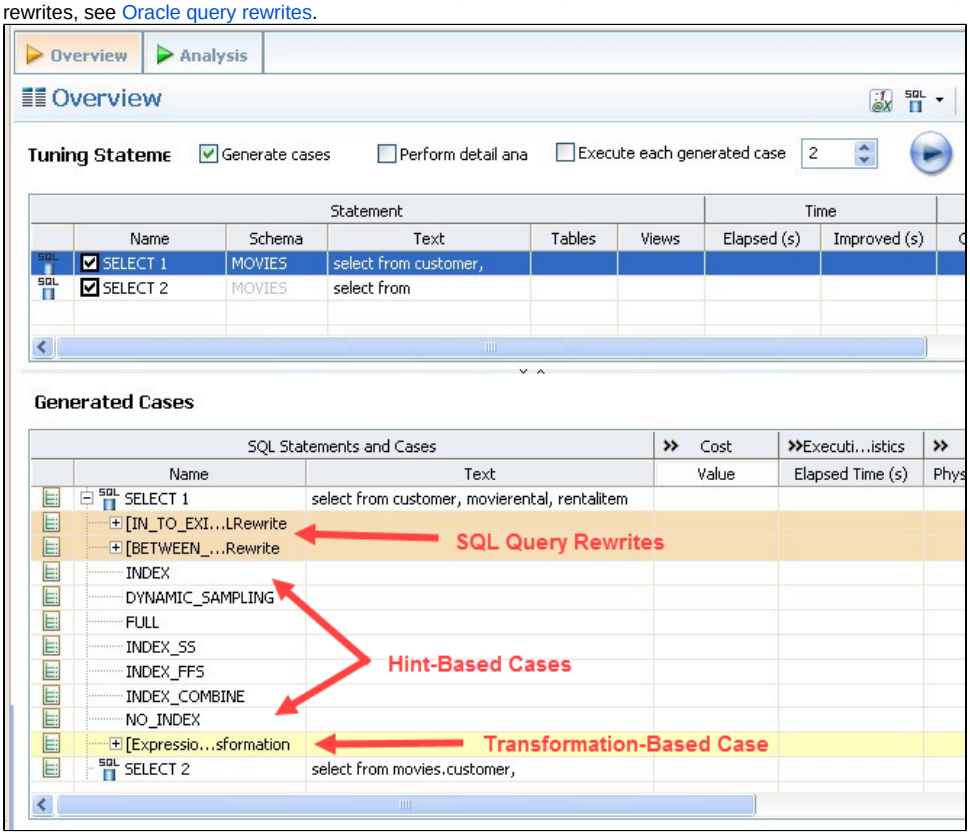
Modify tuning results

As you add SQL source to the Overview tab of a tuning job, the supported DML statements are automatically parsed out and a numbered statement record for each statement is added to the Overview tab.

Cases generated from tuning candidates are alternative forms of the original statement that have been optimized or otherwise "fixed" by the tuning function. Once you have executed a tuning job, tuning automatically generates all SQL optimizer hint-based variations that can be applied to the statement: If you change the schema of a statement

- All SQL Optimizer hint-based variations that can be applied to a statement.
- A transformation-based case, if any of the eight common quick fixes can be applied to an SQL statement. This feature leverages the SQL Query Tuner Code Quality Check functionality. See Understanding Code Quality Checks for more information on the eight quick fixes. A transformation case, in turn, has its own set of SQL Optimizer hint cases. For information on query rewrites, see Oracle query rewrites. For information on other transformations, see Examples of Transformations and SQL Query Rewrites.
- SQL Query Rewrites may be suggested when tuning. For example, a recommended rewrite for EXISTS may be IN. For information on query rewrites, see Oracle query rewrites.



Hint-based cases and the transformation-based cases are a special case of the statement records added to the Overview tab as you add candidates to a tuning job. With the exception of the Text, Source, and Index Analysis fields, cases are identical to the standard statement record. Similarly, execution, statistics collection, and other options available for basic statement records are available for individual cases.

Once cases have been generated, if you have the required permissions on the specified data source, you can apply the changes suggested by hint and transformation based cases in the Overview table.

Oracle query rewrites

The following query rewrites or transformations may be recommended during tuning.

Before	After
select * from t1 where EXISTS (select null from t2 where t2.key=t1.key);	select * from t1 where t1.key IN (select t2.key from t2);
select * from t1 where NOT EXISTS (select null from t2 where t2.key=t1.key);	select * from t1 where t1.key NOT IN (select t2.key from t2 where t2.key is not null);
select * from t1 where t1.key IN (select t2.key from t2);	select * from t1 where EXISTS (select null from t2 where t2.key = t1.key);

select * from t1 where t1.key NOT IN (select t2.key from t2 where t2.key is not null);	select * from t1 where NOT EXISTS (select null from t2 where t2.key = t1.key);
select * from t1 where NOT EXISTS (select null from t2 where t2.key = t1.key);	select t1.* from t1, t2 where t1.key = t2.key(+) and t2.key is null
select * from t1 where t1.key NOT IN (select t2.key from t2 where t2.key is not null);	select t1.* from t1, t2 where t1.key = t2.key(+) and t2.key is null;
select column BETWEEN X AND Y	select (column <= X AND column >= Y)
select column NOT BETWEEN X AND Y	select (column < X AND column > Y)
select (column<= X AND column >= Y)	select column BETWEEN X AND Y
select (column < X AND column > Y)	select column NOT BETWEEN X AND Y
select t1.* from t1, t2 where t1.key = t2.key and t2.col = 10;	select t1.* from t1, (select * from t2 where t2.col = 10) inline_alias where t1.key= inline_alias.key;
select t2.* from t1, t2 where t1.key = t2.key and t1.col is null	select * from t2 where t2.key IN (select t1.key from t1 where t1.col is null)