Normal CriteriaQuery

```
CriteriaBuilder cb = em.getCriteriaBuilder();

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

// Find All
Root<MotorColorMaster> b = query.from(MotorColorMaster.class);

// Select
query.select(b);

// Where
Predicate n1 = cb.equal(b.get("status"), "Y");
Predicate n2 = cb.equal(b.get("colorId"), req.getColorId());
query.where(n1, n2);

// Get Result
TypedQuery<MotorColorMaster> result = em.createQuery(query);
list = result.getResultList();

List<MotorColorMasterRes> resList = new
ArrayList<MotorColorMasterRes>();

for(MotorColorMaster data : list){
MotorColorMasterRes res = new MotorColorMasterRes();
mapper.map(data,res);
resList.add(res);
}
```

Criteria with Subquery

```
Calendar cal = new GregorianCalendar();
cal.setTime(req.getEffectiveDateStart());
cal.set(Calendar.HOUR_OF_DAY, 23);
cal.set(Calendar.MINUTE, 59);
Date startDate = cal.getTime();
Date today = new Date();
cal.setTime(req.getEffectiveDateStart());
cal.set(Calendar.HOUR_OF_DAY, today.getHours());
cal.set(Calendar.MINUTE, today.getMinutes());
```

```
Date oldEndDate = cal.getTime();
cal.setTime(req.getEffectiveDateStart());
cal.set(Calendar.HOUR_OF_DAY, today.getHours());
cal.set(Calendar.MINUTE, today.getMinutes());
Date effDate = cal.getTime();

CriteriaBuilder cb = em.getCriteriaBuilder();
CriteriaQuery<MotorColorMaster> query =
cb.createQuery(MotorColorMaster.class);

// Find All
Root<MotorColorMaster> b = query.from(MotorColorMaster.class);

// Select
query.select(b);

// Effective Date Max Filter
Subquery<Long> effectiveDate = query.subquery(Long.class);
Root<MotorColorMaster> ocpm1 =
effectiveDate.from(MotorColorMaster.class);
effectiveDate.select(cb.max(ocpm1.get("effectiveDateStart")));
Predicate a1 = cb.equal(ocpm1.get("colorId"), b.get("colorId"));
Predicate a2 = cb.lessThanOrEqualTo(ocpm1.get("effectiveDateStart"),
startDate);

effectiveDate.where(a1, a2);

// Where
Predicate n1 = cb.equal(b.get("status"), "Y");
Predicate n2 = cb.equal(b.get("effectiveDateStart"), effectiveDate);
Predicate n3 = cb.equal(b.get("colorId"), req.getColorId());

query.where(n1, n2, n3);

// Get Result
TypedQuery<MotorColorMaster> result = em.createQuery(query);
list = result.getResultList();

for(MotorColorMaster data : list){
MotorColorMasterRes res = new MotorColorMasterRes();
mapper.map(data,res);
resList.add(res);
}
```

Criteria Tuple

```
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<ClientDetails> clientDetails =
clientrepo.findByInsCompanyIdAndBranchCodeOrderByLastVisitedDateDesc(
paging, req.getInsId(),
req.getBranchCode());

List<String> clienIds =    clientDetails.getContent().stream().map(ClientDetails
:: getClientRefNo ).collect(Collectors.toList())  ;

//Criteria
CriteriaBuilder cb = em.getCriteriaBuilder();
CriteriaQuery<Tuple> query = cb.createQuery(Tuple.class);

Root<ClientDetails>  c = query.from(ClientDetails.class);
List<Tuple> list = new ArrayList<Tuple>();

// Lead Count SubQuery
Subquery<Long> leadCount = query.subquery(Long.class);
Root<LeadDetails> l = leadCount.from(LeadDetails.class);
leadCount.select(cb.count(l));
javax.persistence.criteria.Predicate l1 = cb.equal(l.get("clientRefNo"),
c.get("clientRefNo"));
leadCount.where(l1);

// 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("clientRefNo"),
c.get("clientRefNo"));
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("clientRefNo"),
c.get("clientRefNo"));
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("clientRefNo"),
c.get("clientRefNo"));
policyCount.where(p1);

Expression<String>e0= c.get("clientRefNo");
Predicate n1 = e0.in(clienIds);

List<Order> orderList = new ArrayList<Order>();
orderList.add(cb.desc(c.get("lastVisitedDate")));
query.multiselect(c.get("clientRefNo").alias("ClientRefNo") ,
leadCount.alias("LeadCount") , enquiryCount.alias("EnquiryCount") ,
quoteCount.alias("QuoteCount") ,  policyCount.alias("PolicyCount")) ;

query.where(n1).groupBy(c.get("clientRefNo")).orderBy(orderList);
TypedQuery<Tuple> result = em.createQuery(query);
list =  result.getResultList();

List<ClientDetailsGridRes> clientList = new
ArrayList<ClientDetailsGridRes>();
for (ClientDetails data : clientDetails.getContent()) {
ClientDetailsGridRes clientData = new ClientDetailsGridRes();
clientData = mapper.map(data, ClientDetailsGridRes.class);

List<Tuple> filterList   =  list.stream().filter(o ->
o.get("ClientRefNo").equals(data.getClientRefNo())
).collect(Collectors.toList());
if(filterList.size()>0    ) {
Tuple counts = filterList.get(0) ;
clientData.setLeadCount(counts.get("LeadCount")==null?"0":counts.get("Lead
Count").toString());
clientData.setEnquieryCount(counts.get("EnquiryCount")==null?"0":counts.get
("EnquiryCount").toString());
```

```
clientData.setQuoteCount(counts.get("QuoteCount")==null?"0":counts.get("Qu
oteCount").toString());
clientData.setPolicyCount(counts.get("PolicyCount")==null?"0":counts.get("Po
licyCount").toString());

} else {
clientData.setLeadCount("0");
clientData.setEnquieryCount("0");
clientData.setQuoteCount("0");
clientData.setPolicyCount("0");
}
clientList.add(clientData);
}
res.setClientDetails(clientList);
Long count1 =
clientrepo.countByInsCompanyIdAndBranchCode(req.getInsId(),req.getBranch
Code());
res.setClientCount(count1);

} catch (Exception e) {
e.printStackTrace();
log.info(e.getMessage());
return null;

}
return res;
}
```