**Query 1**

SELECT nvl(MIN\_PREMIUM\_AMOUNT,0) into lc\_minprem FROM LOGIN\_USER\_DETAILS lud WHERE agency\_code in

(select decode(USERTYPE,'RSAIssuer',agency\_code,oa\_code) from login\_master lm where lm.LOGIN\_ID = P\_LOGIN\_ID

) and lud.amend\_id=(select max(amend\_id) from LOGIN\_USER\_DETAILS ld where ld.LOGIN\_ID=lud.LOGIN\_ID and ld.product\_id=P\_PRODUCTID)

and product\_id=;

**Query 2**

select nvl(ocm.min\_premium,0) into lc\_minprem from open\_cover\_master ocm,position\_master pm,open\_cover\_position\_master ocpm

where PM.missippi\_opencover\_no=ocpm.open\_cover\_no and ocm.proposal\_no=ocpm.proposal\_no and

ocm.amend\_id=(select max(amend\_id) from open\_cover\_master oc where

oc.proposal\_no=ocm.proposal\_no) and pm.login\_id=P\_LOGIN\_ID

and pm.product\_id=P\_PRODUCTID and pm.quote\_no=P\_QUOTE\_NO;

**Criteria Query Formation 1**

public String getPremium(String productId, String loginID ) {

String res = "";

try {

LoginMaster data = login.findByLoginId(loginID);

String a = data.getLoginid().toString();

String code;

if(a.equalsIgnoreCase("RSAIssuer")){

code = data.getAgencycode();

}

else {

code = data.getOacode();

}

// Criteria

CriteriaBuilder cb = em.getCriteriaBuilder();

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

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

// Find All

Root<LoginUserDetails> c = query.from(LoginUserDetails.class);

// Select

query.multiselect(c.get("minpremiumamount").alias("lc\_minprem") );

// Amend Id Max Filter

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

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

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

javax.persistence.criteria.Predicate a1 = cb.equal(c.get("loginId"),ocpm1.get("loginId") );

javax.persistence.criteria.Predicate a2 = cb.equal(c.get("productId"),ocpm1.get("productId") );

amendId.where(a1,a2);

// Order By

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

orderList.add(cb.asc(c.get("AgencyCode")));

// Where

javax.persistence.criteria.Predicate n1 = cb.equal(c.get("productId"), "3" );

javax.persistence.criteria.Predicate n2 = cb.equal(c.get("agencyCode"), code );

query.where(n1,n2).orderBy(orderList);

// Get Result

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

list = result.getResultList();

res =list.get(0).get("lc\_minprem")==null?"":list.get(0).get("lc\_minprem").toString();

} catch(Exception e) {

e.printStackTrace();

return null;

}

return res ;

}

**Criteria Query Formation 2**

public String getPremium(String proposalno, String productId, String loginID, String quoteno ) {

String res = "";

try {

// Criteria

CriteriaBuilder cb = em.getCriteriaBuilder();

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

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

// Find All

Root<OpenCoverMaster> c = query.from(OpenCoverMaster.class);

Root<PositionMaster> d = query.from(PositionMaster.class);

Root<OpenCoverPositionMaster> e = query.from(OpenCoverPositionMaster.class);

// Select

query.multiselect(c.get("minpremiumamount").alias("lc\_minprem") );

// Amend Id Max Filter

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

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

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

javax.persistence.criteria.Predicate a1 = cb.equal(c.get("proposalno"),ocpm1.get("proposalno") );

amendId.where(a1);

// Order By

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

orderList.add(cb.asc(c.get("EntryDate")));

// Where

javax.persistence.criteria.Predicate n1 = cb.equal(d.get("productId"), "11" );

javax.persistence.criteria.Predicate n2 = cb.equal(d.get("missippiopencoverno"), e.get("opencoverno"));

javax.persistence.criteria.Predicate n3 = cb.equal(c.get("proposalno"), e.get("proposalno"));

query.where(n1,n2,n3).orderBy(orderList);

// Get Result

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

list = result.getResultList();

res =list.get(0).get("lc\_minprem")==null?"":list.get(0).get("lc\_minprem").toString();

} catch(Exception e) {

e.printStackTrace();

return null;

}

return res ;

}

}