**public** List<CompanyProductMasterRes> getallNonSelectedProducts(AcExecutiveNonSelectedReq req) {

List<CompanyProductMasterRes> resList = **new** ArrayList<CompanyProductMasterRes>();

ModelMapper mapper = **new** ModelMapper();

**try** {

Date today = **new** Date();

Calendar cal = **new** GregorianCalendar();

cal.setTime(today);

cal.set(Calendar.***HOUR\_OF\_DAY***, 23);

cal.set(Calendar.***MINUTE***, 1);

today = cal.getTime();

List<CompanyProductMaster> productList = **new** ArrayList<CompanyProductMaster>();

// Criteria

CriteriaBuilder cb = em.getCriteriaBuilder();

CriteriaQuery<CompanyProductMaster> query = cb.createQuery(CompanyProductMaster.**class**);

// Find All

Root<CompanyProductMaster> b = query.from(CompanyProductMaster.**class**);

// Select

query.select(b);

// AmendId Max Filter

Subquery<Long> amendId = query.subquery(Long.**class**);

Root<CompanyProductMaster> ocpm1 = amendId.from(CompanyProductMaster.**class**);

amendId.select(cb.max(ocpm1.get("amendId")));

Predicate a1 = cb.equal(ocpm1.get("productId"), b.get("productId"));

Predicate a2 = cb.equal(ocpm1.get("companyId"), b.get("companyId"));

amendId.where(a1, a2);

// Order By

List<Order> orderList = **new** ArrayList<Order>();

orderList.add(cb.desc(b.get("effectiveDateStart")));

// Amend Id Date Max Filter

Subquery<Long> cover = query.subquery(Long.**class**);

Root<AcExecutiveProductMaster> ps = cover.from(AcExecutiveProductMaster.**class**);

Subquery<Long> amendId2 = query.subquery(Long.**class**);

Root<AcExecutiveProductMaster> ocpm2 = amendId2.from(AcExecutiveProductMaster.**class**);

amendId2.select(cb.max(ocpm2.get("amendId")));

Predicate eff1 = cb.equal(ocpm2.get("acExecutiveId"), ps.get("acExecutiveId"));

Predicate eff2 = cb.equal(ocpm2.get("productId"), ps.get("productId"));

Predicate eff3 = cb.equal(ocpm2.get("companyId"), ps.get("companyId"));

amendId2.where(eff1, eff2, eff3);

// Product Filter

cover.select(ps.get("productId"));

Predicate ps1 = cb.equal(ps.get("companyId"), req.getCompanyId());

Predicate ps2 = cb.equal(ps.get("amendId"), amendId2);

cover.where(ps1, ps2);

// Where

Expression<String> e0 = b.get("productId");

Predicate n1 = cb.equal(b.get("amendId"), amendId);

Predicate n3 = cb.equal(b.get("status"), "Y");

Predicate n4 = e0.in(cover).not();

query.where(n1, n3, n4).orderBy(orderList);

// Get Result

TypedQuery<CompanyProductMaster> result = em.createQuery(query);

productList = result.getResultList();

Map<Integer, List<CompanyProductMaster>> groupByProductId = productList.stream()

.collect(Collectors.*groupingBy*(w -> w.getProductId()));

// Map

**for** (Integer data : groupByProductId.keySet()) {

CompanyProductMaster coverData = groupByProductId.get(data).get(0);

CompanyProductMasterRes res = **new** CompanyProductMasterRes();

res = mapper.map(coverData, CompanyProductMasterRes.**class**);

res.setProductId(coverData.getProductId().toString());

resList.add(res);

}

resList.sort(Comparator.*comparing*(CompanyProductMasterRes::getProductName));

} **catch** (Exception e) {

e.printStackTrace();

log.info(e.getMessage());

**return** **null**;

}

**return** resList;

}