SQL

Note: Standard SQL is used (same flavor as Google Cloud) with the tables denoted `calendar` (calendar.csv) and `listings` (listings.csv).

1. How many different listings were there on 2021-01-10? By how many different hosts?
Ans: 2693 listings by 1515 hosts. "Distinct" is not strictly necessary on listing_id, but it makes it more clear. "Distinct" is necessary on the host_id, as the outer join results in a host_id for each listing_id, and hosts can have several listings.

```
query = """
SELECT
  COUNT(DISTINCT listing_id) total_listings,
  COUNT(DISTINCT host_id) total_hosts
FROM
  `calendar` c
RIGHT OUTER JOIN `listings` l
ON c.listing_id = l.id
WHERE date = '2021-01-10'
"""
```

2. What are the top 10 most expensive (price-wise) listings?
Ans: There are 2 ways to approach the fact that each listing has a price for each day. The first is to compute the average price for a listing, and rank by it. This gives:

```
listing_id  avg_price
35995111    10000
10808594    5800
24844520    5000
45760212    3656.99
12991169    2681.64
17793546    2500
44288008    2230.51
7971599     2200
42244853    2000
19635807    1950
```

```
query = """
SELECT
  listing_id,
  round(avg(price),2) avg_price
FROM
  `calendar` c
GROUP BY listing_id
ORDER BY avg_price DESC
LIMIT 10
"""
```

The second way, which is more in-line with the notions of "expensive" and "top 10" in the question, is to compute the maximum price for each listing and rank by it. It's worth doing both to compare the results.

Interestingly, 5 of the top 10 listings are in both results (top by average and by max), while the top 4 is the same in in both lists.
This should be considered the final answer:

```
listing_id  max_price
35995111    10000
10808594    5800
24844520    5000
45760212    3800
41438534    3650
47077864    3044
44288008    2799
12991169    2769
17793546    2500
32500751    2401
```

```
query = """
SELECT
  distinct listing_id,
  max(price) max_price
FROM
  `calendar` c
GROUP BY listing_id
ORDER BY max_price DESC
LIMIT 10
"""
```

3. Which listing has the lowest Calendar vacancy rate?
Defining vacancy rate as the per-listing average of availability (where true is 1 and false is 0) over days, there are 151 listings with 0.0 vacancy rate which is the minimum. They are:

```
43133911,   43205670,   16191206,   33995060,   42228153,   12336073,
35344264,   19115703,   19715806,   29122523,   41383666,
3638801,    31811462,   31811979,   31812032,   42391359,   42413256,
42413324,   42438323,   42470958,   42471045,   42471179,42471250,
4264068,    32334103,   15438649,   8713048,    26920636,   41382430,
43462272,   29633744,   20422836,
39469357,   46494819,   39469121,   36363128,   10552656,   18893223,
19113545,   19718121,   21516425,   21524326,
27171064,   811391, 1918473,    13125187,   14354337,   39469422,
39469496,   39469555,   39469694,   31041010,
39469191,   39469223,   9670646,    23714456,   18315183,   31994027,
41292044,   17319003,   21091824,   43308918,
15042474,   47274359,   47274445,   47274251,   26875367,   26929465,
28783484,   13626296,   33201678,   41380138,   10054772,
15411365,   21665829,   46024860,   43612610,   30530829,   43080491,
21515689,   27636121,   28737035,   46396917,   37006062,
26445399,   31282503,   32237293,   43879593,   902703, 17306953,
24604083,   25834755,   42201200,   7246559,    30204463,
19295436,   40902129,   22083603,   44451819,   796558, 45830133,   508665,
22177169,   21814576,   1651294,    31591006,
9614278,    35425933,   27704044,   10989292,   13174377,   19030799,
23207929,   4304385,    45750839,   24523456,   21385023,
35912710,   19886665,   17389415,   3192298,    21295501,   2551716,
39891669,   28467145,   22792726,   41574844,   13607511,
24610231,   12808843,   6077649,    22573629,   45939232,   27293391,
39036834,   22788746,   46316747,   2062951,    8092234,
23257279,   34960546,   30084826,   36807152,   20215418,   40265537,
19763845,   10969969,   39442652,   16002247,   21379689,45850114
query = """
SELECT
listing_id
FROM(
SELECT
listing_id,
avg(avail_numeric) vacancy_rate
FROM(
SELECT
  listing_id,
  CASE
     when available = false then 0
     when available = true then 1
  END avail_numeric
FROM
  `calendar` c
GROUP BY listing_id, available)
GROUP BY listing_id
ORDER BY vacancy_rate ASC
) WHERE vacancy_rate = 0
"""
```

4. What 5 listings have had the most frequent day-over-day price increases?
Ans: Though it appears each listing in the calendar dataset has 365 days of data, we should not assume this is always going to be the case (e.g. midway through the year, or if a host only lists during the summer, etc.) Therefore when we define frequency of day-over-day price increases, we need to use the number of days listed (ie in the dataset) as the denominator. We also need to ensure this denominator is greater than 0, and using the safe_divide provides further protection against infinite results. Using this approach, we find that the 5 listings with the most frequent day-over-day price increases are:

```
listing_id  num_pos_changes num_days_total  price_increase_frequency
29059567    227 365 0.621917808
47077864    224 365 0.61369863
20774449    218 365 0.597260274
41913610    214 365 0.58630137
47157179    213 365 0.583561644
```

```
query = """
WITH
  num_pos AS (
  SELECT
    listing_id,
    COUNT(price_difference_1day) num_pos_changes
  FROM (
    SELECT
      listing_id,
      date,
      price - LAG(price) OVER(PARTITION BY listing_id ORDER BY DATE ASC)
price_difference_1day
    FROM
      `calendar`)
  WHERE
    price_difference_1day > 0
  GROUP BY
    listing_id),

  num_total AS(
  SELECT
    listing_id,
    COUNT(date) num_days_total
  FROM
    `calendar`
  GROUP BY
    listing_id )
SELECT
  num_pos.listing_id,
  num_pos_changes,
  num_days_total,
  SAFE_DIVIDE(num_pos_changes,
    num_days_total) AS price_increase_frequency
FROM
  num_pos
INNER JOIN
  num_total
ON
  num_pos.listing_id = num_total.listing_id
WHERE
  num_days_total > 0
ORDER BY price_increase_frequency DESC
LIMIT 5
"""
```