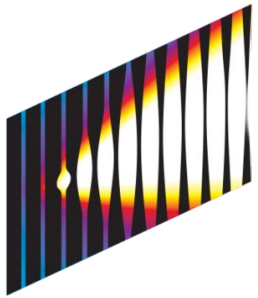
BnB Data Set

Data Analysis Assessment



Sony Pictures Entertainment

Candidate: Ryan Ha (1/26/2023)

SPE - People Analytics

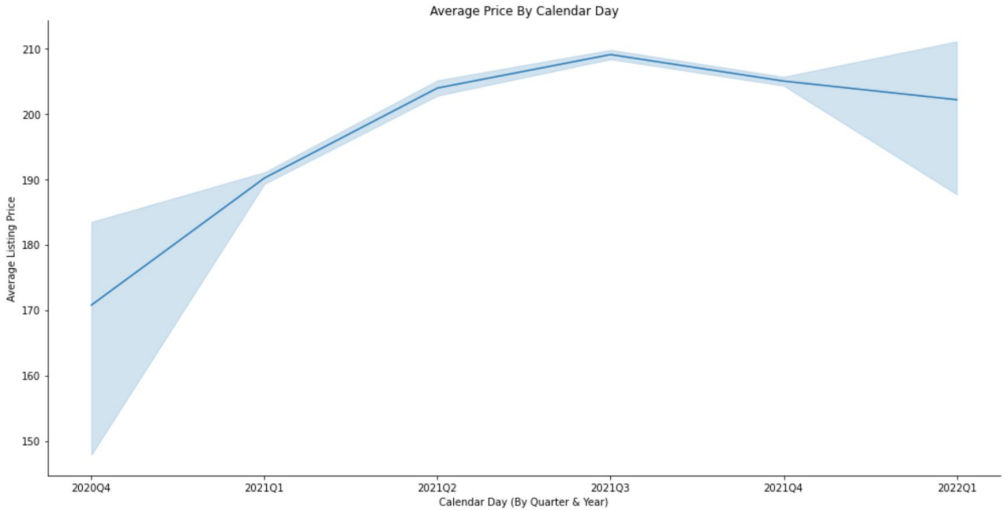
Intuition:

BnB listing prices are highest in **Q3**, as summer vacation plans are underway, where prices are lowest in **Q4**.

```
gfg = sns.relplot(data=price_avg,x='quarterc',y='avg_price' , kind='line', height=7, aspect=2)

# add label to the axis and label to the plot
gfg.set(xlabel ="Calendar Day (By Quarter & Year)",
        ylabel = "Average Listing Price", title ='Average Price By Calendar Day')
```

```
<seaborn.axisgrid.FacetGrid at 0x7f78095f3f70>
```



See [line 59] within the Jupyter Notebook on Git.

Q: If Bnb charges 10% commission to host, how to increase the commission revenue? (based on data)

Based on our “listings” & “calendar” data sets **alone**, the results would be *inconclusive*, and would require additional data.

Luckily, **data is all around us.**

We can use **public data** sets or **conduct additional surveys** to get an idea of how customers & hosts feel about the service, *including other details such as*:

- Best times to be a BnB Host
- BnB tips for travelers & hosts, off season
- Peak & Slowest Seasons for travel (flights)
- Additional revenue streams for BnB Hosts, such as “BnB Plus” homes (lower supply, more expensive stays)

Bottomline...

Insights & Takeaways.

To **increase** **revenue** by commission, or service charge, we have a couple levers to focus on:

Peak Seasons *(Late Q2 - Late Q3):*

1. **Travel demand is high.** Lots of solo travelers & small groups, ranging from group stays to 1-2 bed entire apartment requests.
2. To incentivize hosts to fill the gap & provide accommodations, *focusing the service charge rate on customers is advised.*

Slow Season *(Mid Q4 - Late Q1):*

1. **Travel demand is low.** Family-focused traveling, including large groups, are popular during this time, however are not as common & expensive. Also the supply is lower.
2. More expensive “BnB Plus” products are more popular in this time. ***(e.g., largers homes, cabins, etc.)***
3. To incentivize hosts to fill the gap & provide accommodations, *focusing the service charge rate on hosts is advised.*

SQL Prompts

Link to .sql file on Git

```sql
--  Please provide the SQL statement for these questions:

--  1) How many different listings were there on 2021-01-10? By how many different hosts?
select count(distinct l.id) as listings_unique,
       count(distinct l.host_id) as hosts_unique
from listings l inner join calendar c
    on l.id = c.listing_id
where date(concat(c.year,'-',c.month,'-',c.day)) is date('2021-01-10')
group by 1,2


--  2) What are the top 10 most expensive (pricewise) listings?

select listing_id, price
from (select
        listing_id,
        price,
        rank() over (partition by listing id order by price desc) as ranking   -- ranking
      from calendar
      ) as a    -- subquery alias
where a.ranking<11
order by a.profits desc
```

Link to .sql file on Git

```sql
--  3) Which listing has the lowest Calendar vacancy rate?

with vacancy_ratios as (
    select listing_id,
           date(date) as date,

           --  total units, partitioned by listing, ordered by date
           count(available) over (partition by listing_id order by date) as total_units,

           --  total units vacant, partitioned by
           count (case when available='t' then 1 else null end)
             over (partition by listing_id order by date) as vacancy_total
    from calendar
),

select v.listing_id,
       v.date,
       v.vacancy_total/v.total_units as vacancy_rate
from vacancy_ratios v
group by 1,2,3


--  4) What 5 listings have had the most frequent day-over-day price increases?

-- lag / lead problem
-- Assumptions:  "most" as in "largest" price increase ?
-- Top 5 listings ?

with cte as (
    select listing_id,
        date(date) as date,
        price as date_price,
        lag(price) over (order by date) as yday_price,
        price - lag(price) as price_difference,
        dense_rank(price - lag(price))
          over (partition by listing_id order by date) as dense_ranking   -- ranking
from calendar
where dense_ranking<6
)

select listing_id,
       date,
       price_difference
from cte
order by 1,2,3 desc
```