Item 3 – Queries

Level C

Query C/1	
Query specification	The average, the minimum, the maximum, and the standard deviation of the number of fix-up tasks per user.
JPQL Statement	select avg (c.fixUpTask.size), min(c.fixUpTask.size), max(c.fixUpTask.size), stddev(c.fixUpTask.size) from Customer c;
Short description	This query calculates: - avg is used for the average calculation of the number of fix-up tasks per user - min means we are looking for the customer who has the mínimum amount of fixUpTask. - max means we are looking for the customer who has the maximum amount of fixUpTask. - stddev is used for the standard deviation calculation of the number of fix-up tasks per user
Results	Max: 3 Min: 1 Media: 1.8 Des.Tip: 0.9798

Query C/2	
Query specification	The average, the minimum, the maximum, and the standard deviation of the number of applications per fix-up task.
JPQL Statement	select stddev(f.application.size), min(f.application.size), max(f.application.size), avg(f.application.size) from FixUpTask f;
Short description	This query calculates: - avg is used for the average calculation of the number of application per fix-up task - min means we are looking for the fix-up task which has the mínimum amount of application. - max means we are looking for the fix-up task which has the máximum amount of application. - stddev is used for the standard deviation calculation of the number of application per fix-up task
Results	Max: 3 Min: 0 Media: 0.6667 Des.Tip: 0.9428

Query C/3	
Query specification	The average, the minimum, the maximum, and the standard deviation of the maximum price of the fix-up tasks.
JPQL Statement	select avg(f.maximumPrice), min(f.maximumPrice), max(f.maximumPrice), stddev(f.maximumPrice) from FixUpTask f;
Short description	This query calculates: - avg is used for the average calculation of the number of the maximum price of the fix-up tasks. - min is used for the calculation of the mínimum price (bewteen the máximum prices) of the fix-up-task - max is used for the calculation of the máximum price (bewteen the máximum prices) of the fix-up-task - stddev is used for the calculation of the number of the máximum price of the fix-up-tasks.
Results	Max: 40.0 Min: 2.0 Media: 12.0 Des.Tip: 11.8227

Query C/4	
Query specification	The average, the minimum, the maximum, and the standard deviation of the price offered in the applications.
JPQL Statement	select stddev(f.offeredPrice),min(f.offeredPrice), max(f.offeredPrice), avg(f.offeredPrice) from Application f;
Short description	This query calculates: - stddev is used for the calculation of the standard deviation of the prices offered in the applications - min is used for the calculation of the mínimum price (bewteen the possibles prices offered) in the applications. - min is used for the calculation of the maximum price (bewteen the possibles prices offered) in the applications. - avg is used for the calculation of the average of the prices offered in the applications from our system.
Results	Max: 90.0 Min: 14.0 Media: 41.3336 Des.Tip: 25.9208

Query C/5	
Query specification	The ratio of pending applications.
JPQL Statement	select (select count(a) from Application a where a.status = 'pending')*1.0/count(ap) from Application ap;
Short description	First of all, the sub-query introduced calculates the number of applications which have status ’pending’. The result is divided by the total numbers of applications registered on the system.
Results	0.16667 (16%) – Sixteen percent of the applications registered are on state ‘pending’

Query C/6	
Query specification	The ratio of accepted applications.
JPQL Statement	select count(f.status)*1.0/(select count(t)*1.0 from Application t) from Application f where f.status = 'accepted';
Short description	First of all, the sub-query introduced calculates the number of applications which have status 'accepted'. The result is divided by the total numbers of applications registered on the system.
Results	0.5 (50%) – Fifteen percent of the applications registered are on state 'accepted'

Query C/7	
Query specification	The ratio of rejected applications.
JPQL Statement	select count(f.status)*1.0/(select count(t)*1.0 from Application t) from Application f where f.status = 'rejected';
Short description	First of all, the sub-query introduced calculates the number of applications which have status 'rejected'. The result is divided by the total numbers of applications registered on the system.
Results	0.333 (33%) – Thirty three percent of the applications registered are on state 'rejected'

Query C/8	
Query specification	The ratio of pending applications that cannot change its status because their time period's elapsed.
JPQL Statement	select (count(a)*1.0/(select count(ap) from Application ap)) from Application a where current_date() > a.momentElapsed and a.status = 'pending';
Short description	First of all, the sub-query introduced calculates the number of applications which have status 'pending'. The result is divided by the total numbers of applications registered on the system. Finally, this query is conditioned to check if the current date is later than the elapsed date configured on the application.
Results	0.16667 (16%) – Sixteen percent of the applications registered are on state 'pending'. So they can be modified due to its elapsed date is later than current.

Query C/9	
Query specification	The listing of customers who have published at least 10% more fix-up tasks than the average, ordered by number of applications.
JPQL Statement	select f from Customer f join f.fixUpTask t where f.fixUpTask.size > (select avg(f.fixUpTask.size)+(avg(f.fixUpTask.size)/10)*1.0 from Customer f) order by t.application.size;
Short description	This query compared if the size of the fixUpTask of the system is greater than the average of the size mentioned before plus a ten percent of the same average. Finally, the query returns the results ordered by number of applications that each one has.
Results	The result list is: [customer5, customer2]

Query C/10	
Query specification	The listing of handy workers who have got accepted at least 10% more applications than the average, ordered by number of applications.
JPQL Statement	select f from HandyWorker f join f.application a where a.status='accepted' and f.application.size > (select avg(f.application.size)+(avg(f.application.size)/10)*1.0 from HandyWorker f) order by f.application.size;
Short description	This query returns the Handy workers whose applications status is accepted and their application size is greater than the average of them plus the ten percent of the same average. Finally, the result is ordered by the applications size.
Results	The result list is: [handyworker1]

Level B

Query B/1	
Query specification	The minimum, the maximum, the average, and the standard deviation of the number of complaints per fix-up task.
JPQL Statement	select min(f.complaint.size), max(f.complaint.size), avg(f.complaint.size), stddev(f.complaint.size) from FixUpTask f;
Short description	This query is composed of: - min. This function calculates the minimun number of complaints per fix-up-task - max. This function calculates the maximum number of of complaints per fix-up-task. - stddev. This function calculates the standard deviation of the number of complaints per fix-up-task. - avg. This function calculates the average of the number of complaints per fix-up-task.
Results	Max: 2 Min: 0 Media: 0.7778 Des.Tip: 0.6285

Query B/2	
Query specification	The minimum, the maximum, the average, and the standard deviation of the number of notes per referee report.
JPQL Statement	select stddev(r.notes.size), min(r.notes.size), max(r.notes.size), avg(r.notes.size) from Report r;
Short description	This query is composed of: - min. This function calculates the minimun number of the number of notes per referee report. - max. This function calculates the maximum number of the number of notes per referee report. - stddev. This function calculates the standard deviation of the number of notes per referee report. avg. This function calculates the average of the number of notes per referee report.
Results	Max: 2 Min: 1 Media: 1.5 Des.Tip: 0.5

Query B/3	
Query specification	The ratio of fix-up tasks with a complaint.
JPQL Statement	select count(a)*1.0/(select count(t)*1.0 from FixUpTask t) from FixUpTask a where a.complaint.size > 0;
Short description	This query counts first the number complaints greater than 0 at first. The result before is divided by the total of fix-up-task registered on the system.
Results	0.6667 (66%) – Sixty six percent of applications registered on the system have complaints

Query B/4	
Query specification	The top-three customers in terms of complaints
JPQL Statement	select c from Customer c order by c.complaint.size DESC;
Short description	This query returns the customers ordered by their number of complaints created on the system.
Results	The result list is: [customer1, customer2, customer3, customer4, customer5]

Query B/5	
Query specification	The top-three handy workers in terms of complaints.
JPQL Statement	select distinct h from HandyWorker h join h.application t join t.fixUpTask f join f.complaint c group by h.id order by sum(f.complaint.size) DESC;
Short description	This query returns the handyworkers who have more complaints registered on the system. Some things to take into account are: - distinct. This is set on the query in order to avoid multiple repeated values. - According to our uml model, if we want to get to complaints from Handy worker, we must to join with his applications and then join this last one with the fix-up-task they have saved on the system information.
Results	The result list is: [handyworker1, handyworker3, handyworker2]

* Queries B/4 and B/5 will finally get done when controllers get implemented due to some limitations of JPQL. This technology, as difference of SQL, doesn´t support LIMIT parameter. The following url support this: https://forum.hibernate.org/viewtopic.php?f=1&t=1043789&p=2490865 .