

jpa_jpql_v1.pdf

Task 1

A

```
public List<SalesEvent> retrieveSmallSales(double limit) {  
  
    EntityManager em = emf.createEntityManager();  
    em.getTransaction().begin();  
    List<SalesEvent> result = null;  
  
    Query query = em.createQuery("SELECT e FROM SalesEvent e WHERE  
e.amount < :limit");  
    query.setParameter("limit", limit);  
  
    result = query.getResultList();  
  
    em.getTransaction().commit();  
    em.close();  
    return result;  
}
```

SalesEvent [eventId=1, register=Register [registerId=4, name=Kassapiste_4], amount=3.18] at register Register [registerId=4, name=Kassapiste_4]
SalesEvent [eventId=6, register=Register [registerId=3, name=Kassapiste_3], amount=17.31] at register Register [registerId=3, name=Kassapiste_3]
SalesEvent [eventId=8, register=Register [registerId=5, name=Kassapiste_5], amount=3.59] at register Register [registerId=5, name=Kassapiste_5]

	Q	* eventId int(11)	amount double	register int(11)
		Filter	Filter	Filter
<input type="checkbox"/>	>	1	3.18	4
<input type="checkbox"/>	>	2	25.32	4
<input type="checkbox"/>	>	3	41.71	2
<input type="checkbox"/>	>	4	51.7	2
<input type="checkbox"/>	>	5	25.94	5
<input type="checkbox"/>	>	6	17.31	3
<input type="checkbox"/>	>	7	49.52	5
<input type="checkbox"/>	>	8	3.59	5
<input type="checkbox"/>	>	9	73.89	5
<input type="checkbox"/>	>	10	80.78	4

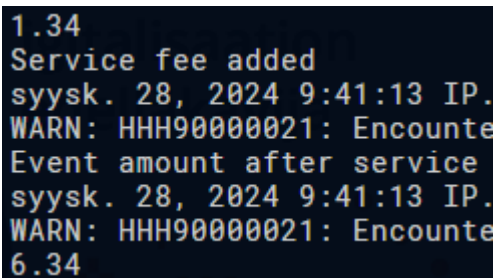
B

```
public void addServiceFee(double fee) {
    EntityManager em = emf.createEntityManager();
    em.getTransaction().begin();

    Query query = em.createQuery("UPDATE SalesEvent e SET e.amount = e.amount
+ :fee");
    query.setParameter("fee", fee);

    query.executeUpdate();

    em.getTransaction().commit();
    em.close();
}
```



1.34
Service fee added
syysk. 28, 2024 9:41:13 IP.
WARN: HHH90000021: Encounte
Event amount after service
syysk. 28, 2024 9:41:13 IP.
WARN: HHH90000021: Encounte
6.34

In this example the service fee is set to 5

C

```
public void deleteAllEvents() {
    EntityManager em = emf.createEntityManager();
    em.getTransaction().begin();

    Query query = em.createQuery("DELETE FROM SalesEvent e");
    query.executeUpdate();

    em.getTransaction().commit();
    em.close();
}
```

Task 2

A

```
public List<SalesEvent> retrieveSmallSales(double limit) {

    EntityManager em = emf.createEntityManager();
    em.getTransaction().begin();
    List<SalesEvent> result = null;
```

```
CriteriaBuilder cb = em.getCriteriaBuilder();
CriteriaQuery<SalesEvent> cq = cb.createQuery(SalesEvent.class);

Root<SalesEvent> salesEvent = cq.from(SalesEvent.class);
cq.select(salesEvent);

Predicate predicate = cb.lessThanOrEqualTo(salesEvent.get("amount"),
limit);
cq.where(predicate);

TypedQuery<SalesEvent> query = em.createQuery(cq);

result = query.getResultList();

em.getTransaction().commit();
em.close();
return result;
}
```

B

```
public void addServiceFee(double fee) {
    EntityManager em = emf.createEntityManager();
    em.getTransaction().begin();

    CriteriaBuilder cb = em.getCriteriaBuilder();
    CriteriaUpdate<SalesEvent> cu = cb.createCriteriaUpdate(SalesEvent.class);

    Root<SalesEvent> salesEvent = cu.from(SalesEvent.class);
    cu.set(salesEvent.get("amount"), cb.sum(salesEvent.get("amount"), fee));

    Query query = em.createQuery(cu);
    query.executeUpdate();

    em.getTransaction().commit();
    em.close();
}
```

C

```
public void deleteAllEvents() {
    EntityManager em = emf.createEntityManager();
    em.getTransaction().begin();

    CriteriaBuilder cb = em.getCriteriaBuilder();
    CriteriaDelete<SalesEvent> cd = cb.createCriteriaDelete(SalesEvent.class);

    Root<SalesEvent> salesEvent = cd.from(SalesEvent.class);
```

```
Query query = em.createQuery(cd);
query.executeUpdate();

em.getTransaction().commit();
em.close();
}
```