시간 타입 비교

시간 타입 설명

- **SQL Date:** 날짜만 표현 (시간 정보 없음)
  - JDBC API가 제공하는 타입, JPA 엔티티에 JDBC API 클래스를 넣는 것은 권장하지는 않음
- **SQL Time:** 시간만 표현 (날짜 정보 없음)
  - JDBC API가 제공하는 타입, JPA 엔티티에 JDBC API 클래스를 넣는 것은 권장하지는 않음
- **SQL Timestamp:** 날짜와 시간을 모두 표현
  - JDBC API가 제공하는 타입, JPA 엔티티에 JDBC API 클래스를 넣는 것은 권장하지는 않음
  - MySQL 8.0.27 까지는 **Timestamp가 32비트로 표현되어 "2038-01-19 03:14:07"이 나면 값이 오버플로우 되서 시간이 0으로 설정**됨
  - 8.0.28+ 부터는 MySQL에서 timestamp의 최댓값을 32,536,771,199로 변경(long형 8bytes지만 윈도우와 호환성을 위해 상수로 설정) => 일부 개선
  - MySQL 개발 정책에서는 **"3001-01-18 23:59:59"를 한참 넘어서는 데이터를 다룬다면 datetime 권장**
  - 데이터를 삽입해보니 "2038-01-19 03:14:07"까지만 삽입 가능?
- **Util Date:** 날짜와 시간을 모두 표현, 레거시 타입
  - JDBC API에 의존하지 않는 장점, @Temporal 명시해야 하는 단점
- **LocalDate:** 날짜만 표현 (Java 8+)
  - 정적 팩토리 메서드 지원으로 util.Date보다 생성하기 쉬움
- **LocalDateTime:** 날짜와 시간을 표현
  - 시간대 정보 없음 (Java 8+), 정적 팩토리 메서드 지원으로 util.Date보다 생성하기 쉬움
- **OffsetDateTime:** 날짜, 시간, UTC 오프셋 포함 (Java 8+)
  - hibernate에서는 **offset을 별도로 저장하지 않기에** LocalDateTime 사용 추천
- **ZonedDateTime:** 날짜, 시간, 시간대 정보 포함 (Java 8+)
  - **offset을 별도로 저장하지 않지만** 시스템 시간대로 변환

ID	SQL Date	SQL Time	SQL Timestamp	Util Date	LocalDate	LocalDateTime	LocalDateTime-Timestamp	OffsetDateTime	ZonedDateTime
9	2013-09-29	01:24:43	2013-09-29 01:24:43.0	2013-09-29	2024-10-12	2024-10-12T09:30:37	2038-01-19T03:14:07	2024-10-12T09:30:37Z	2024-10-12T09:30:37Z
10	2013-09-29	01:24:43	2013-09-29 01:24:43.0	2013-09-29	2024-10-12	2024-10-12T09:30:39	2038-01-19T03:14:07	2024-10-12T09:30:39Z	2024-10-12T09:30:39Z
11	2013-09-29	01:24:43	2013-09-29 01:24:43.0	2013-09-29	2024-10-12	2024-10-12T09:30:41	2038-01-19T03:14:07	2024-10-12T09:30:41Z	2024-10-12T09:30:41Z
12	2013-09-29	01:24:43	2013-09-29 01:24:43.0	2013-09-29	2024-10-12	2024-10-12T09:30:49	2038-01-19T03:14:07	2024-10-12T09:30:49Z	2024-10-12T09:30:49Z
13	2013-09-29	01:24:43	2013-09-29 01:24:43.0	2013-09-29	2024-10-12	2024-10-12T09:30:50	2038-01-19T03:14:07	2024-10-12T09:30:50Z	2024-10-12T09:30:50Z

저장