**Query C/1:**

**The average, the minimum, the maximum, and the standard deviation of the number of fix-up tasks per user.**

select avg(c.fixUpTasks.size),

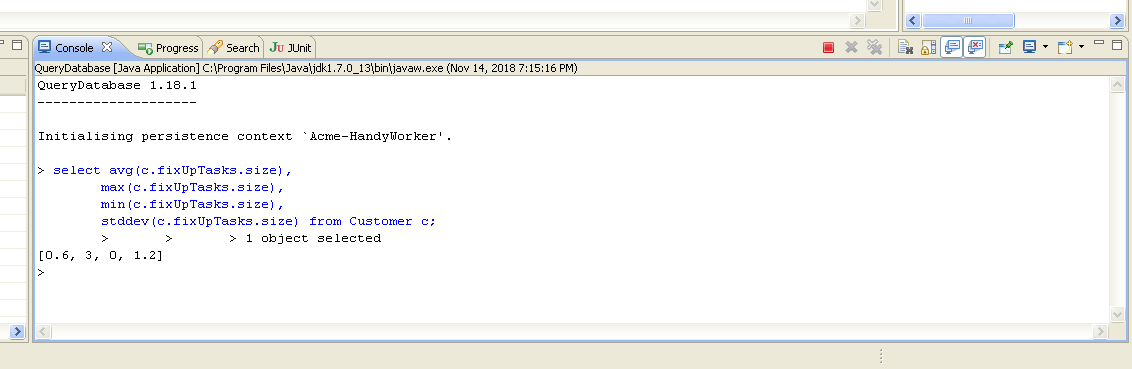
max(c.fixUpTasks.size),

min(c.fixUpTasks.size),

stddev(c.fixUpTasks.size) from Customer c;

**Description:**

**We solve this query using default functions from JPQL, in this case the functions are avg(average), max(maximum), min(minimum) and stddev(standard deviation). Iterating over the amount of fix-up tasks per customer.**

****

**Query C/2:**

**The average, the minimum, the maximum, and the standard deviation of the number of applications per fix-up task.**

select avg(fut.applications.size),

max(fut.applications.size),

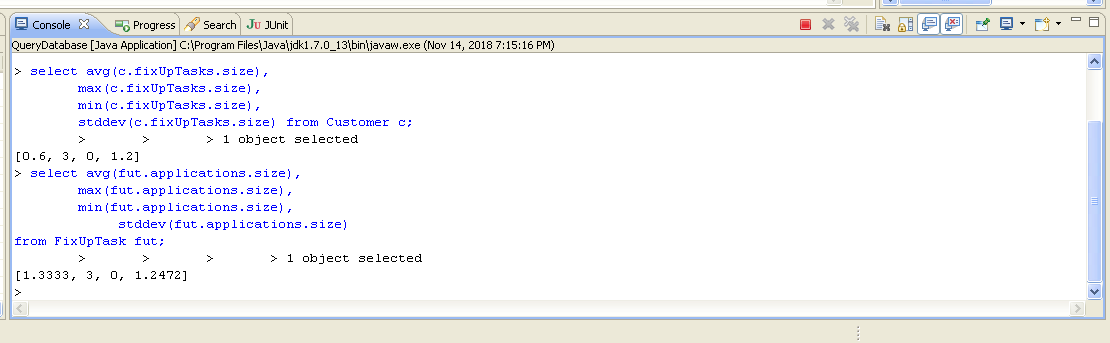
min(fut.applications.size),

stddev(fut.applications.size)

from FixUpTask fut;

**Description:**

**We solve this query using default functions from JPQL, in this case the functions are avg(average), max(maximum), min(minimum) and stddev(standard deviation). Iterating over the amount of applications per fix-up taks.**



**Query C/3:**

**The average, the minimum, the maximum, and the standard deviation of the maximum price of the fix-up tasks.**

select avg(fut.maxPrice.amount),

max(fut.maxPrice.amount),

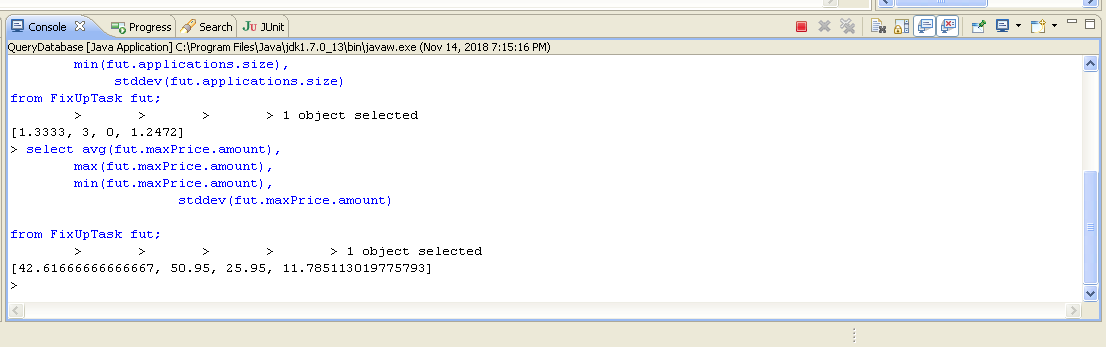
min(fut.maxPrice.amount),

stddev(fut.maxPrice.amount)

from FixUpTask fut;

**Description:**

**We solve this query using default functions from JPQL, in this case the functions are avg(average), max(maximum), min(minimum) and stddev(standard deviation). Iterating over the maximum price of the fix-up tasks.**



**Query C/4:**

**The average, the minimum, the maximum, and the standard deviation of the price offered in the applications.**

select avg(a.offeredPrice.amount),

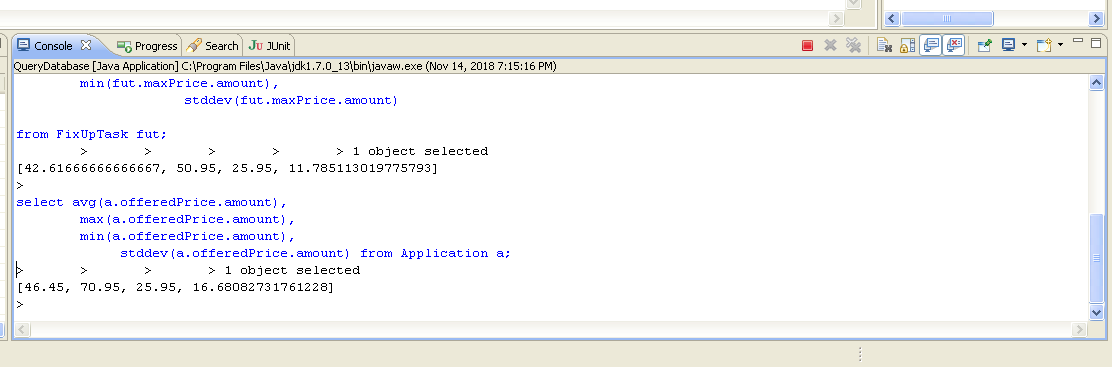
max(a.offeredPrice.amount),

min(a.offeredPrice.amount),

stddev(a.offeredPrice.amount) from Application a;

**Description:**

**We solve this query using default functions from JPQL, in this case the functions are avg(average), max(maximum), min(minimum) and stddev(standard deviation). Iterating over the price offered in the applications.**



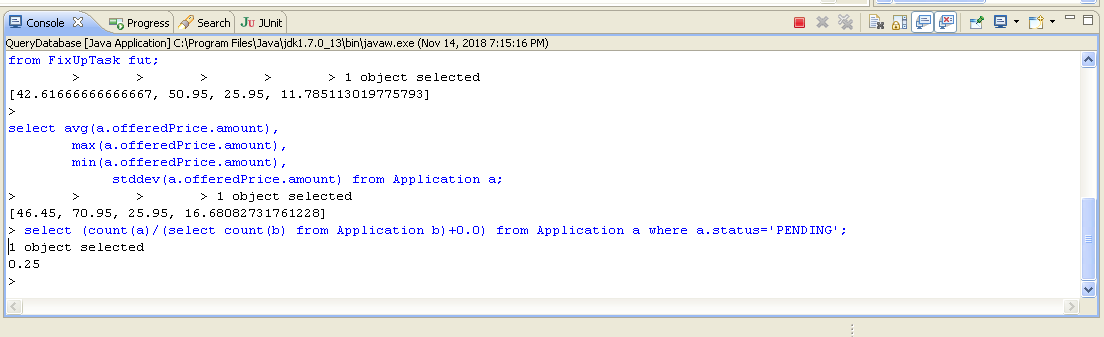
**Query C/5:**

**The ratio of pending applications.**

select (count(a)/(select count(b) from Application b)+0.0) from Application a where a.status='PENDING';

**Description:**

**We solve this query using default functions from JPQL, in this case we calculate the ratio by counting the applications with status ‘pending’ divided by the total amount of applications.**

****

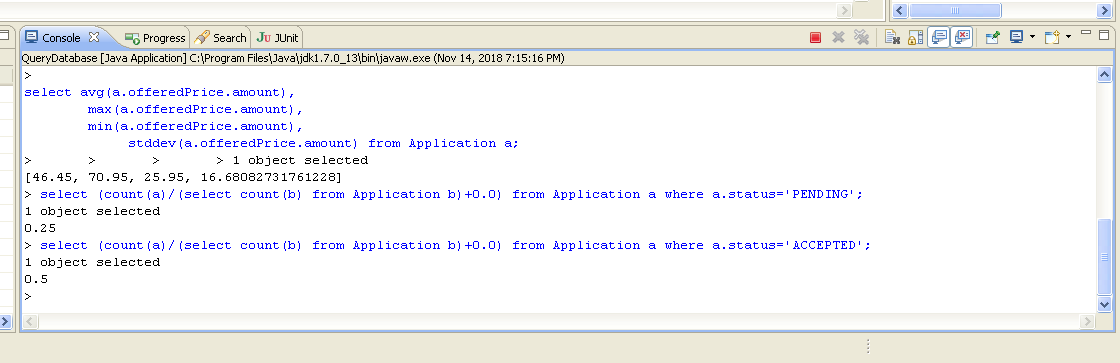
**Query C/6:**

**The ratio of accepted applications.**

select (count(a)/(select count(b) from Application b)+0.0) from Application a where a.status='ACCEPTED';

**Description:**

**We solve this query using default functions from JPQL, in this case we calculate the ratio by counting the applications with status ‘accepted’ divided by the total amount of applications.**



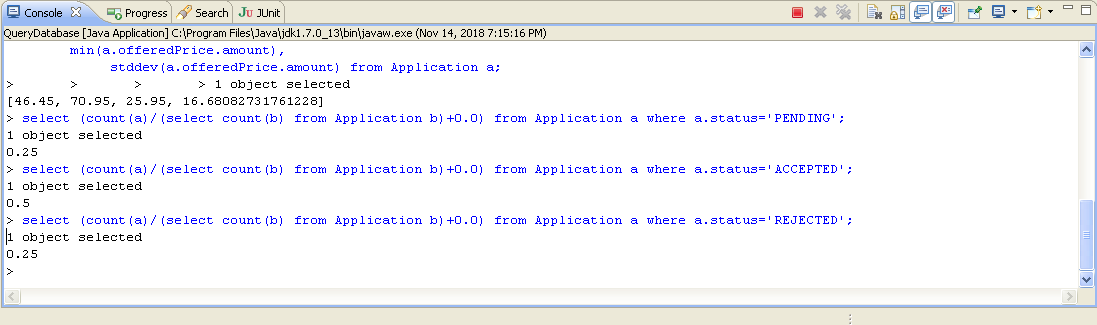
**Query C/7:**

**The ratio of rejected applications**.

select (count(a)/(select count(b) from Application b)+0.0) from Application a where a.status='REJECTED';

**Description:**

**We solve this query using default functions from JPQL, in this case we calculate the ratio by counting the applications with status ‘rejected’ divided by the total amount of applications.**



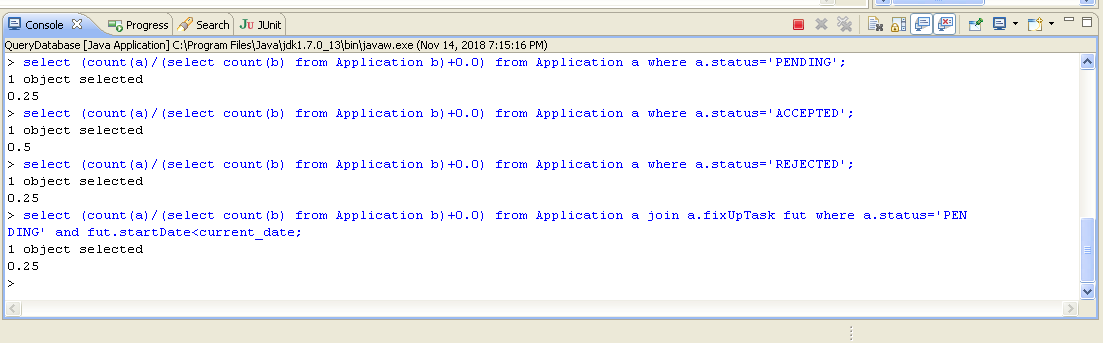
**Query C/8:**

**The ratio of pending applications that cannot change its status because their time period’s elapsed.**

select (count(a)/(select count(b) from Application b)+0.0) from Application a join a.fixUpTask fut where a.status='PENDING' and fut.startDate<current\_date;

**Description:**

**We solve this query using default functions from JPQL, in this case we calculate the ratio by counting the applications with status ‘pending’ that their time period has elapsed divided by the total amount of applications.**



**Query C/9:**

**The listing of customers who have published at least 10% more fix-up tasks than the average, ordered by number of applications.**

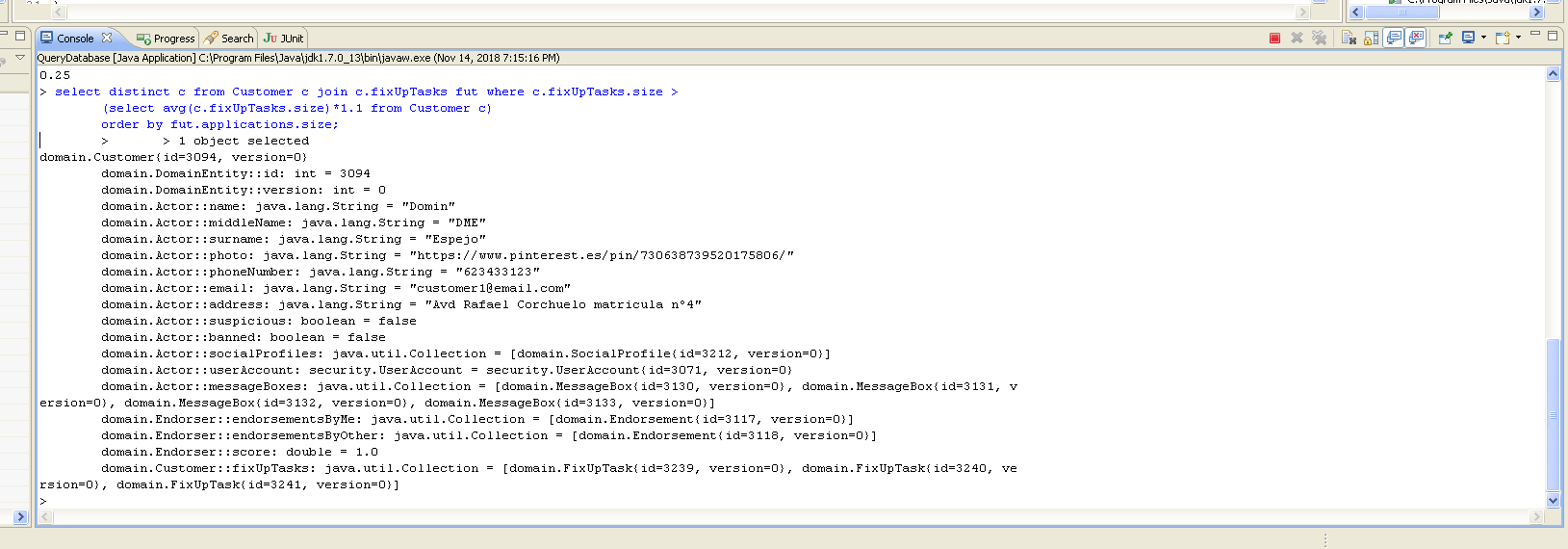
select distinct c from Customer c join c.fixUpTasks fut where c.fixUpTasks.size >

(select avg(c.fixUpTasks.size)\*1.1 from Customer c)

order by fut.applications.size;

**Description:**

**We solve this query using default functions from JPQL, in this case we select the customers that have 10% more fix-up tasks assigned than the average and order them by their number of applications.**



**Query C/10:**

**The listing of handy workers who have got accepted at least 10% more applications than the average, ordered by number of applications.**

select distinct hw from HandyWorker hw join hw.applications a where

(select count(\*) from Application a where a.handyWorker=hw and

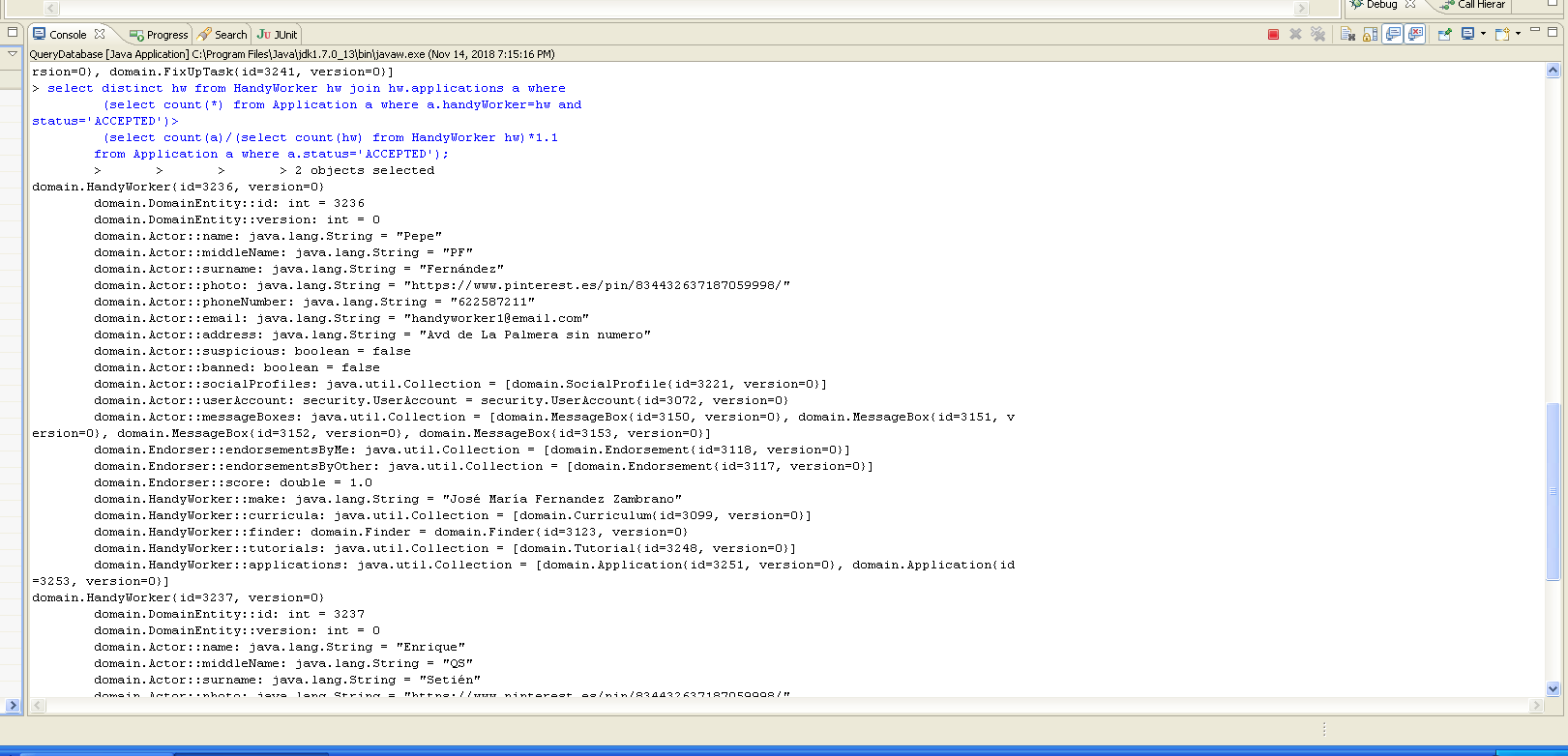
status='ACCEPTED')>

(select count(a)/(select count(hw) from HandyWorker hw)\*1.1

from Application a where a.status='ACCEPTED');

**Description:**

**We solve this query using default functions from JPQL, in this case we select the handy workers that have 10% more fix-up tasks assigned than the average and order them by their number of applications.**



**Query B/1:**

**The minimum, the maximum, the average, and the standard deviation of the number of complaints per fix-up task.**

select avg(fut.complaints.size),

max(fut.complaints.size),

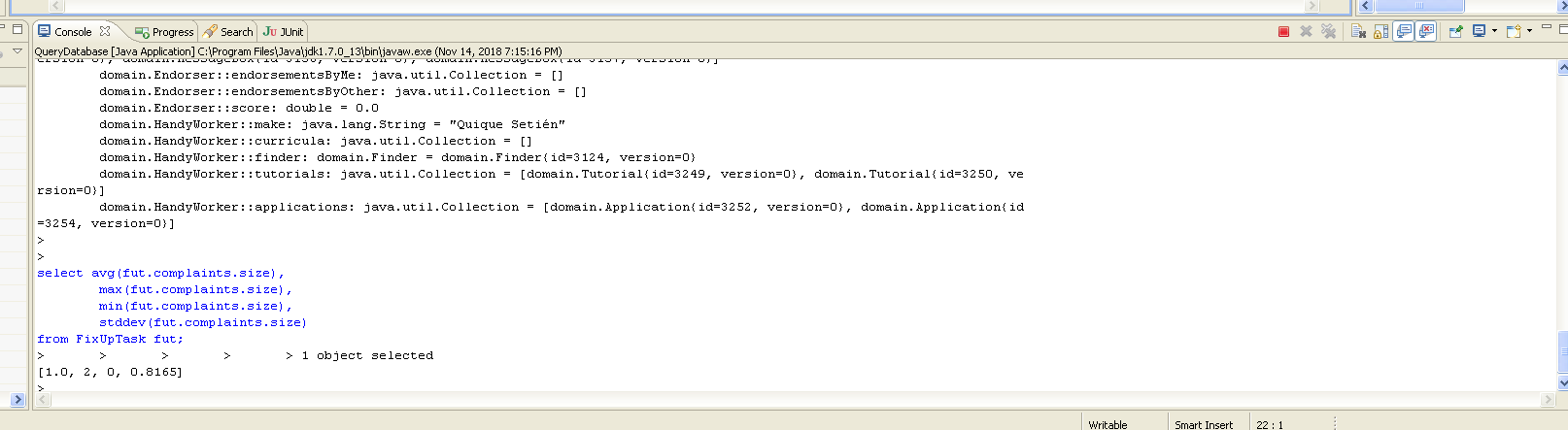
min(fut.complaints.size),

stddev(fut.complaints.size)

from FixUpTask fut;

**Description:**

**We solve this query using default functions from JPQL, in this case the functions are avg(average), max(maximum), min(minimum) and stddev(standard deviation). Iterating over the amount of complaints per fix-up tasks.**



**Query B/2:**

**The minimum, the maximum, the average, and the standard deviation of the number of notes per referee report.**

select avg(r.notes.size),

max(r.notes.size),

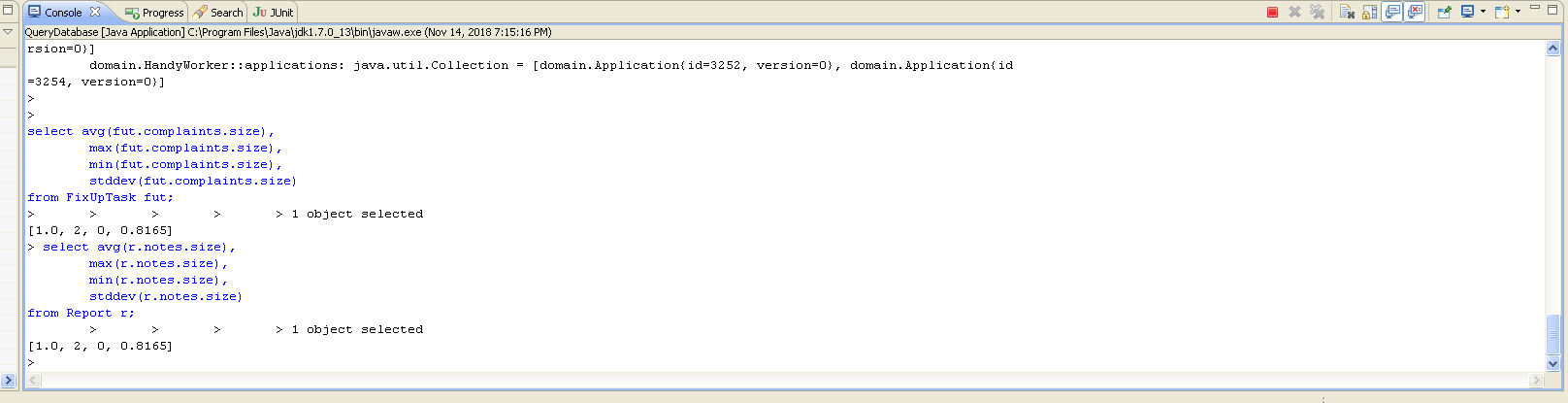
min(r.notes.size),

stddev(r.notes.size)

from Report r;

**Description:**

**We solve this query using default functions from JPQL, in this case the functions are avg(average), max(maximum), min(minimum) and stddev(standard deviation). Iterating over the amount of notes per referee report.**



**Query B/3:**

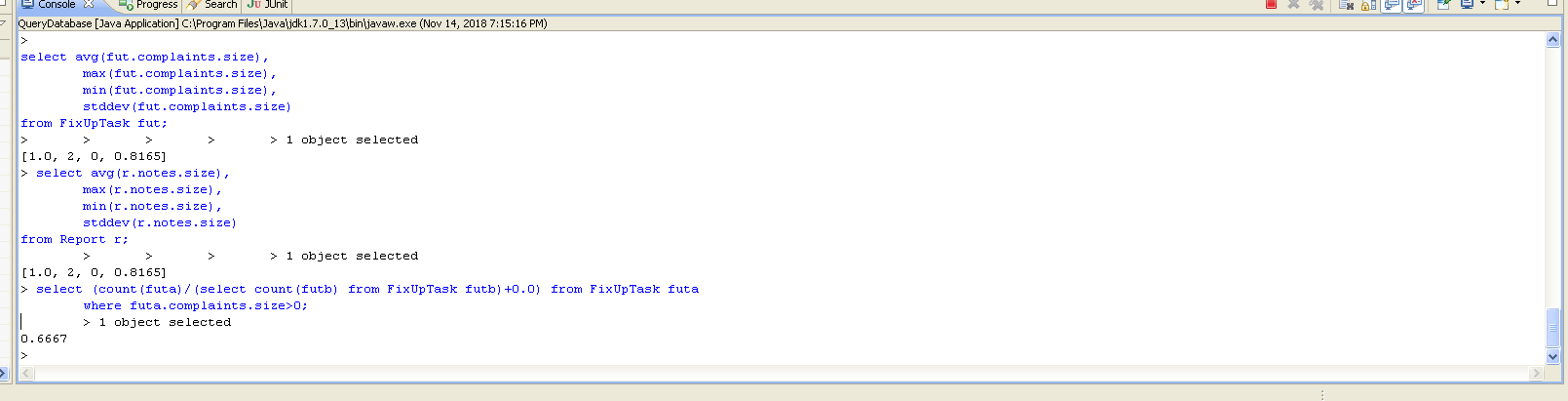
**The ratio of fix-up tasks with a complaint.**

select (count(futa)/(select count(futb) from FixUpTask futb)+0.0) from FixUpTask futa

where futa.complaints.size>0;

**Description:**

**We solve this query using default functions from JPQL, in this case we calculate the ratio by counting the fix-up tasks that have at least one complaint divided by the total amount of applications.**

****

**Query B/4:**

**The top-three customers in terms of complaints.**

select c from Customer c join c.fixUpTasks fut

group by c order by sum(fut.complaints.size) desc;

(HAY QUE LIMITARLO A 3 RESULTADOS)

**Description:**

**We solve this query using default functions from JPQL, in this case we select the three customers with the largests amounts of complaints associated.**

Imagen que contiene captura de pantalla

Descripción generada automáticamente

**Query B/5:**

**The top-three handy workers in terms of complaints.**

select hw from HandyWorker hw join hw.applications a join a.fixUpTask fut

where a.status='ACCEPTED' group by hw order by sum(fut.complaints.size) desc;

(HAY QUE LIMITARLO A 3 RESULTADOS)

**Description:**

**We solve this query using default functions from JPQL, in this case we select the three handy workers with the largests amounts of complaints associated.**

Imagen que contiene captura de pantalla

Descripción generada automáticamente