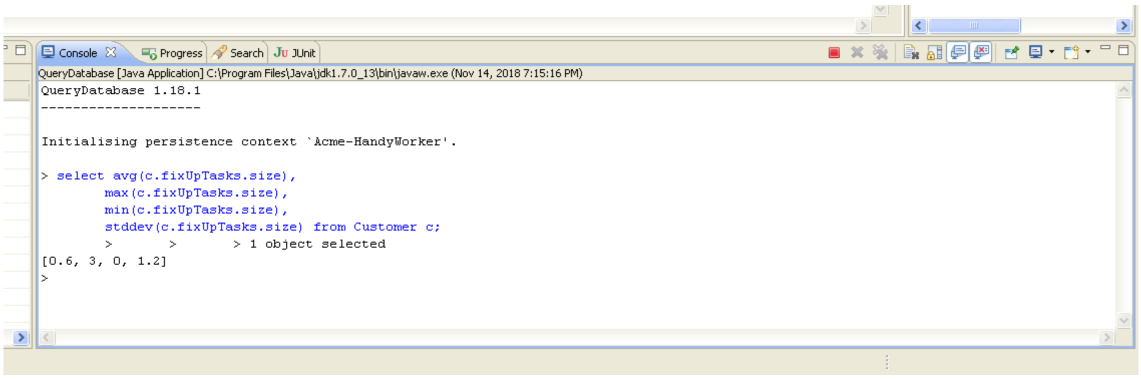
<u>Query C/1:</u>

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.



```
QueryDatabase [Java Application] C:\Program Files\Java\jdk1.7.0_13\bin\javaw.exe (Nov 14, 2018 7:15:16 PM)
QueryDatabase 1.18.1
--------------------

Initialising persistence context `Acme-HandyWorker'.

> select avg(c.fixUpTasks.size),
        max(c.fixUpTasks.size),
        min(c.fixUpTasks.size),
        stddev(c.fixUpTasks.size) from Customer c;
        >       >       > 1 object selected
[0.6, 3, 0, 1.2]
>
```

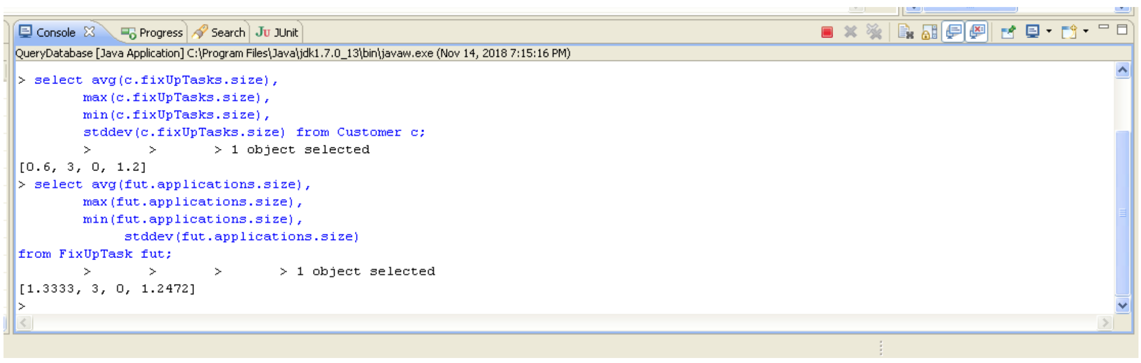
<u>Query C/2:</u>

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.



```
QueryDatabase [Java Application] C:\Program Files\Java\jdk1.7.0_13\bin\javaw.exe (Nov 14, 2018 7:15:16 PM)
> select avg(c.fixUpTasks.size),
        max(c.fixUpTasks.size),
        min(c.fixUpTasks.size),
        stddev(c.fixUpTasks.size) from Customer c;
        >       >       > 1 object selected
[0.6, 3, 0, 1.2]
> select avg(fut.applications.size),
        max(fut.applications.size),
        min(fut.applications.size),
             stddev(fut.applications.size)
from FixUpTask fut;
        >       >       >       > 1 object selected
[1.3333, 3, 0, 1.2472]
>
```

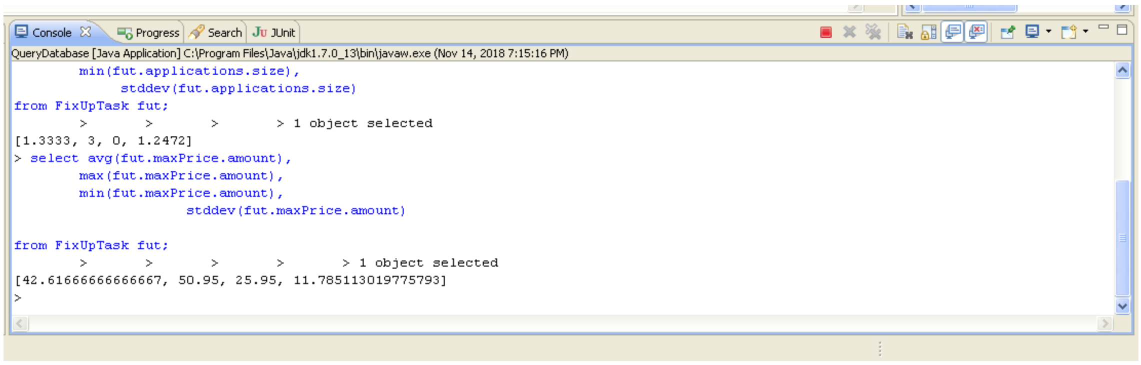
Query C/3:

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.



```
QueryDatabase [Java Application] C:\Program Files\Java\jdk1.7.0_13\bin\javaw.exe (Nov 14, 2018 7:15:16 PM)
        min(fut.applications.size),
             stddev(fut.applications.size)
from FixUpTask fut;
        >       >       >       > 1 object selected
[1.3333, 3, 0, 1.2472]
> select avg(fut.maxPrice.amount),
        max(fut.maxPrice.amount),
        min(fut.maxPrice.amount),
                     stddev(fut.maxPrice.amount)

from FixUpTask fut;
        >       >       >       >       > 1 object selected
[42.61666666666667, 50.95, 25.95, 11.785113019775793]
>
```

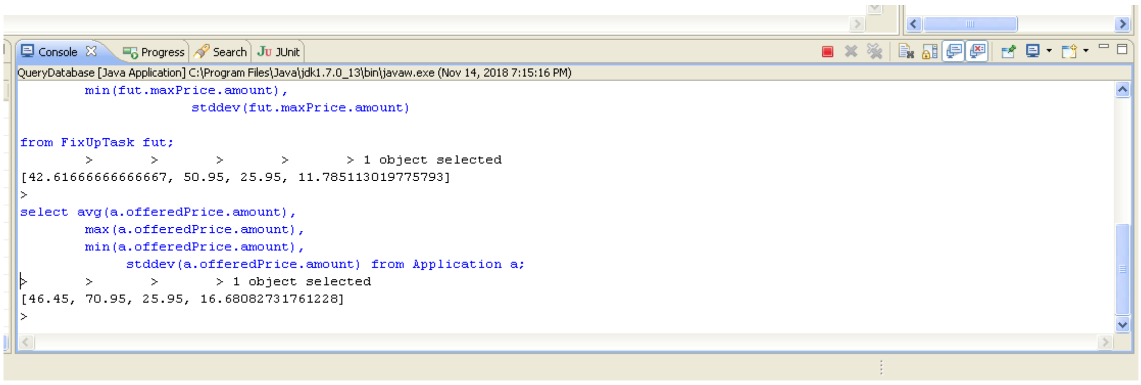
Query C/4:

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.



Console | Progress | Search | JUnit

QueryDatabase [Java Application] C:\Program Files\Java\jdk1.7.0_13\bin\javaw.exe (Nov 14, 2018 7:15:16 PM)

```
        min(fut.maxPrice.amount),
                     stddev(fut.maxPrice.amount)

from FixUpTask fut;
        >       >       >       >       > 1 object selected
[42.61666666666667, 50.95, 25.95, 11.785113019775793]
>
select avg(a.offeredPrice.amount),
        max(a.offeredPrice.amount),
        min(a.offeredPrice.amount),
             stddev(a.offeredPrice.amount) from Application a;
>       >       >       > 1 object selected
[46.45, 70.95, 25.95, 16.68082731761228]
>
```

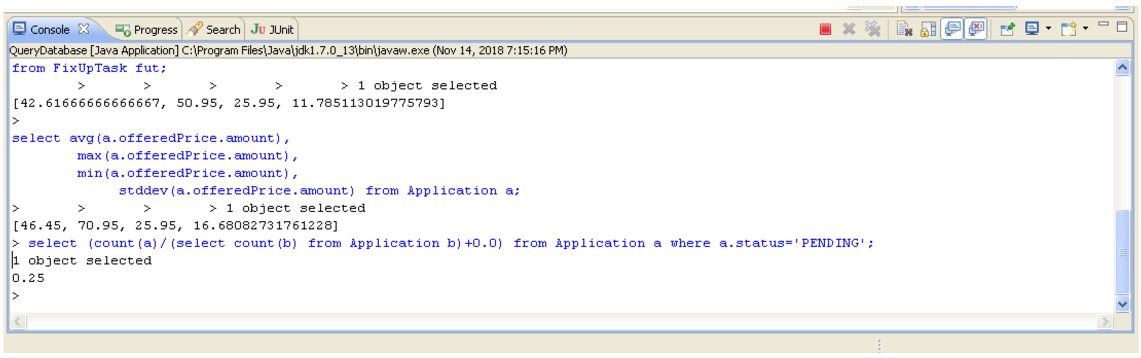
Query C/5:

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.



Console | Progress | Search | JUnit

QueryDatabase [Java Application] C:\Program Files\Java\jdk1.7.0_13\bin\javaw.exe (Nov 14, 2018 7:15:16 PM)

```
from FixUpTask fut;
        >       >       >       >       > 1 object selected
[42.61666666666667, 50.95, 25.95, 11.785113019775793]
>
select avg(a.offeredPrice.amount),
        max(a.offeredPrice.amount),
        min(a.offeredPrice.amount),
             stddev(a.offeredPrice.amount) from Application a;
>       >       >       > 1 object selected
[46.45, 70.95, 25.95, 16.68082731761228]
> select (count(a)/(select count(b) from Application b)+0.0) from Application a where a.status='PENDING';
1 object selected
0.25
>
```

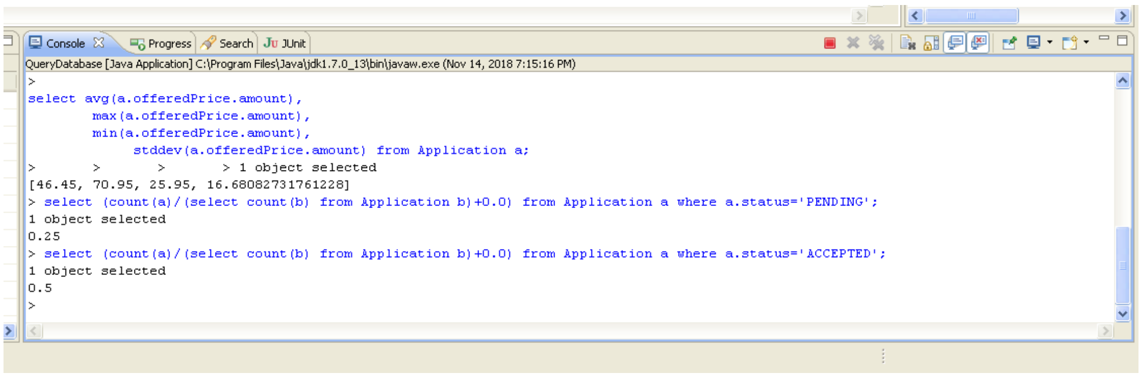
Query C/6:

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.



QueryDatabase [Java Application] C:\Program Files\Java\jdk1.7.0_13\bin\javaw.exe (Nov 14, 2018 7:15:16 PM)

```
>
select avg(a.offeredPrice.amount),
        max(a.offeredPrice.amount),
        min(a.offeredPrice.amount),
             stddev(a.offeredPrice.amount) from Application a;
>       >       >       > 1 object selected
[46.45, 70.95, 25.95, 16.680827317612 28]
> select (count(a)/(select count(b) from Application b)+0.0) from Application a where a.status='PENDING';
1 object selected
0.25
> select (count(a)/(select count(b) from Application b)+0.0) from Application a where a.status='ACCEPTED';
1 object selected
0.5
>
```

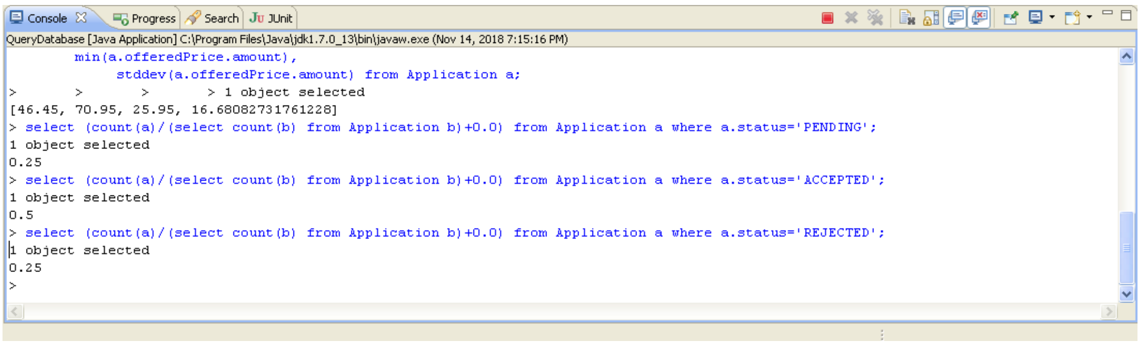
Query C/7:

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.



QueryDatabase [Java Application] C:\Program Files\Java\jdk1.7.0_13\bin\javaw.exe (Nov 14, 2018 7:15:16 PM)

```
        min(a.offeredPrice.amount),
             stddev(a.offeredPrice.amount) from Application a;
>       >       >       > 1 object selected
[46.45, 70.95, 25.95, 16.680827317612 28]
> select (count(a)/(select count(b) from Application b)+0.0) from Application a where a.status='PENDING';
1 object selected
0.25
> select (count(a)/(select count(b) from Application b)+0.0) from Application a where a.status='ACCEPTED';
1 object selected
0.5
> select (count(a)/(select count(b) from Application b)+0.0) from Application a where a.status='REJECTED';
1 object selected
0.25
>
```

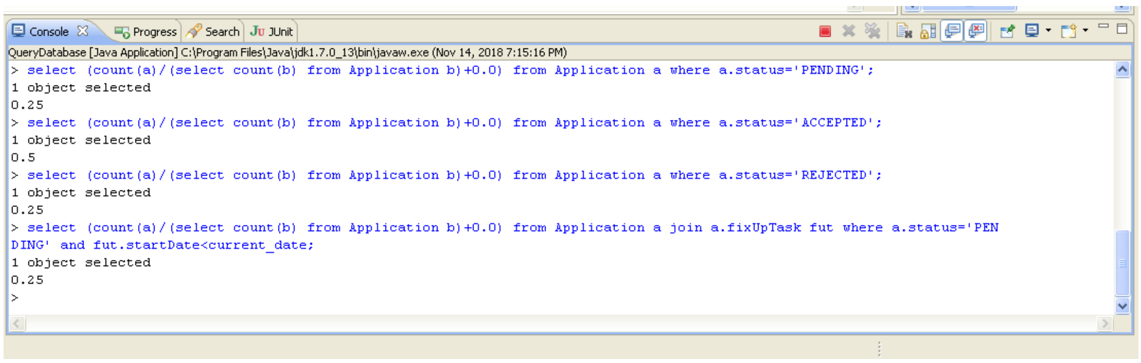
Query C/8:

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.



QueryDatabase [Java Application] C:\Program Files\Java\jdk1.7.0_13\bin\javaw.exe (Nov 14, 2018 7:15:16 PM)

```
> select (count(a)/(select count(b) from Application b)+0.0) from Application a where a.status='PENDING';
1 object selected
0.25
> select (count(a)/(select count(b) from Application b)+0.0) from Application a where a.status='ACCEPTED';
1 object selected
0.5
> select (count(a)/(select count(b) from Application b)+0.0) from Application a where a.status='REJECTED';
1 object selected
0.25
> select (count(a)/(select count(b) from Application b)+0.0) from Application a join a.fixUpTask fut where a.status='PEN
DING' and fut.startDate<current_date;
1 object selected
0.25
>
```

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.

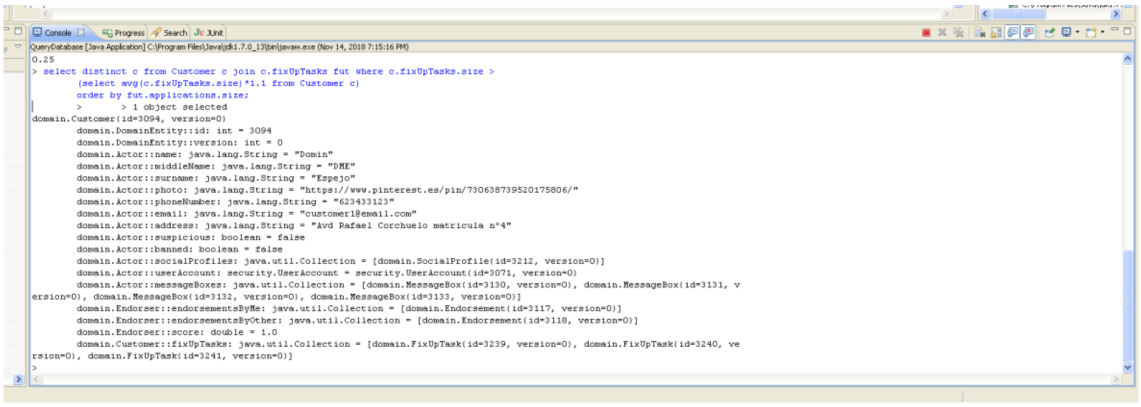
select distinct c from Customer c join c.fixUpTasks fut where c.fixUpTasks.size >

(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.



QueryDatabase [Java Application] C:\Program Files\Java\jdk1.7.0_13\bin\javaw.exe (Nov 14, 2018 7:15:16 PM)

```
0.25
> 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;
        >       > 1 object selected
domain.Customer{id=3094, version=0}
        domain.DomainEntity::id: int = 3094
        domain.DomainEntity::version: int = 0
        domain.Actor::name: java.lang.String = "Domin"
        domain.Actor::middleName: java.lang.String = "DME"
        domain.Actor::surname: java.lang.String = "Espejo"
        domain.Actor::photo: java.lang.String = "https://www.pinterest.es/pin/730638739520175806/"
        domain.Actor::phoneNumber: java.lang.String = "623433123"
        domain.Actor::email: java.lang.String = "customer1@email.com"
        domain.Actor::address: java.lang.String = "Avd Rafael Corchuelo matricula n°4"
        domain.Actor::suspicious: boolean = false
        domain.Actor::banned: boolean = false
        domain.Actor::socialProfiles: java.util.Collection = [domain.SocialProfile{id=3212, version=0}]
        domain.Actor::userAccount: security.UserAccount = security.UserAccount{id=3071, version=0}
        domain.Actor::messageBoxes: java.util.Collection = [domain.MessageBox{id=3130, version=0}, domain.MessageBox{id=3131, v
ersion=0}, domain.MessageBox{id=3132, version=0}, domain.MessageBox{id=3133, version=0}]
        domain.Endorser::endorsementsByMe: java.util.Collection = [domain.Endorsement{id=3117, version=0}]
        domain.Endorser::endorsementsByOther: java.util.Collection = [domain.Endorsement{id=3118, version=0}]
        domain.Endorser::score: double = 1.0
        domain.Customer::fixUpTasks: java.util.Collection = [domain.FixUpTask{id=3239, version=0}, domain.FixUpTask{id=3240, ve
rsion=0}, domain.FixUpTask{id=3241, version=0}]
>
```

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

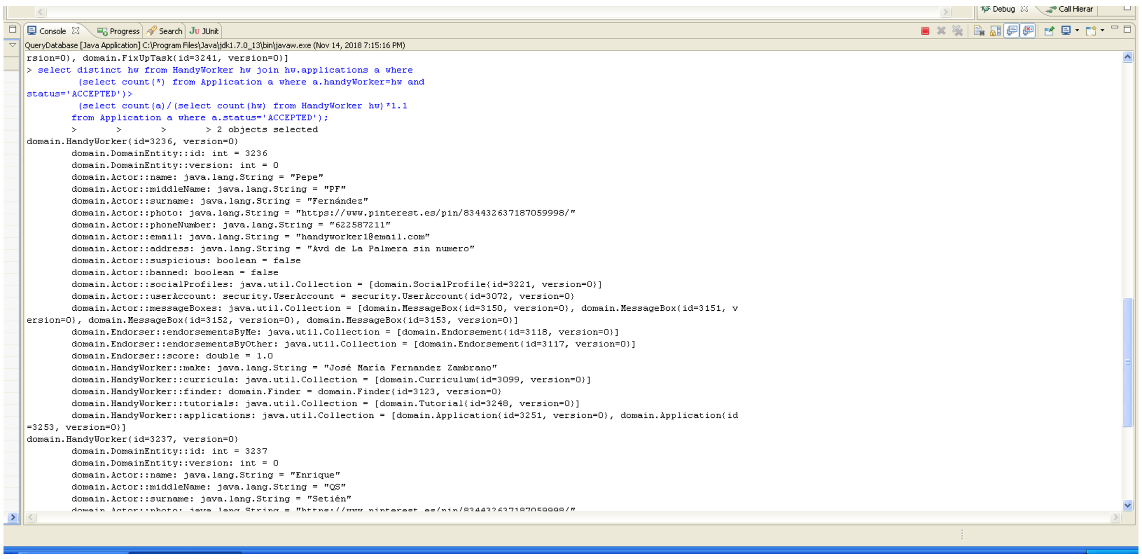
status='ACCEPTED')>

(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.



```
QueryDatabase [Java Application] C:\Program Files\Java\jdk1.7.0_13\bin\javaw.exe (Nov 14, 2018 7:15:16 PM)
rsion=0}, domain.FixUpTask{id=3241, version=0}]
> 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');
        >       >       >       > 2 objects selected
domain.HandyWorker{id=3236, version=0}
        domain.DomainEntity::id: int = 3236
        domain.DomainEntity::version: int = 0
        domain.Actor::name: java.lang.String = "Pepe"
        domain.Actor::middleName: java.lang.String = "PF"
        domain.Actor::surname: java.lang.String = "Fernández"
        domain.Actor::photo: java.lang.String = "https://www.pinterest.es/pin/834432637187059998/"
        domain.Actor::phoneNumber: java.lang.String = "622587211"
        domain.Actor::email: java.lang.String = "handyworker1@email.com"
        domain.Actor::address: java.lang.String = "Avd de La Palmera sin numero"
        domain.Actor::suspicious: boolean = false
        domain.Actor::banned: boolean = false
        domain.Actor::socialProfiles: java.util.Collection = [domain.SocialProfile{id=3221, version=0}]
        domain.Actor::userAccount: security.UserAccount = security.UserAccount{id=3072, version=0}
        domain.Actor::messageBoxes: java.util.Collection = [domain.MessageBox{id=3150, version=0}, domain.MessageBox{id=3151, v
ersion=0}, domain.MessageBox{id=3152, version=0}, domain.MessageBox{id=3153, version=0}]
        domain.Endorser::endorsementsByMe: java.util.Collection = [domain.Endorsement{id=3118, version=0}]
        domain.Endorser::endorsementsByOther: java.util.Collection = [domain.Endorsement{id=3117, version=0}]
        domain.Endorser::score: double = 1.0
        domain.HandyWorker::make: java.lang.String = "José María Fernandez Zambrano"
        domain.HandyWorker::curricula: java.util.Collection = [domain.Curriculum{id=3099, version=0}]
        domain.HandyWorker::finder: domain.Finder = domain.Finder{id=3123, version=0}
        domain.HandyWorker::tutorials: java.util.Collection = [domain.Tutorial{id=3248, version=0}]
        domain.HandyWorker::applications: java.util.Collection = [domain.Application{id=3251, version=0}, domain.Application{id
=3253, version=0}]
domain.HandyWorker{id=3237, version=0}
        domain.DomainEntity::id: int = 3237
        domain.DomainEntity::version: int = 0
        domain.Actor::name: java.lang.String = "Enrique"
        domain.Actor::middleName: java.lang.String = "QS"
        domain.Actor::surname: java.lang.String = "Setién"
```

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),

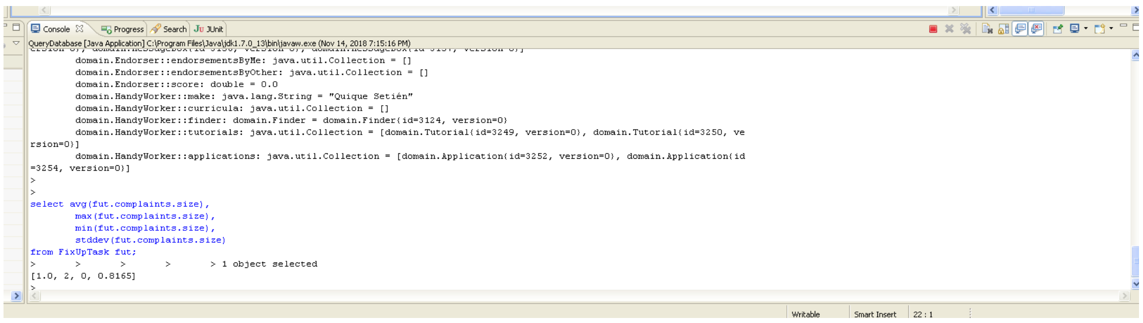
min(fut.complaints.size),

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.



QueryDatabase [Java Application] C:\Program Files\Java\jdk1.7.0_13\bin\javaw.exe (Nov 14, 2018 7:15:16 PM)

```
        domain.Endorser::endorsementsByMe: java.util.Collection = []
        domain.Endorser::endorsementsByOther: java.util.Collection = []
        domain.Endorser::score: double = 0.0
        domain.HandyWorker::make: java.lang.String = "Quique Setién"
        domain.HandyWorker::curricula: java.util.Collection = []
        domain.HandyWorker::finder: domain.Finder = domain.Finder{id=3124, version=0}
        domain.HandyWorker::tutorials: java.util.Collection = [domain.Tutorial{id=3249, version=0}, domain.Tutorial{id=3250, ve
rsion=0}]
        domain.HandyWorker::applications: java.util.Collection = [domain.Application{id=3252, version=0}, domain.Application{id
=3254, version=0}]
>
>
select avg(fut.complaints.size),
        max(fut.complaints.size),
        min(fut.complaints.size),
        stddev(fut.complaints.size)
from FixUpTask fut;
>       >       >       >       > 1 object selected
[1.0, 2, 0, 0.8165]
>
```

Writable   Smart Insert   22 : 1

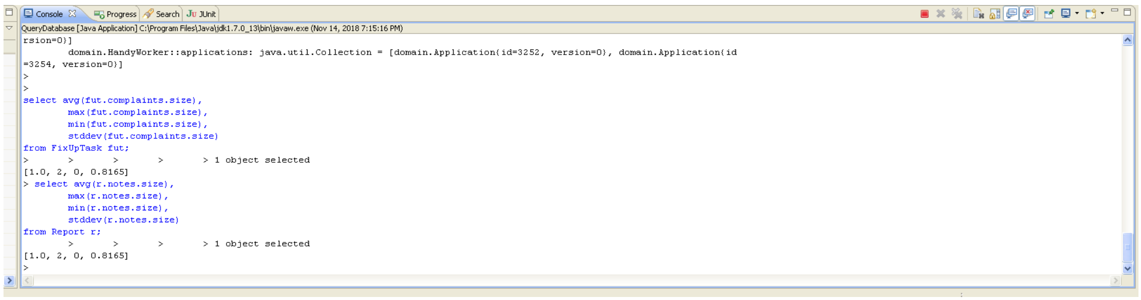
Query B/2:

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.



QueryDatabase [Java Application] C:\Program Files\Java\jdk1.7.0_13\bin\javaw.exe (Nov 14, 2018 7:15:16 PM)

```
rsion=0}]
        domain.HandyWorker::applications: java.util.Collection = [domain.Application{id=3252, version=0}, domain.Application{id
=3254, version=0}]
>
>
select avg(fut.complaints.size),
        max(fut.complaints.size),
        min(fut.complaints.size),
        stddev(fut.complaints.size)
from FixUpTask fut;
>       >       >       >       > 1 object selected
[1.0, 2, 0, 0.8165]
> select avg(r.notes.size),
        max(r.notes.size),
        min(r.notes.size),
        stddev(r.notes.size)
from Report r;
        >       >       >       > 1 object selected
[1.0, 2, 0, 0.8165]
>
```

Query B/3:

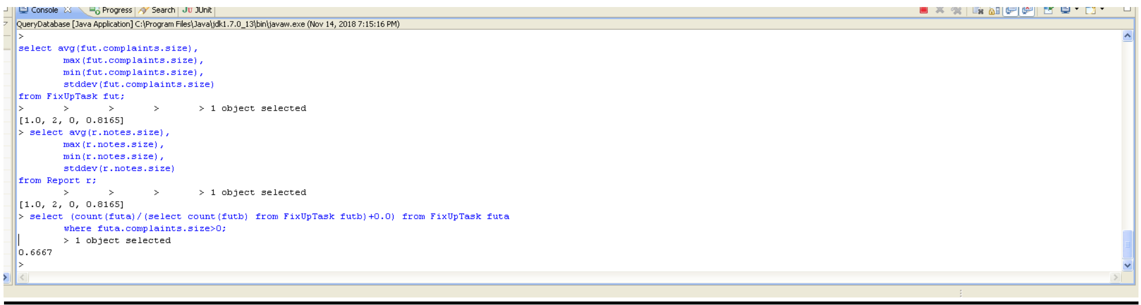
The ratio of fix-up tasks with a complaint.

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.



```
QueryDatabase [Java Application] C:\Program Files\Java\jdk1.7.0_13\bin\javaw.exe (Nov 14, 2018 7:15:16 PM)
>
select avg(fut.complaints.size),
        max(fut.complaints.size),
        min(fut.complaints.size),
        stddev(fut.complaints.size)
from FixUpTask fut;
>       >       >       >       > 1 object selected
[1.0, 2, 0, 0.8165]
> select avg(r.notes.size),
        max(r.notes.size),
        min(r.notes.size),
        stddev(r.notes.size)
from Report r;
        >       >       >       > 1 object selected
[1.0, 2, 0, 0.8165]
> select (count(futa)/(select count(futb) from FixUpTask futb)+0.0) from FixUpTask futa
        where futa.complaints.size>0;
        > 1 object selected
0.6667
>
```

Query B/4:

The top-three customers in terms of complaints.

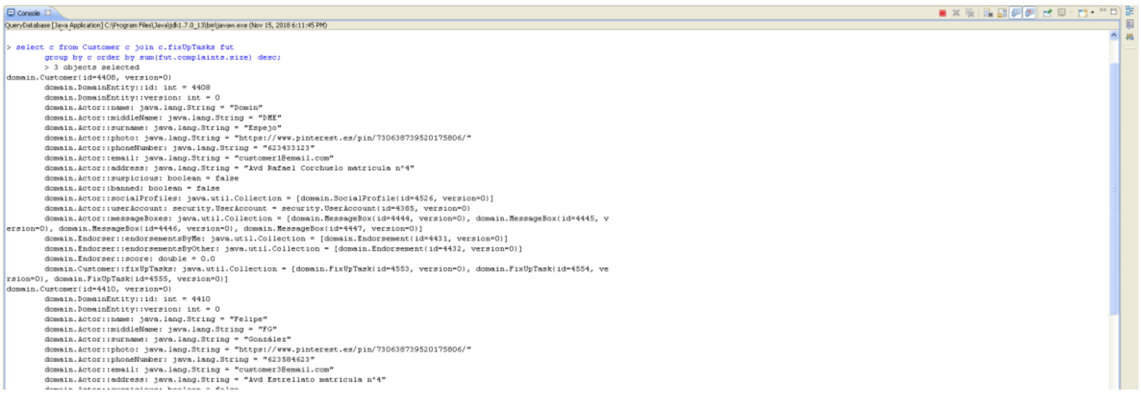
select c from Customer c join c.fixUpTasks fut

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.



```
QueryDatabase [Java Application] C:\Program Files\Java\jdk1.7.0_13\bin\javaw.exe (Nov 15, 2018 6:11:45 PM)

> select c from Customer c join c.fixUpTasks fut
        group by c order by sum(fut.complaints.size) desc;
        > 3 objects selected
domain.Customer(id=4408, version=0)
        domain.DomainEntity::id: int = 4408
        domain.DomainEntity::version: int = 0
        domain.Actor::name: java.lang.String = "Domin"
        domain.Actor::middleName: java.lang.String = "DME"
        domain.Actor::surname: java.lang.String = "Espejo"
        domain.Actor::photo: java.lang.String = "https://www.pinterest.es/pin/730638739520175806/"
        domain.Actor::phoneNumber: java.lang.String = "623433123"
        domain.Actor::email: java.lang.String = "customer1@email.com"
        domain.Actor::address: java.lang.String = "Avd Rafael Corchuelo matricula n°4"
        domain.Actor::suspicious: boolean = false
        domain.Actor::banned: boolean = false
        domain.Actor::socialProfiles: java.util.Collection = [domain.SocialProfile(id=4526, version=0)]
        domain.Actor::userAccount: security.UserAccount = security.UserAccount(id=4385, version=0)
        domain.Actor::messageBoxes: java.util.Collection = [domain.MessageBox(id=4444, version=0), domain.MessageBox(id=4445, v
ersion=0), domain.MessageBox(id=4446, version=0), domain.MessageBox(id=4447, version=0)]
        domain.Endorser::endorsementsByMe: java.util.Collection = [domain.Endorsement(id=4431, version=0)]
        domain.Endorser::endorsementsByOther: java.util.Collection = [domain.Endorsement(id=4432, version=0)]
        domain.Endorser::score: double = 0.0
        domain.Customer::fixUpTasks: java.util.Collection = [domain.FixUpTask(id=4553, version=0), domain.FixUpTask(id=4554, ve
rsion=0), domain.FixUpTask(id=4555, version=0)]
domain.Customer(id=4410, version=0)
        domain.DomainEntity::id: int = 4410
        domain.DomainEntity::version: int = 0
        domain.Actor::name: java.lang.String = "Felipe"
        domain.Actor::middleName: java.lang.String = "FG"
        domain.Actor::surname: java.lang.String = "González"
        domain.Actor::photo: java.lang.String = "https://www.pinterest.es/pin/730638739520175806/"
        domain.Actor::phoneNumber: java.lang.String = "623584623"
        domain.Actor::email: java.lang.String = "customer3@email.com"
        domain.Actor::address: java.lang.String = "Avd Estrellato matricula n°4"
```

Query B/5:

The top-three handy workers in terms of complaints.

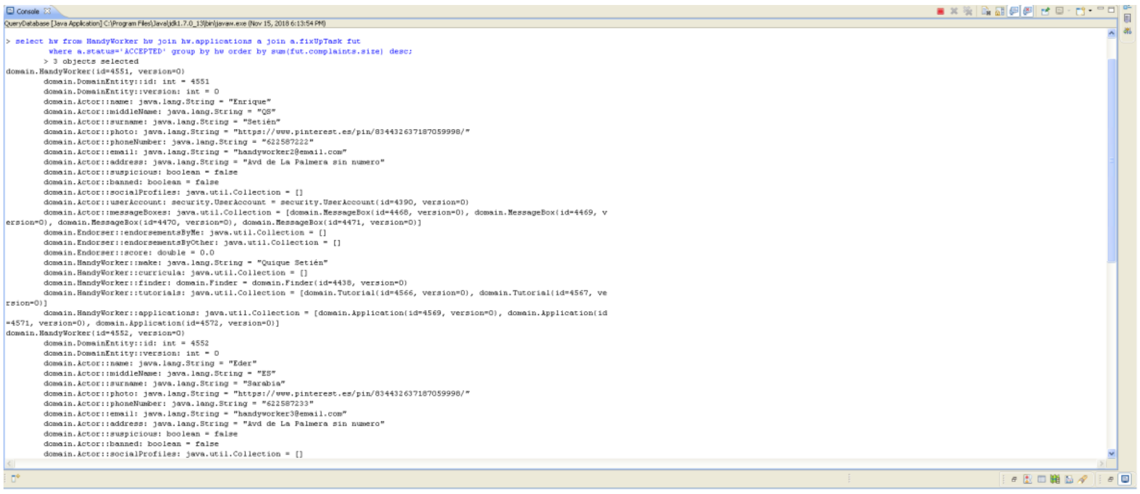
select hw from HandyWorker hw join hw.applications a join a.fixUpTask fut

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.



Console

QueryDatabase [Java Application] C:\Program Files\Java\jdk1.7.0_13\bin\javaw.exe (Nov 15, 2018 6:13:54 PM)

```
> 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;
        > 3 objects selected
domain.HandyWorker{id=4551, version=0}
        domain.DomainEntity::id: int = 4551
        domain.DomainEntity::version: int = 0
        domain.Actor::name: java.lang.String = "Enrique"
        domain.Actor::middleName: java.lang.String = "QS"
        domain.Actor::surname: java.lang.String = "Setién"
        domain.Actor::photo: java.lang.String = "https://www.pinterest.es/pin/834432637187059998/"
        domain.Actor::phoneNumber: java.lang.String = "622587222"
        domain.Actor::email: java.lang.String = "handyworker2@email.com"
        domain.Actor::address: java.lang.String = "Avd de La Palmera sin numero"
        domain.Actor::suspicious: boolean = false
        domain.Actor::banned: boolean = false
        domain.Actor::socialProfiles: java.util.Collection = []
        domain.Actor::userAccount: security.UserAccount = security.UserAccount{id=4390, version=0}
        domain.Actor::messageBoxes: java.util.Collection = [domain.MessageBox{id=4468, version=0}, domain.MessageBox{id=4469, v
ersion=0}, domain.MessageBox{id=4470, version=0}, domain.MessageBox{id=4471, version=0}]
        domain.Endorser::endorsementsByMe: java.util.Collection = []
        domain.Endorser::endorsementsByOther: java.util.Collection = []
        domain.Endorser::score: double = 0.0
        domain.HandyWorker::make: java.lang.String = "Quique Setién"
        domain.HandyWorker::curricula: java.util.Collection = []
        domain.HandyWorker::finder: domain.Finder = domain.Finder{id=4438, version=0}
        domain.HandyWorker::tutorials: java.util.Collection = [domain.Tutorial{id=4566, version=0}, domain.Tutorial{id=4567, ve
rsion=0}]
        domain.HandyWorker::applications: java.util.Collection = [domain.Application{id=4569, version=0}, domain.Application{id
=4571, version=0}, domain.Application{id=4572, version=0}]
domain.HandyWorker{id=4552, version=0}
        domain.DomainEntity::id: int = 4552
        domain.DomainEntity::version: int = 0
        domain.Actor::name: java.lang.String = "Eder"
        domain.Actor::middleName: java.lang.String = "ES"
        domain.Actor::surname: java.lang.String = "Sarabia"
        domain.Actor::photo: java.lang.String = "https://www.pinterest.es/pin/834432637187059998/"
        domain.Actor::phoneNumber: java.lang.String = "622587233"
        domain.Actor::email: java.lang.String = "handyworker3@email.com"
        domain.Actor::address: java.lang.String = "Avd de La Palmera sin numero"
        domain.Actor::suspicious: boolean = false
        domain.Actor::banned: boolean = false
        domain.Actor::socialProfiles: java.util.Collection = []
```