Reference: https://towardsdatascience.com/a-beginners-guide-to-sentiment-analysis-in-python-95e354ea84f6

In [9]:

```
#Importing packages for data analysis
import pandas as pd
import nltk
from wordcloud import WordCloud, STOPWORDS
import matplotlib.pyplot as plt
import plotly.express as px
import warnings; warnings.simplefilter('ignore')
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix,classification_report
```

In [10]:

```
#Importing Disneyland reviews csv file
Disney_Reviews = pd.read_csv('Disneylandreviews.csv')
Disney_Reviews
```

Out[10]:

| | Review_ID | Rating | Year_Month | Reviewer_Location | Review_Text | Branch |
|---|---|---|---|---|---|---|
| **0** | 670772142 | 4 | 2019-4 | Australia | If you've ever been to Disneyland anywhere you... | Disneyland_HongKong |
| **1** | 670682799 | 4 | 2019-5 | Philippines | Its been a while since d last time we visit HK... | Disneyland_HongKong |
| **2** | 670623270 | 4 | 2019-4 | United Arab Emirates | Thanks God it wasn t too hot or too humid wh... | Disneyland_HongKong |
| **3** | 670607911 | 4 | 2019-4 | Australia | HK Disneyland is a great compact park. Unfortu... | Disneyland_HongKong |
| **4** | 670607296 | 4 | 2019-4 | United Kingdom | the location is not in the city, took around 1... | Disneyland_HongKong |
| **...** | ... | ... | ... | ... | ... | ... |
| **42651** | 1765031 | 5 | missing | United Kingdom | i went to disneyland paris in july 03 and thou... | Disneyland_Paris |
| **42652** | 1659553 | 5 | missing | Canada | 2 adults and 1 child of 11 visited Disneyland ... | Disneyland_Paris |
| **42653** | 1645894 | 5 | missing | South Africa | My eleven year old daughter and myself went to... | Disneyland_Paris |
| **42654** | 1618637 | 4 | missing | United States | This hotel, part of the Disneyland Paris compl... | Disneyland_Paris |

| | Review_ID | Rating | Year_Month | Reviewer_Location | Review_Text | Branch |
|---|---|---|---|---|---|---|
| **42655** | 1536786 | 4 | missing | United Kingdom | I went to the Disneyparis resort, in 1996, wit... | Disneyland_Paris |

42656 rows × 6 columns

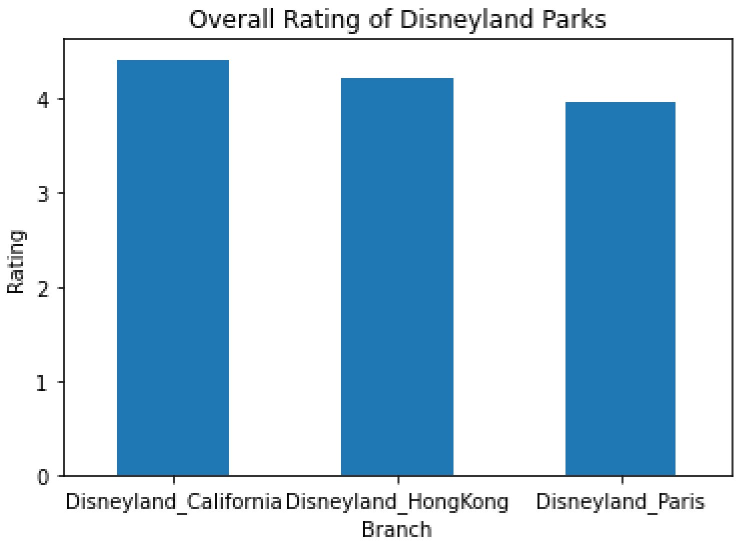
Which Disneyland park has the best reviews?

```
In [11]: #Creating dataframe of average ratings per branch
         Mean_Ratings = Disney_Reviews.groupby('Branch')['Rating'].mean()
         Mean_Ratings
```

```
Out[11]: Branch
         Disneyland_California    4.405339
         Disneyland_HongKong      4.204158
         Disneyland_Paris         3.960088
         Name: Rating, dtype: float64
```

```
In [12]: #Plotting average ratings per branch
         ax = Mean_Ratings.plot.bar(x='Branch', y='Mean', rot=0)
         ax.title.set_text('Overall Rating of Disneyland Parks')
         ax.set_ylabel('Rating')
```

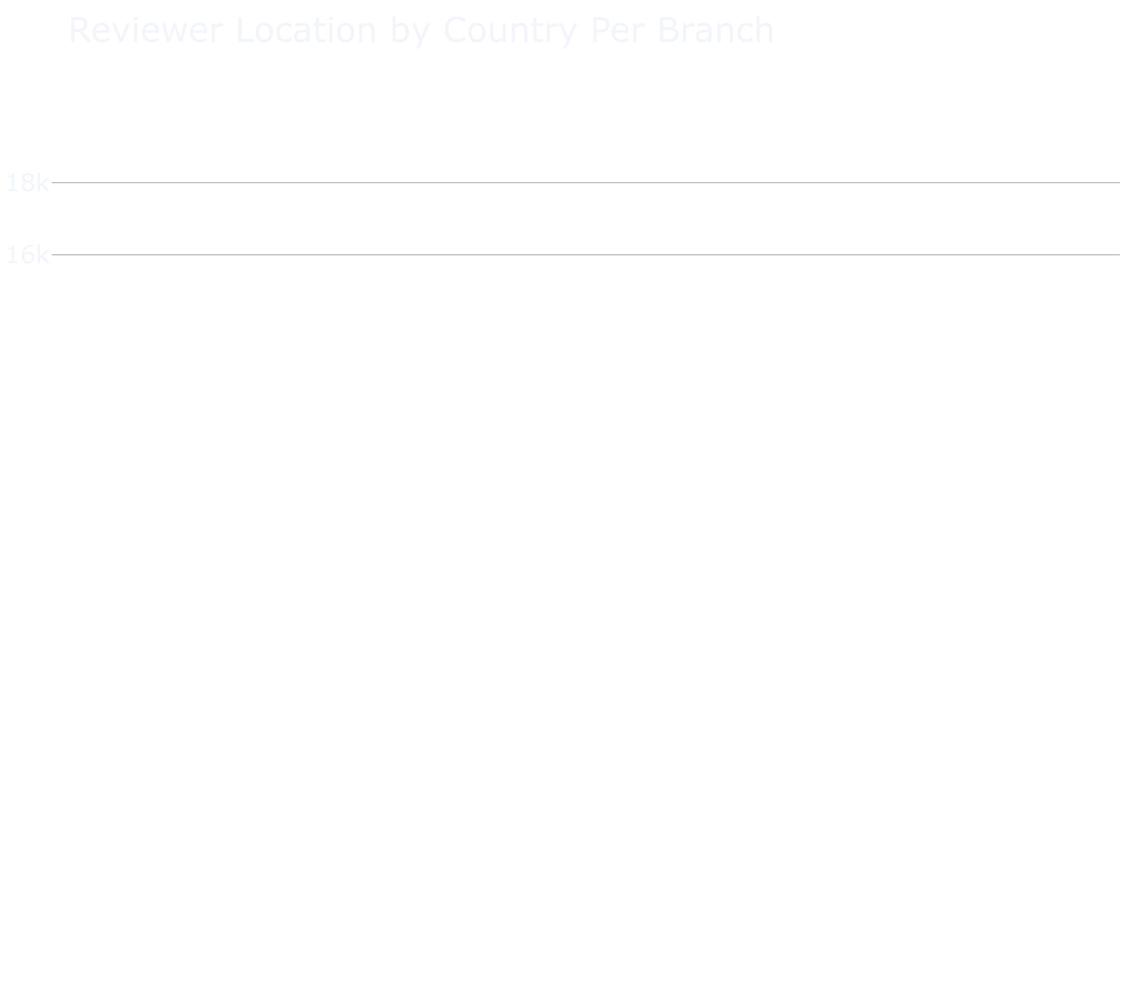
```
Out[12]: Text(0, 0.5, 'Rating')
```



Which Country has left the most reviews for Disneyland Locations?

```
In [13]: #Creating new dataframe to get n largest reviewer countries by branch
         Location_Counts = Disney_Reviews.groupby(['Branch']).Reviewer_Location.value_counts().n
         Locations = pd.DataFrame(Location_Counts)
         Locations = Locations.rename(columns={"Reviewer_Location": "Location_Counts"})
         Locations = Locations.reset_index()
         Locations = Locations.sort_values('Location_Counts')
```

```
In [14]: #Plotting reviewer location counts by branch using n largest dataframe
         fig = px.bar(Locations, x='Branch', y="Location_Counts", template = 'plotly_dark',
                      color = 'Reviewer_Location', title="Reviewer Location by Country Per Bran
         fig.show()
```

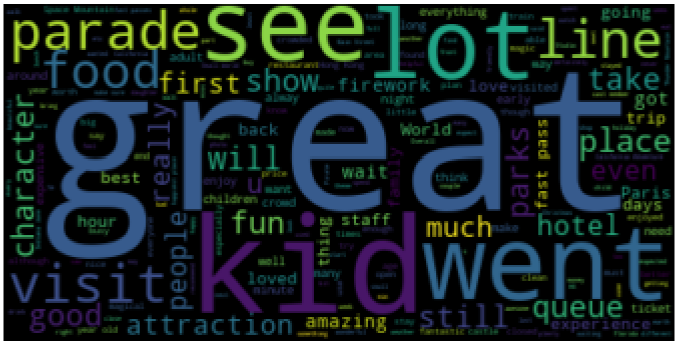


What is the most common positive feedback left in the reviews?

In [15]:
```
#Creating new dataframe for sentiment analysis, deleting ratings of 3 as neutral and as
#and ratings <3 as negative (-1)
Disney_Reviews = Disney_Reviews[Disney_Reviews['Rating'] != 3]
Disney_Reviews['sentiment'] = Disney_Reviews['Rating'].apply(lambda rating : +1 if rati
```

In [18]:
```
#Creating a positive and negative dataframe based on setiment to create word clouds
positive = Disney_Reviews[Disney_Reviews['sentiment'] == 1]
negative = Disney_Reviews[Disney_Reviews['sentiment'] == -1]
```

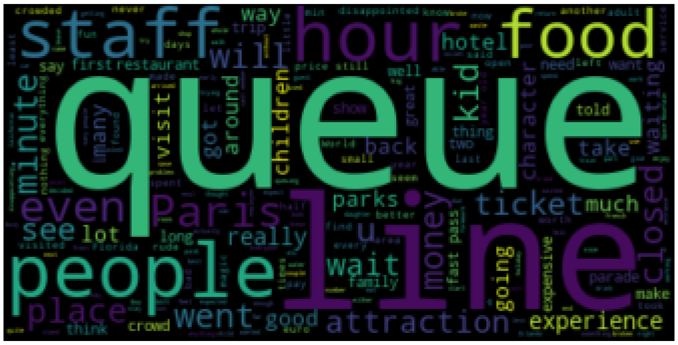
In [19]:
```
#Creating positive word cloud using stop words which will be removed from the analysis
stopwords = set(STOPWORDS)
stopwords.update(["park", "ride", "Disneyland", "Disney", "rides", "time", "go", "day",
pos = " ".join(Review for Review in positive.Review_Text) #pulling together all the wor
wordcloud2 = WordCloud(stopwords=stopwords).generate(pos) #generating word cloud
plt.title('Positive Review Word Cloud')
plt.imshow(wordcloud2, interpolation='bilinear')
plt.axis("off")
plt.show()
```


Positive Review Word Cloud

What is the most common negative feedback left in the reviews?

In [20]:

```python
neg = " ".join(Review for Review in negative.Review_Text) #pulling together all the wor
wordcloud3 = WordCloud(stopwords=stopwords).generate(neg) #generating word cloud
plt.title('Negative Review Word Cloud')
plt.imshow(wordcloud3, interpolation='bilinear')
plt.axis("off")
plt.show()
```


Negative Review Word Cloud

Can we predict the review rating based on reviewer feedback?

In [21]:

```python
#removing puncuation from text
def remove_punctuation(text):
    final = "".join(u for u in text if u not in ("?", ".", ";", ":",  "!",'"'))
    return final
Disney_Reviews['Review_Text'] = Disney_Reviews['Review_Text'].apply(remove_punctuation)
Disney_Reviews = Disney_Reviews.dropna(subset=['Review_Text']) #Removing missing review
Disney_Reviews
```

Out[21]:

|   | Review_ID | Rating | Year_Month | Reviewer_Location | Review_Text | Branch | sentimen |
|---|---|---|---|---|---|---|---|
| 0 | 670772142 | 4 | 2019-4 | Australia | If you've ever been to Disneyland anywhere you... | Disneyland_HongKong |  |
| 1 | 670682799 | 4 | 2019-5 | Philippines | Its been a while since d last time we visit HK... | Disneyland_HongKong |  |

| | Review_ID | Rating | Year_Month | Reviewer_Location | Review_Text | Branch | sentimen |
|---|---|---|---|---|---|---|---|
| **2** | 670623270 | 4 | 2019-4 | United Arab Emirates | Thanks God it wasn t too hot or too humid wh... | Disneyland_HongKong | |
| **3** | 670607911 | 4 | 2019-4 | Australia | HK Disneyland is a great compact park Unfortun... | Disneyland_HongKong | |
| **4** | 670607296 | 4 | 2019-4 | United Kingdom | the location is not in the city, took around 1... | Disneyland_HongKong | |
| **...** | ... | ... | ... | ... | ... | ... | |
| **42651** | 1765031 | 5 | missing | United Kingdom | i went to disneyland paris in july 03 and thou... | Disneyland_Paris | |
| **42652** | 1659553 | 5 | missing | Canada | 2 adults and 1 child of 11 visited Disneyland ... | Disneyland_Paris | |
| **42653** | 1645894 | 5 | missing | South Africa | My eleven year old daughter and myself went to... | Disneyland_Paris | |
| **42654** | 1618637 | 4 | missing | United States | This hotel, part of the Disneyland Paris compl... | Disneyland_Paris | |
| **42655** | 1536786 | 4 | missing | United Kingdom | I went to the Disneyparis resort, in 1996, wit... | Disneyland_Paris | |

37547 rows × 7 columns

In [22]:

```
Reviews = Disney_Reviews[['Review_Text','sentiment']] #Creating new df just for reviews
Reviews.head()
```

Out[22]:

| | Review_Text | sentiment |
|---|---|---|
| **0** | If you've ever been to Disneyland anywhere you... | 1 |
| **1** | Its been a while since d last time we visit HK... | 1 |

| | Review_Text | sentiment |
|---|---|---|
| 2 | Thanks God it wasn t too hot or too humid wh... | 1 |
| 3 | HK Disneyland is a great compact park Unfortun... | 1 |
| 4 | the location is not in the city, took around 1... | 1 |

In [23]:

```
Reviews.groupby('sentiment')['sentiment'].sum()
```

Out[23]:

```
sentiment
-1    -3626
 1    33921
Name: sentiment, dtype: int64
```

In [24]:

```
#Split train and test data using random number generation
index = Reviews.index
Reviews['random_number'] = np.random.randn(len(index))
train = Reviews[Reviews['random_number'] <= 0.8]
test = Reviews[Reviews['random_number'] > 0.8]
```

In [25]:

```
#Vectorizing the reviewer text so that each word can be counted and represented as a ve
vectorizer = CountVectorizer(token_pattern=r'\b\w+\b')
train_matrix = vectorizer.fit_transform(train['Review_Text'])
test_matrix = vectorizer.transform(test['Review_Text'])
```

In [26]:

```
#Creating train and test matrices and labels
X_train = train_matrix
X_test = test_matrix
y_train = train['sentiment']
y_test = test['sentiment']
```

In [27]:

```
#Setting up logistic regresion and creating model on the train data set
lr = LogisticRegression()
lr.fit(X_train,y_train)
```

Out[27]:

```
LogisticRegression()
```

In [28]:

```
#Using the model to make predictions on the test data set
predictions = lr.predict(X_test)
```

In [29]:

```
#Checking the results of the model via confusion matrix to get precision, accuracy and
new = np.asarray(y_test)
confusion_matrix(predictions,y_test)
```

Out[29]:

```
array([[ 565,  127],
       [ 230, 7167]], dtype=int64)
```

In [30]:

```
#Printing the results of the model
print(classification_report(predictions,y_test))
```

```
              precision    recall  f1-score   support

          -1       0.71      0.82      0.76       692
           1       0.98      0.97      0.98      7397

    accuracy                           0.96      8089
   macro avg       0.85      0.89      0.87      8089
```

```
weighted avg       0.96      0.96      0.96      8089
```

```
In [ ]:
```