Query Methods Cheat Sheet

```
    List<Employee> findAllByOrderByEname();
    List<Employee> findAllByOrderByEnameDesc();
    List<Employee> findAllByOrderBySalDesc();
    List<Employee> findAllByOrderBySalAsc();
    List<Employee> findAllByOrderBySal();

    List<Employee> findByEname(String name);
    List<Employee> findByEnameIs(String name);
    List<Employee> findByEnameEquals(String name);
    List<Employee> findByEnameIsNot(String name);

    List<Employee> findByCommIsNull();
    List<Employee> findByCommIsNotNull();

    List<Employee> findByEnameStartingWith(String prefix);
    List<Employee> findByEnameStartingWithAndEnameEndingWith(String prefix,String
suffix);
    List<Employee> findByEnameContaining(String infix);
    List<Employee> findByEnameEndingWith(String suffix);

    List<Employee> findByEnameLike(String likePattern);
    List<Employee> findByEnameNotLike(String likePattern);

    List<Employee> findBySalLessThan(BigDecimal sal);
    List<Employee> findBySalLessThanEqual(BigDecimal sal);
    List<Employee> findBySalGreaterThanEqual(BigDecimal sal);
    List<Employee> findBySalGreaterThan(BigDecimal bigDecimal);
    List<Employee> findBySalLessThanOrderBySalAsc(BigDecimal bigDecimal);
    List<Employee> findBySalLessThanAndCommNotNull(BigDecimal bigDecimal);
    List<Employee> findBySalLessThanAndCommNotNullOrderByComm(BigDecimal bigDecimal);

    List<Employee> findByDeptno(int i);
    List<Employee> findBySalBetween(BigDecimal bigDecimal, BigDecimal bigDecimal2);
    List<Employee> findByDeptnoIn(List<Integer> collect);

    Optional<Employee> getByEname(String name);

    boolean existsByEname(String value);

    Employee getByUsername(String name);

    List<User> findByActiveTrue();
    List<User> findByActiveFalse();

    List<User> findByBirthDateAfter(ZonedDateTime birthDate);
    List<User> findByBirthDateBefore(ZonedDateTime birthDate);

    List<User> findByNameOrBirthDate(String name, ZonedDateTime birthDate);
    List<User> findByNameOrBirthDateAndActive(String name, ZonedDateTime birthDate,
Boolean active);
```

```
    Student findByName(String string);

    List<Student> findByDobAfter(LocalDate of);

    List<Student> findByDobBefore(LocalDate of);

    List<Student> findByActiveTrue();

    List<Student> findByActiveFalse();

    List<Student> findByOrderByName();

    List<Student> findByOrderByCgpaDesc();

    List<Student> findByOrderByDob();

    Student findFirstByOrderByName();

    Optional<Student> findFirstByOrderByCgpa();

    Optional<Student> findTopByOrderByCgpaDesc();

    List<Student> findTop3ByOrderByCgpaDesc();

    Optional<Student> findFirstByName(String string);

    Optional<Student> findFirstByGender(Gender gender);

    Optional<Student> findFirstByGenderOrderByCgpaDesc(Gender gender);

    Optional<Student> findFirstByGenderOrderByCgpa(Gender gender);

    List<Student> findFirst3ByOrderByCgpaDesc();

    List<Student> findAll(Sort sort);
```

```
// RETRIVE METHODS

    boolean existsByEname(String name);
    boolean existsByEnameContainsIgnoreCase(String name);
    boolean existsByEnameStartsWithIgnoreCase(String name);
    int countByEnameContainsIgnoreCase(String name);

    @Query("select new java.lang.String(e.ename) from Employee e")
    public List<String> findAllEmployeeNames();

    List<Employee> findByEnameContainsIgnoreCase(String name);

    List<Employee> findByEnameStartingWith(String name);

    List<Employee> findByEnameStartingWithIgnoreCase(String name);
```

```
    List<Employee> findByEnameIgnoreCase(String name);

    // DELETE METHODS

    @Transactional
    int deleteByEname(String name);

    @Transactional
    int deleteById(int id);

    @Transactional
    long deleteByEnameIgnoreCase(String name);

    @Transactional
    List<Employee> removeByEnameIgnoreCase(String lastname);

    // UPDATE METHODS

    @Modifying
    @Transactional
    @Query("update Employee e set e.sal = :sal where e.empno = :id")
    int updateSalaryById(int id, BigDecimal sal);


    @Modifying
    @Transactional
    @Query("update Employee e set e.sal = :sal where e.ename like :name")
    int updateSalaryByNameLike(String name, BigDecimal sal);
```

```
    @Query(value = "SELECT * FROM EMP WHERE ENAME LIKE %:pattern%", nativeQuery =
true)
    List<Employee> searchBy(String pattern);

    @Modifying
    @Query(value = "DELETE FROM EMP WHERE ENAME LIKE :name", nativeQuery = true)
    int deleteByName(String name);

    @Query("SELECT e.ename, d.dname FROM Employee e JOIN e.dept d")
    List<?> findEmployeesWithDept();

    @Query("SELECT e.ename FROM Employee e where e.id = :id")
    Optional<String> findEnameById(int id);

    @Query("SELECT e.ename FROM Employee e where e.mgr = :mgr")
    List<String> findFirstByMgr(int mgr);

    @Query("SELECT e.job, max(sal) from Employee e GROUP BY e.job")
    List<?> groupByJobMax();

    @Query("SELECT e.job, min(sal) from Employee e GROUP BY e.job")
    List<?> groupByJobMin();
```

```
@Query("SELECT e.job, count(sal) from Employee e GROUP BY e.job")
List<?> groupByJobCount();

@Query("SELECT e.job, avg(sal) from Employee e GROUP BY e.job")
List<?> groupByJobAvg();

@Query("SELECT e.job, sum(sal) from Employee e GROUP BY e.job")
List<?> groupByJobSum();
```