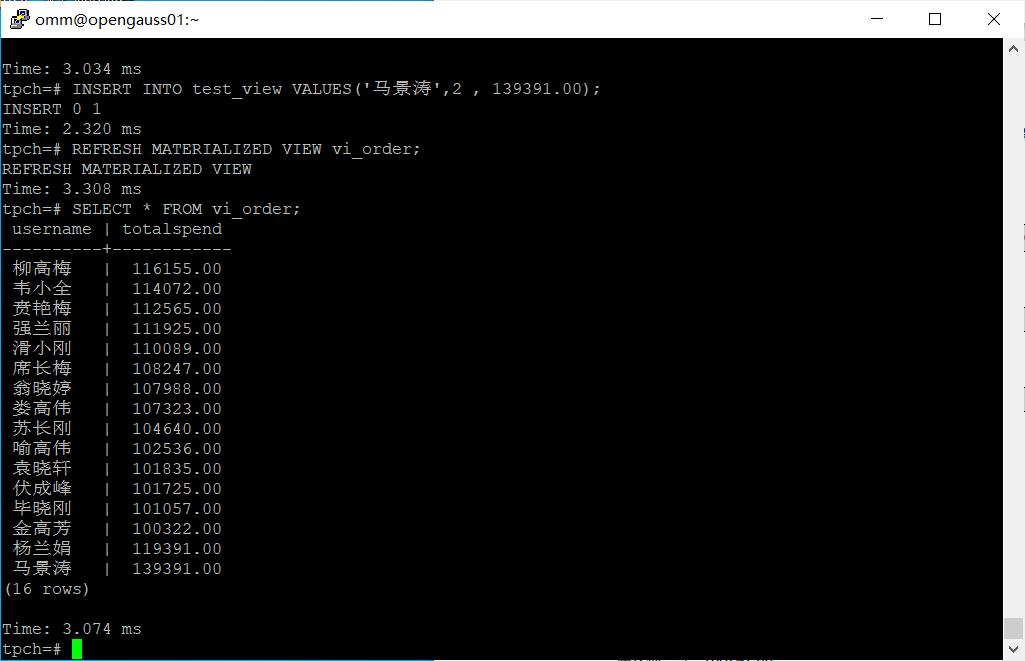
4. 创建增量物化视图，查询物化视图结果，将执行结果截图。

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;



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？

行存表将数据按行存储在磁盘上，每个行记录包含该行的所有字段。当执行涉及到单个或少量行的查询时，行存表可以快速访问所需的数据，因为它可以直接读取整行数据。

列存表将数据按列存储在磁盘上，每个列值被存储在一起。这种存储方式可以带来数据压缩和更高的查询性能。当执行需要处理大量列的查询时，列存表可以通过只读取所需的列数据而实现更高的效率。这在涉及聚合、分组和大数据量的查询时尤为明显。

行存表在执行以下类型的 SQL 时效率更高：

需要访问整个行记录的查询，例如按主键查询、返回所有字段的查询等。

需要频繁更新和插入数据的场景，因为行存表在更新和插入单个行时更高效。

列存表在执行以下类型的 SQL 时效率更高：

需要处理大量列数据的查询，例如聚合函数（SUM、AVG 等）、分组查询和数据分析等。

需要返回特定列数据的查询，而不需要读取整个行记录的情况下。

行存表和列存表的性能优劣还受多种因素的影响，例如数据量、查询复杂性、系统配置等。因此，在选择存储方式时，需要根据具体的业务需求和查询模式进行评估和测试。

实践思考题2：全量物化视图与增量物化视图有哪些差别？

全量物化视图和增量物化视图是在数据库中创建的两种类型的物化视图，它们之间有以下主要差异：

1. 数据更新方式：

- 全量物化视图：全量物化视图在刷新时重新计算和加载所有数据，无论数据是否发生变化。这意味着每次刷新都会执行完整的计算和加载过程，无论数据是否发生变化。这通常适用于静态或少量更新的场景。

- 增量物化视图：增量物化视图仅计算和加载发生变化的数据。它通过跟踪源表的变化来确定需要更新的数据，并将这些变化应用于物化视图。这种方式可以减少刷新过程的开销，并提供更高的性能。增量物化视图适用于大量数据更新和频繁变化的场景。

2. 刷新频率：

- 全量物化视图：全量物化视图的刷新频率通常较低，可以是定期或手动触发的。由于需要重新计算和加载所有数据，全量刷新可能需要较长时间。

- 增量物化视图：增量物化视图的刷新频率通常较高，可以在源表发生变化时自动触发刷新。增量刷新只需计算和加载变化的数据，因此刷新过程更快。

3. 存储和维护成本：

- 全量物化视图：全量物化视图需要存储完整的数据集，因此在存储方面需要更多的空间。此外，由于每次刷新都需要重新计算和加载所有数据，因此在维护方面可能需要更多的计算资源。

- 增量物化视图：增量物化视图只存储变化的数据，因此在存储方面需要较少的空间。此外，增量刷新只处理变化的数据，因此在维护方面需要较少的计算资源。

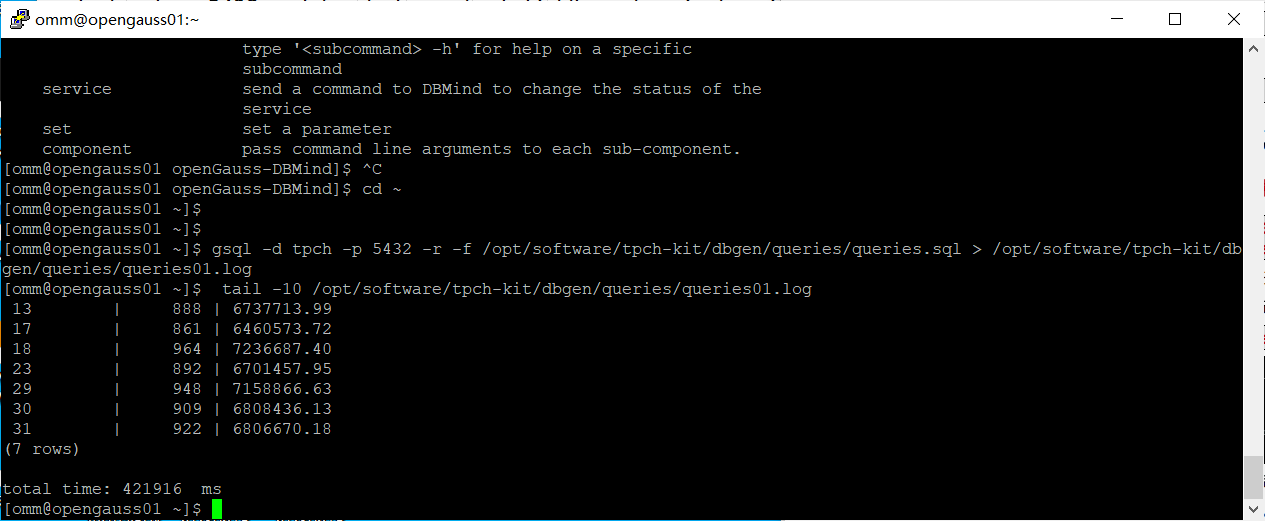
全量物化视图适用于静态或少量更新的场景，而增量物化视图适用于大量数据更新和频繁变化的场景。选择适合的物化视图类型应考虑到数据更新频率、刷新性能要求以及存储和维护成本等因素。

# 关卡三、openGauss的AI4DB特性应用

任务一：使用X-Tuner进行参数优化

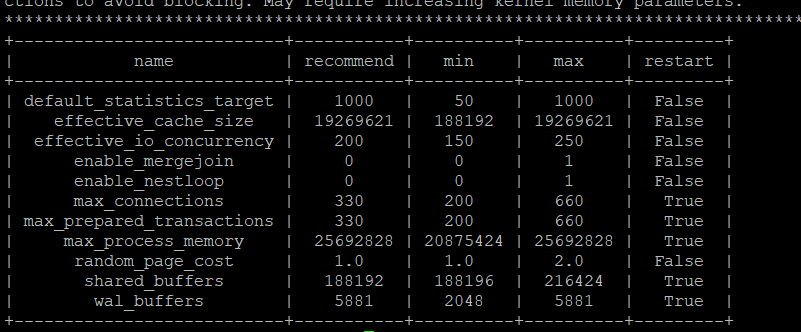
1. 执行TPCH脚本，获得测试时间，将执行结果截图：

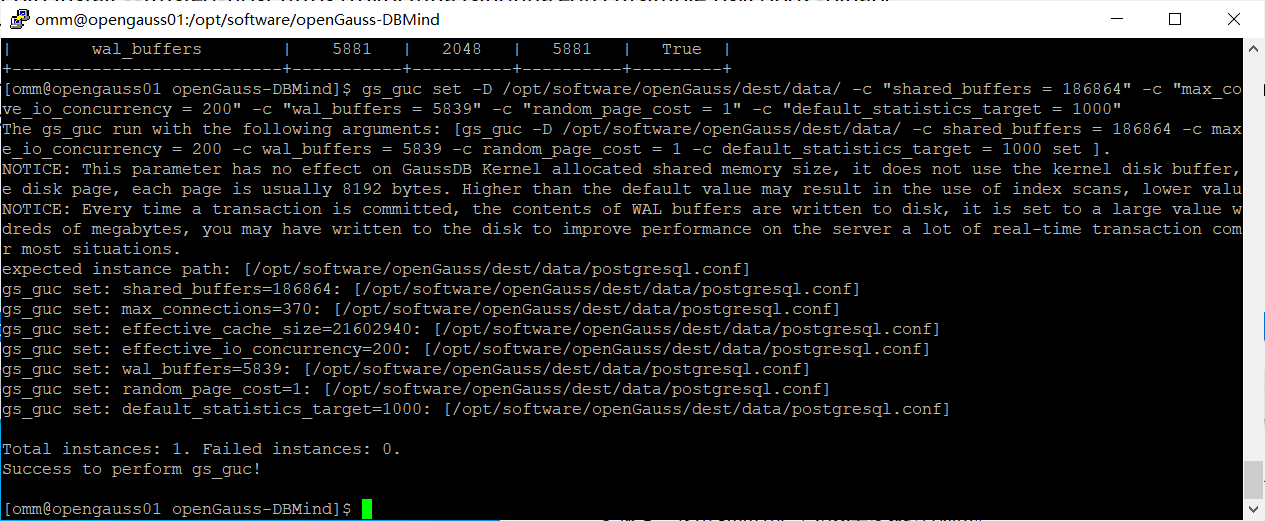
gsql -d tpch -p 5432 -r -f /opt/software/tpch-kit/dbgen/queries/queries.sql > /opt/software/tpch-kit/dbgen/queries/queries01.log



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

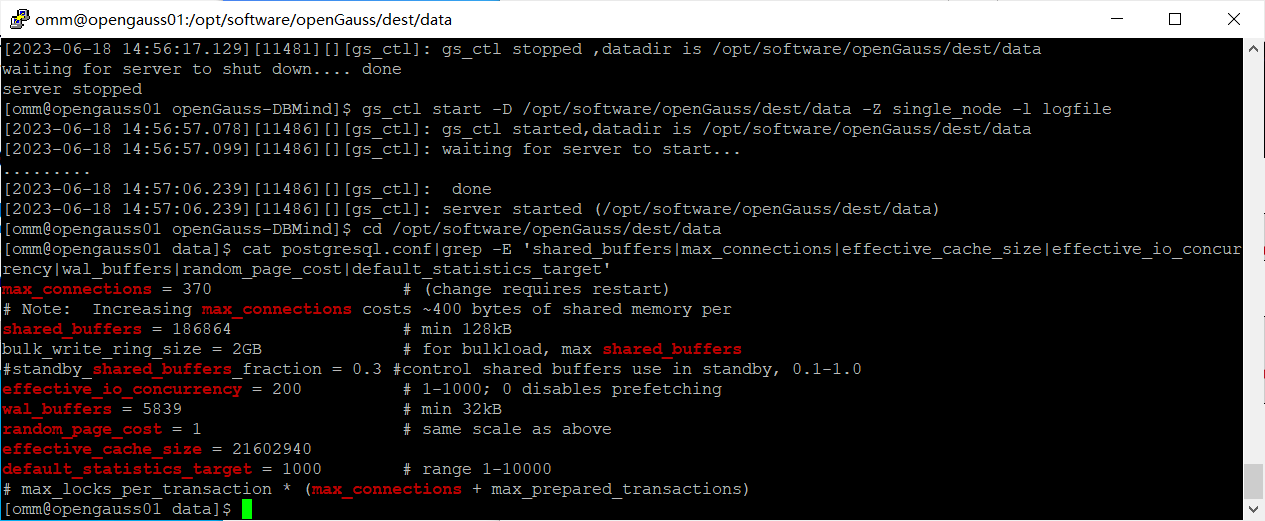




3.重启完成后，获取参数值：

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

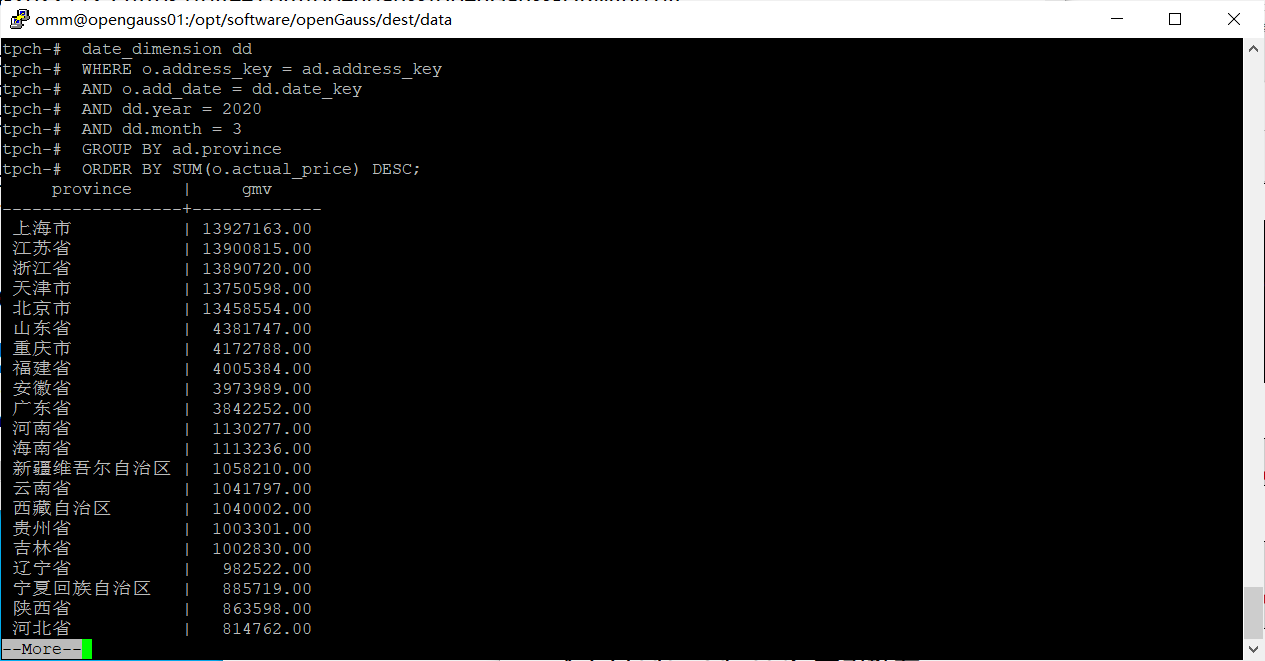
AND o.add\_date = dd.date\_key

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

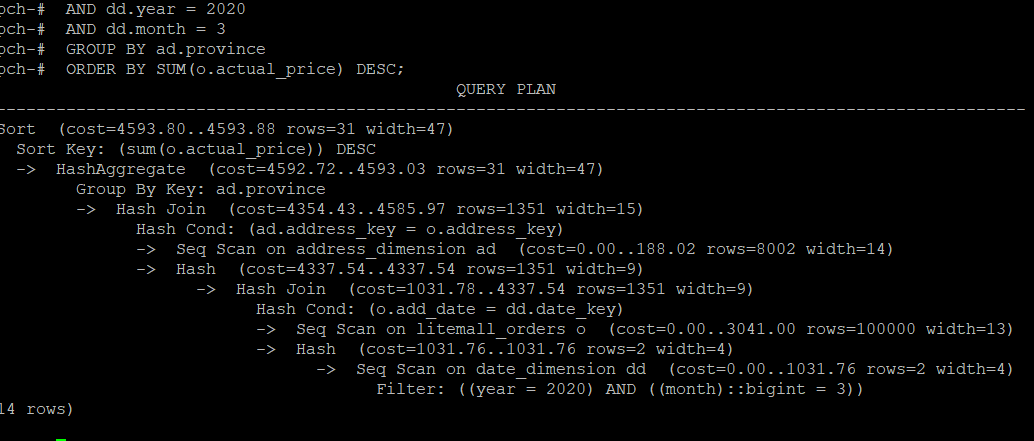
AND o.add\_date = dd.date\_key

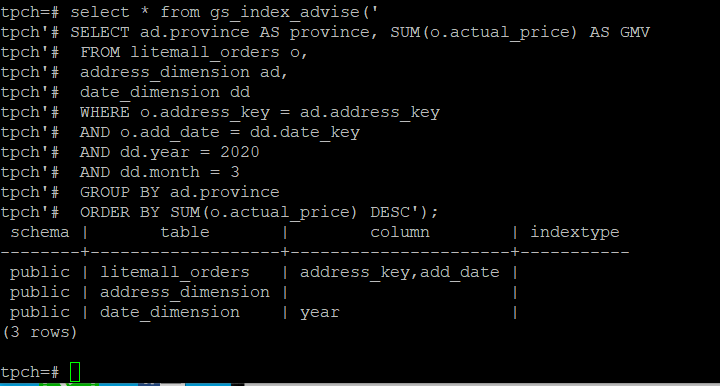
AND dd.year = 2020

AND dd.month = 3

GROUP BY ad.province

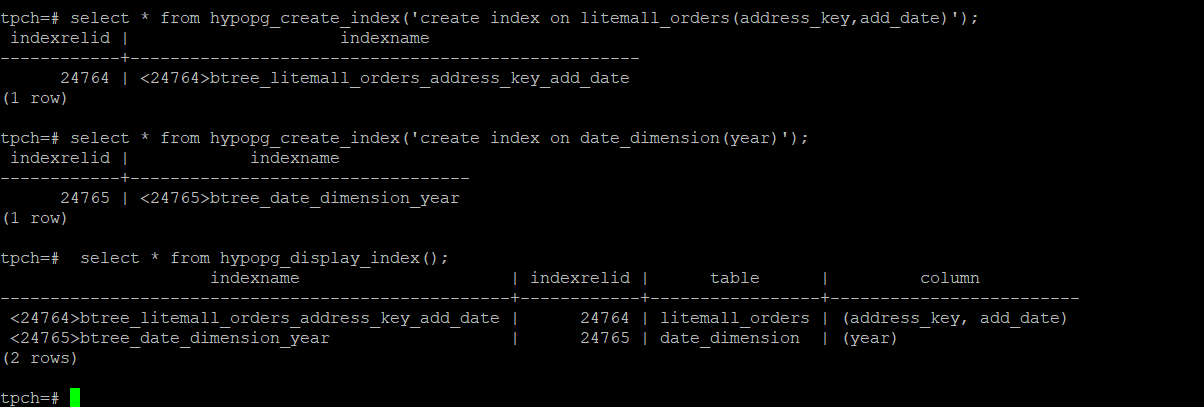
ORDER BY SUM(o.actual\_price) DESC');





3. 查看创建的虚拟索引列，将执行结果截图。

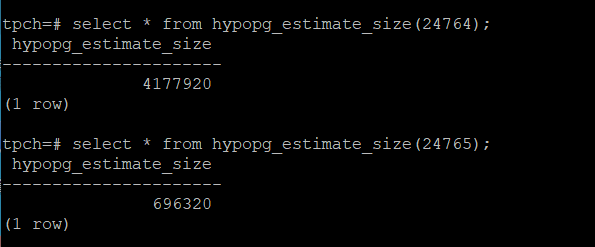
select \* from hypopg\_display\_index();



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

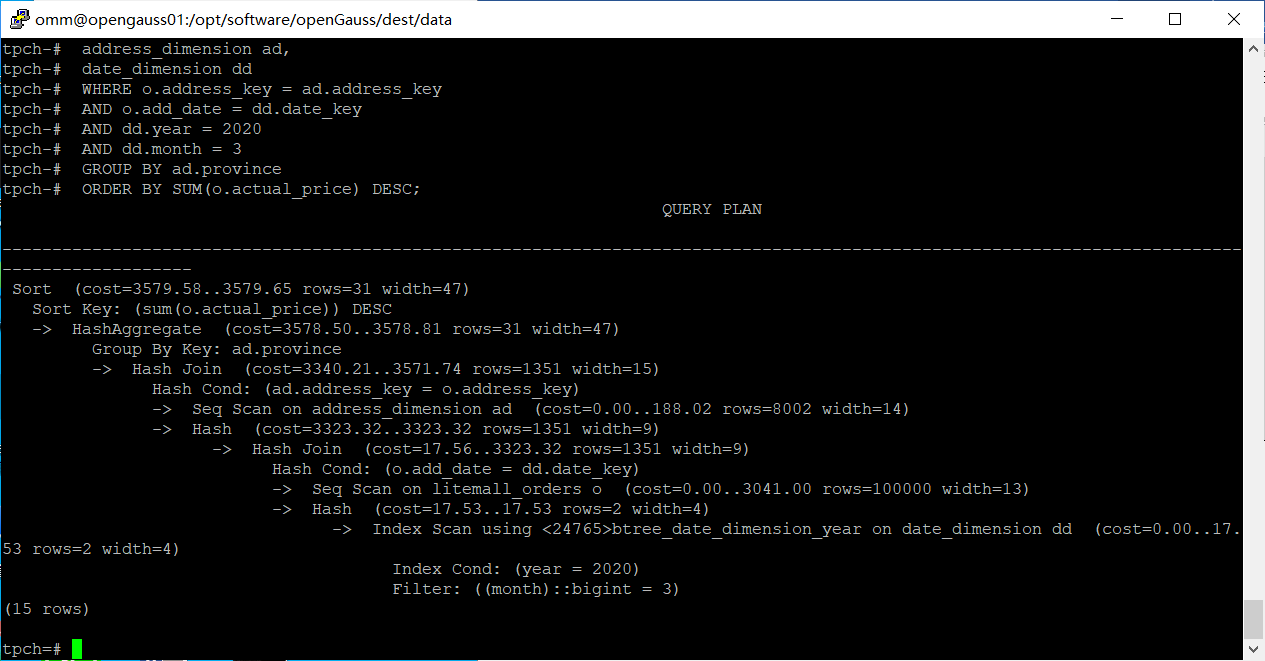
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

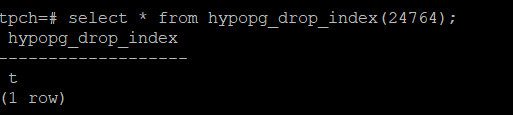
GROUP BY ad.province

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



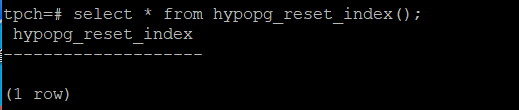
6. 删除某一个索引虚拟列，将执行结果截图。

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



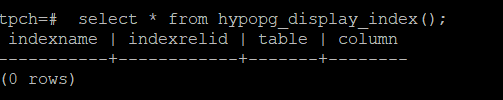
7. 删除某一个索引虚拟列，将执行结果截图。

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



8. 查看索引虚拟列，将执行结果截图。

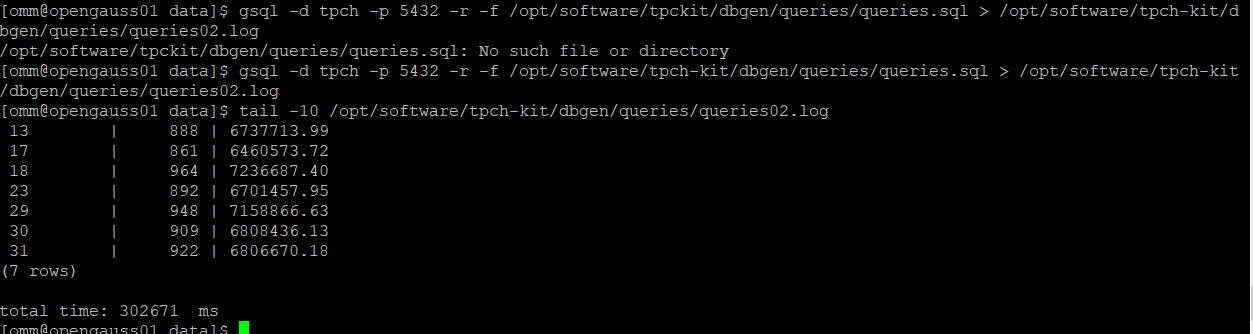
select \* from hypopg\_display\_index();



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：

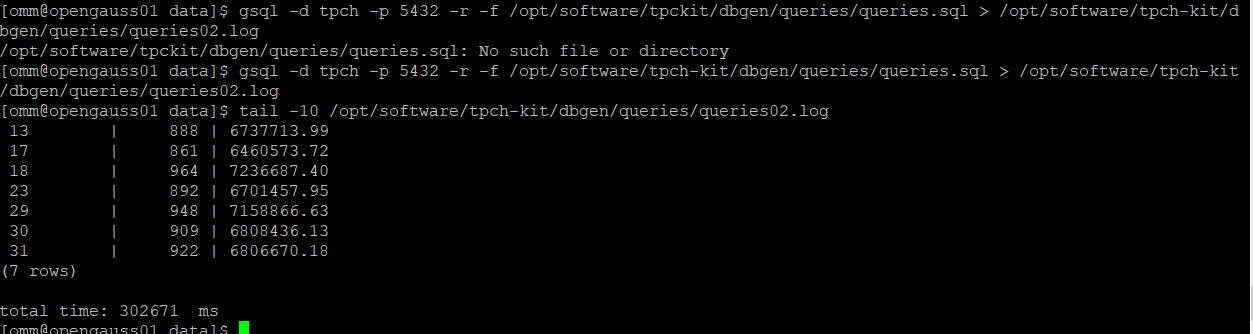
gsql -d tpch -p 5432 -r -f /opt/software/tpch-kit/dbgen/queries/queries.sql > /opt/software/tpch-kit/dbgen/queries/queries02.log



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？

通常情况下，X-Tuner 会根据数据库的实际工作负载和配置情况，针对以下参数进行优化：

1. shared\_buffers：共享缓冲区大小。这个参数控制了数据库在内存中分配给共享缓冲区的空间，用于存储常用数据的块。通过优化这个参数，可以提高对磁盘的访问效率，减少磁盘IO操作，从而提升查询性能。
2. max\_connections：最大连接数。这个参数定义了数据库系统能够同时处理的最大连接数。通过优化这个参数，可以确保数据库能够适应并发连接的需求，避免连接被拒绝或系统过载。
3. effective\_cache\_size：有效缓存大小。这个参数定义了数据库在查询时使用的预计可用内存量。通过优化这个参数，可以使数据库更有效地利用内存，提高缓存命中率，从而加快数据查询速度。
4. effective\_io\_concurrency：有效IO并发数。这个参数定义了数据库在执行IO操作时能够同时使用的并发IO数。通过优化这个参数，可以提高数据库对磁盘IO的并发处理能力，加快IO操作速度，提高整体性能。
5. wal\_buffers：WAL缓冲区大小。这个参数定义了WAL（Write-Ahead Logging）日志的缓冲区大小。通过优化这个参数，可以减少WAL写入磁盘的频率，提高写入性能和日志记录效率。
6. random\_page\_cost：随机页面成本。这个参数定义了随机页面访问相对于顺序页面访问的成本。通过优化这个参数，可以使查询优化器更准确地评估查询计划的成本，选择更高效的执行路径。
7. default\_statistics\_target：默认统计目标。这个参数定义了系统在自动收集统计信息时使用的抽样行数。通过优化这个参数，可以提高查询优化器对数据分布的准确性，提高查询计划的准确性和性能。

这些参数的优化是为了根据具体的工作负载和系统配置，使数据库能够更好地适应应用需求，提高查询性能、资源利用率和整体系统效率。优化这些参数可以使数据库在不同的场景下达到更好的性能和响应能力。

实践思考题2：索引的使用对于执行 SQL 语句有以下好处：

1. 提高查询性能：索引可以加速数据的查找和检索过程，减少数据库需要扫描的数据量，从而加快查询的执行速度。

2. 减少磁盘IO操作：索引可以减少对磁盘的IO操作，因为通过索引可以直接定位到所需数据的位置，而不需要全表扫描。

3. 优化排序和分组操作：对于排序和分组操作，使用索引可以避免数据库对整个表进行排序和分组，而只需针对索引数据进行操作，提高性能。

4. 支持唯一性约束：通过在列上创建唯一索引，可以确保数据的唯一性，避免重复数据的插入。

除了使用索引和参数优化，还有其他方面可以对数据库进行优化，包括：

1. 数据库设计：良好的数据库设计可以避免冗余数据、规范数据类型和关系，提高查询效率和数据的一致性。

2. 表结构优化：合理的表结构设计可以提高查询性能，包括选择适当的数据类型、合理划分表和分区、使用正确的主键和外键等。

3. 正确使用事务：合理使用事务可以保证数据的一致性和完整性，同时避免锁竞争和死锁等问题，提高并发处理能力和系统性能。

4. 缓存策略：合理使用缓存可以减少对数据库的访问，提高系统的响应速度和吞吐量，如使用缓存服务器、查询结果缓存等。

5. 查询优化：优化 SQL 查询语句的编写，包括选择合适的查询方式、避免全表扫描、合理使用连接和子查询、避免使用过多的函数和不必要的数据转换等。

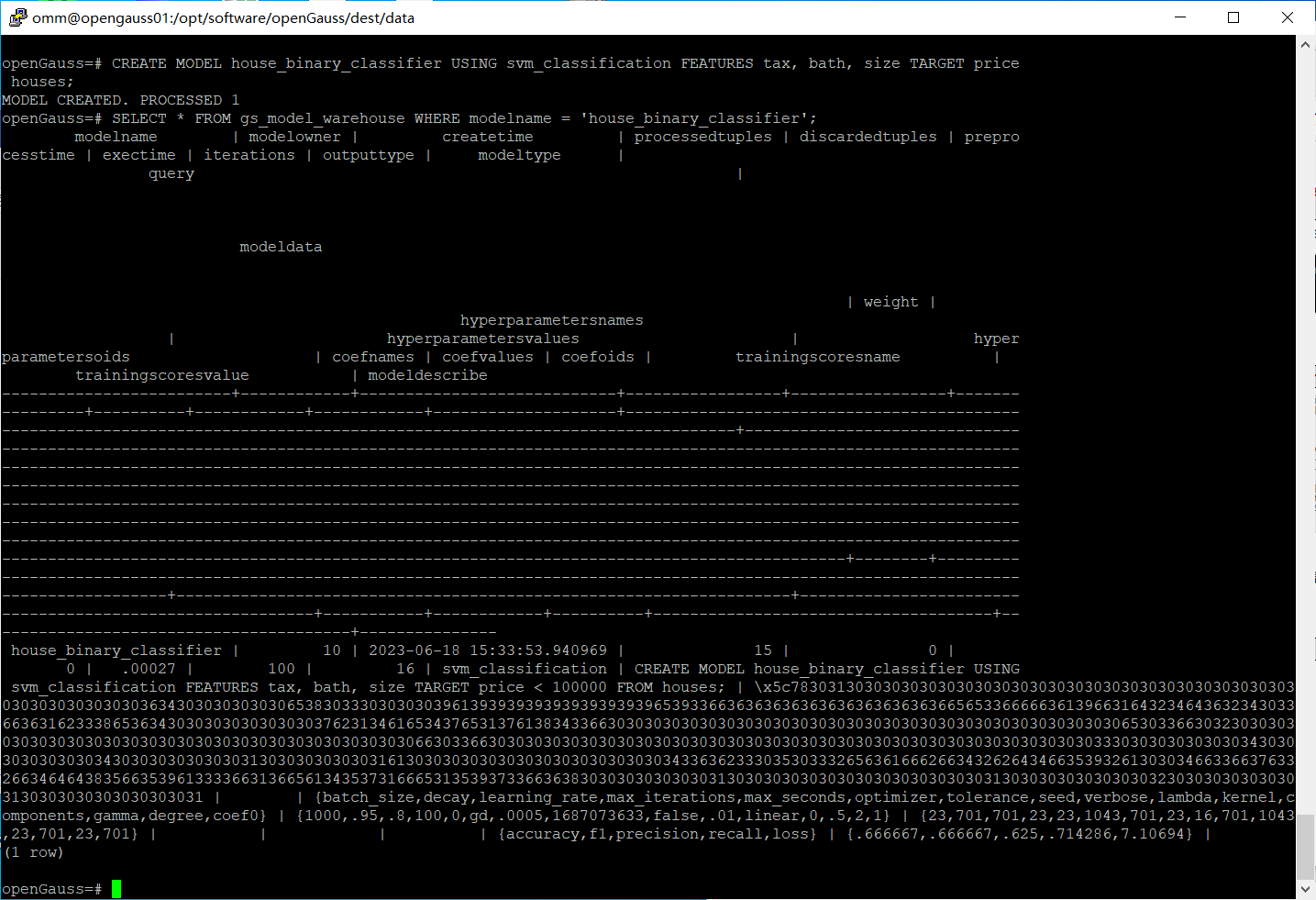
6. 硬件和存储优化：合理选择硬件配置和存储设备，如使用高速磁盘、RAID 阵列、SSD 等，提供更高的IO性能和吞吐量。

7. 定期维护和监控：定期进行数据库的维护工作，包括数据备份和恢复、数据库统计信息的收集、索引的重建和优化等，同时监控数据库的性能指标，及时发现和解决问题。

# 关卡四、openGauss的DB4AI特性应用

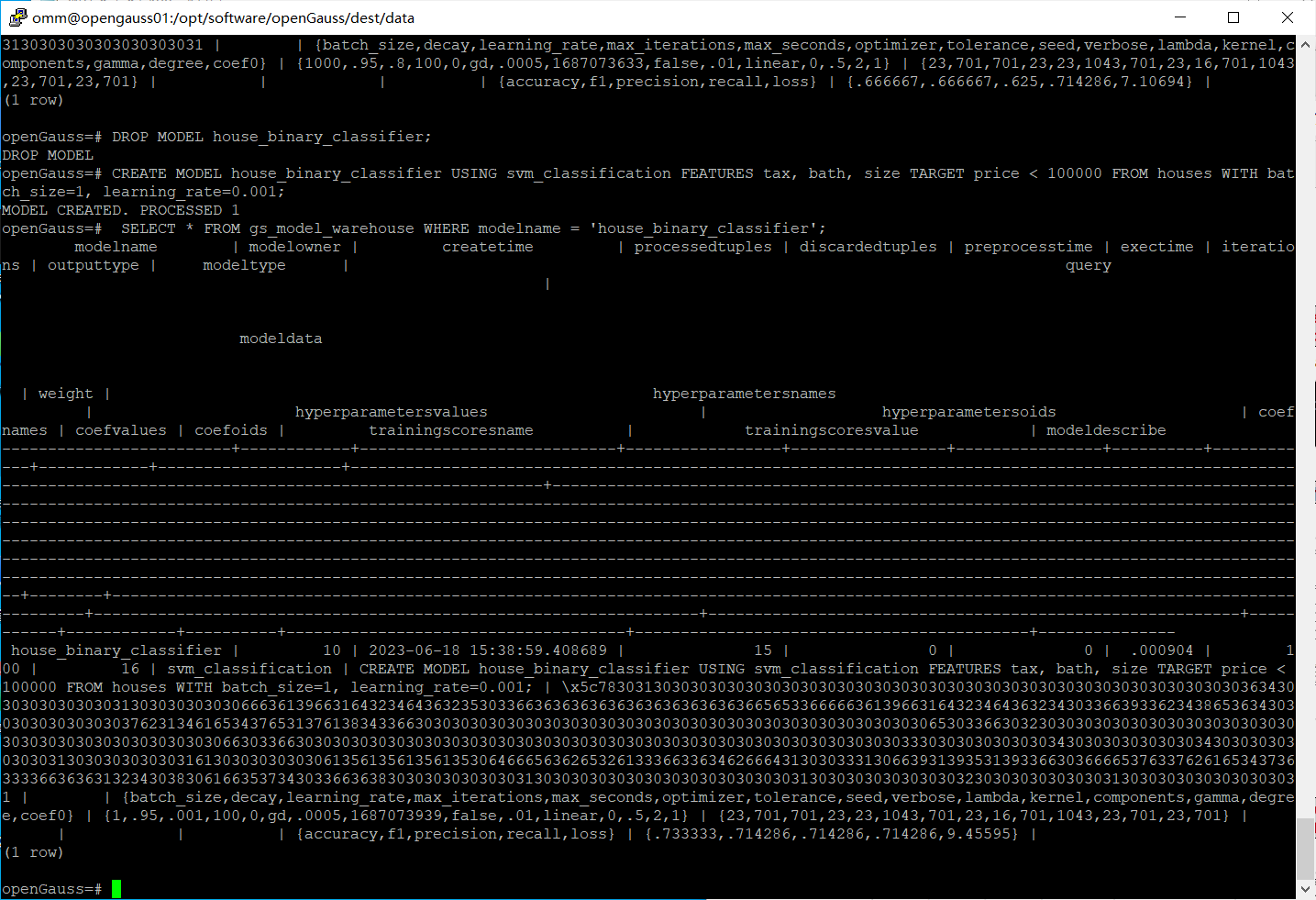
任务一：在gs\_model\_warehouse系统表中查看训练后的模型信息，将执行结果截图：

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



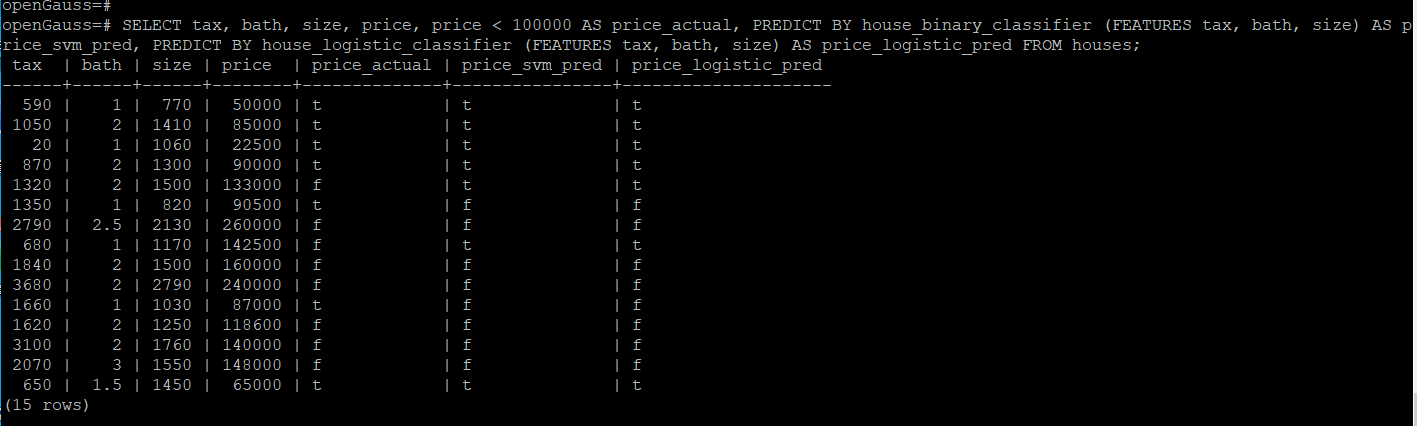
任务二：观察新模型的信息，将执行结果截图。

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？

分类模型和回归模型是机器学习中两种不同类型的预测模型，它们在目标变量类型和模型输出方面存在一些关键的区别：

1. 目标变量类型：

- 分类模型：分类模型用于预测离散的类别或标签。目标变量通常是具有固定类别的变量，例如二元分类问题（如判断邮件是否为垃圾邮件）或多类分类问题（如手写数字识别）。

- 回归模型：回归模型用于预测连续的数值。目标变量通常是连续的实数值，例如房价预测或销售额预测。

2. 模型输出：

- 分类模型：分类模型的输出是对样本进行分类的概率或类别标签。对于二元分类问题，通常输出两个类别中的一个概率，或者根据一个阈值将概率转换为类别标签。对于多类分类问题，输出是每个类别的概率分布或最可能的类别标签。

- 回归模型：回归模型的输出是对目标变量的数值预测。输出可以是实数或连续值，例如预测房价的模型可以输出具体的价格。

3. 模型算法：

- 分类模型：常见的分类模型算法包括逻辑回归、决策树、随机森林、支持向量机、朴素贝叶斯和深度学习中的卷积神经网络（CNN）和循环神经网络（RNN）等。

- 回归模型：常见的回归模型算法包括线性回归、多项式回归、岭回归、Lasso回归、决策树回归、随机森林回归和梯度提升回归（GBM）等。

总的来说，分类模型用于预测离散的类别或标签，输出是类别的概率或标签；而回归模型用于预测连续的数值，输出是目标变量的数值预测。选择使用分类模型还是回归模型取决于目标变量的类型和预测任务的性质。

实践思考题2：什么是SVM算法？

SVM（支持向量机）是一种监督学习算法，主要用于分类和回归分析。它在机器学习领域具有广泛应用，并被认为是一种强大且灵活的算法。

SVM 的核心思想是找到一个最优的超平面来将不同类别的样本分隔开。在二分类问题中，该超平面将样本空间划分为两个部分，使得两个类别的样本尽可能地分开，并且离超平面最近的一些样本点被称为支持向量。SVM 算法的目标是找到这个最优的超平面，使得支持向量到超平面的距离最大化。

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？

1. 准确率（Accuracy）：

准确率是最常用的分类评价指标，表示分类器正确预测的样本占总样本数的比例。计算公式为：

准确率 = (预测正确的样本数) / (总样本数)

2. 精确率（Precision）：

精确率衡量分类器在预测为正类的样本中有多少是真正的正类。它关注的是模型预测为正类的准确性。计算公式为：

精确率 = (真正的正类数) / (预测为正类的样本数)

3. 召回率（Recall）：

召回率衡量分类器在所有正类样本中预测正确的比例。它关注的是模型对正类样本的覆盖程度。计算公式为：

召回率 = (真正的正类数) / (所有真实的正类数)

4. F1 分数（F1-Score）：

F1 分数是精确率和召回率的调和均值，综合考虑了分类器的准确性和覆盖率。它是精确率和召回率的一个综合指标，适用于不平衡数据集。计算公式为：

F1 分数 = 2 \* (精确率 \* 召回率) / (精确率 + 召回率)

5. 特异度（Specificity）：

特异度衡量分类器在预测为负类的样本中有多少是真正的负类。它关注的是模型对负类样本的判断能力。计算公式为：

特异度 = (真正的负类数) / (所有真实的负类数)

6. AUC-ROC（Area Under the Receiver Operating Characteristic Curve）：

AUC-ROC 是衡量分类器性能的曲线下面积，表示分类器在不同阈值下的整体性能。ROC 曲线以真正类率（召回率）为纵轴，假正类率（1 - 特异度）为横轴。AUC-ROC 越接近于 1，表示分类器性能越好。

这些评价指标可以根据具体的问题和需求选择使用。准确率适用于样本类别平衡的情况，而精确率和召回率更适用于类别不平衡的情况。F1 分数综合考虑了精确率和召回率，适用于兼顾准确性和覆盖率的情况。特异度衡量了对负类样本的判断能力，AUC-ROC 综合了分类器在所有阈值下的性能。根据具体问题的需求，选择合适的评价指标来评估分类器的性能。

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？

1. 平均绝对误差（Mean Absolute Error, MAE）：

平均绝对误差是预测值与真实值之间差值的绝对值的平均值。它衡量了预测值和真实值之间的平均误差大小，数值越小表示模型预测的准确性越高。

2. 均方误差（Mean Squared Error, MSE）：

均方误差是预测值与真实值之间差值的平方的平均值。它衡量了预测值和真实值之间的平均偏差的平方，数值越小表示模型预测的准确性越高。MSE 对异常值比较敏感。

3. 均方根误差（Root Mean Squared Error, RMSE）：

均方根误差是均方误差的平方根，它对均方误差进行了标准化，与原始数据的单位一致。RMSE 与 MSE 相比更易于解释和理解。

4. R平方（Coefficient of Determination, R-squared）：

R平方衡量了回归模型对观测数据的拟合程度，即预测值与实际值的变异程度的比例。取值范围在0到1之间，越接近1表示模型对数据的解释能力越好。

5. 可决系数（Adjusted R-squared）：

可决系数对模型复杂度进行了调整，考虑了自变量的数量和样本量，用于衡量回归模型的拟合优度。可决系数可以避免在模型中增加不具有实际意义的变量而导致 R 平方增加的情况。

6. 相对绝对误差（Relative Absolute Error, RAE）：

相对绝对误差是预测值与真实值之间差值的绝对值的平均值与真实值的绝对值之比。它衡量了预测值和真实值之间的平均误差占真实值的比例。

这些评价指标可以根据具体的回归问题和需求选择使用。MAE、MSE、RMSE 衡量了预测值和真实值之间的误差大小，R平方和可决系数衡量了模型对数据的拟合程度，RAE 衡量了平均误差与真实值的相对关系。根据具体问题的需求，选择合适的评价指标来评估回归模型的性能。