



## **1. Introduction**

Yelp is a digital platform that provides users the ability to discover and obtain information about various businesses, such as their location, contact information, operating hours, and more. Additionally, it offers a feature where users can read reviews and ratings about businesses posted by fellow users, as well as the opportunity to write and share their own evaluations and remarks. By incorporating social networking elements, Yelp encourages its users to exchange their business experiences and viewpoints, aiding others in identifying superior services and establishments. Established in 2004 in the United States, Yelp has since broadened its reach to several other countries, including Canada, the United Kingdom, Australia, France, Germany, Italy, and Switzerland, among others.

## **2. Background**

This report analyzes a series of tables from the Yelp database. The database contains a total of 6 tables, which include information on businesses, business operating hours, business registration, user information, reviews, and tips.

From the tables, we can determine that the database's statistics start from 2004 and end in 2017. All the businesses included in the statistics are located in the United States, including information on businesses that have closed. In addition to the location, type of business, and operating hours, the data also provides the platform's annual ratings for businesses. For users, the platform mainly displays information through reviews and user ratings. The platform also provides a table specifically to display review content, ratings, and the businesses and users associated with the reviews.

### **3. Tables Interpretation**

Yelp database contains the following 6 tables:

1. **yelp\_business**: It has 174567 rows of data. This table contains information about business on Yelp, including their ID, name, neighborhood, address, city, state, postal code, latitude, longitude, stars, review count, operation status, and categories.
2. **yelp\_business\_hours**: It has 174567 rows of data. This table contains information about the hours of operation of business, including the opening and closing time for each day of the week.
3. **yelp\_checkin**: It has 146350 rows of data. This table contains check-in information for businesses, such as the time and date when users check in.
4. **yelp\_review**: It has 5261668 rows of data. This table contains reviews of business, including the user ID, business\_id, stars for the business, date of the review, text of the review, 'useful' count for the review, 'funny' count for the review, and 'cool' count for the review.
5. **yelp\_tip**: It has 1098324 rows of data. This table contains tips and suggestions from users about businesses, such as recommended dishes or things to watch out for. It contains the following columns: text, date, likes, business\_id, and user\_id.
6. **yelp\_user**: It has 1326100 rows of data. This table contains information about yelp users, including their ID, name, registration time, average rating, review count, and various review stats.

### **4. Analysis Objective**

To analyze the specifics of this online sale and to propose improvements for the next sale, I have analyzed the data from the following three perspectives:

**1. Customer perspective:**

(1) On average, how many reviews per month do regular users post since creating their account, and how many do elite users post? How often are reviews considered useful, funny, or cool in terms of numbers and percentages?

(2) Do reviews with a high number of useful votes tend to be more positive or negative? What are the length of the reviews? What about funny and cool votes?

(3) What is the average score of user reviews? What is the distribution of restaurant star ratings at each review score?

(4) What is the relationship between the number of reviews, number of photos, and the number of followers? What is the relationship between the number of followers and elite users?

(5) Which cities are most frequently reviewed by users?

**2. Restaurant perspective:**

(1) What is the number of restaurants at each star level? What is the average star rating of restaurants? What is the relationship between a restaurant's star rating and the number of reviews, average review score, and the ratio of good to bad reviews?

(2) In which states are there more restaurants? Which states have a higher proportion of five-star restaurants? Which states have a high rate of positive restaurant reviews? What is the relationship to the geographical location?

(3) What is the relationship between the star rating of a restaurant and the number of days it is open for business?

(4) What are the characteristics of the operation days and star ratings of the restaurants that have closed?

**3. Platform perspective:**

(1) How many users registered each year and each month? How many user reviews?

(2) How many elite customers are there each year, and what is their percentage?

(3) What is the annual retention rate of users? What about the annual retention rate of elite users?

## **5. Data Preprocessing**

Before conducting data analysis, It is necessary to clean the data, I will examine if there is any missing or invalid data in each table one by one:

**1. yelp\_business:**

First, filter out all the invalid data, as follows in the code:

```
-- Select all columns from the 'yelp_business' table
SELECT *
FROM `yelp_business`
-- Filter the results where 'name' is NULL
-- or 'state' is NULL
-- or 'stars' is less than 0 or greater than 5
WHERE
    name IS NULL
    OR state IS NULL
    OR stars < 0
    OR stars > 5;
```

0 rows, so means all the rows of data in the table is valid and logic:

business_id	name	neighborhood	address	city	state	postal_code	latitude	longitude	stars
varchar(100)	varchar(100)	varchar(100)	varchar(200)	varchar(100)	varchar(100)	varchar(100)	varchar(100)	varchar(100)	decimal(2,1)

## 5.1 no invalid data from yelp\_business

### 2. yelp\_business\_hours:

See if there is any business that do not operate all seven days of the week and business without operating hours information, as follows in the code:

```
-- Select the data that have invalid operating hours
SELECT *
FROM `yelp_business_hours`
WHERE
  monday = "None"
  and tuesday = "None"
  and wednesday = "None"
  and thursday = "None"
  and friday = "None"
  and saturday = "None"
  and sunday = "None";
```

The result is shown below:

business_id	monday	tuesday	wednesday	thursday	friday	saturday	sunday
_4vsNZHVKIHPNsEzluQf	None	None	None	None	None	None	None
_4wwCUQy2stAL4LE6lsy	None	None	None	None	None	None	None
_5_6LzL1rBouQPOAVjdV-	None	None	None	None	None	None	None

## 5.2 yelp\_business\_hours

There are 45307 rows of data. It means that original dataset contains some business are permanently closed.

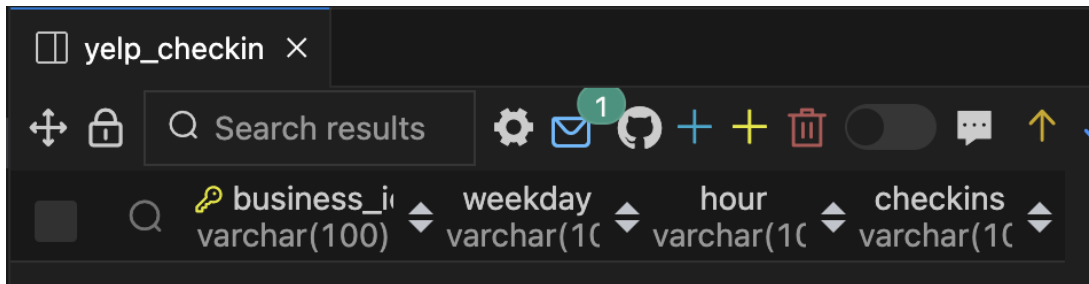
### 3. yelp\_checkin:

See if there is any invalid data. The code is below:

```
- see if there is any null values in yelp_checkin:
```

```
SELECT *  
FROM `yelp_checkin`  
WHERE  
    weekday IS NULL  
    OR hour IS NULL  
    OR checkins IS NULL;
```

No result, shows all the data is valid and logic:



### 5.3: yelp\_checkins

### 4. yelp\_review:

See if there is any invalid data. The code is below:

```
-- See if there is any null values in 'yelp_review' table
```

```
SELECT *  
FROM yelp_review  
  
-- Filter the results where 'review_id' is NULL  
-- or 'business_id' is NULL  
-- or 'user_id' is NULL  
-- or 'date' is NULL  
-- or 'stars' is less than 0 or greater than 5  
WHERE  
    review_id IS NULL  
    OR business_id IS NULL  
    OR user_id IS NULL
```

```
OR date IS NULL
OR stars < 0
OR stars > 5;
```

No results, means all the data is valid and logic:

yelp_review										
Search results										
Cost: 3s < 1 > Total 0										
	review_id	user_id	business_id	stars	date	text	useful	funny	cool	
	varchar(10)	varchar(10)	varchar(100)	int	date	longtext	int	int	int	

## 5.4: yelp\_review

### 5. yelp\_tip:

See if there is any invalid data. The code is below:

```
-- See if there is any null values in 'yelp_tip' table
SELECT *
FROM yelp_tip
-- Filter the results where 'text' is NULL
-- or 'business_id' is NULL
-- or 'user_id' is NULL
-- or 'date' is NULL
-- or 'likes' is less than 0
WHERE
text IS NULL
OR business_id IS NULL
OR user_id IS NULL
OR date IS NULL
OR likes < 0;
```

yelp_tip					
Search results					
Cost: 388ms < 1 > Total 0					
	text	date	likes	business_id	user_id
	varchar(10)	date	int	varchar(100)	varchar(10)

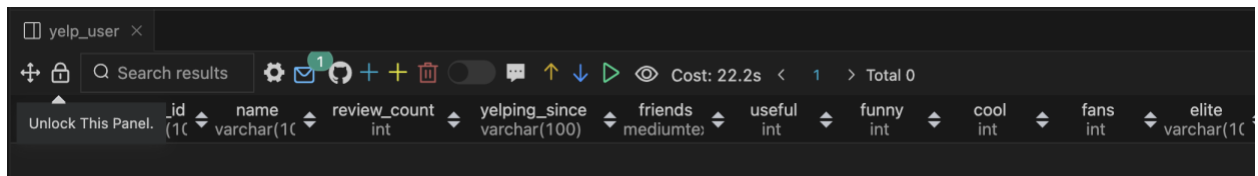


## 5.5: yelp\_tip

### 6. yelp\_user:

See if there is any invalid data. The code is below:

```
-- See if there is any null values in 'yelp_user' table
SELECT *
FROM `yelp_user`
-- Filter the results where any of the following conditions are met:
-- 'useful' is less than 0
-- 'funny' is less than 0
-- 'cool' is less than 0
-- 'fans' is less than 0
-- 'average_stars' is less than 0
-- 'average_stars' is greater than 5
WHERE
    useful < 0
    OR funny < 0
    OR cool < 0
    OR fans < 0
    OR average_stars < 0
    OR average_stars > 5;
```



Unlock This Panel.	_id	name	review_count	yelping_since	friends	useful	funny	cool	fans	elite
	(10)	varchar(100)	int	varchar(100)	mediumint	int	int	int	int	varchar(100)

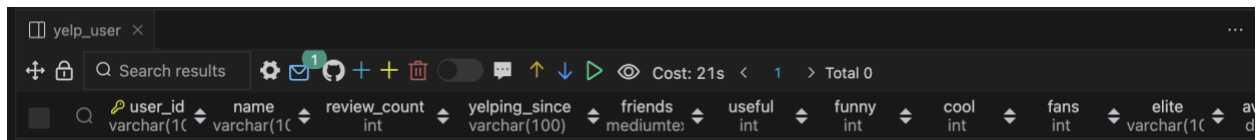
## 5.6: yelp\_user

No result, let's see if there is any row of data that has missing information on name, review, and registration.

```
-- Select all columns from the 'yelp_user' table
-- Where 'name', 'review_count', and 'yelping_since' is NULL
SELECT *
FROM `yelp_user`
```

```
WHERE
  name IS NULL
or review_count IS NULL
or yelping_since IS NULL;
```

0 rows, so all the data in the yelp\_user is valid:



The screenshot shows a database query interface with a table named 'yelp\_user'. The table has the following columns: user\_id (varchar(10)), name (varchar(100)), review\_count (int), yelping\_since (varchar(100)), friends (mediumtext), useful (int), funny (int), cool (int), fans (int), elite (varchar(10)), and average\_rating (float). The interface includes a search bar, a settings icon, a notification icon, and a toolbar with various actions like insert, update, delete, and export. The cost of the query is 21s, and the total number of rows is 0.

5.7 yelp\_user

## 6. Data Analysis

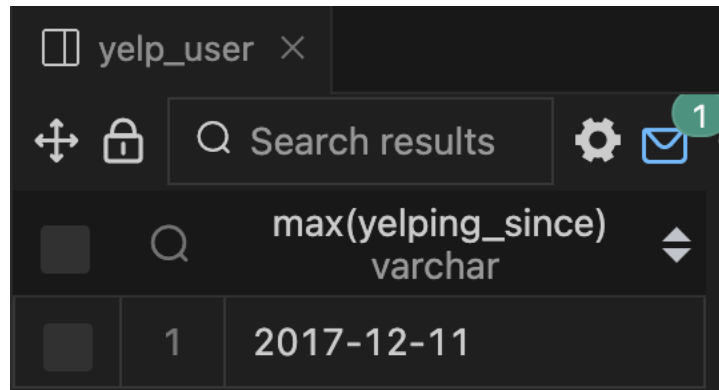
### 1. Customer perspective:

**(1). On average, how many reviews per month do regular users post since creating their account, and how many do elite users post? How often are reviews considered useful, funny, or cool in terms of numbers and percentages?**

First, I need to find out the final date of the data collection. Here, it is represented by the latest registration time of the users:

```
SELECT max(yelping_since)
FROM `yelp_user`;
```

The result is shown below:



#### 6.1.1.1 End date of the data

Display the registration days and average daily comment count for non-elite users:

```
-- Display the registration days and average daily comment count for non-elite users
SELECT
  user_id,
  DATEDIFF("2017-12-11", yelping_since) as register_time,
  (review_count / DATEDIFF("2017-12-11", yelping_since)) as avg_review_day
FROM `yelp_user`
WHERE elite = "None";
```

And calculate the average daily review for non elite users:

```
-- Calculate the average of 'avg_review_day' values for non-elite users.
SELECT AVG(sub.avg_review_day) as avg_review_day_nelite
FROM (
  -- Subquery to display 'avg_review_day' for non-elite users
  SELECT
    user_id,
    DATEDIFF("2017-12-11", yelping_since) as register_time,
    (review_count / DATEDIFF("2017-12-11", yelping_since)) as avg_review_day
  FROM `yelp_user`
  WHERE elite = "None"
) as sub;
```

We obtain the following results:

avg_review_day_elite newdecimal
0.00959149

#### 6.1.1.2 Average review posted by non elite users.

Count the number of comments that are rated as 'useful', 'funny' or 'cool ':

```
-- Calculate the rates of 'useful','funny', and 'cool' votes per review:
```

```
SELECT
```

```
    SUM(review_count) as total_review_count,  
    SUM(useful) / SUM(review_count) as useful_rate_ne,  
    SUM(funny) / SUM(review_count) as funny_rate_ne,  
    SUM(cool) / SUM(review_count) as cool_rate_ne
```

```
FROM `yelp_user`
```

```
WHERE elite = "None";
```

It was found that the counts of useful, funny, and cool evaluations are respectively 0.59, 0.21, and 0.21 times the total number of evaluations:

Q	total_review_count newdecimal	useful_rate_ne newdecimal	funny_rate_ne newdecimal	cool_rate_ne newdecimal
1	16857521	0.5905	0.2141	0.2072

#### 6.1.1.3 non elite users post stats

Using the same method, analyze the elite users. First, find out the number of comments made by elite users per day:

```
-- Calculate the average of 'avg_review_day' values for elite users.
```

```
SELECT AVG(sub.avg_review_day) as avg_review_day_elite
```

```
FROM (
```

```
    -- Subquery to calculate 'avg_review_day' for elite users
```

```
    SELECT
```

```

user_id,
DATEDIFF("2017-12-11", yelping_since) as register_time,
(review_count / DATEDIFF("2017-12-11", yelping_since)) as avg_review_day
FROM `yelp_user`
WHERE elite <> "None"
) as sub;

```

It can be seen that the average number of comments made by elite users per day is 0.0979, which is approximately 10.2 times that of non elite users:

	avg_review_day_elite newdecimal
1	0.09788794

#### 6.1.1.4 Average daily post for elite users

Next, calculate the proportions of useful, funny, and cool ratings received by comments from elite users:

```

-- Calculate the total review count for elite users,
-- as well as the rates of 'useful,' 'funny,' and 'cool' votes per review.
SELECT
SUM(review_count) as total_review_count,
SUM(useful) / SUM(review_count) as useful_rate_e,
SUM(funny) / SUM(review_count) as funny_rate_e,
SUM(cool) / SUM(review_count) as cool_rate_e
FROM `yelp_user`
WHERE elite <> "None";

```

The result is shown below:

total_review_count newdecimal	useful_rate_e newdecimal	funny_rate_e newdecimal	cool_rate_e newdecimal
13798162	2.0988	1.1426	1.6309

#### 6.1.1.5: Elite user post stats

It is observed that the numbers of useful, funny, and cool ratings received by elite users are respectively 2.10, 1.14, and 1.63 times the number of reviews they posted, which are 3.56, 5.43, and 7.76 times those of nonelite users, respectively. Elite users receive more rating for their reviews than the reviews they post. Elite users visit more shops and post significantly more reviews than ordinary users, and the likelihood of their reviews being praised is higher, indicating that their reviews are generally better written.

It was also discovered that, regardless of the type of user, the probability of a review being rated as useful is much higher than that of being rated as funny or cool. This indicates that when users write reviews, they pay more attention to the content rather than the style of writing or choice of words; similarly, when other users read reviews, they are more concerned with whether the content of the review is useful.

**(2). Do reviews with a high number of useful votes tend to be more positive or negative? What are the length of the reviews? What about funny and cool votes?**

Count the number of comments that received more than 1000, 500, 200, and 100 useful ratings, respectively, the average rating these comments gave to the restaurants, and the average length of these comments.

When 'useful' values greater than 1000

```
-- Calculate the count of reviews with 'useful' values greater than 1000,
SELECT
COUNT(sub.text) as review_count,
ROUND(AVG(sub.stars), 1) as avg_stars,
```

```

ROUND(AVG(sub.len), 0) as avg_length
FROM (
  -- Subquery to select relevant data from yelp_review
  SELECT stars, text, LENGTH(text) as len, useful
  FROM yelp_review
  WHERE useful > 1000
) as sub;

```

For comments with more than 1000 useful votes, there are a total of 11, with an average rating of 1.4 stars given to the restaurants, and the average comment length is 2204 words.

review_count bigint	avg_stars newdecimal	avg_length newdecimal
11	1.3	2204

#### 6.1.2.1 Average rating and comment length for users when the useful count is greater than 1000

When 'useful' values greater than 500:

```

-- Calculate the count of reviews with 'useful' values greater than 500,
SELECT
  COUNT(sub.text) as review_count,
  ROUND(AVG(sub.stars), 1) as avg_stars,
  ROUND(AVG(sub.len), 0) as avg_length
FROM (
  -- Subquery to select relevant data from yelp_review
  SELECT stars, text, LENGTH(text) as len, useful
  FROM yelp_review
  WHERE useful > 500
) as sub;

```

For comments with more than 500 useful votes, there are a total of 38, with an average rating of 1.3 stars given to the restaurants, and the average comment length is 1456 words.

	review_count bigint	avg_stars newdecimal	avg_length newdecimal
1	38	1.3	1456

#### 6.1.2.2 Average rating and comment length for users when the useful count is greater than 500

When 'useful' count greater than 200:

```
-- Calculate the count of reviews with 'useful' values greater than 200,
SELECT
  COUNT(sub.text) as review_count,
  ROUND(AVG(sub.stars), 1) as avg_stars,
  ROUND(AVG(sub.len), 0) as avg_length
FROM (
  -- Subquery to select relevant data from yelp_review
  SELECT stars, text, LENGTH(text) as len, useful
  FROM yelp_review
  WHERE useful > 200
) as sub;
```

For comments with more than 200 useful votes, there are a total of 138, with an average rating of 1.8 starts given to the restaurants, and the average comment length is 1342 words.

review_count bigint	avg_stars newdecimal	avg_length newdecimal
138	1.8	1342

#### 6.1.2.3 Average rating and comment length for users when the useful count is greater than 100

When 'useful' count greater than 100:



```

-- Calculate the count of reviews with 'useful' values greater than 100,
SELECT
  COUNT(sub.text) as review_count,
  ROUND(AVG(sub.stars), 1) as avg_stars,
  ROUND(AVG(sub.len), 0) as avg_length
FROM (
  -- Subquery to select relevant data from yelp_review
  SELECT stars, text, LENGTH(text) as len, useful
  FROM yelp_review
  WHERE useful > 100
) as sub;

```

For comments with more than 100 useful votes, there are a total of 408, with an average rating of 2.4 stars given to the restaurants, and the average comment length is 1479 words.

review_count bigint	avg_stars newdecimal	avg_length newdecimal
408	2.4	1479

### 6.1.2.3 Average rating and comment length for users when the useful count is greater than 100

From the data above, it is evident that there is a positive correlation between the length of comments and the likelihood of receiving useful votes. Moreover, comments with a higher number of useful votes tend to give lower average ratings to restaurants. This suggests that compared to positive reviews, people often consider negative reviews to provide more useful information about restaurants.

Use the same method to analyze the cases for funny and cool ratings. Since users prefer to rate comments as useful and there are fewer funny ratings, this report selects 500, 200, and 100 as thresholds for funny.

When 'funny' count greater than 500:

```
-- Calculate the count of reviews with 'funny' values greater than 500,
select count(sub.text), round(avg(sub.stars),1), round(avg(sub.len),0)
from(SELECT stars, text, length(text) as len, funny
FROM yelp_review
WHERE funny>500
) as sub
```

For comments with more than 500 funny votes, there are a total of 16, with an average rating of 3.3 stars given to the restaurants, and the average comment length is 1752 words.

count(sub.text) bigint	round(avg(sub.stars),1) newdecimal	round(avg(sub.len),0) newdecimal
16	3.3	1752

#### 6.1.2.4 Average rating and comment length for users when the funny count is greater than 500

When 'funny' count greater than 200:

```
-- Calculate the count of reviews with 'funny' values greater than 200,
select count(sub.text), round(avg(sub.stars),1), round(avg(sub.len),0)
from(SELECT stars, text, length(text) as len, funny
FROM yelp_review
WHERE funny>200
) as sub
```

For comments with more than 200 funny votes, there are a total of 86, with an average rating of 3.6 stars given to the restaurants, and the average comment length is 1323 words.

	count(sub.text) bigint	round(avg(sub.stars),1) newdecimal	round(avg(sub.len),0) newdecimal
1	86	3.6	1323

#### 6.1.2.5 Average rating and comment length for users when the funny count is greater than 200

When 'funny' count greater than 100:

```
-- Calculate the count of reviews with 'funny' values greater than 100,
select count(sub.text), round(avg(sub.stars),1), round(avg(sub.len),0)
from(SELECT stars, text, length(text) as len, funny
FROM yelp_review
WHERE funny>100
) as sub
```

For comments with more than 100 funny votes, there are a total of 261, with an average rating of 3.6 stars given to the restaurants, and the average comment length is 1386 words.

count(sub.text) bigint	round(avg(sub.stars),1) newdecimal	round(avg(sub.len),0) newdecimal
261	3.6	1386

#### 6.1.2.6 Average rating and comment length for users when the funny count is greater than 100

The number of cool votes is even fewer, also using 500,200, and 100 as the thresholds:

```
-- Calculate the count of reviews with 'cool' values greater than 500,
select count(sub.text), round(avg(sub.stars),1), round(avg(sub.len),0)
from(SELECT stars, text, length(text) as len, cool
FROM yelp_review
WHERE cool>500
) as sub
```

For comments with more than 500 cool votes, there is only 1, with an average rating of 1.0 star given to the restaurant, and the average comment length is 4133 words. This comment also received evaluations for useful and funny. This comment can be considered an outlier, and in this

context, it is not representative for data statistics, therefore it will not be considered in subsequent analyses.

	count(sub.text) bigint	round(avg(sub.stars),1) newdecimal	round(avg(sub.len),0) newdecimal
1	1	1.0	4133

#### 6.1.2.7 Average rating and comment length for users when the cool count is greater than 500

When 'cool count greater than 200:

```
-- Calculate the count of reviews with 'cool' values greater than 200,  
select count(sub.text), round(avg(sub.stars),1), round(avg(sub.len),0)  
from(SELECT stars, text, length(text) as len, cool  
FROM yelp_review  
WHERE cool>200  
 ) as sub
```

For comments with more than 200 cool votes, there are a total of 26, with an average rating of 3.4 stars given to the restaurants, and the average comment length is 1584 words.

count(sub.text) bigint	round(avg(sub.stars),1) newdecimal	round(avg(sub.len),0) newdecimal
26	3.4	1584

#### 6.1.2.8 Average rating and comment length for users when the cool count is greater than 200

When 'cool count greater than 100:

```
-- Calculate the count of reviews with 'cool' values greater than 100,  
select count(sub.text), round(avg(sub.stars),1), round(avg(sub.len),0)  
from(SELECT stars, text, length(text) as len, cool  
FROM yelp_review  
WHERE cool>100  
 ) as sub
```

For comments with more than 100 cool votes, there are a total of 151, with an average rating of 3.6 stars given to the restaurants, and the average comment length is 1721 words.

count(sub.text) bigint	round(avg(sub.stars),1) newdecimal	round(avg(sub.len),0) newdecimal
151	3.6	1721

#### 6.1.2.9 Average rating and comment length for users when the cool count is greater than 100

Based on the data and observation, for funny and cool ratings, the length of review and the star rating given to restaurants do not significantly influence whether users perceive a comment as cool or funny. This differs from how users assess whether a comment is useful or not.

### **(3) What is the average score of user reviews? What is the distribution of restaurant star ratings at each review score?**

First, for restaurants that have been rated with different scores by users, calculate the average star rating for these restaurants.

```
-- Calculate the average business star rating for each unique review star rating,  
-- ordering the results by review star rating in descending order.  
SELECT  
    yelp_review.stars,  
    ROUND(AVG(yelp_business.stars), 2) as avg_busi_star  
FROM  
    yelp_review  
LEFT JOIN  
    yelp_business  
ON  
    yelp_review.business_id = yelp_business.business_id  
GROUP BY
```

```
yelp_review.stars
ORDER BY
yelp_review.stars DESC;
```

The table's left side showing the user rating and the right side showing the average star rating of restaurants under that user rating:

stars int	avg_busi_star newdecimal
5	4.08
4	3.74
3	3.52
2	3.37
1	3.03

Table 6.1.3.1: The distribution of restaurant star ratings at each user rating



Graph 6.1.3.2: Distribution of restaurant star ratings at each user rating level

As we can observe, the higher the star rating given by users, the higher the average score of the restaurants, and the distribution is about uniform, which is consistent with common sense understanding.

Now count the proportion of each star rating category among restaurants rated as 5 stars by users:

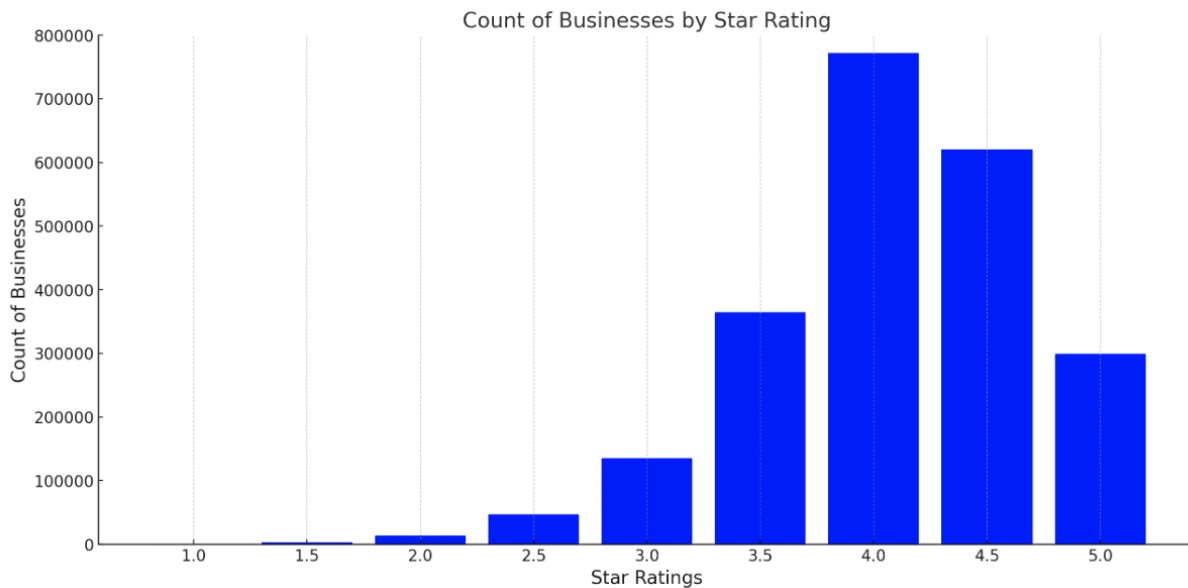
```
-- Calculate the count of businesses, count of reviews with a star rating of 5,  
SELECT  
  yelp_business.stars,  
  COUNT(yelp_business.business_id) AS count_busi  
FROM  
  yelp_review  
LEFT JOIN  
  yelp_business  
ON
```

```

yelp_review.business_id = yelp_business.business_id
WHERE
yelp_review.stars = 5
GROUP BY
yelp_business.stars
ORDER BY
yelp_business.stars DESC;

```

The bar plot of the result is shown below:



Graph 6.1.3.3: Proportion of restaurant star ratings in 5-star user reviews

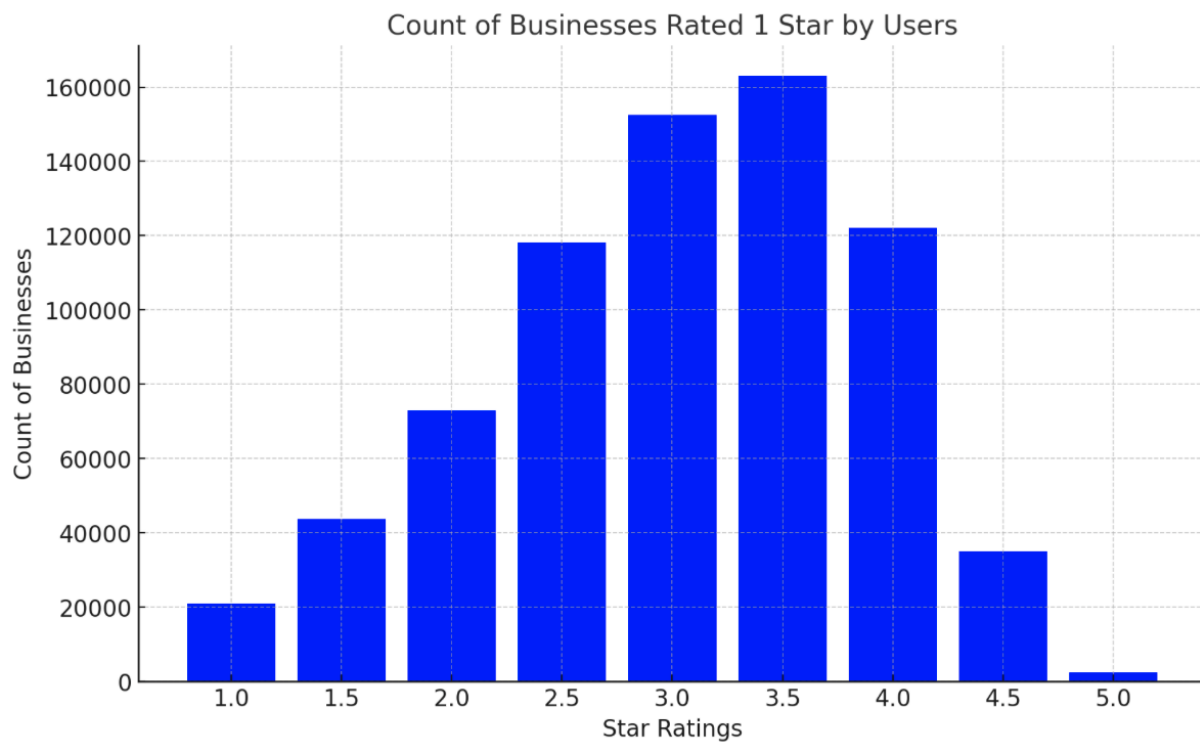
There is a certain gap between the star rating distribution of restaurants rated with 5 stars and the overall distribution of all restaurants star ratings. Firstly, 5-star reviews are more concentrated in restaurants with a rating of 4.0 and above, while restaurants below 3 stars hardly receive any 5-star reviews.

Now, Count the proportion of each star rating category among restaurants rated as 1 star by users:



```
-- Calculate the count of businesses, count of reviews with a star rating of 1,
SELECT
  yelp_business.stars,
  COUNT(yelp_business.business_id) AS count_busi
FROM
  yelp_review
LEFT JOIN
  yelp_business
ON
  yelp_review.business_id = yelp_business.business_id
WHERE
  yelp_review.stars = 1
GROUP BY
  yelp_business.stars
ORDER BY
  yelp_business.stars DESC;
```

The bar plot of the result is shown below:



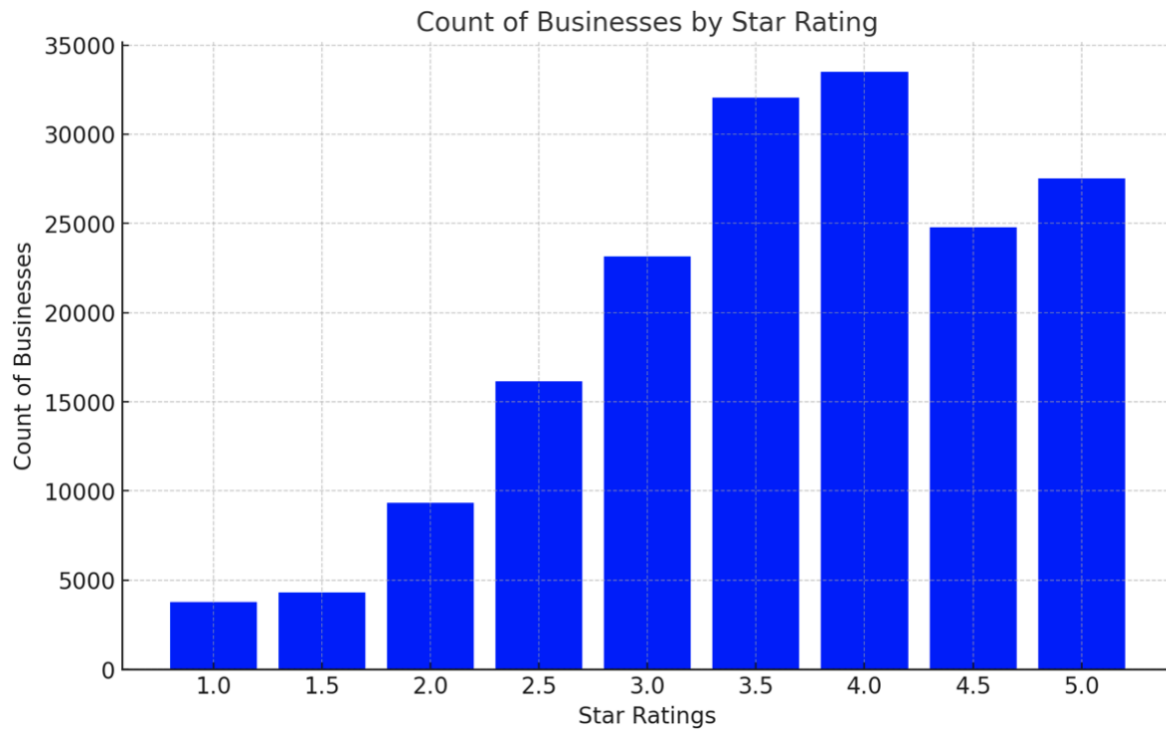
#### Graph 6.1.3.4: Proportion of restaurant star ratings in 1-star user reviews

The star rating distribution of restaurants rated with 1 star is relatively similar to the overall distribution of all restaurant star ratings, indicating that negative reviews from users are not greatly influenced by the restaurant's own star rating. However, the rate of negative reviews for five-star restaurants is exceptionally low, suggesting that the rate of negative reviews may also be an important indicator in the evaluation of five-star restaurants.

The code for overall distribution of all restaurant star rating:

```
-- Calculate the count of restaurant respect to star rating
select
  yelp_business.stars,
  count(yelp_business.business_id) as count_busi
from yelp_business
group by
  yelp_business.stars
order by
  yelp_business.stars DESC;
```

The bar plot of the resulted table from code is shown below:



**(4) What is the relationship between the number of reviews, number of photos, and the number of followers? What is the relationship between the number of followers and elite users?**

First, to analyze the relationship between the number of reviews, number of photos, and number of followers for elite users:

```
-- Relationship between number of reviews, number of photos, and number of followers for elite users
SELECT sum(fans), sum(review_count), sum(compliment_photos), sum(review_count)/sum(
fans)as fans_re_rate_e, sum(compliment_photos)/sum(fans)as fans_ph_rate_e
FROM `yelp_user`
WHERE elite<>"None";
```

The results are as follows, with the columns in the table sequentially representing "Total Number of Followers," "Total Number of Reviews," "Total Number of Photos," "Ratio of Reviews to Followers," and "Ratio of Photos to Followers":

sum(fans) newdecimal	sum(review_count) newdecimal	sum(compliment_photos) newdecimal	fans_re_rate_e newdecimal	fans_ph_rate_e newdecimal
1334249	13798162	1347109	10.3415	1.0096

#### 6.1.4.1 Elite users

It is apparent that the number of reviews posted by elite users is about 10 times the number of their followers, while the number of followers is nearly equal to the number of photos posted by elite users.

Now calculate for non-elite users:

```
-- Calculate for non-elite users
SELECT sum(fans), sum(review_count), sum(compliment_photos), sum(review_count)/sum(
fans)as fans_re_rate_e, sum(compliment_photos)/sum(fans)as fans_ph_rate_e
FROM `yelp_user`
WHERE elite="None";
```

The result is shown below

sum(fans) newdecimal	sum(review_count) newdecimal	sum(compliment_photos) newdecimal	fans_re_rate_e newdecimal	fans_ph_rate_e newdecimal
598242	16857521	227730	28.1784	0.3807

#### 6.1.4.2 Number of review, pictures, and fans of non-elite users

It is observed that the number of reviews posted by non-elite users is about 28 times the number of their followers, while the number of photos posted is about 0.38 times the number of their followers.

Comparing the data of elite users with that of non-elite users, it is found that the total number of reviews posted by both elite and non-elite users is roughly the same, but the reviews of elite users are more likely to attract followers, with a higher probability of gaining followers. Elite users post far more photos than non-elite users, about 6 times more; however, the total

number of followers is less than 3 times that of ordinary users, which suggests that compared to reviews, photos are less likely to attract the attention of other users and have a lower probability of gaining followers.

Obtain the number of fans for non-elite users:

```
-- Number of Non-elite users' fans
SELECT count(user_id), sum(fans), sum(fans)/count(user_id) as fans_rate_ne
FROM `yelp_user`
WHERE elite="None";
```

The result is shown below:

count(user_id) bigint	sum(fans) newdecimal	fans_rate_ne newdecimal
1265282	598242	0.4728

#### 6.1.4.3 Proportion of fan to non-elite users

The total number of followers for non-elite users is about 0.47 times the number of non-elite users, with an average of less than one follower per person, indicating that a large number of non-elite users have no followers.

Now, let's calculate the number of followers for elite users:

```
-- Number of fans for elite users
SELECT count(user_id), sum(fans), sum(fans)/count(user_id) as fans_rate_e
FROM `yelp_user`
WHERE elite<>"None";
```

count(user_id) bigint	sum(fans) newdecimal	fans_rate_e newdecimal
60818	1334249	21.9384

### 6.1.4.3 Proportion of fan to elite users

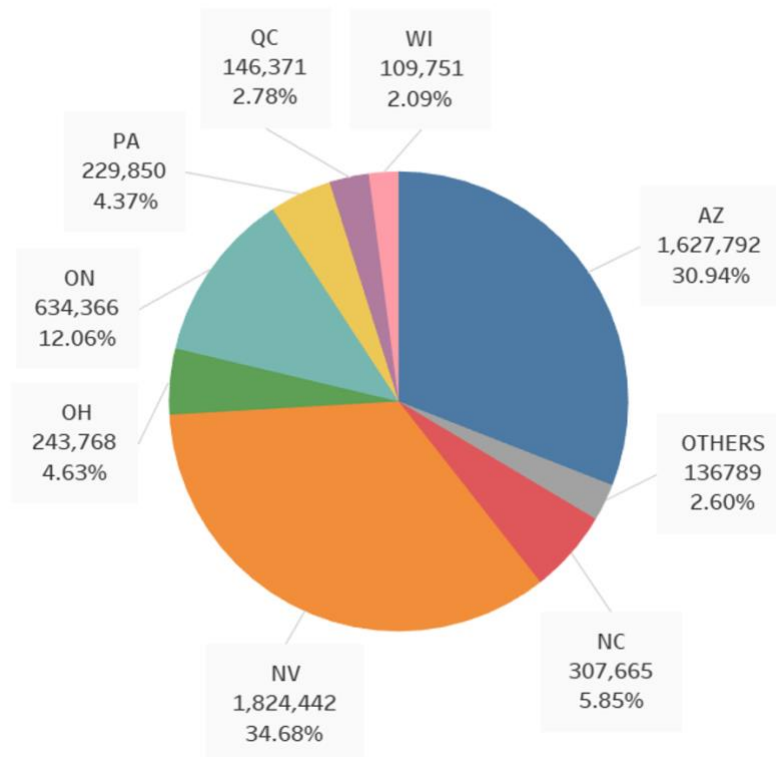
Each elite user has an average of about 22 followers, which is more than 45 times that of non-elite users. It is speculated that when Yelp selects elite users each year, the number of followers a user has is an important metric. The number of followers not only indicates a user's high dependency on the platform and that they provide more valuable information, but it also brings more trends, which is essential for the platform.

#### **(5) Which cities are most frequently reviewed by users?**

First, let's calculate the number of reviews for each state:

```
--Number of reviews for each state
SELECT yelp_business.state, COUNT(yelp_review.review_id) AS count_re_state
FROM yelp_review LEFT JOIN yelp_business
ON yelp_review.business_id = yelp_business.business_id
GROUP BY yelp_business.state
ORDER BY count_re_state DESC;
```

	state varchar	count_re_state bigint
1	NV	1824442
2	AZ	1627792
3	ON	634366
4	NC	307665
5	OH	243768
6	PA	229850
7	QC	146371
8	WI	109751
9	EDH	47889
10	IL	36467
11	BW	35400



6.1.5.1 Percentage of states most frequently reviewed by users

The top two states most frequently reviewed by users account for 65.62% of all reviews, with Nevada accounting for 34.68% and Arizona for 30.94%. Following them, with high proportions, are North Carolina, Ohio, Pennsylvania, etc. The high number of reviews in these states also indicates that users are mainly active in these cities.

## 2. Restaurant perspective:

**(1) What is the number of restaurants at each star level? What is the average star rating of restaurants? What is the relationship between a restaurant's star rating and the number of reviews, average review score, and the ratio of good to bad reviews?**

First, calculate the average star rating of restaurants:

```
-- Average stars for restaurants  
SELECT round(avg(stars),1)  
FROM yelp_business;
```

The result is below:

round(avg(stars),1) newdecimal
3.6

### 6.2.1.1 Average star rating for restaurant

Since the star rating levels range from 1 to 5 stars, with each half-star representing a tier, if the distribution of restaurant star ratings is approximately normal, then the average star rating should be around 3 stars. The data shows that the average is higher than 3 stars, indicating that most restaurants have received a higher star rating above 3. This suggests that the platform tends not to give low star ratings to restaurants.

Now, count the number of restaurants for each star rating:

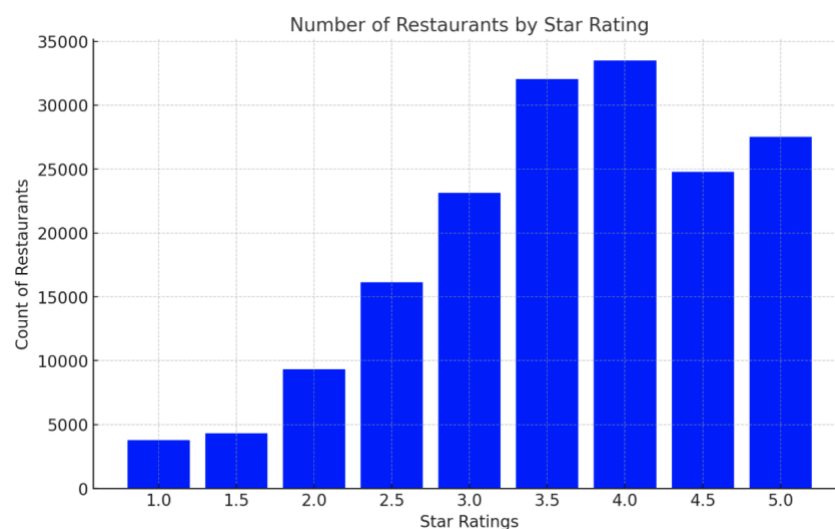


```
-- Number of restaurant for each star rating
SELECT stars, count(business_id)
FROM yelp_business
GROUP BY stars
ORDER BY stars DESC;
```

The result is shown below, with the left side showing the restaurant star rating and the right side showing the corresponding number of restaurants:

stars	count(business_id)
newdecimal	bigint
5.0	27540
4.5	24796
4.0	33492
3.5	32038
3.0	23142
2.5	16148
2.0	9320
1.5	4303
1.0	3788

#### 6.2.1.2 Number of restaurants by star rating



### 6.2.1.3 plot of number of restaurants by star rating

The histogram roughly shows a left-skewed distribution, with more higher-rated restaurants than lower-rated ones, which corresponds with the information presented by the average value. It can be observed that restaurants with a 4.0 rating are the most numerous, with 33492 of them, followed closely by those with a 3.5 rating, with 32038. The majority of restaurants have achieved a level of customer satisfaction that is generally acceptable.

It is noteworthy that there is a very high number of 5-star restaurants, with 15.78% of all restaurants achieving this rating, even surpassing the number of 4.5-star restaurants. On one hand, a user's rating can affect a restaurant's star rating, and subjectively, when users have an excellent dining experience, they tend to give full marks to express their appreciation, support, and fondness without paying much attention to the restaurant's flaws, which in such cases are usually considered as "minor blemishes" or "uniquely characteristic." On the other hand, the platform tends to promote 5-star restaurants over those with 4.5 stars, which is beneficial for advertising restaurants on the platform, as a perfect score is a powerful "hook" in user promotion compared to 4.5 stars. At the same time, restaurants promoted by the platform can reciprocate by offering exclusive deals on Yelp, bringing significant traffic to both the businesses themselves and the platform.

For each star level of the restaurants, calculate the average score from user reviews and the number of review:

```
-- Calculate the average review star rating and count of reviews for each business star rating
SELECT
  yelp_business.stars,
  ROUND(AVG(yelp_review.stars), 2) as avg_re_star,
  COUNT(review_id) as count_re
```

```

FROM
  yelp_business
RIGHT JOIN
  yelp_review
ON
  yelp_business.business_id = yelp_review.business_id
GROUP BY
  yelp_business.stars
ORDER BY
  yelp_business.stars DESC;

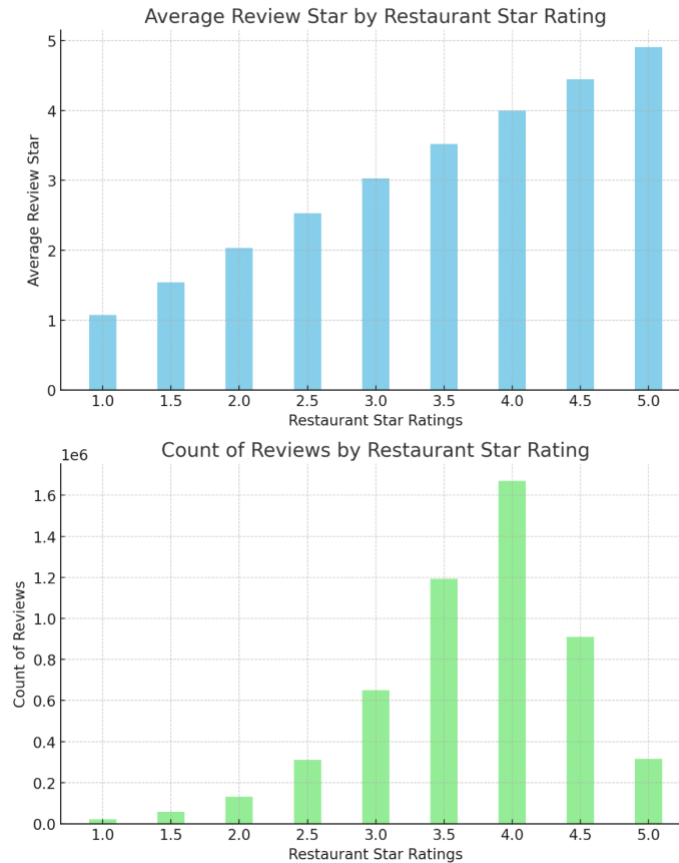
```

The result is below, with the left side showing the restaurant star rating, the middle showing the average user rating from reviews, and the right side showing the number of user reviews:

stars newdecimal	avg_re_star newdecimal	count_re bigint
5.0	4.91	315730
4.5	4.45	909666
4.0	4.00	1670164
3.5	3.52	1193114
3.0	3.03	650162
2.5	2.53	311991
2.0	2.03	130194
1.5	1.54	58598
1.0	1.07	22049

6.2.1.4 Table of user rating for restaurant of different stars

6.2.1.4 Table of user rating for restaurant of different stars



#### 6.2.1.5 Bar plot of user rating and number of review for restaurant of different stars

The upper plot present a histogram of the average user rating as it varies with the restaurant star levels. The average user rating increases with the restaurant star level, displaying a roughly uniform distribution, which aligns with common sense.

The lower side shows the trend for the number of user reviews in relation to the restaurant star levels, generally displaying a left-skewed distribution and peaking at the 4.0-star rating. This is partly because there are more restaurants with a 4.0-star rating, making them the most accessible, and also indicates that 4.0-star restaurants meet most users' daily dining, exploring, and socializing needs. The chart also shows that the number of reviews for 5-star restaurants is low, especially in relation to the large base number of 5-star restaurants. In combination with the above analysis, this further suggests that the selection of 5-star restaurants

is largely influenced by the platform. This may also be due to rigid factors such as generally higher prices and limited seating capacity at 5-star restaurants.

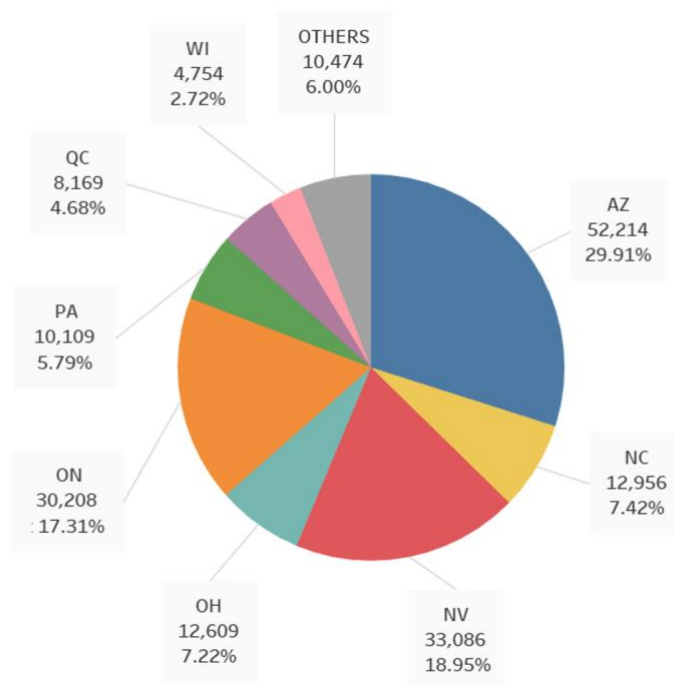
**(2) In which states are there more restaurants? Which states have a higher proportion of five-star restaurants? Which states have a high rate of positive restaurant reviews? What is the relationship to the geographical location?**

First, calculate the total number of restaurants in each state:

```
-- Calculate the total number of restaurants in each state:
```

```
SELECT state, count(state)
FROM yelp_business
GROUP BY state
ORDER BY count(state) DESC;
```

Display it as a pie chart:



#### 6.2.2.1 Percentage of restaurant numbers by state

The results are consistent with those for the states most frequently reviewed by users. The top two account for 48.86% of all the reviews, still led by Arizona with 29.91%, followed by

Nevada with 18.95%. Next in line with a high proportion are North Carolina, Ohio, Pennsylvania, etc., which also match up with the number of user reviews. For restaurants, being established in these states means potentially receiving more favor from users, but it also means facing more intense competition with rivals.

Next, calculate the proportion of restaurants with different star rating in each state.

```
-- Calculate statistics related to star ratings for businesses grouped by state.
SELECT
  state,
  COUNT(state) as count_state,
  SUM(CASE stars WHEN 5.0 THEN 1 ELSE 0 END) as sum_5,
  (SUM(CASE stars WHEN 5.0 THEN 1 ELSE 0 END) / COUNT(state)) as rate_5,
  SUM(CASE stars WHEN 4.5 THEN 1 ELSE 0 END) as sum_4_5,
  (SUM(CASE stars WHEN 4.5 THEN 1 ELSE 0 END) / COUNT(state)) as rate_4_5,
  SUM(CASE stars WHEN 4.0 THEN 1 ELSE 0 END) as sum_4,
  (SUM(CASE stars WHEN 4.0 THEN 1 ELSE 0 END) / COUNT(state)) as rate_4,
  SUM(CASE stars WHEN 3.5 THEN 1 ELSE 0 END) as sum_3_5,
  (SUM(CASE stars WHEN 3.5 THEN 1 ELSE 0 END) / COUNT(state)) as rate_3_5,
  SUM(CASE stars WHEN 3.0 THEN 1 ELSE 0 END) as sum_3,
  (SUM(CASE stars WHEN 3.0 THEN 1 ELSE 0 END) / COUNT(state)) as rate_3,
  SUM(CASE stars WHEN 2.5 THEN 1 ELSE 0 END) as sum_2_5,
  (SUM(CASE stars WHEN 2.5 THEN 1 ELSE 0 END) / COUNT(state)) as rate_2_5,
  SUM(CASE stars WHEN 2.0 THEN 1 ELSE 0 END) as sum_2,
  (SUM(CASE stars WHEN 2.0 THEN 1 ELSE 0 END) / COUNT(state)) as rate_2,
  SUM(CASE stars WHEN 1.5 THEN 1 ELSE 0 END) as sum_1_5,
  (SUM(CASE stars WHEN 1.5 THEN 1 ELSE 0 END) / COUNT(state)) as rate_1_5,
  SUM(CASE stars WHEN 1.0 THEN 1 ELSE 0 END) as sum_1,
  (SUM(CASE stars WHEN 1.0 THEN 1 ELSE 0 END) / COUNT(state)) as rate_1
FROM
  yelp_business
GROUP BY
  state
ORDER BY
  count_state DESC;
```

The results are shown below, listed in order are the number of restaurants in each state, the number of restaurants of each star rating, and their respective proportions.

	state	count_state	sum_5	rate_5	sum_4_5	rate_4_5	sum_4	rate_4	sum_3_5	rate_3_5	sum_3	rate_3	sum_2_5	rate_2_5	sum_2	rate_2	sum_1_5	rate_1_5	sum_1	rate_1
	varchar	bigint	newdecimal	newdecimal	newdecimal	newdecimal	newdecimal	newdecimal	newdecimal	newdecimal	newdecimal	newdecimal	newdecimal	newdecimal	newdecimal	newdecimal	newdecimal	newdecimal	newdecimal	newdecimal
1	AZ	52214	11698	0.2240	7441	0.1425	8928	0.1710	8207	0.1572	5995	0.1148	4781	0.0916	3204	0.0618	2122	0.0408	1387	0.0267
2	NV	33086	6631	0.2004	4924	0.1488	5978	0.1807	5426	0.1640	4040	0.1221	2819	0.0852	1921	0.0590	1298	0.0392	827	0.0250
3	ON	30208	2219	0.0735	3268	0.1082	5954	0.1971	6724	0.2226	5258	0.1741	3912	0.1290	2946	0.0912	2187	0.0659	1545	0.0481
4	NC	12956	1893	0.1461	1706	0.1317	2422	0.1869	2445	0.1887	1767	0.1364	1298	0.0998	954	0.0736	703	0.0543	545	0.0421
5	OH	12609	1578	0.1251	1676	0.1329	2473	0.1961	2474	0.1962	1711	0.1357	1298	0.0998	954	0.0736	703	0.0543	545	0.0421
6	PA	10109	1310	0.1296	1561	0.1544	1934	0.1913	2011	0.1989	1372	0.1357	989	0.0975	783	0.0772	594	0.0587	461	0.0454
7	QC	8169	542	0.0663	1579	0.1933	2128	0.2605	1687	0.2065	1064	0.1302	687	0.0841	477	0.0584	311	0.0381	234	0.0286
8	WI	4754	673	0.1416	699	0.1470	993	0.2089	923	0.1942	570	0.1199	404	0.0861	266	0.0561	177	0.0375	127	0.0270
9	EDH	3795	229	0.0603	745	0.1963	1230	0.3241	836	0.2203	435	0.1146	218	0.0576	117	0.0304	59	0.0156	29	0.0076
10	BW	3118	330	0.1058	654	0.2097	799	0.2563	614	0.1969	421	0.1350	271	0.0871	174	0.0558	117	0.0375	77	0.0247
11	IL	1852	222	0.1199	265	0.1431	313	0.1690	338	0.1825	279	0.1506	218	0.0841	177	0.0584	141	0.0543	112	0.0443
12	SC	679	132	0.1944	85	0.1252	115	0.1694	111	0.1635	103	0.1517	96	0.1473	89	0.1443	82	0.1340	75	0.1252

#### 6.2.2.1 Count of each star rating restaurants by state

This table contains a lot of information. From the table, we can see that no state has a particularly high proportion of low-starred restaurants. The proportions of restaurants of each star rating, except for five-star restaurants, fluctuate within a narrow range and are not related to the total number of restaurants in the state. Both Arizona and Nevada not only have a large number of restaurants, but also a higher proportion of five-star restaurants. This indicates that the selection of five-star restaurants is influenced by factors related to the state where the restaurant is located.

Arizona and Nevada are both located in the southwestern corner of the United States, are relatively large in area, and are close to California and Hawaii, while Yelp's headquarters is located in California. The coastal positions of these two states, influenced by the flow of people from neighboring California and with the advantage of being closer to Yelp's headquarters, may

also have prompted the platform to focus on developing its business in these two states, thus having more restaurants registered on the platform and a higher number of five-star restaurants.

### **(3) What is the relationship between the star rating of a restaurant and the number of days it is open for business?**

First, let's generate the number of days of operations. Begin by calculating the number of operating days each week for restaurants:

```
with open_days as
(SELECT
  business_id,
  (
    (CASE WHEN monday = 'None' THEN 0 ELSE 1 END) +
    (CASE WHEN tuesday = 'None' THEN 0 ELSE 1 END) +
    (CASE WHEN wednesday = 'None' THEN 0 ELSE 1 END) +
    (CASE WHEN thursday = 'None' THEN 0 ELSE 1 END) +
    (CASE WHEN friday = 'None' THEN 0 ELSE 1 END) +
    (CASE WHEN saturday = 'None' THEN 0 ELSE 1 END) +
    (CASE WHEN sunday = 'None' THEN 0 ELSE 1 END)
  ) AS open_days
FROM
  yelp_business_hours)
select * from open_days;
```

I built a CTE called open\_days. The result is below:



business_id varchar	open_days bigint
__1uG7MLxWGFiv2fCGPiQQ	6
__3l-DDkqM9XjLH1cJl3VA	7
__3qOwWFBUE8mdOTol7YrQ	0
__47_7H-yK3HChO5vyut_Q	0
__6jYJ6Hm-Qq8XQEGDrOGQ	0
__8j8yhsmE98wNWHJNyAgw	7
__aKnGBedQ51_hEc3D9ARw	7
__bIIPRrsfEoaioSPj1oIQ	0
__bqGGnOjtY9eEhrZAUsG	7
__briK1e5l1BCEwGYijhFg	3
__CQ2SE4NXFFjYfrB_TJ6w	6
__D6AVR_hLpW_bott0-upA	6
__eb2f_wEBrEl0xCyLqDeQ	0
__FFoyg0XmJluBBNE0QP0w	6
__fMLrmv9M1_W4kBvR2VnQ	7
__fyRzU8kL6HkVV3wgxfmQ	7
__G0Ug3CK2yCDdQLYpd0wv	7

### 6.2.3.1 Restaurant operating days

Next, use ‘open\_days’ to analyze the relationship between a restaurant’s star rating and its operating days:

```
-- Calculate the count of businesses with different star ratings
-- group by the number of open days.
with open_days as
(SELECT
  business_id,
  (
    (CASE WHEN monday = 'None' THEN 0 ELSE 1 END) +
    (CASE WHEN tuesday = 'None' THEN 0 ELSE 1 END) +
    (CASE WHEN wednesday = 'None' THEN 0 ELSE 1 END) +
```

```

        (CASE WHEN thursday = 'None' THEN 0 ELSE 1 END) +
        (CASE WHEN friday = 'None' THEN 0 ELSE 1 END) +
        (CASE WHEN saturday = 'None' THEN 0 ELSE 1 END) +
        (CASE WHEN sunday = 'None' THEN 0 ELSE 1 END)
    ) AS open_days
FROM
    yelp_business_hours)
SELECT
    open_days.open_days,
    SUM(CASE yelp_business.stars WHEN 5.0 THEN 1 ELSE 0 END) as sum_5,
    SUM(CASE yelp_business.stars WHEN 4.5 THEN 1 ELSE 0 END) as sum_4_5,
    SUM(CASE yelp_business.stars WHEN 4.0 THEN 1 ELSE 0 END) as sum_4,
    SUM(CASE yelp_business.stars WHEN 3.5 THEN 1 ELSE 0 END) as sum_3_5,
    SUM(CASE yelp_business.stars WHEN 3.0 THEN 1 ELSE 0 END) as sum_3,
    SUM(CASE yelp_business.stars WHEN 2.5 THEN 1 ELSE 0 END) as sum_2_5,
    SUM(CASE yelp_business.stars WHEN 2.0 THEN 1 ELSE 0 END) as sum_2,
    SUM(CASE yelp_business.stars WHEN 1.5 THEN 1 ELSE 0 END) as sum_1_5,
    SUM(CASE yelp_business.stars WHEN 1.0 THEN 1 ELSE 0 END) as sum_1
FROM
    yelp_business
LEFT JOIN
    open_days
ON
    yelp_business.business_id = open_days.business_id
GROUP BY
    open_days.open_days;

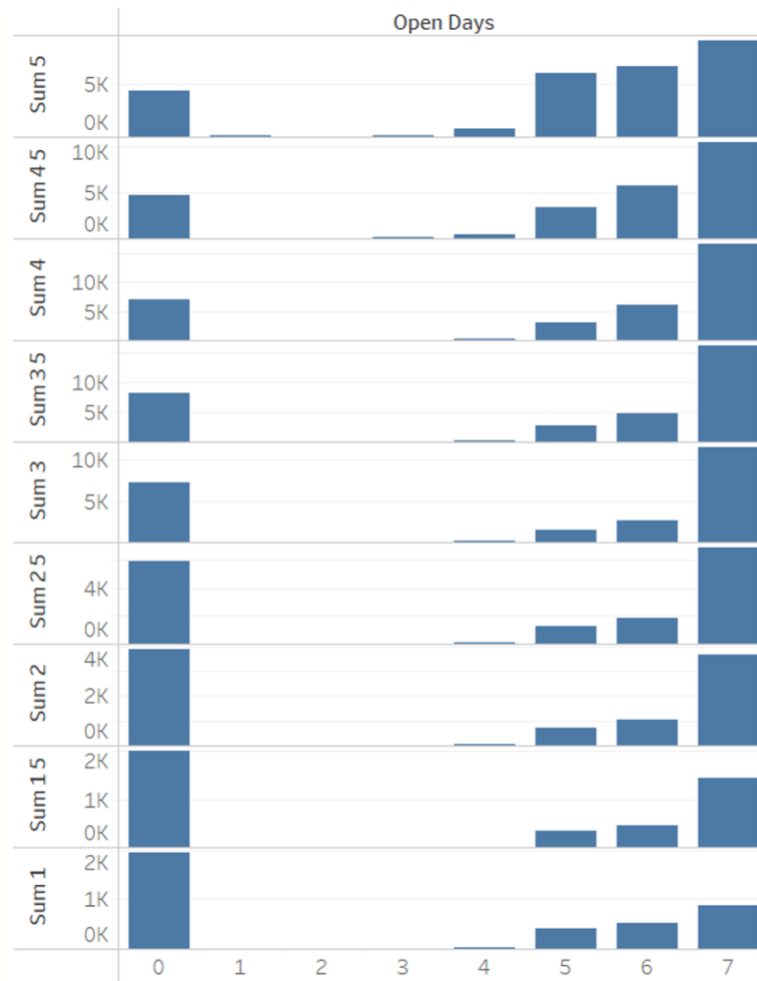
```

The results are as follows, with each row representing the number of operating days per week and the corresponding numbers of restaurants at each star rating.

open_days int	sum_5 newdecimal	sum_4_5 newdecimal	sum_4 newdecimal	sum_3_5 newdecimal	sum_3 newdecimal	sum_2_5 newdecimal	sum_2 newdecimal	sum_1_5 newdecimal	sum_1 newdecimal
7	9220	10456	16667	16165	11577	6964	3636	1450	858
6	6676	5709	6139	4671	2652	1839	1023	460	529
5	6108	3338	3143	2640	1418	1272	691	343	414
4	801	392	303	207	137	70	42	13	19
3	193	90	80	85	53	38	15	9	8
2	64	54	59	61	37	25	15	4	6
1	73	69	62	47	34	13	10	4	10
0	4405	4688	7039	8162	7234	5927	3888	2020	1944

6.2.3.2: The number of operating days per week and the corresponding number of restaurants for each star rating

Use the number of operating days per week as the x axis and the number of restaurants as the y axis to sequentially display the distribution of operating days for restaurants of each star ratings:



6.2.3.3: The number of operating days per week and the corresponding number of restaurants for each star rating

We can disregard the restaurant with 0 days operation. It means that there is no data. Observing the above graph reveals that it is common for five-star restaurants to operate 5, 6, or 7 days a week. Conversely, in restaurants rated between 3 to 5 stars, the proportion of restaurants operating 5 or 6 days gradually decreases with the reduction in star rating, while the majority operate 7 days a week. This indicates that a restaurant's star rating is not solely accumulated through operating hours. Ensuring a certain number of operating days (at least 5 days) is important, but other factors of the restaurant are more crucial. Observing 1 to 2-star restaurants, it is found that the proportion of restaurants operating 5 or 6 days increases compared to those

rated 3 to 4 stars. This suggests that for average restaurants, operating every day can to some extent guarantee the restaurant's reputation if it is not possible to improve the facilities and other factors.

**(4) What are the characteristics of the operation days and star ratings of the restaurants that have closed?**

Calculate the average number of operating days per week and the average star rating for restaurants that have closed:

```
with open_days as
(SELECT
  business_id,
  (
    (CASE WHEN monday = 'None' THEN 0 ELSE 1 END) +
    (CASE WHEN tuesday = 'None' THEN 0 ELSE 1 END) +
    (CASE WHEN wednesday = 'None' THEN 0 ELSE 1 END) +
    (CASE WHEN thursday = 'None' THEN 0 ELSE 1 END) +
    (CASE WHEN friday = 'None' THEN 0 ELSE 1 END) +
    (CASE WHEN saturday = 'None' THEN 0 ELSE 1 END) +
    (CASE WHEN sunday = 'None' THEN 0 ELSE 1 END)
  ) AS open_days
FROM
  yelp_business_hours)
SELECT
  ROUND(AVG(open_days.open_days), 1) as avg_open,
  ROUND(AVG(yelp_business.stars), 2) as avg_stars
FROM
  yelp_business
LEFT JOIN
  open_days
ON
  yelp_business.business_id = open_days.business_id
WHERE
  yelp_business.is_open = 0;
```

The result is below:

avg_open newdecimal	avg_stars newdecimal
3.9	3.51

On average, they operate about 4 days per week, with the average star rating of the restaurants being about 3.5. Compared with the data for all restaurants, both figures are slightly lower, indicating that the closure of a restaurant is mildly associated with the number of operating days and the average star rating, with the three factors influencing each other.

### 3. Platform perspective

#### (1) How many users registered each year and each month? How many user reviews?

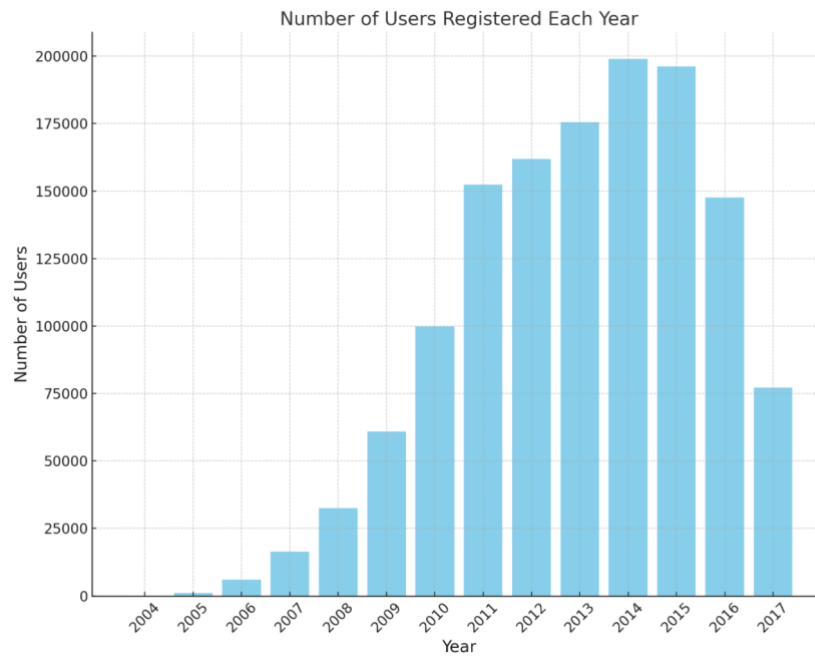
Calculate the number of users registered in each year:

```
-- Calculate the number of users registered in each year
SELECT
  YEAR(yelping_since) AS year_since,
  COUNT(user_id) AS user_count
FROM
  yelp_user
GROUP BY
  YEAR(yelping_since)
ORDER BY
  year_since ASC;
```

year_since int	user_count bigint
2004	75
2005	979
2006	5951
2007	16364
2008	32433
2009	60905
2010	99785
2011	152353
2012	161880
2013	175483
2014	198976
2015	196149
2016	147593
2017	77174

#### 6.3.1.1 Registration of new user per year

The bar plot is shown below:



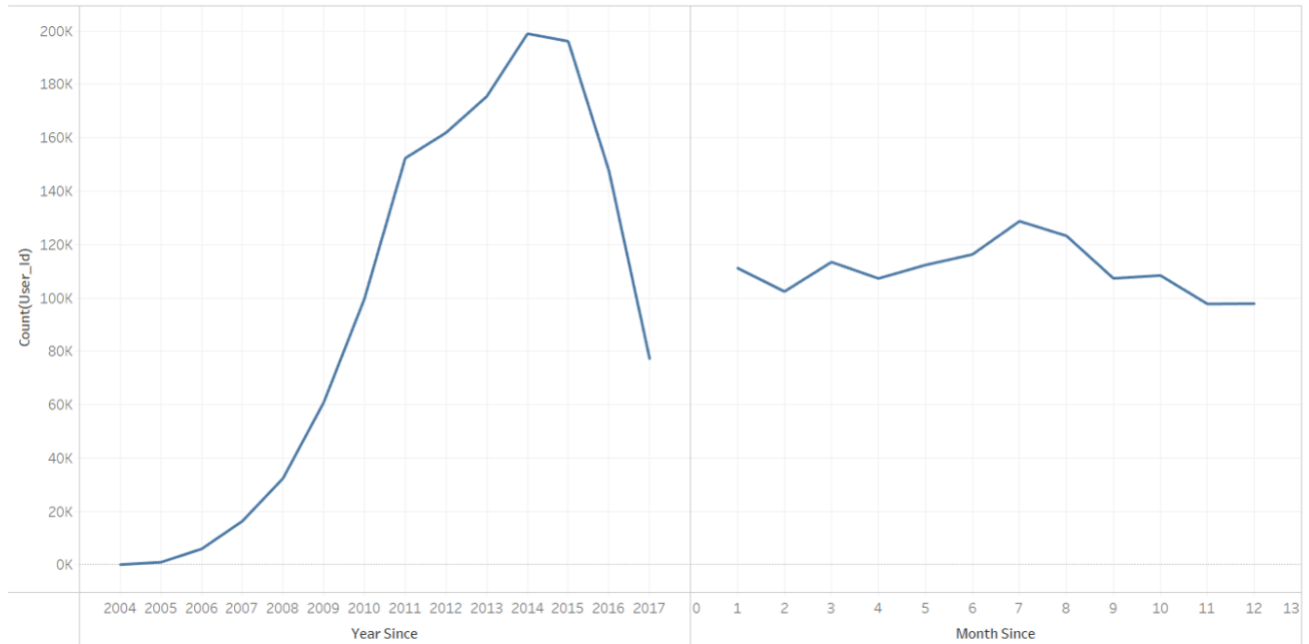
#### 6.3.1.1 Registration of new user per year

It is evident that the number of user registrations was on an upward trend until 2014, after which it declined. The year with the most significant growth in platform user numbers was 2014, with a total of 198,976 new users, and the number of new users in 2015 was almost the same, reaching 196,149 people. The decline after 2015 is quite noticeable. Although the statistical data for 2017 only goes up to December 11th, as shown in the first subsection of this chapter, it is unlikely that there would be a surge in new users in the remaining days of the year before its end, so it can be inferred that the number of registrations for the entire year will still show a downward trend.

Calculate the number of users registered in each month of each year:

```
-- Retrieve the count of users who joined Yelp in each year and month
SELECT
    YEAR(yelping_since) as year_since,
    MONTH(yelping_since) as month_since,
    COUNT(user_id)
FROM
    yelp_user
GROUP BY
    YEAR(yelping_since),
    MONTH(yelping_since);
```





#### 6.3.1.2 Trend chart of the number of users each year and each month

For the growth of new users each month, it can be seen that there is a peak period around July, which may be influenced by the hot summer weather and a dense period of vacations. During this time, people's desire and frequency to dine in a restaurant increases, which in turn raises the demand and usage intensity for the app.

Calculate the total number of reviews posted by users on the platform each year:

```
-- Calculate the total number of reviews posted by users on the platform each YEAR
SELECT YEAR(date) as year_review, count(review_id)
FROM yelp_review
GROUP BY YEAR(date)
ORDER BY YEAR(date) ASC;
```

The result is shown below:

year_review int	count(review_id) bigint
2004	14
2005	870
2006	5669
2007	23020
2008	61553
2009	98288
2010	187073
2011	290933
2012	350381
2013	472595
2014	678351
2015	911487
2016	1052916
2017	1128518

6.3.1.3 Total number of reviews posted by users on the platform each year



#### 6.3.1.4 Total number of reviews posted by users on the platform each year

The number of reviews users post on the platform is gradually increasing, and the rate of growth has not diminished over the years. The usage of the platform by users is quite favorable.

#### (2) How many elite customers are there each year, and what is their percentage?

First, filter out a list of users who have been selected as elite users for at least one year:

```
-- users who have been selected as elite users for at least one year:  
SELECT user_id, elite  
FROM yelp_user  
WHERE elite <> 'None'
```

The result is shown below:

user_id varchar(100)	elite varchar(100)
__-f____fto1Q	2016, 2017
__-jTbcqlU4pwDy4BZ9JIQ	2012
__05rytNjsye9MBhqB0DMA	2011, 2012, 2017, 2009, 20
__2Xu2F0Z1gAodYpIdOsCQ	2014, 2011, 2013, 2010, 20
__3Lm1VjoOK5WHL2tt4Z6w	2015
__48dJJcPvNgqUIEozwtpw	2015, 2016, 2014, 2017
__8GaZVjxvWHm1Hip5dH8w	2014, 2016, 2015
__Bk3fmzhWYAY4MRrml65Q	2014, 2013
__bMs0nf3_hnhitK91gT4A	2016, 2017
__d3AvbtFvdKFPEgQIYQaQ	2007, 2009, 2010
__Ksf1CCgaKdS3Gi5f6PcQ	2012, 2013
__L7M05VxbYZBhRAQNABJw	2015, 2016, 2017
__LHU2ohslRR7_9uu00rYg	2014, 2016, 2017, 2015

Table 6.3.2.1 Elite users

The result is shown below:

2004_elite newdecimal	2005_elite newdecimal	2006_elite newdecimal	2007_elite newdecimal	2008_elite newdecimal	2009_elite newdecimal	2010_elite newdecimal	2011_elite newdecimal	2012_elite newdecimal
0	140	887	2363	3621	6536	10485	13185	17777

2013_elite newdecimal	2014_elite newdecimal	2015_elite newdecimal	2016_elite newdecimal	2017_elite newdecimal
19841	20488	26018	30856	34928

The select of users began in 2005. The number of people selected as elite users has been increasing year by year, but the rate of increase has not changed significantly since 2008,

fluctuating between 2000 and 6000, mostly around 4000, which is much less than the rate of change for new users.

**(3) What is the annual retention rate of users? What about the annual retention rate of elite users?**

For all users, use the year of the review as the basis for retention. Calculate the user IDs and the number of reviews each year.

```
-- Calculate the count of reviews posted by each user in each year from 2008 to 2017.
SELECT
  user_id,
  SUM(CASE WHEN YEAR(yelp_review.date) = 2008 THEN 1 ELSE 0 END) as 2008_exist,
  SUM(CASE WHEN YEAR(yelp_review.date) = 2009 THEN 1 ELSE 0 END) as 2009_exist,
  SUM(CASE WHEN YEAR(yelp_review.date) = 2010 THEN 1 ELSE 0 END) as 2010_exist,
  SUM(CASE WHEN YEAR(yelp_review.date) = 2011 THEN 1 ELSE 0 END) as 2011_exist,
  SUM(CASE WHEN YEAR(yelp_review.date) = 2012 THEN 1 ELSE 0 END) as 2012_exist,
  SUM(CASE WHEN YEAR(yelp_review.date) = 2013 THEN 1 ELSE 0 END) as 2013_exist,
  SUM(CASE WHEN YEAR(yelp_review.date) = 2014 THEN 1 ELSE 0 END) as 2014_exist,
  SUM(CASE WHEN YEAR(yelp_review.date) = 2015 THEN 1 ELSE 0 END) as 2015_exist,
  SUM(CASE WHEN YEAR(yelp_review.date) = 2016 THEN 1 ELSE 0 END) as 2016_exist,
  SUM(CASE WHEN YEAR(yelp_review.date) = 2017 THEN 1 ELSE 0 END) as 2017_exist
FROM
  yelp_review
GROUP BY
  user_id;
```

The result is shown below, it will be stored as CTE.

user_id varchar(100)	2008_exist newdecimal	2009_exist newdecimal	2010_exist newdecimal	2011_exist newdecimal	2012_exist newdecimal	2013_exist newdecimal	2014_exist newdecimal	2015_exist newdecimal	2016_exist newdecimal
___DPmKJsBF2X6ZKgAeGqg	0	0	0	0	0	0	2	1	0
___fEWlOijtPaZ-pK0eq9g	0	0	0	0	0	0	0	0	1
___l9ZYdYGkZ6dMYxwJEIQ	0	0	0	0	4	0	0	0	0
___MTsBloH4jvybJ5DrTYw	0	0	0	0	0	0	1	0	0
___QCazm0YrHLd3uNUPYMA	0	0	0	0	0	0	2	2	0
___-8WEg7xD0CwCDu04MzB/	0	0	0	0	1	0	3	3	4
___-C_dSG8Ky98M6EVPYehA	0	0	0	0	0	1	0	0	0
___-FcRiBzu8tqWYGofTo1Q	0	0	0	0	0	0	0	0	1
___-FvQQtlLqqe5s4ClEzfA	0	0	0	0	0	0	0	1	0
___-jTbcqlU4pwDy4BZ9JIQ	0	0	0	1	0	0	0	0	0

Calculate the proportion of users who posted reviews in each year and continued to post reviews in the following year, as an annual retention rate.

```
-- Calculate the proportion of users who posted reviews in consecutive years for each pair of years from 2008 to 2017.
with user_keep as(
SELECT
    user_id,
    SUM(CASE WHEN YEAR(yelp_review.date) = 2008 THEN 1 ELSE 0 END) as 2008_exist,
    SUM(CASE WHEN YEAR(yelp_review.date) = 2009 THEN 1 ELSE 0 END) as 2009_exist,
    SUM(CASE WHEN YEAR(yelp_review.date) = 2010 THEN 1 ELSE 0 END) as 2010_exist,
    SUM(CASE WHEN YEAR(yelp_review.date) = 2011 THEN 1 ELSE 0 END) as 2011_exist,
    SUM(CASE WHEN YEAR(yelp_review.date) = 2012 THEN 1 ELSE 0 END) as 2012_exist,
    SUM(CASE WHEN YEAR(yelp_review.date) = 2013 THEN 1 ELSE 0 END) as 2013_exist,
    SUM(CASE WHEN YEAR(yelp_review.date) = 2014 THEN 1 ELSE 0 END) as 2014_exist,
    SUM(CASE WHEN YEAR(yelp_review.date) = 2015 THEN 1 ELSE 0 END) as 2015_exist,
    SUM(CASE WHEN YEAR(yelp_review.date) = 2016 THEN 1 ELSE 0 END) as 2016_exist,
    SUM(CASE WHEN YEAR(yelp_review.date) = 2017 THEN 1 ELSE 0 END) as 2017_exist
FROM
    yelp_review
GROUP BY
    user_id
)
SELECT
    ROUND(SUM(CASE WHEN (2008_exist <> 0 AND 2009_exist <> 0) THEN 1 ELSE 0 END) / SUM(CASE WHEN
2008_exist <> 0 THEN 1 ELSE 0 END), 2) AS exist_2009,
```

```

ROUND(SUM(CASE WHEN (2009_exist <> 0 AND 2010_exist <> 0) THEN 1 ELSE 0 END) / SUM(CASE WHEN
2009_exist <> 0 THEN 1 ELSE 0 END), 2) AS exist_2010,
ROUND(SUM(CASE WHEN (2010_exist <> 0 AND 2011_exist <> 0) THEN 1 ELSE 0 END) / SUM(CASE WHEN
2010_exist <> 0 THEN 1 ELSE 0 END), 2) AS exist_2011,
ROUND(SUM(CASE WHEN (2011_exist <> 0 AND 2012_exist <> 0) THEN 1 ELSE 0 END) / SUM(CASE WHEN
2011_exist <> 0 THEN 1 ELSE 0 END), 2) AS exist_2012,
ROUND(SUM(CASE WHEN (2012_exist <> 0 AND 2013_exist <> 0) THEN 1 ELSE 0 END) / SUM(CASE WHEN
2012_exist <> 0 THEN 1 ELSE 0 END), 2) AS exist_2013,
ROUND(SUM(CASE WHEN (2013_exist <> 0 AND 2014_exist <> 0) THEN 1 ELSE 0 END) / SUM(CASE WHEN
2013_exist <> 0 THEN 1 ELSE 0 END), 2) AS exist_2014,
ROUND(SUM(CASE WHEN (2014_exist <> 0 AND 2015_exist <> 0) THEN 1 ELSE 0 END) / SUM(CASE WHEN
2014_exist <> 0 THEN 1 ELSE 0 END), 2) AS exist_2015,
ROUND(SUM(CASE WHEN (2015_exist <> 0 AND 2016_exist <> 0) THEN 1 ELSE 0 END) / SUM(CASE WHEN
2015_exist <> 0 THEN 1 ELSE 0 END), 2) AS exist_2016,
ROUND(SUM(CASE WHEN (2016_exist <> 0 AND 2017_exist <> 0) THEN 1 ELSE 0 END) / SUM(CASE WHEN
2016_exist <> 0 THEN 1 ELSE 0 END), 2) AS exist_2017
FROM
user_keep;

```

The result is shown below:

exist_2009 newdecimal	exist_2010 newdecimal	exist_2011 newdecimal	exist_2012 newdecimal	exist_2013 newdecimal	exist_2014 newdecimal	exist_2015 newdecimal	exist_2016 newdecimal	exist_2017 newdecimal
0.30	0.31	0.31	0.28	0.30	0.33	0.33	0.32	0.32

#### 6.3.3.1: Annual retention rates of all users from 2009 to 2017.

As can be seen from the graph, the annual retention rate of ordinary users has remained at around 30% over the past decade.

To calculate the annual retention rate for elite users, it is determined by whether the elite users from the previous year still retain their elite status in the following year. The code for this calculation is as follows:

```

-- Calculate the proportion of users who had elite status in consecutive years for each pair of years from 2005 to
2017.

```

```

with elite as(
    SELECT user_id, elite
FROM yelp_user
WHERE elite <> 'None'
)
SELECT
    ROUND(SUM(CASE WHEN (elite LIKE '%2005%' AND elite LIKE '%2006%') THEN 1 ELSE 0 END) / SUM(CASE
WHEN elite LIKE '%2005%' THEN 1 ELSE 0 END), 2) AS exist_2006,
    ROUND(SUM(CASE WHEN (elite LIKE '%2006%' AND elite LIKE '%2007%') THEN 1 ELSE 0 END) / SUM(CASE
WHEN elite LIKE '%2006%' THEN 1 ELSE 0 END), 2) AS exist_2007,
    ROUND(SUM(CASE WHEN (elite LIKE '%2007%' AND elite LIKE '%2008%') THEN 1 ELSE 0 END) / SUM(CASE
WHEN elite LIKE '%2007%' THEN 1 ELSE 0 END), 2) AS exist_2008,
    ROUND(SUM(CASE WHEN (elite LIKE '%2008%' AND elite LIKE '%2009%') THEN 1 ELSE 0 END) / SUM(CASE
WHEN elite LIKE '%2008%' THEN 1 ELSE 0 END), 2) AS exist_2009,
    ROUND(SUM(CASE WHEN (elite LIKE '%2009%' AND elite LIKE '%2010%') THEN 1 ELSE 0 END) / SUM(CASE
WHEN elite LIKE '%2009%' THEN 1 ELSE 0 END), 2) AS exist_2010,
    ROUND(SUM(CASE WHEN (elite LIKE '%2010%' AND elite LIKE '%2011%') THEN 1 ELSE 0 END) / SUM(CASE
WHEN elite LIKE '%2010%' THEN 1 ELSE 0 END), 2) AS exist_2011,
    ROUND(SUM(CASE WHEN (elite LIKE '%2011%' AND elite LIKE '%2012%') THEN 1 ELSE 0 END) / SUM(CASE
WHEN elite LIKE '%2011%' THEN 1 ELSE 0 END), 2) AS exist_2012,
    ROUND(SUM(CASE WHEN (elite LIKE '%2012%' AND elite LIKE '%2013%') THEN 1 ELSE 0 END) / SUM(CASE
WHEN elite LIKE '%2012%' THEN 1 ELSE 0 END), 2) AS exist_2013,
    ROUND(SUM(CASE WHEN (elite LIKE '%2013%' AND elite LIKE '%2014%') THEN 1 ELSE 0 END) / SUM(CASE
WHEN elite LIKE '%2013%' THEN 1 ELSE 0 END), 2) AS exist_2014,
    ROUND(SUM(CASE WHEN (elite LIKE '%2014%' AND elite LIKE '%2015%') THEN 1 ELSE 0 END) / SUM(CASE
WHEN elite LIKE '%2014%' THEN 1 ELSE 0 END), 2) AS exist_2015,
    ROUND(SUM(CASE WHEN (elite LIKE '%2015%' AND elite LIKE '%2016%') THEN 1 ELSE 0 END) / SUM(CASE
WHEN elite LIKE '%2015%' THEN 1 ELSE 0 END), 2) AS exist_2016,
    ROUND(SUM(CASE WHEN (elite LIKE '%2016%' AND elite LIKE '%2017%') THEN 1 ELSE 0 END) / SUM(CASE
WHEN elite LIKE '%2016%' THEN 1 ELSE 0 END), 2) AS exist_2017
FROM
    elite;

```

The result is shown below:



exist_2006 newdecimal	exist_2007 newdecimal	exist_2008 newdecimal	exist_2009 newdecimal	exist_2010 newdecimal	exist_2011 newdecimal	exist_2012 newdecimal	exist_2013 newdecimal	exist_2014 newdecimal	exist_2015 newdecimal	exist_2016 newdecimal	exist_2017 newdecimal
0.94	0.89	0.72	0.82	0.83	0.80	0.80	0.85	0.77	0.82	0.84	0.82

Since the year following the platform's initiation of selecting elite users, which is 2006, up to the 2017, the annual retention rate for elite users has consistently remained around 80%, achieving a high standard and significantly surpassing the annual retention rate of all users.

## 7. Conclusion

### 1. Customer perspective

For users, the most important aspect might be how to quickly become an elite user, in order to enjoy more promotion and additional benefits brought by the platform's traffic. From the analysis, users first need to post many reviews and photos, especially reviews, as the quantity is a relatively important indicator. For each review, users need to not only focus on posting longer comments but also on enhancing the content of the comments. Rather than pondering over the style of language, it's more important to provide information valuable to the public. Especially if one can pinpoint issues with highly rated restaurants, their comments are more likely to attract attention from other users. Being based in states like Nevada and Arizona could also increase the chances of being rated as an elite user, as these states have a higher concentration of businesses with significant foot traffic.

For non-elite users who do not care about becoming elite users, choosing delicious restaurants that meet their needs is their ultimate goal in using Yelp. Overall, the star ratings of restaurants and user ratings roughly align and can to some extent reflect the quality of restaurants, allowing users to make selections based on their needs. It's worth noting that five-

star restaurants may have more "padding" in their ratings, necessitating discernment based on user feedback.

## **2. Restaurant perspective**

From the perspective of businesses looking to attract more customers, opening branches in larger states such as Nevada, Arizona, North Carolina, Ohio, and Pennsylvania could be considered. If aiming to upgrade to a five-star establishment, special focus should be placed on Arizona and Nevada. Beyond the basic guarantees of foot traffic and operating days, five-star restaurants also place greater emphasis on other conditions that could be targeted for improvement; simultaneously, it's very important to lower the rate of negative reviews. For restaurants that don't wish to receive low stars and are aiming for higher ratings, the choice of location isn't as critical, but it's advisable to have more operating days per week, and attention must be paid to ensuring the rate of positive reviews isn't too low.

## **3. Platform perspective**

The platform still has a considerable attraction to existing users, who are using it satisfactorily. However, the registration of new users has been declining year by year. Platform might need to think out some measures to slower the decline, such as collaborating with restaurants to promote the app during the summer when people prefer to dine out, or through promotional discounts. At the same time, the yearly decline in registration numbers may also signal that the current market is becoming saturated. Expanding the business to more states, or even more countries, could help take the platform to the next level.

For old users, especially those non-elite users who are still active, the platform could tailor periodic activities that encourage users to increase platform activity for rewards, or offer benefits to returning users, based on each user's registration timeline and activity patterns.

Additionally, further enhancing the benefits for elite users could attract more non-elite users to reach the standards of elite users.



