**Criteria For Single Table**

@PersistenceContext

**private** EntityManager em;

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();

for(MotorColorMaster data : list) {

mapper.map(data,res);

}

return res;

**Criteria for Multiple Table**

@PersistenceContext

**private** EntityManager em;

CriteriaBuilder cb = em.getCriteriaBuilder();

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

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

//Find all

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

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

//Sub Query

Subquery<String> branchName = query.subquery(String.class);

branchName.select(b.get(“branchName”));

Predicate b1 = cb.equal(c.get(“companyId”),b.get(“companyId”));

branchName.where(b1);

//Select

query.multiselect(c.get(“entryDate”).alias(“ENTRY\_DATE”),

c.get(“productId”).alias(“PRODUCT\_ID”),

b.get(“branchName”).alias(“BRANCH\_NAME”));

//where

Predicate n1 = cb.equal(c.get(“productId”),b.get(“productId”));

Predicate n2 = cb.equal(c.get(“productName”), b.get(“productName”));

query.where(n1,n2);

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

list = result.getResultList();

for(Tuple data : list) {

mapper.map(data,res);

}

return res;

**Criteria Queries**

**1. Greater than**

cr.select(root).where(cb.gt(root.get("itemPrice"), 1000));

**2. Less than**

cr.select(root).where(cb.lt(root.get("itemPrice"), 1000));

**3. Item Name contain chair**

cr.select(root).where(cb.like(root.get("itemName"), "%chair%"));

**4. Item price between 100 to 200**

cr.select(root).where(cb.between(root.get("itemPrice"), 100, 200));

**5. Items having itemName in Skate Board, Paint and Glue:**

cr.select(root).where(root.get("itemName").in("Skate Board", "Paint", "Glue"));

**6. Given Property is null**

cr.select(root).where(cb.isNull(root.get("itemDescription")));

**7. Given Property is not null**

cr.select(root).where(cb.isNotNull(root.get("itemDescription")));

**8. Add Two Expression**

Predicate[] predicates = **new** **Predicate**[2]; predicates[0] = cb.isNull(root.get("itemDescription")); predicates[1] = cb.like(root.get("itemName"), "chair%"); cr.select(root).where(predicates);

**9. Add Two Expression**

**Predicate** greaterThanPrice = cb.gt(root.get("itemPrice"), 1000); **Predicate** chairItems = cb.like(root.get("itemName"), "Chair%");

**10. Logical OR Condition**

cr.select(root).where(cb.or(greaterThanPrice, chairItems));

**11. And Condition**

cr.select(root).where(cb.and(greaterThanPrice, chairItems));

**12. Sorting**

cr.orderBy( cb.asc(root.get("itemName")), cb.desc(root.get("itemPrice")));

**13. Row Count**

CriteriaQuery<Long> cr = cb.createQuery(Long.class);

Root<Item> root = cr.from(Item.class);

cr.select(cb.count(root));

Query<Long> query = session.createQuery(cr);

List<Long> itemProjected = query.getResultList();

**14. Average**

CriteriaQuery<Double> cr = cb.createQuery(Double.class);

Root<Item> root = cr.from(Item.class); cr.select(cb.avg(root.get("itemPrice")));

Query<Double> query = session.createQuery(cr);

**List** avgItemPriceList = query.getResultList();

**15. Criteria Update**

CriteriaUpdate<Item> criteriaUpdate = cb.createCriteriaUpdate(Item.class); Root<Item> root = criteriaUpdate.from(Item.class); criteriaUpdate.set("itemPrice", newPrice); criteriaUpdate.where(cb.equal(root.get("itemPrice"), oldPrice));

**Transaction** transaction = session.beginTransaction(); session.createQuery(criteriaUpdate).executeUpdate();

transaction.commit();

**16. Criteria Delete**

CriteriaDelete<Item> criteriaDelete = cb.createCriteriaDelete(Item.class); Root<Item> root = criteriaDelete.from(Item.class); criteriaDelete.where(cb.greaterThan(root.get("itemPrice"), targetPrice)); **Transaction** transaction = session.beginTransaction(); session.createQuery(criteriaDelete).executeUpdate();

transaction.commit();