# Example 12 - Regular SQL Selects

This example shows how to run regular (free) SQL selects.

SQL selects must be configured in the configuration file. See the reference section for details on their syntax and how to define their parameters.

SQL selects are exposed in the DAOs as simple java methods with the configured java parameters.

Regular SQL selects shares the syntax with Native SQL and Dynamic SQL. They can be combined all at once when creating SQL statements.

SQL selects can include <column> tags to force the Java name and Java type of the result set. When included they need to be included outside the CDATA section, if any.

## How to Run this example

The **Example 12** is included in the download package. To run this example please refer to the section How to Run the Examples above.

## Case #1: Columns from a single table

The configuration file includes the SQL select that retrieves clients created between two dates using the following SQL statement:

<select java-class-name=*"CreatedClient"*>

<![CDATA[

select \* from client

{\*

where created\_at between #{since,javaType=java.sql.Date,jdbcType=DATE}

and #{until,javaType=java.sql.Date,jdbcType=DATE}

\*}

]]>

</select>

The SQL select can be run using the select(java.sql.Date since, java.sql.Date until) Java method on the DAO CreatedClient.

## Case #2: Join returns fully-typed, fully-named columns from multiple tables

A new DAO Java class is generated to include the fully-typed, fully-named returned columns for both tables on the SQL select. The configuration file includes the SQL select performs the SQL join:

<select java-class-name=*"ClientWithPurchase"*>

<![CDATA[

select p.\*, c.vip, c.state from purchase p

join client c on (c.id) = (p.client\_id)

{\*

where p.purchase\_date =

#{purchaseDate,javaType=java.sql.Date,jdbcType=DATE}

\*}

]]>

</select>

The SQL select can be run using the select(java.sql.Date purchaseDate) Java method on the DAO ClientWithPurchase.

## Case #3: Grouping and adding extra columns

A new DAO Java class is generated to include the fully-typed, fully-named returned columns from the SQL select. The configuration file includes the following SQL select:

<select java-class-name=*"DailyTotal"*>

<![CDATA[

select

purchase\_date,

count(\*) as number\_of\_purchases,

sum(vehicle\_price) as price,

sum(extras\_price) as extras,

sum(discount) as discount,

sum(tax) as taxes,

sum(final\_price) as revenue

from purchase

{\*

where purchase\_date between #{from,javaType=java.sql.Date,jdbcType=DATE}

and #{to,javaType=java.sql.Date,jdbcType=DATE}

group by purchase\_date

\*}

]]>

</select>

The SQL select can be run using the select(java.sql.Date from, java.sql.Date to) Java method on the DAO DailyTotal.

## Case #4: Subqueries

A new DAO Java class is generated to include the fully-typed, fully-named returned columns from the SQL select. The configuration file includes the following SQL select:

<select java-class-name=*"ClientNeverOfferedDiscount"*>

<![CDATA[

select \* from client where id not in

(select client\_id from purchase where discount > 0)

]]>

</select>

The SQL select can be using the select() Java method on the DAO ClientNeverOfferedDiscount.