Not in Criteria Query

```
@Override
public List<SectionCoverMasterGetAllRes>
getallSectionCoverBroker(SectionCoverMasterGetAllReq req) {
// TODO Auto-generated method stub
List<SectionCoverMasterGetAllRes> resList = new
ArrayList<SectionCoverMasterGetAllRes>();
DozerBeanMapper mapper = new DozerBeanMapper();
try {
Date today = req.getEffectiveDateStart()!=null ?req.getEffectiveDateStart() :
new Date();
Calendar cal = new GregorianCalendar();
cal.setTime(today);
cal.set(Calendar.HOUR_OF_DAY, 23);
cal.set(Calendar.MINUTE, 1);
today = cal.getTime();
cal.set(Calendar.HOUR_OF_DAY, 1);
cal.set(Calendar.MINUTE, 1);
Date todayEnd = cal.getTime();
List<SectionCoverMaster> list = new ArrayList<SectionCoverMaster>();
// Find Latest Record
CriteriaBuilder cb = em.getCriteriaBuilder();
CriteriaQuery<SectionCoverMaster> query =
cb.createQuery(SectionCoverMaster.class);
// Find All
Root<SectionCoverMaster> b = query.from(SectionCoverMaster.class);
// Select
query.select(b);
// Effective Date Max Filter
Subquery<Long> effectiveDate = query.subquery(Long.class);
Root<SectionCoverMaster> ocpm1 =
effectiveDate.from(SectionCoverMaster.class);
effectiveDate.select(cb.max(ocpm1.get("effectiveDateStart")));
Predicate a1 = cb.equal(ocpm1.get("coverId"), b.get("coverId"));
Predicate a2 = cb.equal(ocpm1.get("subCoverId"), b.get("subCoverId"));
Predicate a3 = cb.equal(ocpm1.get("sectionId"), b.get("sectionId"));
Predicate a4 = cb.equal(ocpm1.get("productId"), b.get("productId"));
Predicate a5 = cb.equal(ocpm1.get("companyId"), b.get("companyId"));
Predicate a6 = cb.lessThanOrEqualTo(ocpm1.get("effectiveDateStart"), today);
Predicate a7 = cb.equal(ocpm1.get("agencyCode"), b.get("agencyCode"));
Predicate a8 = cb.equal(ocpm1.get("branchCode"), b.get("branchCode"));
effectiveDate.where(a1,a2,a3,a4,a5,a6,a7,a8);
```

```
Subquery<Long> effectiveDate2 = query.subquery(Long.class);
Root<SectionCoverMaster> ocpm2 =
effectiveDate2.from(SectionCoverMaster.class);
effectiveDate2.select(cb.max(ocpm2.get("effectiveDateEnd")));
Predicate a9 = cb.equal(ocpm2.get("coverId"), b.get("coverId"));
Predicate a10 = cb.equal(ocpm2.get("subCoverId"), b.get("subCoverId"));
Predicate a11 = cb.equal(ocpm2.get("sectionId"), b.get("sectionId"));
Predicate a12 = cb.equal(ocpm2.get("productId"), b.get("productId"));
Predicate a13 = cb.equal(ocpm2.get("companyId"), b.get("companyId"));
Predicate a14 = cb.greaterThanOrEqualTo(ocpm2.get("effectiveDateEnd"),
todayEnd);
Predicate a15 = cb.equal(ocpm2.get("agencyCode"), b.get("agencyCode"));
Predicate a16 = cb.equal(ocpm2.get("branchCode"), b.get("branchCode"));
effectiveDate2.where(a9,a10,a11,a12,a13,a14,a15,a16);
// Order By
List<Order> orderList = new ArrayList<Order>();
orderList.add(cb.asc(b.get("coverName")));
Subquery<Long> cover = query.subquery(Long.class);
Root<SectionCoverMaster> ps = cover.from(SectionCoverMaster.class);
Subquery<Long> effectiveDate3 = query.subquery(Long.class);
Root<SectionCoverMaster> ocpm3 =
effectiveDate3.from(SectionCoverMaster.class);
effectiveDate3.select(cb.max(ocpm3.get("effectiveDateStart")));
Predicate eff1 = cb.equal(ocpm3.get("coverId"), ps.get("coverId"));
Predicate eff2 = cb.equal(ocpm3.get("subCoverId"), ps.get("subCoverId"));
Predicate eff3 = cb.equal(ocpm3.get("sectionId"), ps.get("sectionId"));
Predicate eff4 = cb.equal(ocpm3.get("productId"), ps.get("productId"));
Predicate eff5 = cb.equal(ocpm3.get("companyId"), ps.get("companyId"));
Predicate eff6 = cb.lessThanOrEqualTo(ocpm3.get("effectiveDateStart"),
today);
Predicate eff7 = cb.equal(ocpm3.get("agencyCode"), ps.get("agencyCode"));
Predicate eff8 = cb.equal(ocpm3.get("branchCode"), ps.get("branchCode"));
effectiveDate3.where(eff1,eff2,eff3,eff4,eff5,eff6,eff7,eff8);
// Filter
cover.select(ps.get("coverId"));
Predicate ps1 = cb.equal(ps.get("companyId"), req.getInsuranceId());
Predicate ps2 = cb.equal(ps.get("branchCode"), req.getBranchCode());
Predicate ps3 = cb.equal(ps.get("productId"), req.getProductId());
Predicate ps4 = cb.equal(ps.get("sectionId"), req.getSectionId());
Predicate ps5 = cb.equal(ps.get("agencyCode"), req.getAgencyCode());
cover.where(ps1,ps2,ps3,ps4,ps5);
// Where
Expression<String>e0= b.get("coverId");
```

```
// Where
Predicate n1 = cb.equal(b.get("effectiveDateStart"),effectiveDate);
Predicate n2 = cb.equal(b.get("subCoverId"),"0");
Predicate n3 = cb.equal(b.get("productId"), req.getProductId());
Predicate n4 = cb.equal(b.get("companyId"), req.getInsuranceId());
Predicate n5 = cb.equal(b.get("sectionId"), req.getSectionId());
Predicate n6 = cb.equal(b.get("agencyCode"), req.getAgencyCode());
Predicate n7 = cb.equal(b.get("agencyCode"), "99999");
Predicate n8 = cb.or(n6,n7);
Predicate n9 = cb.equal(b.get("branchCode"), req.getBranchCode());
Predicate n10 = cb.equal(b.get("branchCode"), "99999");
Predicate n11 = cb.or(n9,n10 );
Predicate n12 = cb.equal(b.get("effectiveDateEnd"),effectiveDate2);
Predicate n13 = e0.in(cover).not();
query.where(n1,n2,n3,n4,n5,n8,n11,n12,n13).orderBy(orderList);
// Get Result
TypedQuery<SectionCoverMaster> result = em.createQuery(query);
list = result.getResultList();
list.sort( Comparator.comparing(SectionCoverMaster :: getAgencyCode
).thenComparing(SectionCoverMaster :: getBranchCode ) );;
list = list.stream().filter(distinctByKey(o -> Arrays.asList(o.getCoverId()
,o.getSubCoverId() ))).collect(Collectors.toList());
Map<Integer,List<SectionCoverMaster>> groupByCover =
list.stream().collect(Collectors.groupingBy(SectionCoverMaster :: getCoverId));
// Map
for (Integer data : groupByCover.keySet()) {
List<SectionCoverMaster> datas = groupByCover.get(data);
datas.sort(Comparator.comparing(SectionCoverMaster :: getStatus).reversed());
SectionCoverMasterGetAllRes res = new SectionCoverMasterGetAllRes();
res = mapper.map(datas.get(0), SectionCoverMasterGetAllRes.class);
resList.add(res);
}
} catch (Exception e) {
e.printStackTrace();
log.info(e.getMessage());
return null;
}
return resList;
}
```