

TDC Time And Floating Point Numbers

- Single-precision floating point numbers:
 - Cannot precisely represent all real numbers
 - **0.1** stored becomes **0.100000001490116119384765625** exactly
 - Testing for safe division is problematic: Checking that divisor != zero does not guarantee that a division will not overflow.
 - Testing for equality is problematic. Two computational sequences that are mathematically equal may well produce different floating-point values.
- Problem with Testing for equality of floats seen in queries

$$1.2345 = \underbrace{12345}_{\text{mantissa}} \times \overbrace{10^{-4}}^{\text{exponent}}$$

GROUP BY tdcTime

```
SELECT tdcTime, COUNT(*)  
FROM run_004064_R001.Hit  
WHERE (detectorName='H1B') AND  
      (spillID=13)  
GROUP BY tdcTime;
```

GROUP BY groups together matching tdcTimes, but due to inaccuracy in FLOAT's, not all tdcTime's match.

#	tdcTime	COUNT
15	499.556	1
16	500	3
17	500.444	1
18	500.444	2
19	500.444	3
20	500.448	1
21	500.888	1
22	500.888	2
23	500.888	1
24	500.888	3
25	500.888	2
26	501.332	1
27	501.332	1
28	501.332	1

GROUP BY tdcTime

```
SELECT ROUND(tdcTime,3) AS `roundTime`, COUNT(*)  
FROM run_004064_R001.Hit  
WHERE (detectorName='H1B') AND  
      (spillID=13)  
GROUP BY `roundTime`;
```

- ROUND or TRUNCATE functions yields desired results, but adds computation.
- Could consider changing type to DECIMAL

#	time	COUNT(*)
7	499.556	9
8	500.000	3
9	500.444	6
10	500.448	1
11	500.888	9
12	501.332	4
13	501.336	4
14	501.776	5