openGauss AI特性创新实践课



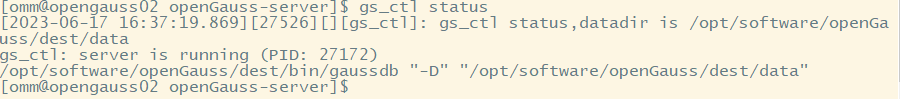
华为技术有限公司

# 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：

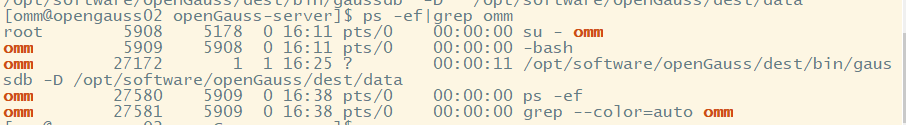
任务一：数据库状态验证

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？

回答：

通过源码编译安装数据库有以下几个原因：

1. 定制化：通过源码编译安装数据库可以进行各种自定义配置和优化，以满足特定需求。这使得安装的数据库更适合特定的应用场景，提高了性能和灵活性。
2. 最新版本：源码编译安装可以确保安装的是最新版本的数据库软件。这样可以获得最新的功能、性能改进和安全更新。
3. 平台兼容性：通过源码编译安装可以适应不同的操作系统和平台，以满足特定环境的要求。

安装数据库步骤：

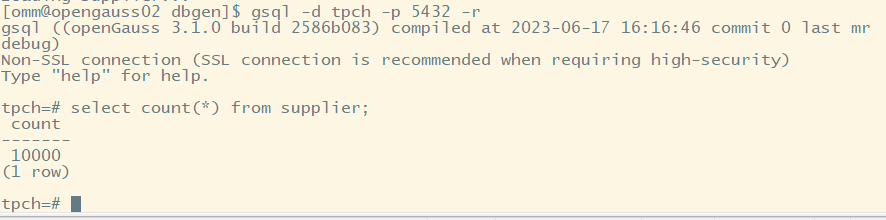
1. 下载源代码：从数据库的官方网站或开源社区下载数据库的源代码包。
2. 安装编译工具：确保系统上安装了编译所需的工具，如编译器、链接器和构建工具等。
3. 解压源代码：将下载的源代码包解压到适当的目录。
4. 配置编译选项：进入源代码目录，运行配置脚本或命令，以设置编译选项和配置参数。可以根据需要选择不同的编译选项，如启用特定的功能或禁用不需要的功能。
5. 编译源代码：运行编译命令开始编译源代码。这将根据系统配置和选项编译数据库的可执行文件和库文件。
6. 安装数据库：在编译完成后，运行安装命令，将编译生成的文件安装到系统中的适当位置。这通常包括可执行文件、库文件、配置文件和其他必要的文件。
7. 配置数据库：根据需要进行数据库的配置，包括设置数据库的根目录、网络连接、用户权限等。
8. 启动数据库：运行启动命令，启动安装好的数据库服务。

# 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：

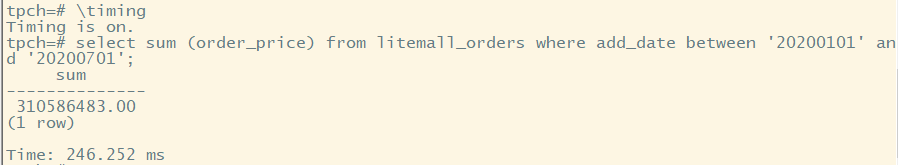
select count(\*) from supplier;;



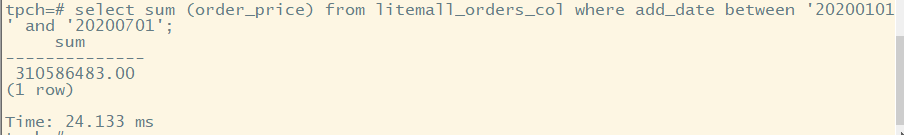
任务二：行存表与列存表执行效率对比

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';

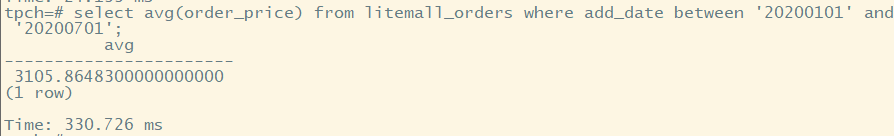


select sum (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

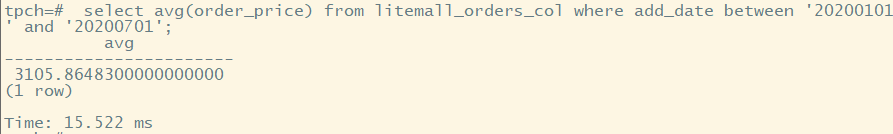


2. 2020年上半年litemall\_orders行存表与litemall\_orders\_col列存表中的order\_price的平均值查询，并对比执行效率截图

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

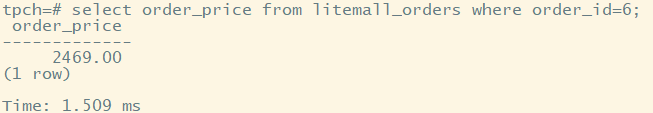


select avg (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

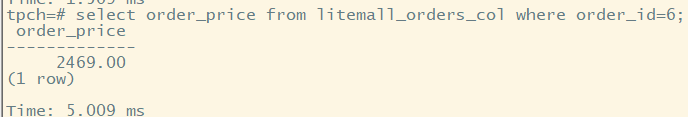


3. 查询litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price的值，并对比执行效率截图。

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

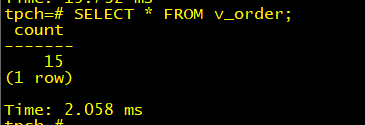
1. 创建物化视图所需要的表后，对表内容进行查询，对查询结果截图：

SELECT \* FROM test\_view;



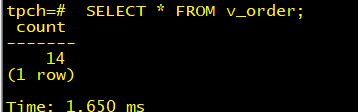
2. 使用物化视图统计人数，查询物化视图结果，将执行结果截图。

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？

1. 行存表（Row-based table）将数据按行存储在磁盘上。当执行 SQL 查询时，数据库需要读取整行的数据，包括查询条件所涉及的所有列，然后再进行筛选和计算。这对于需要获取整行数据的查询和更新操作效率较高，例如需要返回完整记录的查询、频繁进行更新和插入操作的场景。
2. 列存表（Columnar-based table）将数据按列存储在磁盘上。当执行 SQL 查询时，数据库只需要读取查询所需的列数据，而不需要读取整行数据。这可以减少磁盘 I/O 的开销，并提高查询性能，尤其是当查询只需要涉及部分列数据时。列存表适用于大规模数据分析和聚合查询，例如统计查询、数据挖掘和报表生成等。

在以下类型的 SQL 查询中，行存表效率更高：

1. 需要获取完整行数据的查询：如果查询需要返回整行数据，包括所有列，那么行存表可以直接读取整行数据，效率更高。
2. 频繁进行更新和插入操作的场景：由于行存表的数据存储方式更接近于传统的存储结构，对于频繁进行更新和插入操作的场景，行存表通常能够提供更好的性能。

在以下类型的 SQL 查询中，列存表效率更高：

1. 统计查询和聚合操作：当需要进行大规模数据的统计查询和聚合操作时，列存表可以只读取所需的列数据，减少了磁盘 I/O 的开销，提高查询性能。
2. 数据挖掘和报表生成：对于需要从大规模数据集中提取特定列数据进行数据挖掘、生成报表或执行复杂计算的场景，列存表能够以更高的效率进行列级别的数据读取和处理。

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：

1. 数据刷新方式：全量物化视图的数据是通过完全刷新（Full Refresh）的方式进行更新。完全刷新意味着在刷新时重新计算和加载整个物化视图的数据，通常会执行与原始数据源相应的查询，并将结果存储在物化视图中。
2. 维护频率：全量物化视图通常需要定期或手动执行完全刷新操作，以确保物化视图中的数据与原始数据源保持同步。这可能会导致在刷新过程中的一段时间内物化视图不可用。
3. 存储需求：由于需要存储完整的数据副本，全量物化视图通常需要更多的存储空间。

增量物化视图：

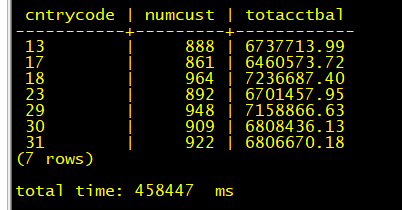
1. 数据刷新方式：增量物化视图的数据是通过增量刷新（Incremental Refresh）的方式进行更新。增量刷新只更新物化视图中发生更改的数据，而不需要重新计算和加载整个物化视图。这可以减少刷新时间和资源消耗。
2. 维护频率：增量物化视图可以根据需要定期或持续地进行增量刷新，以使物化视图保持与原始数据源的同步。这使得物化视图的可用性更高，因为它们可以在刷新期间进行查询。
3. 存储需求：由于只存储更改的数据，增量物化视图通常需要较少的存储空间，尤其是在处理大型数据集时

# 关卡三、openGauss的AI4DB特性应用

任务一：使用X-Tuner进行参数优化

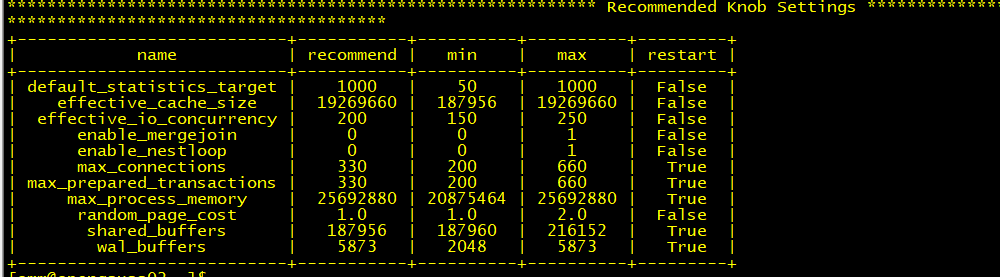
1. 执行TPCH脚本，获得测试时间，将执行结果截图：

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

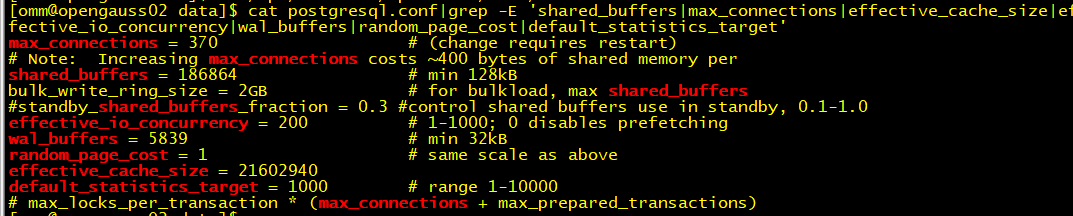
gs\_xtuner recommend --db-name tpch --db-user omm --port 5432 --host 127.0.0.1 --host-user omm



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

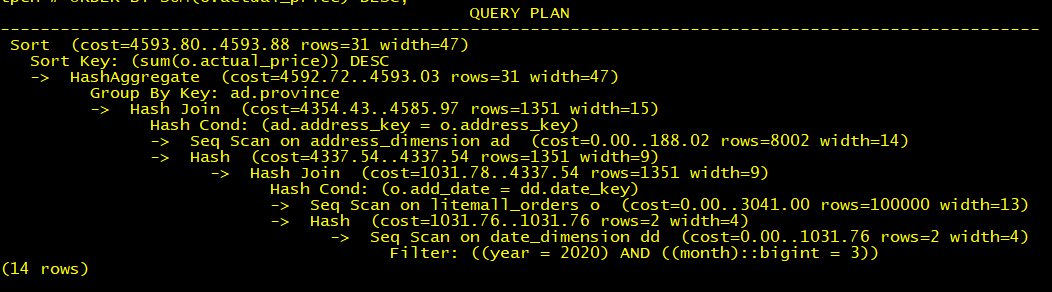
AND o.add\_date = dd.date\_key

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

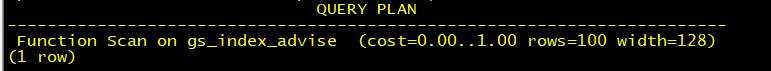
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

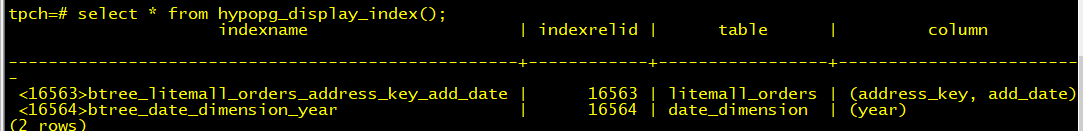
GROUP BY ad.province

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。

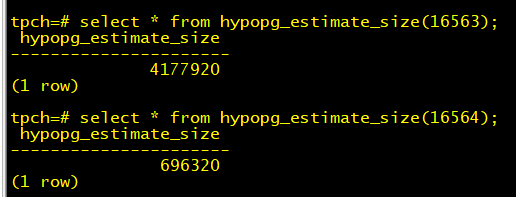
select \* from hypopg\_display\_index();



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

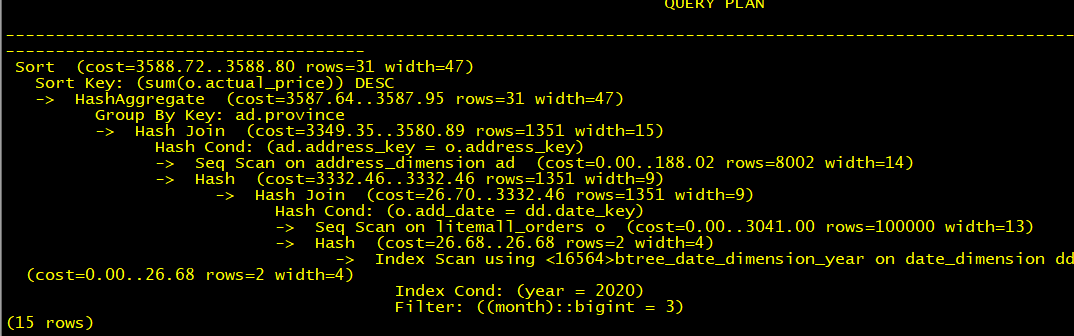
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

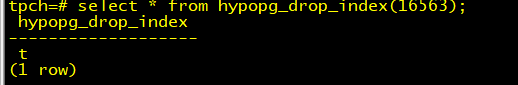
GROUP BY ad.province

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



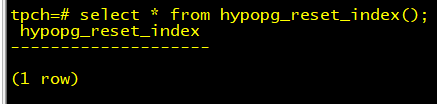
6. 删除某一个索引虚拟列，将执行结果截图。

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



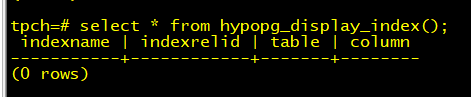
7. 删除某一个索引虚拟列，将执行结果截图。

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



8. 查看索引虚拟列，将执行结果截图。

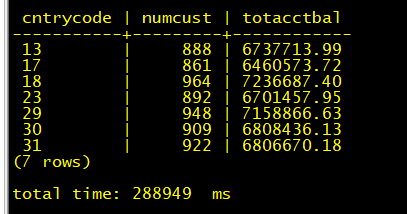
select \* from hypopg\_display\_index();



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？

uner 会关注以下几个方面的参数进行优化：

1. 缓冲区参数：优化缓冲区参数可以提高数据库系统的性能和响应时间。例如，可以调整数据库的 shared\_buffers 参数来控制内存中用于缓存数据的大小，从而提高查询的执行速度。
2. 并发参数：优化并发参数可以提高数据库系统的并发性能和资源利用率。例如，可以调整 max\_connections 参数来限制同时连接到数据库的客户端数量，以避免过多的连接导致性能下降。
3. 查询优化参数：优化查询优化参数可以改善数据库系统的查询执行计划和性能。例如，可以调整 work\_mem 参数来控制每个查询使用的内存大小，以提高排序和哈希操作的性能。
4. 自动化参数调整：X-Tuner 还可以自动化地优化一些参数，例如自动调整 cost-based optimizer 的参数，以便根据查询的复杂性和数据分布进行动态优化，从而提高查询性能。

这些参数的优化是为了根据实际工作负载和系统环境的要求，提供最佳的性能和资源利用率。通过对这些参数进行优化，可以改善数据库系统的查询响应时间、并发处理能力和资源利用效率，从而提供更好的用户体验和系统性能。优化这些参数需要根据具体情况进行评估和测试，以找到最适合应用场景的配置值。X-Tuner 提供了自动化的参数调整和优化功能，简化了这个过程，并提供了更好的性能调优结果。

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？

索引的使用对于执行 SQL 查询有以下好处：

1. 提高查询性能：索引可以加快查询的速度，通过在数据库表上创建索引，可以减少数据库的扫描操作，快速定位到满足查询条件的数据行，从而提高查询效率。
2. 减少磁盘 I/O：索引可以减少磁盘 I/O 操作，因为索引存储在内存中或占用较小的磁盘空间，可以减少对磁盘的访问次数。这对于大型数据集和频繁查询的场景尤为重要。
3. 提高并发性能：通过使用索引，可以减少数据的锁定范围，提高并发性能。当多个事务同时对数据库进行读取操作时，索引可以帮助减少锁冲突，提高并发性能。

除了使用索引和参数来优化数据库性能外，还有以下几个方面可以对数据库进行优化：

1. 数据库设计：合理的数据库设计是优化数据库性能的重要基础。包括正确定义表结构、合理划分数据表和数据字段、建立适当的关系和约束等。
2. 查询优化：通过优化 SQL 查询语句，可以减少查询的执行时间和资源消耗。包括选择合适的查询方式、优化连接操作、避免全表扫描、合理使用聚合函数等。
3. 数据库配置和参数调优：调整数据库的配置和参数设置可以提高性能和资源利用率。例如，调整内存配、并发连接数、缓冲区大小等。
4. 磁盘和存储优化：合理的磁盘和存储配置可以提高数据库的读写性能。包括使用合适的磁盘类型和 RAID 配置、优化文件系统、分离数据和日志文件等。
5. 定期维护和优化：定期进行数据库维护任务，如备份和恢复、索引重建、统计信息更新等，可以保持数据库的健康状态和良好性能。
6. 垂直和水平扩展：根据需求和负载情况，可以考虑垂直扩展（增加单台服务器的资源）或水平扩展（增加服务器数量）来提高数据库性能和可扩展性。

# 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';



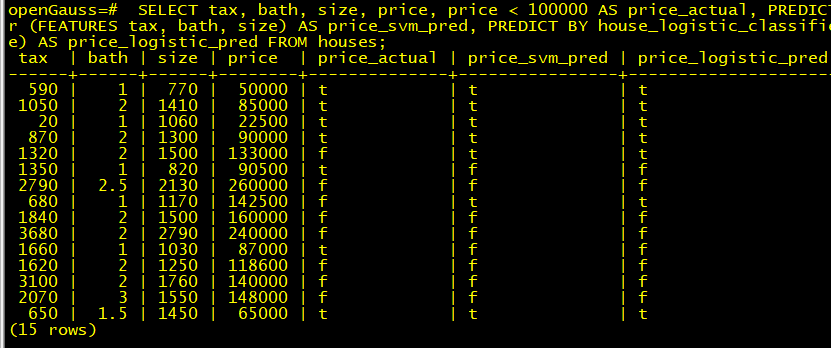
任务二：观察新模型的信息，将执行结果截图。

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（Classification Model）用于对离散的类别进行预测。其目标是将输入数据映射到预定义的类别标签。分类模型的输出是离散的类别标签或类别概率。

回归模型（Regression Model）用于对连续的数值进行预测。其目标是根据输入数据的特征，预测一个连续的数值输出。回归模型的输出是一个实数或实数范围内的值。

主要的不同点如下：

1. 输出类型：分类模型的输出是离散的类别标签或类别概率，而回归模型的输出是连续的数值。
2. 目标：分类模型旨在预测样本所属的类别，即对离散的类别进行分类。而回归模型旨在预测一个数值，例如房价、销售量等。
3. 损失函数：分类模型通常使用交叉熵（Cross Entropy）等损失函数来衡量预测结果与真实类别之间的差异。回归模型常用的损失函数包括均方误差（Mean Squared Error）和平均绝对误差（Mean Absolute Error）等。
4. 模型选择：根据问题的性质和数据的特点选择合适的模型。如果目标是进行类别预测，例如垃圾邮件分类、图像分类等，则选择分类模型。如果目标是进行数值预测，例如房价预测、销售额预测等，则选择回归模型。

实践思考题2：什么是SVM算法？

支持向量机（Support Vector Machine，SVM）是一种常用的监督学习算法，用于进行分类和回归任务。SVM的主要目标是找到一个最优的超平面或决策边界，以将不同类别的样本尽可能地分开。

在二分类问题中，SVM将样本表示为特征空间中的点，并试图找到一个最优的超平面来将两个类别的样本分开。这个超平面在特征空间中是具有最大间隔（Margin）的，即离两个类别最近的样本点的距离最大化。这些离超平面最近的样本点被称为支持向量（Support Vectors），它们对于确定超平面起到关键作用。

SVM的工作原理可以归结为以下步骤：

1. 特征转换：将原始的输入数据通过一定的方式转换为高维特征空间，使得样本在新的特征空间中容易分开。
2. 寻找最优超平面：在特征空间中寻找一个超平面，使得间隔（Margin）最大化。间隔是指离两个类别最近的样本点到超平面的距离，SVM试图找到最大间隔的超平面。
3. 分类或回归：通过计算样本点到超平面的位置，将新样本进行分类或回归预测。

SVM在分类问题中常用的核函数有线性核、多项式核、径向基函数（Radial Basis Function，RBF）等，这些核函数可以将数据映射到更高维的特征空间，以处理线性不可分或非线性的问题。

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？

在分类问题中，有几个常用的评价指标用于评估分类模型的性能。以下是常见的分类评价指标及其含义：

1. 准确率（Accuracy）：准确率是最常见的分类评价指标之一，表示分类正确的样本数占总样本数的比例。它计算为分类正确的样本数除以总样本数。
2. 精确率（Precision）：精确率衡量模型预测为正类的样本中有多少是真正的正类样本。它计算为真正的正类样本数除以预测为正类的样本总数。
3. 召回率（Recall）：召回率衡量真正的正类样本中有多少被模型正确地预测为正类。它计算为真正的正类样本数除以实际为正类的样本总数。
4. F1 分数（F1-Score）：F1 分数是精确率和召回率的加权调和平均值，用于综合考虑两者的表现。它计算为精确率和召回率的调和平均值。
5. ROC 曲线和 AUC：ROC 曲线（Receiver Operating Characteristic Curve）是以真正类率（TPR）为纵轴，假正类率（FPR）为横轴绘制的曲线。ROC 曲线能够直观地展示分类器在不同阈值下的性能。AUC（Area Under the ROC Curve）是 ROC 曲线下方的面积，用于度量分类器的整体性能。AUC 越大，分类器的性能越好。
6. 混淆矩阵（Confusion Matrix）：混淆矩阵是一种表格，用于展示分类模型的预测结果与真实类别之间的关系。它包括真正类（True Positive，TP）、假正类（False Positive，FP）、真负类（True Negative，TN）和假负类（False Negative，FN）。从混淆矩阵可以计算其他评价指标，如准确率、精确率和召回率。

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？

在回归问题中，常用的评价指标用于评估回归模型的性能。以下是一些常见的回归评价指标及其含义：

1. 均方误差（Mean Squared Error，MSE）：均方误差是预测值与真实值之间差异的平方的平均值。它衡量了模型预测结果的平均误差大小。MSE 越小，表示模型的预测越准确。
2. 均方根误差（Root Mean Squared Error，RMSE）：均方根误差是均方误差的平方根，它与原始数据的单位相匹配。RMSE 同样衡量了模型预测结果的平均误差大小。RMSE 越小，表示模型的预测越准确。
3. 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异的绝对值的平均值。它衡量了模型预测结果的平均误差的绝对大小。MAE 越小，表示模型的预测越准确。
4. 决定系数（Coefficient of Determination，R²）：决定系数衡量了模型对观测值的拟合程度，即预测值与真实值之间的方差比例。它的取值范围在 0 到 1 之间，越接近 1 表示模型的拟合程度越好。
5. 相关系数（Correlation Coefficient）：相关系数度量了预测值与真实值之间的线性相关性。它的取值范围在 -1 到 1 之间，正值表示正相关，负值表示负相关，绝对值越接近 1 表示相关性越强。