@Override

**public** LeadGetAllCountRes getallLeadCount(LeadGetallCountReq req) {

LeadGetAllCountRes res = **new** LeadGetAllCountRes();

List<Tuple> list = **new** ArrayList<Tuple>();

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

mapper.getConfiguration().setAmbiguityIgnored(**true**);

**try** {

// Limit , Offset

Integer limit = StringUtils.*isBlank*(req.getLimit()) ? 0 : Integer.*valueOf*(req.getLimit());

Integer offset = StringUtils.*isBlank*(req.getOffset()) ? 100 : Integer.*valueOf*(req.getOffset());

Pageable paging = PageRequest.*of*(limit, offset);

// Find All

Page<LeadDetails> leadDetails = repository.findByInsCompanyIdAndBranchCodeOrderByUpdatedDateDesc(

paging, req.getInsId(), req.getBranchCode());

List<String> leadIds = leadDetails.getContent().stream().map(LeadDetails::getLeadId)

.collect(Collectors.*toList*());

// Criteria

CriteriaBuilder cb = em.getCriteriaBuilder();

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

Root<LeadDetails> l = query.from(LeadDetails.**class**);

// Enquiry Count SubQuery

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

Root<EnquiryDetails> e = enquiryCount.from(EnquiryDetails.**class**);

enquiryCount.select(cb.count(e));

javax.persistence.criteria.Predicate e1 = cb.equal(e.get("leadId"), l.get("leadId"));

enquiryCount.where(e1);

// Quote Count SubQuery

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

Root<QuoteDetails> q = quoteCount.from(QuoteDetails.**class**);

quoteCount.select(cb.count(q));

javax.persistence.criteria.Predicate q1 = cb.equal(q.get("leadId"), l.get("leadId"));

quoteCount.where(q1);

// Policy Count SubQuery

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

Root<PolicyDetails> p = policyCount.from(PolicyDetails.**class**);

policyCount.select(cb.count(p));

javax.persistence.criteria.Predicate p1 = cb.equal(p.get("leadId"), l.get("leadId"));

policyCount.where(p1);

Expression<String> e0 = l.get("leadId");

Predicate n1 = e0.in(leadIds);

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

orderList.add(cb.desc(l.get("updatedDate")));

query.multiselect(l.get("leadId").alias("LeadId"),enquiryCount.alias("EnquiryCount") , quoteCount.alias("QuoteCount") ,

policyCount.alias("PolicyCount") );

query.where(n1).groupBy(l.get("leadId")).orderBy(orderList);

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

list = result.getResultList();

List<CrmLeadRes> leadList = **new** ArrayList<CrmLeadRes>();

**for** (LeadDetails data : leadDetails.getContent()) {

CrmLeadRes leadData = **new** CrmLeadRes();

leadData = mapper.map(data, CrmLeadRes.**class**);

List<Tuple> filterList = list.stream().filter(o -> o.get("LeadId").equals(data.getLeadId())).collect(Collectors.*toList*());

**if**(filterList.size()>0 ) {

Tuple counts = filterList.get(0) ;

leadData.setEnquiryCount(counts.get("EnquiryCount")==**null**?"0":counts.get("EnquiryCount").toString());

leadData.setQuoteCount(counts.get("QuoteCount")==**null**?"0":counts.get("QuoteCount").toString());

leadData.setPolicyCount(counts.get("PolicyCount")==**null**?"0":counts.get("PolicyCount").toString());

} **else** {

leadData.setEnquiryCount("0");

leadData.setQuoteCount("0");

leadData.setPolicyCount("0");

}

leadList.add(leadData);

}

res.setLeadDetails(leadList);

Long count1 = repository.countByInsCompanyIdAndBranchCode(req.getInsId(), req.getBranchCode());

res.setLeadCount(count1);

} **catch** (Exception e) {

e.printStackTrace();

log.info(e.getMessage());

**return** **null**;

}

**return** res;