



QUERY REPORT

Álvaro Calle González
Julia García Gallego
María Jiménez Vega
Antonio Nolé Anguita
Fernando Manuel Ruiz Pliego

Contenido

LEVEL B QUERIES..... 3

LEVEL C QUERIES..... 5

LEVEL B QUERIES

Query B/1: The average, the minimum, the maximum, and the standard deviation of the number of complaints per fix-up task.

```
select min(f.complaints.size), max(f.complaints.size), avg(f.complaints.size), sqrt(sum
(f.complaints.size * f.complaints.size) / count (f.complaints.size)- avg(f.complaints.size)
*avg(f.complaints.size)) from FixUpTask f;
```

This query calculates the average, the minimum, the maximum and the standard deviation of the number of complaints per fix-up task. To make this query max, min, avg, count and sqrt statistical functions have been used.

```
> select min(f.complaints.size), max(f.complaints.size), avg(f.complaints.size), sqrt(sum (f.complaints.size * f.complaints.size) / count (f.complaints.size)- avg(f.complaints.size) *avg(f.complaints.size)) from FixUpTask f;
1 object selected
[0, 3, 0.75, 1.0897247358851685]
```

Query B/2: The average, the minimum, the maximum, and the standard deviation of the number of notes per referee report.

```
select min(r.notes.size), max(r.notes.size), avg(r.notes.size), sqrt(sum (r.notes.size *
r.notes.size) / count (r.notes.size)- avg(r.notes.size) *avg(r.notes.size)) from Report r;
```

This query calculates the average, the minimum, the maximum and the standard deviation of the number of notes per referee report. To make this query max, min, avg, count and sqrt statistical functions have been used.

```
> select min(r.notes.size), max(r.notes.size), avg(r.notes.size), sqrt(sum (r.notes.size * r.notes.size) / count (r.notes.size)- avg(r.notes.size) *avg(r.notes.size)) from Report r;
1 object selected
[0, 2, 1.0, 0.7071067811865476]
```

Query B/3: The ratio of fix-up tasks with a complaint.

```
select count(c)/(select count(f) from FixUpTask f)*1.0 from FixUpTask c where c.complaints.size=1;
```

This query calculates the ratio of fix-up tasks with a complaint.

```
> select count(c)/(select count(f) from FixUpTask f)*1.0 from FixUpTask c where c.complaints.size=1;
1 object selected
0.125
```

Query B/4: The top-three customers in terms of complaints.

```
select c from Customer c Join c.fixUpTasks f group by f.customer order by f.complaints.size DESC;
```

This query calculates a list of customers sorted descending by their number of complaints. First the query performs a join to calculate all the tasks performed by a customer and then calculates all the complaints associated with each of the customer's tasks. Finally, with an order by order the customers according to the number of complaints they have descending.

```
> select c from Customer c Join c.fixUpTasks f group by f.customer order by f.complaints.size DESC;
5 objects selected
domain.Customer{id=9075, version=0}
  domain.DomainEntity::id: int = 9075
  domain.DomainEntity::version: int = 0
  domain.Actor::name: java.lang.String = "Maria"
  domain.Actor::middleName: java.lang.String = "Antonieta"
  domain.Actor::surname: java.lang.String = "San Mateo"
  domain.Actor::photoLink: java.lang.String = "http://www.link6.com"
  domain.Actor::email: java.lang.String = "customer6@gmail.com"
  domain.Actor::phoneNumber: java.lang.String = "636014701"
  domain.Actor::address: java.lang.String = "Address 6"
  domain.Actor::isSuspicious: boolean = false
  domain.Actor::userAccount: security.UserAccount = security.UserAccount{id=8983, version=0}
  domain.Customer::score: java.lang.Double = null
  domain.Customer::fixUpTasks: java.util.Collection = [domain.FixUpTask{id=9247, version=0}]
domain.Customer{id=9074, version=0}
  domain.DomainEntity::id: int = 9074
```

Query B/5: The top-three handy workers in terms of complaints.

```
select h from HandyWorker h join h.applications a join a.fixUpTask f group by a.handyWorker order by f.complaints.size DESC;
```

This query calculates a list of handy workers sorted descending by their number of complaints. First the query performs a join to calculate all the applications performed by a handy worker, second calculates all the tasks associated with each of the handy worker's applications and third calculates all the complaints associated with each of the handy worker's tasks. Finally, with an order by order the handy workers according to the number of complaints they have descending.

```
> select h from HandyWorker h join h.applications a join a.fixUpTask f group by a.handyWorker order by f.complaints.size
DESC;
6 objects selected
domain.HandyWorker(id=9116, version=0)
  domain.DomainEntity::id: int = 9116
  domain.DomainEntity::version: int = 0
  domain.Actor::name: java.lang.String = "Joanne"
  domain.Actor::middleName: java.lang.String = "Kathelyn"
  domain.Actor::surname: java.lang.String = "Rowling"
  domain.Actor::photoLink: java.lang.String = "http://www.link6.es"
  domain.Actor::email: java.lang.String = "handyworker6@gmail.com"
  domain.Actor::phoneNumber: java.lang.String = "654126078"
  domain.Actor::address: java.lang.String = "Address 6"
  domain.Actor::isSuspicious: boolean = false
  domain.Actor::userAccount: security.UserAccount = security.UserAccount(id=8988, version=0)
  domain.HandyWorker::make: java.lang.String = "Make 6"
  domain.HandyWorker::score: java.lang.Double = null
  domain.HandyWorker::finder: domain.Finder = domain.Finder(id=9110, version=0)
  domain.HandyWorker::curriculum: domain.Curriculum = null
  domain.HandyWorker::applications: java.util.Collection = [domain.Application(id=9265, version=0)]
domain.HandyWorker(id=9115, version=0)
  domain.DomainEntity::id: int = 9115
  domain.DomainEntity::version: int = 0
```

LEVEL C QUERIES

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),
min(c.fixUpTasks.size),
max(c.fixUpTasks.size),
sqrt(sum(c.fixUpTasks.size * c.fixUpTasks.size)/ count(c.fixUpTasks.size) -
avg(c.fixUpTasks.size)*avg(c.fixUpTasks.size))
from Customer c;
```

This query calculates the average, the minimum, the maximum and the standard deviation of the number of fix-up tasks per user. To make this query max, min, avg, count and sqrt statistical functions have been used.

```
Progress Search JUnit Console
QueryDatabase [Java Application] C:\Program Files\Java\jdk1.7.0_13\bin\javaw.exe (Nov 14, 2018 6:44:58 PM)
QueryDatabase 1.18.1
-----
Initialising persistence context 'Acme-HandyWorker'.

> select avg(c.fixUpTasks.size), min(c.fixUpTasks.size), max(c.fixUpTasks.size),
> sqrt(sum(c.fixUpTasks.size * c.fixUpTasks.size)/ count(c.fixUpTasks.size) -
> avg(c.fixUpTasks.size)*avg(c.fixUpTasks.size)) from Customer c;
1 object selected
[1.3333, 0, 3, 0.9428090416999145]
>
```

Query C/2: The average, the minimum, the maximum, and the standard deviation of the number of applications per fix-up task.

```
select avg(f.applications.size), min(f.applications.size), max(f.applications.size),
sqrt(sum(f.applications.size * f.applications.size)/count(f.applications.size) -
avg(f.applications.size)*avg(f.applications.size)) from FixUpTask f;
```

This query calculates the average, the minimum, the maximum and the standard deviation of the number of applications per fix-up tasks. To make this query max, min, avg, count and sqrt statistical functions have been used.

```
Progress Search JUnit Console
QueryDatabase [Java Application] C:\Program Files\Java\jdk1.7.0_13\bin\javaw.exe (Nov 14, 2018 6:53:25 PM)
QueryDatabase 1.18.1
-----
Initialising persistence context 'Acme-HandyWorker'.

> select avg(f.applications.size), min(f.applications.size), max(f.applications.size),
> sqrt(sum(f.applications.size * f.applications.size)/count(f.applications.size) -
> avg(f.applications.size)*avg(f.applications.size)) from FixUpTask f;
1 object selected
[1.5, 1, 2, 0.5]
>
```

Query C/3: The average, the minimum, the maximum, and the standard deviation of the maximum price of the fix-up tasks.

```
select avg(f.maxPrice), min(f.maxPrice),max(f.maxPrice),
sqrt(sum(f.maxPrice*f.maxPrice)/count(f.maxPrice)- avg(f.maxPrice)*avg(f.maxPrice)) from FixUpTask
f;
```

This query calculates the average, the minimum, the maximum and the standard deviation of the maximum price of the fix-up tasks. To make this query max, min, avg, count and sqrt statistical functions have been used.

```
Progress Search JUnit Console
QueryDatabase [Java Application] C:\Program Files\Java\jdk1.7.0_13\bin\javaw.exe (Nov 14, 2018 7:00:09 PM)
QueryDatabase 1.18.1
-----
Initialising persistence context 'Acme-HandyWorker'.
> select avg(a.offeredPrice), min(a.offeredPrice), max(a.offeredPrice),
  > sqrt(sum(a.offeredPrice*a.offeredPrice)/count(a.offeredPrice)-
  > avg(a.offeredPrice)*avg(a.offeredPrice)) from Application a;
1 object selected
[338.79833333333335, 50.0, 1100.0, 342.81005821575434]
>
```

Query C/4: The average, the minimum, the maximum, and the standard deviation of the price offered in the applications.

```
select avg(a.offeredPrice),min(a.offeredPrice),max(a.offeredPrice),sqrt(sum(a.offeredPrice *
a.offeredPrice)/count(a.offeredPrice)-avg(a.offeredPrice)*avg(a.offeredPrice)) from Application a;
```

This query calculates the average, the minimum, the maximum and the standard deviation of the price offered in the applications. To make this query max, min, avg, count and sqrt statistical functions have been used.

```
Progress Search JUnit Console
QueryDatabase [Java Application] C:\Program Files\Java\jdk1.7.0_13\bin\javaw.exe (Nov 14, 2018 6:56:15 PM)
QueryDatabase 1.18.1
-----
Initialising persistence context 'Acme-HandyWorker'.
> select avg(f.maxPrice), min(f.maxPrice),max(f.maxPrice),
  > sqrt(sum(f.maxPrice*f.maxPrice)/count(f.maxPrice)-
  > avg(f.maxPrice)*avg(f.maxPrice)) from FixUpTask f;
1 object selected
[323.72375, 50.0, 1000.0, 301.4978253627006]
>
```

Query C/5: The ratio of pending applications.

```
select (sum(case when a.status='PENDING' then 1.0 else 0 end)/count(*)) from Application a;
```

This query calculates the ratio of pending applications. We have used when, then operator to say that in case the status of applications is pending add one. This sum has been divided among all applications.

```
Progress Search JUnit Console
QueryDatabase [Java Application] C:\Program Files\Java\jdk1.7.0_13\bin\javaw.exe (Nov 12, 2018 11:06:11 AM)
QueryDatabase 1.18.1
-----
Initialising persistence context 'Acme-HandyWorker'.
select (sum(case when a.status='PENDING' then 1.0 else 0 end)/count(*)) from Application a;
|
> 1 object selected
0.25
>
```

Query C/6: The ratio of accepted applications.

```
select (sum(case when a.status='ACCEPTED' then 1.0 else 0 end)/count(*)) from Application a;
```

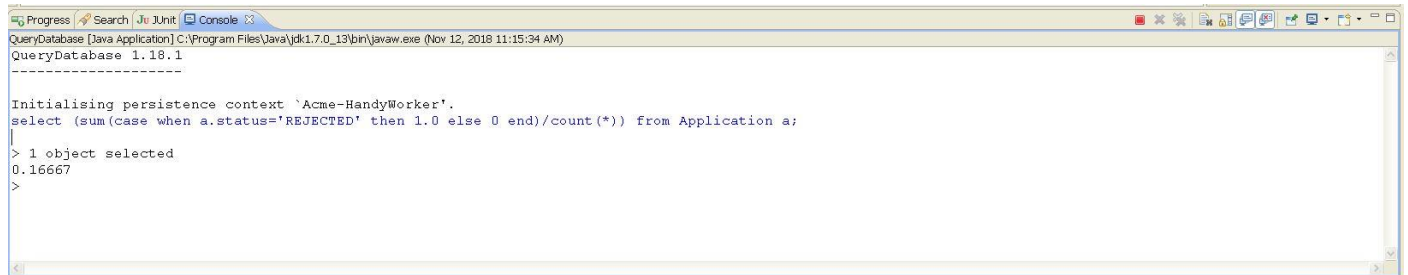
This query calculates the ratio of accepted applications. We have used when, then operator to say that in case the status of applications is accepted add one. This sum has been divided among all applications.

```
Progress Search JUnit Console
QueryDatabase [Java Application] C:\Program Files\Java\jdk1.7.0_13\bin\javaw.exe (Nov 12, 2018 11:11:27 AM)
QueryDatabase 1.18.1
-----
Initialising persistence context 'Acme-HandyWorker'.
select (sum(case when a.status='ACCEPTED' then 1.0 else 0 end)/count(*)) from Application a;
>
1 object selected
0.58333
>
```

Query C/7: The ratio of rejected applications.

```
select (sum(case when a.status='REJECTED' then 1.0 else 0 end)/count(*)) from Application a;
```

This query calculates the ratio of rejected applications. We have used when, then operator to say that in case the status of applications is rejected add one. This sum has been divided among all applications.



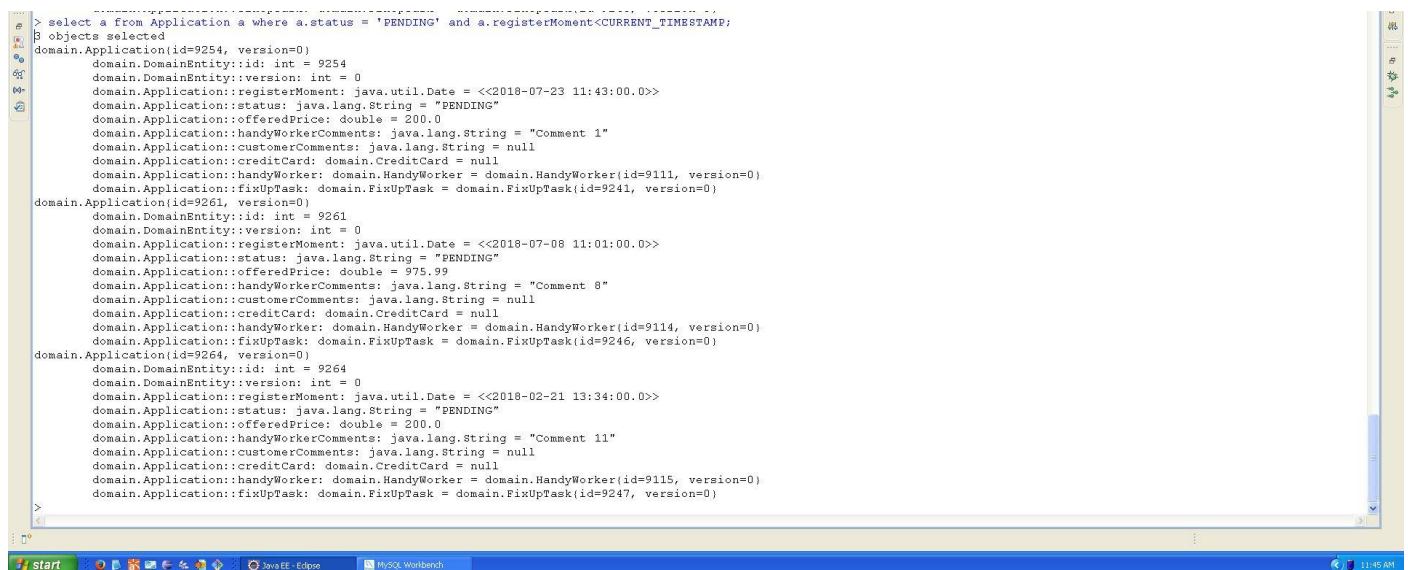
```
Progress Search JUnit Console
QueryDatabase [Java Application] C:\Program Files\Java\jdk1.7.0_13\bin\javaw.exe (Nov 12, 2018 11:15:34 AM)
QueryDatabase 1.18.1

-----
Initialising persistence context 'Acme-HandyWorker'.
select (sum(case when a.status='REJECTED' then 1.0 else 0 end)/count(*)) from Application a;
> 1 object selected
0.16667
>
```

Query C/8: The ratio of pending applications that cannot change its status because their time period's elapsed.

```
select a from Application a where a.status='PENDING' and a.registerMoment <CURRENT_TIMESTAMP;
```

This query calculates the ratio of pending applications that cannot change its status because their time period's elapsed. We have used where operator to say that in case the status of applications is pending and register moment is before to current time.



```
> select a from Application a where a.status = 'PENDING' and a.registerMoment<CURRENT_TIMESTAMP;
3 objects selected
domain.Application(id=9254, version=0)
domain.DomainEntity::id: int = 9254
domain.DomainEntity::version: int = 0
domain.Application:registerMoment: java.util.Date = <<2018-07-23 11:43:00.0>>
domain.Application:status: java.lang.String = "PENDING"
domain.Application:offeredPrice: double = 200.0
domain.Application:handyWorkerComments: java.lang.String = "Comment 1"
domain.Application:customerComments: java.lang.String = null
domain.Application:creditCard: domain.CreditCard = null
domain.Application:handyWorker: domain.HandyWorker = domain.HandyWorker(id=9111, version=0)
domain.Application:fixUpTask: domain.FixUpTask = domain.FixUpTask(id=9241, version=0)
domain.Application(id=9261, version=0)
domain.DomainEntity::id: int = 9261
domain.DomainEntity::version: int = 0
domain.Application:registerMoment: java.util.Date = <<2018-07-08 11:01:00.0>>
domain.Application:status: java.lang.String = "PENDING"
domain.Application:offeredPrice: double = 975.99
domain.Application:handyWorkerComments: java.lang.String = "Comment 8"
domain.Application:customerComments: java.lang.String = null
domain.Application:creditCard: domain.CreditCard = null
domain.Application:handyWorker: domain.HandyWorker = domain.HandyWorker(id=9114, version=0)
domain.Application:fixUpTask: domain.FixUpTask = domain.FixUpTask(id=9246, version=0)
domain.Application(id=9264, version=0)
domain.DomainEntity::id: int = 9264
domain.DomainEntity::version: int = 0
domain.Application:registerMoment: java.util.Date = <<2018-02-21 13:34:00.0>>
domain.Application:status: java.lang.String = "PENDING"
domain.Application:offeredPrice: double = 200.0
domain.Application:handyWorkerComments: java.lang.String = "Comment 11"
domain.Application:customerComments: java.lang.String = null
domain.Application:creditCard: domain.CreditCard = null
domain.Application:handyWorker: domain.HandyWorker = domain.HandyWorker(id=9115, version=0)
domain.Application:fixUpTask: domain.FixUpTask = domain.FixUpTask(id=9247, version=0)
>
```

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 f.customer from FixUpTask f group by f.customer having count(f) >= 1.1*((select count(t) from FixUpTask t)/(select count(c) from Customer c)) order by f.applications.size;
```

This query return list of customer who have published at least 10% more fix-up tasks than the average, ordered by number of applications. To listing customer we have used having operator. We have compare the number of all fix-up task with a operation that count all fix up task and this have been divided by other query in which we calculate if handy worker's applications is more than 10% than the average.

```
Progress Search JUnit Console
QueryDatabase [Java Application] C:\Program Files\Java\jdk1.7.0_13\bin\javaw.exe (Nov 14, 2018 7:35:24 PM)
Initialising persistence context 'Acme-HandyWorker'.

> select c from Customer c where c.fixUpTasks.size/
> (select avg(cl.fixUpTasks.size) from Customer cl)>= 1.1 order by c.fixUpTasks.size;
2 objects selected
domain.Customer(id=9071, version=0)
domain.DomainEntity::id: int = 9071
domain.DomainEntity::version: int = 0
domain.Actor::name: java.lang.String = "Maria"
domain.Actor::middleName: java.lang.String = "Isabel"
domain.Actor::surname: java.lang.String = "Diaz Gomez"
domain.Actor::photoLink: java.lang.String = "http://www.link2.com"
domain.Actor::email: java.lang.String = "customer2@gmail.com"
domain.Actor::phoneNumber: java.lang.String = "632014700"
domain.Actor::address: java.lang.String = "Address 2"
domain.Actor::isSuspicious: boolean = false
domain.Actor::userAccount: security.UserAccount = security.UserAccount(id=8979, version=0)
domain.Customer::score: java.lang.Double = null
domain.Customer::fixUpTasks: java.util.Collection = [domain.FixUpTask(id=9244, version=0), domain.FixUpTask(id=9245, ve
rsion=0)]
domain.Customer(id=9070, version=0)
domain.DomainEntity::id: int = 9070
domain.DomainEntity::version: int = 0
domain.Actor::name: java.lang.String = "Antonio"
domain.Actor::middleName: java.lang.String = ""
domain.Actor::surname: java.lang.String = "Ferrerias"
domain.Actor::photoLink: java.lang.String = "http://www.instagram.es"
domain.Actor::email: java.lang.String = "alrojojovivo@gmail.com"
domain.Actor::phoneNumber: java.lang.String = "632014785"
domain.Actor::address: java.lang.String = "Calle Sexta 6"
domain.Actor::isSuspicious: boolean = false
domain.Actor::userAccount: security.UserAccount = security.UserAccount(id=8975, version=0)
domain.Customer::score: java.lang.Double = null
domain.Customer::fixUpTasks: java.util.Collection = [domain.FixUpTask(id=9241, version=0), domain.FixUpTask(id=9242, ve
rsion=0), domain.FixUpTask(id=9243, version=0)]
>
```

Query C/10: The listing of handy workers who have got accepted at least 10% more ap-plications than the average, ordered by number of applications.

```
select h from HandyWorker h where h.applications.size/ (select avg(h1.applications.size) from
HandyWorker h1)>=1.1 order by h.applications.size;
```

This query return list of handy worker who have got accepted at least 10% more applications than the average, ordered by number of applications. To listing handy workers we have used where operator and the handy worker's applications have been divided by other query in which we calculate if handy worker's applications is more than 10% than the average.

```
Progress Search JUnit Console
QueryDatabase [Java Application] C:\Program Files\Java\jdk1.7.0_13\bin\javaw.exe (Nov 14, 2018 7:29:12 PM)
QueryDatabase 1.18.1
-----
Initialising persistence context 'Acme-HandyWorker'.

> select h from HandyWorker h where h.applications.size/
> (select avg(h1.applications.size) from HandyWorker h1)>=1.1 order by h.applications.size;
1 object selected
domain.HandyWorker(id=9112, version=0)
domain.DomainEntity::id: int = 9112
domain.DomainEntity::version: int = 0
domain.Actor::name: java.lang.String = "Lucia"
domain.Actor::middleName: java.lang.String = null
domain.Actor::surname: java.lang.String = "Del Carmen Fuentes"
domain.Actor::photoLink: java.lang.String = "http://www.link2.es"
domain.Actor::email: java.lang.String = "handyworker2@gmail.com"
domain.Actor::phoneNumber: java.lang.String = "654122078"
domain.Actor::address: java.lang.String = "Address 2"
domain.Actor::isSuspicious: boolean = false
domain.Actor::userAccount: security.UserAccount = security.UserAccount(id=8984, version=0)
domain.HandyWorker::make: java.lang.String = "Make 2"
domain.HandyWorker::score: java.lang.Double = null
domain.HandyWorker::finder: domain.Finder = domain.Finder(id=9106, version=0)
domain.HandyWorker::curriculum: domain.Curriculum = null
domain.HandyWorker::applications: java.util.Collection = [domain.Application(id=9256, version=0), domain.Application(id
=9257, version=0), domain.Application(id=9258, version=0)]
>
```