

Subject: Research Findings for Boston Airbnb Data Analysis

Dear Raj,

We are writing to share the research findings for Boston Airbnb data analysis. Below are the answers to the research questions we addressed:

1. Busiest Times to Visit Boston and Price Spikes

1.1. What are the busiest times to visit Boston?

Answer:

The busiest times of the year to visit Boston are **September** and **October**. These months exhibit the highest average monthly prices, indicating increased demand for accommodations during these periods.

Query:

```
with monthly_avg_price as (  
    select  
        extract(month from c.date) as month,  
        avg(c.price) as avg_monthly_price  
    from calendar_table c  
    group by month  
)  
annual_avg as (select avg(price) as avg_price from calendar_table)  
select  
    m.month,  
    m.avg_monthly_price,  
    round(((m.avg_monthly_price - a.avg_price) / a.avg_price) * 100, 2) as  
price_spike_percentage  
from monthly_avg_price m, annual_avg a  
order by m.month;
```

```
mysql> with monthly_avg_price as (
->   select
->       extract(month from c.date) as month,
->       avg(c.price) as avg_monthly_price
->   from calendar_table c
->   group by month
-> ),
-> annual_avg as (select avg(price) as avg_price from calendar_table)
-> select
->     m.month,
->     m.avg_monthly_price,
->     round(((m.avg_monthly_price - a.avg_price) / a.avg_price) * 100, 2) as price_spike_percentage
->   from monthly_avg_price m, annual_avg a
->   order by m.month;
```

month	avg_monthly_price	price_spike_percentage
1	178.36371694021938	-6.95
2	175.86229537053336	-8.26
3	175.93381779960492	-8.22
4	187.18761433965201	-2.35
5	184.1742723880597	-3.92
6	189.99918454040318	-0.88
7	193.2161454069101	0.8
8	190.21059736145858	-0.77
9	231.52407329780226	20.78
10	230.26755852842808	20.12
11	199.2004822714305	3.92
12	187.96191615981112	-1.95

12 rows in set (4.08 sec)

1.2. How much do the calendar prices increase?

Answer:

In **September** and in **October**, the represent price increases of **+20.78%** and **+20.12%** respectively compared to the annual average price.

Query:

```
with monthly_avg_price as (
  select
    extract(month from c.date) as month,
    avg(c.price) as avg_monthly_price
  from calendar_table c
  group by month
),
annual_avg as (select avg(price) as avg_price from calendar_table)
select
  m.month,
  m.avg_monthly_price,
  round(((m.avg_monthly_price - a.avg_price) / a.avg_price) * 100, 2) as
price_spike_percentage
from monthly_avg_price m, annual_avg a
order by m.month;
```

```
mysql> with monthly_avg_price as (
->   select
->       extract(month from c.date) as month,
->       avg(c.price) as avg_monthly_price
->   from calendar_table c
->   group by month
-> ),
-> annual_avg as (select avg(price) as avg_price from calendar_table)
-> select
->     m.month,
->     m.avg_monthly_price,
->     round(((m.avg_monthly_price - a.avg_price) / a.avg_price) * 100, 2) as price_spike_percentage
->   from monthly_avg_price m, annual_avg a
->   order by m.month;
```

month	avg_monthly_price	price_spike_percentage
1	178.36371694021938	-6.95
2	175.86229537053336	-8.26
3	175.93381779960492	-8.22
4	187.18761433965201	-2.35
5	184.1742723880597	-3.92
6	189.99918454040318	-0.88
7	193.2161454069101	0.8
8	190.21059736145858	-0.77
9	231.52407329780226	20.78
10	230.26755852842808	20.12
11	199.2004822714305	3.92
12	187.96191615981112	-1.95

12 rows in set (4.08 sec)

1.3. What is the price gap between peak and off-peak months?

Answer:

There is a significant price gap between peak months (September and October) and off-peak months (January to March). While peak months see average prices exceeding **\$230**, off-peak months have average prices around **175–178**, indicating a decrease of approximately **7-8%** below the annual average.

Query:

```
with monthly_avg_price as (
    select
        extract(month from c.date) as month,
        avg(c.price) as avg_monthly_price
    from calendar_table c
    group by month
),
annual_avg as (select avg(price) as avg_price from calendar_table)
select
    m.month,
    m.avg_monthly_price,
    round(((m.avg_monthly_price - a.avg_price) / a.avg_price) * 100, 2) as
price_spike_percentage
from monthly_avg_price m, annual_avg a
order by m.month;
```

```
mysql> with monthly_avg_price as (
->   select
->       extract(month from c.date) as month,
->       avg(c.price) as avg_monthly_price
->   from calendar_table c
->   group by month
-> ),
-> annual_avg as (select avg(price) as avg_price from calendar_table)
-> select
->     m.month,
->     m.avg_monthly_price,
->     round(((m.avg_monthly_price - a.avg_price) / a.avg_price) * 100, 2) as price_spike_percentage
->   from monthly_avg_price m, annual_avg a
->   order by m.month;
```

month	avg_monthly_price	price_spike_percentage
1	178.36371694021938	-6.95
2	175.86229537053336	-8.26
3	175.93381779960492	-8.22
4	187.18761433965201	-2.35
5	184.1742723880597	-3.92
6	189.99918454040318	-0.88
7	193.2161454069101	0.8
8	190.21059736145858	-0.77
9	231.52407329780226	20.78
10	230.26755852842808	20.12
11	199.2004822714305	3.92
12	187.96191615981112	-1.95

12 rows in set (4.08 sec)

2. Top Hosts

2.1. Who owns the highest number of listings, and how many?

Answer:

The host with the highest number of listings is **Kara**, who owns **1,088** listings. Other top hosts include **Seamless** with **711** listings, **Mike** with **488** listings, and **Flatbook** with **464** listings.

Query:

```
select h.host_id, h.host_name, count(l.id) as total_listings
from host_table h
join listings_table l on h.host_id = l.host_id
group by h.host_id, h.host_name
order by total_listings desc
limit 10;
```

```
mysql> select h.host_id, h.host_name, count(l.id) as total_listings
-> from host_table h
-> join listings_table l on h.host_id = l.host_id
-> group by h.host_id, h.host_name
-> order by total_listings desc
-> limit 10;
```

host_id	host_name	total_listings
30283594	Kara	1088
25188	Seamless	711
9419684	Mike	488
12243051	Flatbook	464
22348222	Alicia	250
1444340	Will	144
21184200	Paige	144
32532791	Marie	120
18202088	Beantown Suites	102
4962900	Stay Alfred	100

10 rows in set (0.05 sec)

2.2. What is the average number of listings owned by a host?

Answer:

The average number of listings owned by a host is **3.4172**. This calculation is based on the total number of listings divided by the number of active hosts.

Query:

```
select avg(total_listings) as avg_host_listings
from (
  select h.host_id, count(l.id) as total_listings
  from host_table h
  join listings_table l on h.host_id = l.host_id
  group by h.host_id
) subquery;
```

```
mysql> select avg(total_listings) as avg_host_listings
-> from (
->   select h.host_id, count(l.id) as total_listings
->   from host_table h
->   join listings_table l on h.host_id = l.host_id
->   group by h.host_id
-> ) subquery;
ERROR 2006 (HY000): MySQL server has gone away
No connection. Trying to reconnect...
Connection id: 2522
Current database: Boston

+-----+
| avg_host_listings |
+-----+
|          3.4172 |
+-----+
1 row in set (0.70 sec)
```

2.3. Who earns the most, and how much do they earn?

Answer:

Seamless is the top-earning host with estimated earnings of **\$1.5 million**. They are followed by **Jason** with **\$1.3 million** and **Mike** with **\$881,856** in estimated earnings.

Query:

First, delete the "\$" and turn it into float.

```
update listings_table
set price = cast(replace(price, '$', '') as decimal(10, 2))
where price is not null;

alter table listings_table
modify price float;
```

Then, we estimate the historical earnings using number of reviews.

```
select h.host_id, h.host_name, sum(l.price * r.number_of_reviews) as
total_earnings
from host_table h
join listings_table l on h.host_id = l.host_id
join availability_table a on l.id = a.listing_id
join review_table r on l.id = r.listing_id
group by h.host_id, h.host_name
order by total_earnings desc
limit 10;
```

```
mysql> select h.host_id, h.host_name, sum(l.price * r.number_of_reviews) as total_earnings
-> from host_table h
-> join listings_table l on h.host_id = l.host_id
-> join availability_table a on l.id = a.listing_id
-> join review_table r on l.id = r.listing_id
-> group by h.host_id, h.host_name
-> order by total_earnings desc
-> limit 10;
```

host_id	host_name	total_earnings
25188	Seamless	1577592
9410008	Jason	1367885
9419684	Mike	881856
1444340	Will	864174
12243051	Flatbook	692304
324630	Sean	400968
21184200	Paige	365868
5618949	Alan	358602
814298	Brent	350505
508268	Richard	240180

10 rows in set (0.23 sec)

2.4. What is the average earnings per host?

Answer:

The average earnings per host are **8161.55**. This figure is derived from the total estimated earnings of all hosts divided by the number of hosts.

Query:

```
select avg(total_earnings) as avg_host_earnings
from (
    select h.host_id, sum(l.price * r.number_of_reviews) as total_earnings
    from host_table h
    join listings_table l on h.host_id = l.host_id
    join availability_table a on l.id = a.listing_id
    join review_table r on l.id = r.listing_id
    group by h.host_id
) subquery;
```

```
mysql> select avg(total_earnings) as avg_host_earnings
-> from (
->     select h.host_id, sum(l.price * r.number_of_reviews) as total_earnings
->     from host_table h
->     join listings_table l on h.host_id = l.host_id
->     join availability_table a on l.id = a.listing_id
->     join review_table r on l.id = r.listing_id
->     group by h.host_id
-> ) subquery;
```

avg_host_earnings
8161.552957359009

1 row in set (0.08 sec)

2.5. Who has the highest rating, and what is their score?

Answer:

Several hosts have achieved the highest average rating of **100%**. These include **Shahid, Andrew, Kelly**, and others. This perfect rating reflects exceptional service quality and guest satisfaction.

Query:

```
select h.host_id, h.host_name, avg(r.review_scores_rating) as avg_rating
from host_table h
join listings_table l on h.host_id = l.host_id
join review_table r on l.id = r.listing_id
where r.review_scores_rating is not null
group by h.host_id, h.host_name
order by avg_rating desc
limit 10;
```

```
mysql> select h.host_id, h.host_name, avg(r.review_scores_rating) as avg_rating
-> from host_table h
-> join listings_table l on h.host_id = l.host_id
-> join review_table r on l.id = r.listing_id
-> where r.review_scores_rating is not null
-> group by h.host_id, h.host_name
-> order by avg_rating desc
-> limit 10;
+-----+-----+-----+
| host_id | host_name | avg_rating |
+-----+-----+-----+
| 70552156 | Shahid | 100 |
| 26413047 | Andrew | 100 |
| 16379445 | Kelly | 100 |
| 68927890 | Cole | 100 |
| 43779441 | Tanmay | 100 |
| 9350281 | Emily | 100 |
| 69028974 | Christine | 100 |
| 45851126 | Kelly | 100 |
| 26420664 | Michael | 100 |
| 36275619 | Sam | 100 |
+-----+-----+-----+
10 rows in set (0.08 sec)
```

2.6. What is the average rating across all hosts?

Answer:

The average rating across all hosts is **93.28**. This statistic provides an overall view of guest satisfaction within the Boston Airbnb market.

Query:

```
select avg(avg_rating) as avg_host_rating
from (
    select h.host_id, avg(r.review_scores_rating) as avg_rating
    from host_table h
    join listings_table l on h.host_id = l.host_id
    join review_table r on l.id = r.listing_id
    where r.review_scores_rating is not null
    group by h.host_id
) subquery;
```



```
mysql> select avg(avg_rating) as avg_host_rating
-> from (
->   select h.host_id, avg(r.review_scores_rating) as avg_rating
->   from host_table h
->   join listings_table l on h.host_id = l.host_id
->   join review_table r on l.id = r.listing_id
->   where r.review_scores_rating is not null
->   group by h.host_id
-> ) subquery;
+-----+
| avg_host_rating |
+-----+
| 93.28137721011832 |
+-----+
1 row in set (0.09 sec)
```

3. Supply and Demand in Each Neighborhood

3.1. Neighborhoods that have the most number of reviews

Answer:

The neighborhoods with the highest number of reviews are:

- **Jamaica Plain: 9,055** reviews
- **Dorchester: 7,396** reviews
- **South End: 5,714** reviews

These figures indicate high visitor engagement and rental activity in these areas.

Query:

```
select n.neighborhood_id, n.neighbourhood_cleansed AS neighborhood_name,
sum(r.number_of_reviews) AS total_reviews
from neighborhood_table n
join listings_table li on n.neighborhood_id = li.neighborhood_id
join review_table r on li.id = r.listing_id
group by n.neighborhood_id, n.neighbourhood_cleansed
order by total_reviews desc;
```

```
mysql> select n.neighborhood_id, n.neighbourhood_cleansed AS neighborhood_name, sum(r.number_of_reviews) AS total_reviews
-> from neighborhood_table n
-> join listings_table li on n.neighborhood_id = li.neighborhood_id
-> join review_table r on li.id = r.listing_id
-> group by n.neighborhood_id, n.neighbourhood_cleansed
-> order by total_reviews desc;
```

neighborhood_id	neighborhood_name	total_reviews
2	Jamaica Plain	9055
22	Dorchester	7396
10	South End	5714
12	East Boston	5504
8	North End	4483
11	Back Bay	4389
15	Beacon Hill	4098
24	South Boston	3851
9	Roxbury	3775
18	Brighton	3052
25	Allston	2988
17	Fenway	2817
16	Downtown	2535
13	Charlestown	1981
3	Mission Hill	1469
1	Roslindale	1387
23	South Boston Waterfront	744
19	West Roxbury	690
7	Chinatown	669
21	Mattapan	442
4	Longwood Medical Area	393
20	Hyde Park	376
5	Bay Village	241
14	West End	157
6	Leather District	69

25 rows in set (0.11 sec)

3.2. Neighborhoods that have the most listings

Answer:

The neighborhoods with the most listings are:

- **Jamaica Plain: 343** listings
- **South End: 326** listings
- **Back Bay: 302** listings

A higher number of listings suggests strong supply and host interest in these popular neighborhoods.

Query:

```
select n.neighborhood_id, n.neighbourhood_cleansed AS neighborhood_name,
count(li.id) AS total_listings
from neighborhood_table n
join listings_table li on n.neighborhood_id = li.neighborhood_id
group by n.neighborhood_id, n.neighbourhood_cleansed
order by total_listings desc;
```

```
mysql> select n.neighborhood_id, n.neighbourhood_cleansed AS neighborhood_name, count(li.id) AS total_listings
-> from neighborhood_table n
-> join listings_table li on n.neighborhood_id = li.neighborhood_id
-> group by n.neighborhood_id, n.neighbourhood_cleansed
-> order by total_listings desc;
```

neighborhood_id	neighborhood_name	total_listings
2	Jamaica Plain	343
10	South End	326
11	Back Bay	302
17	Fenway	290
22	Dorchester	269
25	Allston	260
15	Beacon Hill	194
18	Brighton	185
24	South Boston	174
16	Downtown	172
12	East Boston	150
9	Roxbury	144
8	North End	143
3	Mission Hill	124
13	Charlestown	111
23	South Boston Waterfront	83
7	Chinatown	71
1	Roslindale	56
14	West End	49
19	West Roxbury	46
20	Hyde Park	31
5	Bay Village	24
21	Mattapan	24
4	Longwood Medical Area	9
6	Leather District	5

25 rows in set (0.04 sec)

Rank Difference

```
select
  tr.neighborhood_id,
  tr.neighborhood_name,
  tr.review_rank as review_rank,
  tl.listing_rank as listing_rank,
  tr.review_rank - tl.listing_rank as rank_difference
from (
  select
    neighborhood_id,
    neighborhood_name,
    total_reviews,
    @review_rank := @review_rank + 1 as review_rank
  from (
    select
      n.neighborhood_id,
      n.neighbourhood_cleansed as neighborhood_name,
      sum(r.number_of_reviews) as total_reviews
    from neighborhood_table n
    join listings_table li on n.neighborhood_id = li.neighborhood_id
    join review_table r on li.id = r.listing_id
    group by n.neighborhood_id, n.neighbourhood_cleansed
    order by total_reviews desc
  ) subquery, (select @review_rank := 0) r
) tr
join (
  select
    neighborhood_id,
    neighborhood_name,
    total_listings,
    @listing_rank := @listing_rank + 1 as listing_rank
  from (
```

```

select
    n.neighborhood_id,
    n.neighbourhood_cleansed as neighborhood_name,
    count(li.id) as total_listings
from neighborhood_table n
join listings_table li on n.neighborhood_id = li.neighborhood_id
group by n.neighborhood_id, n.neighbourhood_cleansed
order by total_listings desc
) subquery, (select @listing_rank := 0) r
) t1
on tr.neighborhood_id = t1.neighborhood_id
order by rank_difference;

```

neighborhood_id	neighborhood_name	review_rank	listing_rank	rank_difference
8	North End	5	13	-8
12	East Boston	4	11	-7
22	Dorchester	2	5	-3
4	Longwood Medical Area	21	24	-3
21	Mattapan	20	23	-3
9	Roxbury	9	12	-3
19	West Roxbury	18	20	-2
1	Roslindale	16	18	-2
24	South Boston	8	9	-1
13	Charlestown	14	15	-1
6	Leather District	25	25	0
2	Jamaica Plain	1	1	0
15	Beacon Hill	7	7	0
3	Mission Hill	15	14	1
23	South Boston Waterfront	17	16	1
20	Hyde Park	22	21	1
5	Bay Village	23	22	1
10	South End	3	2	1
18	Brighton	10	8	2
7	Chinatown	19	17	2
16	Downtown	13	10	3
11	Back Bay	6	3	3
25	Allston	11	6	5
14	West End	24	19	5
17	Fenway	12	4	8

25 rows in set, 4 warnings (0.65 sec)

3.3. Average price per neighborhood

Answer:

Here is the list of neighborhoods with their respective average listing prices: \

1. Bay Village: \$266.83
2. South Boston Waterfront: \$254.90
3. Leather District: \$253.60
4. Back Bay: \$236.81
5. Downtown: \$236.46
6. Chinatown: \$232.35
7. Beacon Hill: \$211.15
8. West End: \$209.59

9. South End: \$200.37
10. Fenway: \$197.49
11. North End: \$195.67
12. Charlestown: \$189.05
13. South Boston: \$181.87
14. Jamaica Plain: \$138.48
15. Longwood Medical Area: \$138.44
16. Roxbury: \$136.61
17. Mission Hill: \$121.97
18. East Boston: \$119.15
19. Brighton: \$118.76
20. Allston: \$112.31
21. West Roxbury: \$107.11
22. Roslindale: \$98.43
23. Dorchester: \$91.63
24. Hyde Park: \$86.54
25. Mattapan: \$75.12

These averages are helpful in understanding the market positioning and affordability of accommodations in each neighborhood.

Query:

```
select
  n.neighborhood_id,
  n.neighbourhood_cleansed AS neighborhood_name,
  avg(l.price) AS average_price
from
  neighborhood_table n
join
  listings_table l
  on n.neighborhood_id = l.neighborhood_id
group by
  n.neighborhood_id,
  n.neighbourhood_cleansed
order by
  average_price desc;
```

```
mysql> select
->     n.neighborhood_id,
->     n.neighbourhood_cleansed AS neighborhood_name,
->     avg(l.price) AS average_price
-> from
->     neighborhood_table n
-> join
->     listings_table l
->     on n.neighborhood_id = l.neighborhood_id
-> group by
->     n.neighborhood_id,
->     n.neighbourhood_cleansed
-> order by
->     average_price desc;
```

neighborhood_id	neighborhood_name	average_price
5	Bay Village	266.83333333333333
23	South Boston Waterfront	254.90361445783134
6	Leather District	253.6
11	Back Bay	236.81456953642385
16	Downtown	236.4593023255814
7	Chinatown	232.35211267605635
15	Beacon Hill	211.15463917525773
14	West End	209.59183673469389
10	South End	200.36503067484662
17	Fenway	197.49310344827586
8	North End	195.67832167832168
13	Charlestown	189.04504504504504
24	South Boston	181.867816091954
2	Jamaica Plain	138.47813411078718
4	Longwood Medical Area	138.44444444444446
9	Roxbury	136.61805555555555
3	Mission Hill	121.96774193548387
12	East Boston	119.15333333333334
18	Brighton	118.76756756756757
25	Allston	112.3076923076923
19	West Roxbury	107.1086956521739
1	Roslindale	98.42857142857143
22	Dorchester	91.63940520446097
20	Hyde Park	86.54838709677419
21	Mattapan	75.125

25 rows in set (0.24 sec)

We are happy to discuss these findings and our insights in more detail at your convenience.

Best regards,

Zhiyi Ji, Xinyi Huang

DBA Team