Recruit Restaurant Visitor Forecasting

1.1 Business Problem

Problem Statement:

Running a thriving local restaurant isn't always as charming as first impressions appear. There are often all sorts of unexpected troubles popping up that could hurt business. **One common predicament is that restaurants need to know how many customers to expect each day to effectively purchase ingredients and schedule staff members.** This forecast isn't easy to make because many unpredictable factors affect restaurant attendance, like weather and local competition. It's even harder for newer restaurants with little historical data. Recruit Holdings has unique access to key datasets that could make automated future customer prediction possible. Specifically, Recruit Holdings owns Hot Pepper Gourmet (a restaurant review service), AirREGI (a restaurant point of sales service), and Restaurant Board (reservation log management software). **In this competition, you're challenged to use reservation and visitation data to predict the total number of visitors to a restaurant for future dates. This information will help restaurants be much more efficient and allow them to focus on creating an enjoyable dining experience for their customers.**

Loss function: RMSLE

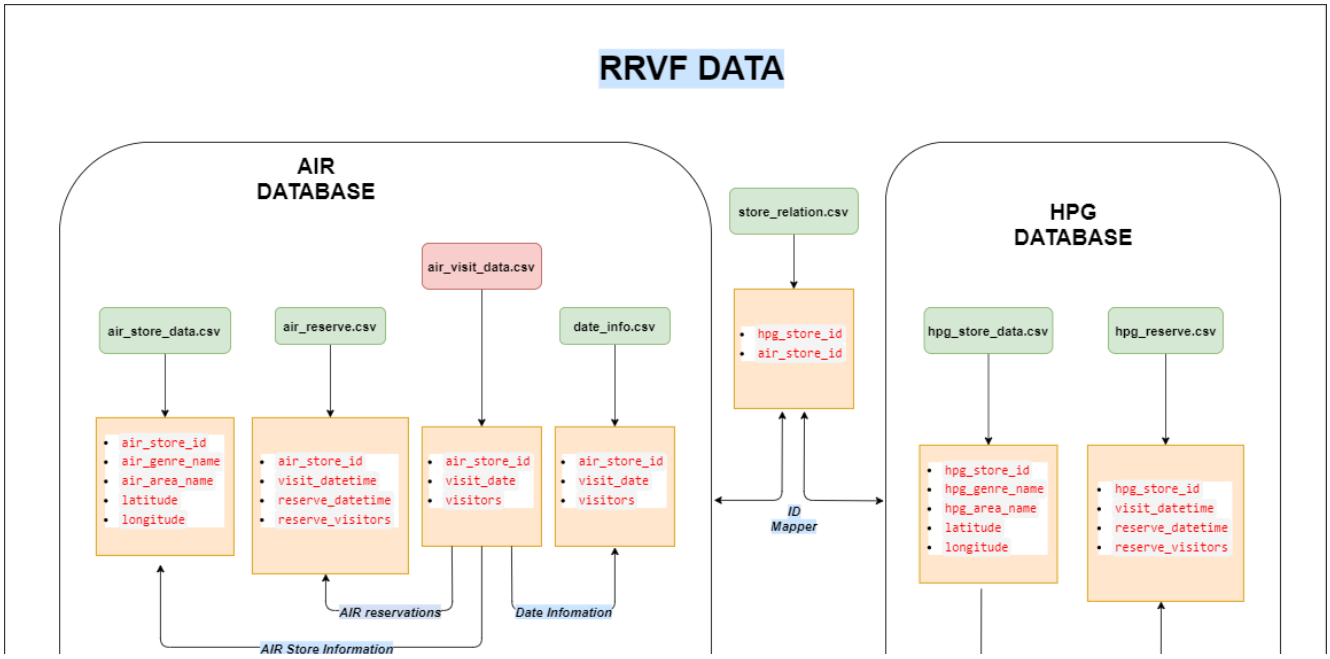
RMSLE has this unique feature of penalizing underprediction compared to overprediction, which is important in this problem since we don't want the restaurants to be underprepared especially in case of small restaurants. Being over prepared has less effects on the business as the extra resources can be stored.

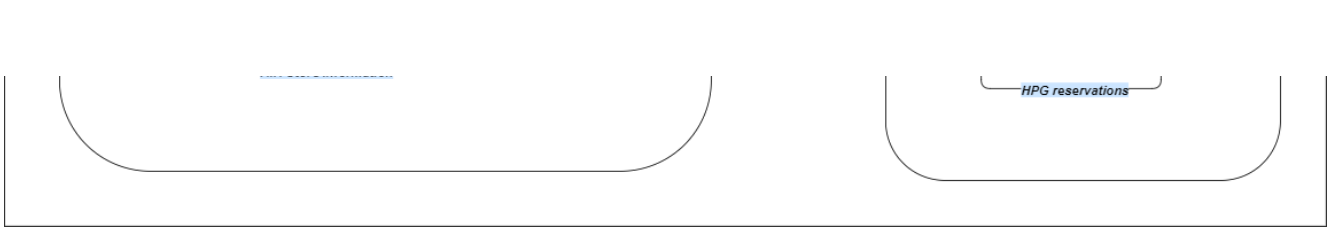
few other advantages:

```
* RMSLE is unaffected by the outlier values while RMSE does
* RMSLE focuses on the relational ratio difference between prediction and actual values while RM
SE focuses on the difference between their magnitudes.
```

read more

1.2 Database





1.3 Data Description

1. **air_visit_data.csv**: This data contains the historical visits done to AIR registered restaurants

    - **air_store_id** : Unique ID for AIR registered restaurants
    - **visit_date** : The date of the day
    - **visitors** : No. of customers visited the restaurant

1. **air_reserve.csv**: This data contains the reservation done using AIR reservation system

    - **air_store_id** : Unique ID for AIR registered restaurants
    - **visit_datetime** : The visiting date done through reservation
    - **reserve_datetime** : The date on which reservation was made.
    - **reserve_visitors** : No. of visitors for the reservation

1. **air_store_data**: This data contains the location and restaurant type information for AIR

    - **air_store_id** : Unique ID for AIR registered restaurants
    - **air_genre_name** : Type of restaurant
    - **air_area_name** : area name of the restaurant
    - **latitude** : lat of the restaurant
    - **longitude** : long of the restaurant

1. **date_info.csv**: This data contains the visting day calendar information

    - **calendar_date** : The date
    - **day_of_week** : what day
    - **holiday_flg** : if the day was holiday? 1:yes; 0:No

1. **hpg_reserve.csv**: This data contains the reservations done through HPG reservation system

    - **hpg_store_id** : Unique ID for the restaurants in HPG database
    - **visit_datetime** : the time of the reservation
    - **reserve_datetime** : the time the reservation was made
    - **reserve_visitors** : the number of visitors for that reservation

1. **hpg_store_info.csv**: This data contains the location and restaurant type information for HPG

    - **hpg_store_id** : Unique ID for HPG registered restaurants
    - **hpg_genre_name** : Type of restaurant
    - **hpg_area_name** : area name of the restaurant
    - **latitude** : lat of the restaurant
    - **longitude** : long of the restaurant

1. **store_id_relation.csv**: This data contains the mapping for HPG restaurants ID to AIR restaurant ID

    - **hpg_store_id** : HPG unique ID
    - **air_store_id** : AIR unique ID

1. **sample_submission.csv**: This is test data for which the predictions has to be done.

    - **id** : The id is formed by concatenating the air_store_id and visit_date with an underscore
    - **visitors** : The number of visitors forecasted for the store and date combination

1.4 Data Preparation

In [2]:

In [3]:

```
# imports
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import os

sns.set_style('whitegrid')

data_path = os.path.join(os.path.curdir, "Kaggle_Data")
processed_data_path = os.path.join(os.path.curdir, "Processed_Data")
```

In [4]:

```
# read all the csv files

# air
air_visit_data = pd.read_csv(data_path + "/air_visit_data.csv", )
air_reservation_data = pd.read_csv(data_path + "/air_reserve.csv")
air_store_info = pd.read_csv(data_path + "/air_store_info.csv")

# date info
date_info = pd.read_csv(data_path + "/date_info.csv")

# hpg
hpg_reservation_data = pd.read_csv(data_path + "/hpg_reserve.csv")
hpg_store_info = pd.read_csv(data_path + "/hpg_store_info.csv")
store_id_relation = pd.read_csv(data_path + "/store_id_relation.csv")

# submission
submission = pd.read_csv(data_path + "/sample_submission.csv")
```

1.4.1: Null values checking

In [3]:

```
def check_null_values(df: pd.DataFrame):
    """
    This function will check for null values in the dataframe
    """

    return np.any(df.isna().sum() > 1)


print("Null values check".center(50, "="))
print()
print("air_visit_data:", check_null_values(air_visit_data))
print("air_reservation_data:", check_null_values(air_reservation_data))
print("air_store_info:", check_null_values(air_store_info))
print("date_info:", check_null_values(date_info))
print("hpg_reservation_data:", check_null_values(hpg_reservation_data))
print("hpg_store_info:", check_null_values(hpg_store_info))
print("store_id_relation:", check_null_values(store_id_relation))
```

```
================Null values check=================

air_visit_data: False
air_reservation_data: False
air_store_info: False
date_info: False
hpg_reservation_data: False
hpg_store_info: False
store_id_relation: False
```

1.4.2 Data Merge

air_visit_data.csv and date_info.csv

In [ ]:

```
# rename the 'calendar_date' column to 'visit date'
date_info.rename(columns={'calendar_date': 'visit_date'}, inplace=True)

# merge the air_visit data and date info on 'visit date'
train_data = pd.merge(air_visit_data, date_info, how='left', on="visit_date")
train_data.sort_values(by='visit_date', inplace=True, ignore_index=True)

train_data.head()
```

Out[ ]:

	air_store_id	visit_date	visitors	day_of_week	holiday_flg
0	air_fab092c35776a9b1	2016-01-01	19	Friday	1
1	air_f26f36ec4dc5adb0	2016-01-01	64	Friday	1
2	air_d97dabf7aae60da5	2016-01-01	102	Friday	1
3	air_39dccf7df20b1c6a	2016-01-01	55	Friday	1
4	air_79f528087f49df06	2016-01-01	42	Friday	1

train_data.csv and air_store_info.csv

In [ ]:

```
# merge the train data and air_store_info on 'air_store_id'
train_data = pd.merge(train_data, air_store_info, how='left', on="air_store_id")

train_data.head()
```

Out[ ]:

	air_store_id	visit_date	visitors	day_of_week	holiday_flg	air_genre_name	air_area_name	latitude	longitude
0	air_fab092c35776a9b1	2016-01-01	19	Friday	1	Cafe/Sweets	Fukuoka-ken Fukuoka-shi Momochi	33.581941	130.348436
1	air_f26f36ec4dc5adb0	2016-01-01	64	Friday	1	Izakaya	Tōkyō-to Shinjuku-ku Kabukichō	35.693840	139.703549
2	air_d97dabf7aae60da5	2016-01-01	102	Friday	1	Cafe/Sweets	Tōkyō-to Shibuya-ku Jingūmae	35.669290	139.707056
3	air_39dccf7df20b1c6a	2016-01-01	55	Friday	1	Izakaya	Hyōgo-ken Takarazuka-shi Tōyōchō	34.799767	135.360073
4	air_79f528087f49df06	2016-01-01	42	Friday	1	Western food	Tōkyō-to Suginami-ku Asagayaminami	35.699566	139.636438

train_data.csv and air_reservation_info.csv

In [ ]:

```
# date time conversion
air_reservation_data.visit_datetime = pd.to_datetime(air_reservation_data.visit_datetime)
air_reservation_data.reserve_datetime = pd.to_datetime(air_reservation_data.reserve_datetime)

# add the hours gap difference between visit time to reservation time
air_reserve_diff = air_reservation_data.visit_datetime - air_reservation_data.reserve_datetime
air_reserve_diff = air_reserve_diff / np.timedelta64(1, 'h')
air_reservation_data['reservation_gap'] = air_reserve_diff

# add the visit date column
air_reservation_data['visit_date'] = air_reservation_data.visit_datetime.dt.date
```

In [ ]:

```
air_reservation_data.to_csv(processed_data_path + "/processed_air_reserve.csv", index=False)
air_reservation_data.head()
```

Out[ ]:

	air_store_id	visit_datetime	reserve_datetime	reserve_visitors	reservation_gap	visit_date
0	air_877f79706adbfb06	2016-01-01 19:00:00	2016-01-01 16:00:00	1	3.0	2016-01-01
1	air_db4b38ebe7a7ceff	2016-01-01 19:00:00	2016-01-01 19:00:00	3	0.0	2016-01-01
2	air_db4b38ebe7a7ceff	2016-01-01 19:00:00	2016-01-01 19:00:00	6	0.0	2016-01-01
3	air_877f79706adbfb06	2016-01-01 20:00:00	2016-01-01 16:00:00	2	4.0	2016-01-01
4	air_db80363d35f10926	2016-01-01 20:00:00	2016-01-01 01:00:00	5	19.0	2016-01-01

In [ ]:

```
# we need to get total reservation visitors on the visit_date per restaurant
reservations_per_day = air_reservation_data.groupby(by=['air_store_id', 'visit_date'], as_index=False).
sum()
reservations_per_day.visit_date = pd.to_datetime(reservations_per_day.visit_date)

# merge train_data and the total reservations done on the visit date per store id
train_data.visit_date = pd.to_datetime(train_data.visit_date)
train_data = pd.merge(train_data, reservations_per_day, how='left', on=['visit_date', 'air_store_id', ]
)
train_data.fillna(0, inplace=True)

# fill no reservation with zero
train_data.head()
```

Out[ ]:

	air_store_id	visit_date	visitors	day_of_week	holiday_flg	air_genre_name	air_area_name	latitude	longitude	reser
0	air_fab092c35776a9b1	2016-01-01	19	Friday	1	Cafe/Sweets	Fukuoka-ken Fukuoka-shi Momochi	33.581941	130.348436	
1	air_f26f36ec4dc5adb0	2016-01-01	64	Friday	1	Izakaya	Tōkyō-to Shinjuku-ku Kabukichō	35.693840	139.703549	
2	air_d97dabf7aae60da5	2016-01-01	102	Friday	1	Cafe/Sweets	Tōkyō-to Shibuya-ku Jingūmae	35.669290	139.707056	
3	air_39dccf7df20b1c6a	2016-01-01	55	Friday	1	Izakaya	Hyōgo-ken Takarazuka-shi Tōyōchō	34.799767	135.360073	
4	air_79f528087f49df06	2016-01-01	42	Friday	1	Western food	Tōkyō-to Suginami-ku Asagayaminami	35.699566	139.636438	

In [ ]:

```
train_data.to_csv(processed_data_path + "/processed_air_data.csv")
```

train_data.csv and hpg_reservation.csv

In [ ]:

```
## we need to extract the reservation made to the same AIR restaurants through HPG reservation system

# get the air_store_id's reservations done throught HPG reservation
hpg_database = pd.merge(hpg_reservation_data, store_id_relation, how='inner', on='hpg_store_id')

# add their store_information
hpg_database = pd.merge(hpg_database, hpg_store_info, how='left', on='hpg_store_id')
```

```
# add the reservation time diff
hpg_database.visit_datetime = pd.to_datetime(hpg_database.visit_datetime)
hpg_database.reserve_datetime = pd.to_datetime(hpg_database.reserve_datetime)
hpg_reserve_diff = hpg_database.visit_datetime - hpg_database.reserve_datetime
hpg_reserve_diff = hpg_reserve_diff / np.timedelta64(1, 'h')
hpg_database['hpg_reserve_gap'] = hpg_reserve_diff
hpg_database['hpg_visit_date'] = hpg_database.visit_datetime.dt.date
```

In [ ]:

```
hpg_database.to_csv(processed_data_path + "/processed_hpg_data.csv")
hpg_database.head()
```

Out[ ]:

	hpg_store_id	visit_datetime	reserve_datetime	reserve_visitors	air_store_id	hpg_genre_name	hpg_area_name
0	hpg_878cc70b1abc76f7	2016-01-01 19:00:00	2016-01-01 15:00:00	4	air_db80363d35f10926	Seafood	Hokkaidō Asahikawa-shi 3 Jōdōri
1	hpg_878cc70b1abc76f7	2016-01-02 19:00:00	2016-01-02 14:00:00	2	air_db80363d35f10926	Seafood	Hokkaidō Asahikawa-shi 3 Jōdōri
2	hpg_878cc70b1abc76f7	2016-01-03 18:00:00	2016-01-02 20:00:00	6	air_db80363d35f10926	Seafood	Hokkaidō Asahikawa-shi 3 Jōdōri
3	hpg_878cc70b1abc76f7	2016-01-06 20:00:00	2016-01-04 22:00:00	3	air_db80363d35f10926	Seafood	Hokkaidō Asahikawa-shi 3 Jōdōri
4	hpg_878cc70b1abc76f7	2016-01-11 18:00:00	2016-01-11 14:00:00	2	air_db80363d35f10926	Seafood	Hokkaidō Asahikawa-shi 3 Jōdōri

In [ ]:

```
# we need to get total reservation visitors on the visit_date per restaurant
hpg_reservations_per_day = hpg_database.groupby(by=['air_store_id', 'hpg_visit_date'], as_index=False)[
['reserve_visitors', 'hpg_reserve_gap']].sum()
hpg_reservations_per_day.rename(columns={'hpg_visit_date': 'visit_date'}, inplace=True)
hpg_reservations_per_day.visit_date = pd.to_datetime(hpg_reservations_per_day.visit_date)

# merge train_data and the total reservations done on the visit date per store id
train_data = pd.merge(train_data, hpg_reservations_per_day, how='left', on=['visit_date', 'air_store_id
'], suffixes=("_air", "_hpg"))
train_data.fillna(0, inplace=True)

# fill no reservation with zero
train_data.head()
```

Out[ ]:

	air_store_id	visit_date	visitors	day_of_week	holiday_flg	air_genre_name	air_area_name	latitude	longitude	reser
0	air_fab092c35776a9b1	2016-01-01	19	Friday	1	Cafe/Sweets	Fukuoka-ken Fukuoka-shi Momochi	33.581941	130.348436	
1	air_f26f36ec4dc5adb0	2016-01-01	64	Friday	1	Izakaya	Tōkyō-to Shinjuku-ku Kabukichō	35.693840	139.703549	
2	air_d97dabf7aae60da5	2016-01-01	102	Friday	1	Cafe/Sweets	Tōkyō-to Shibuya-ku Jingūmae	35.669290	139.707056	
3	air_39dccf7df20b1c6a	2016-01-01	55	Friday	1	Izakaya	Hyōgo-ken Takarazuka-shi Tōyōchō	34.799767	135.360073	
4	air_79f528087f49df06	2016-01-01	42	Friday	1	Western food	Tōkyō-to Suginami-ku Asagayaminami	35.699566	139.636438	

In [ ]:

```
train_data.to_csv(processed_data_path + "/processed_train_data.csv", index=False)
```

submission.csv

In [ ]:

```
submission['air_store_id'] = submission.id.str[:20]
submission['visit_date'] = submission.id.str[21:]
submission.head()
```

Out[ ]:

	id	visitors	air_store_id	visit_date
0	air_00a91d42b08b08d9_2017-04-23	0	air_00a91d42b08b08d9	2017-04-23
1	air_00a91d42b08b08d9_2017-04-24	0	air_00a91d42b08b08d9	2017-04-24
2	air_00a91d42b08b08d9_2017-04-25	0	air_00a91d42b08b08d9	2017-04-25
3	air_00a91d42b08b08d9_2017-04-26	0	air_00a91d42b08b08d9	2017-04-26
4	air_00a91d42b08b08d9_2017-04-27	0	air_00a91d42b08b08d9	2017-04-27

In [ ]:

```
submission.to_csv(processed_data_path + "/processed_test_data.csv", index=False)
```

In [ ]:

```
print()
print("Train Data information".center(50, "="))
print()
print("Train dataset size: ", train_data.shape)
print("Total unique restaurants", len(train_data.air_store_id.unique()))
print("Total unique restaurant genre: ", len(train_data.air_genre_name.unique()))
print(f"Average visitors: {train_data.visitors.mean():.2f}")
print(f"Median visitors: {train_data.visitors.median():.2f}")
print("The average/median vistors for restaurants is around 18-20. Most of these restaurants are small"
)
print("No of days data:", len(train_data.visit_date.unique()))
print(f"Date span: {train_data.visit_date.min().date()} - {train_data.visit_date.max().date()}")

print()
print("Test Data information".center(50, "="))
print()
print("Total unique restarants: ", len(submission.air_store_id.unique()))
print("No of days to be predicted: ",len(submission.visit_date.unique()))
submission.visit_date = pd.to_datetime(submission.visit_date)
print(f"Date span: {submission.visit_date.min().date()} - {submission.visit_date.max().date()}")
```

```
==============Train Data information==============

Train dataset size:  (252108, 13)
Total unique restaurants 829
Total unique restaurant genre:  14
Average visitors: 20.97
Median visitors: 17.00
The average/median vistors for restaurants is around 18-20. Most of these restaurants are small
No of days data: 478
Date span: 2016-01-01 - 2017-04-22

==============Test Data information===============

Total unique restarants:  821
No of days to be predicted:  39
Date span: 2017-04-23 - 2017-05-31
```

1.5 EDA

In [1]:

```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import datetime
sns.set_style('darkgrid')

%matplotlib inline
```

In [5]:

```
train_data = pd.read_csv(processed_data_path + "/processed_train_data.csv")
submission = pd.read_csv(processed_data_path + "/processed_test_data.csv")
train_data.visit_date = pd.to_datetime(train_data.visit_date)
train_data.head()
```

Out[5]:

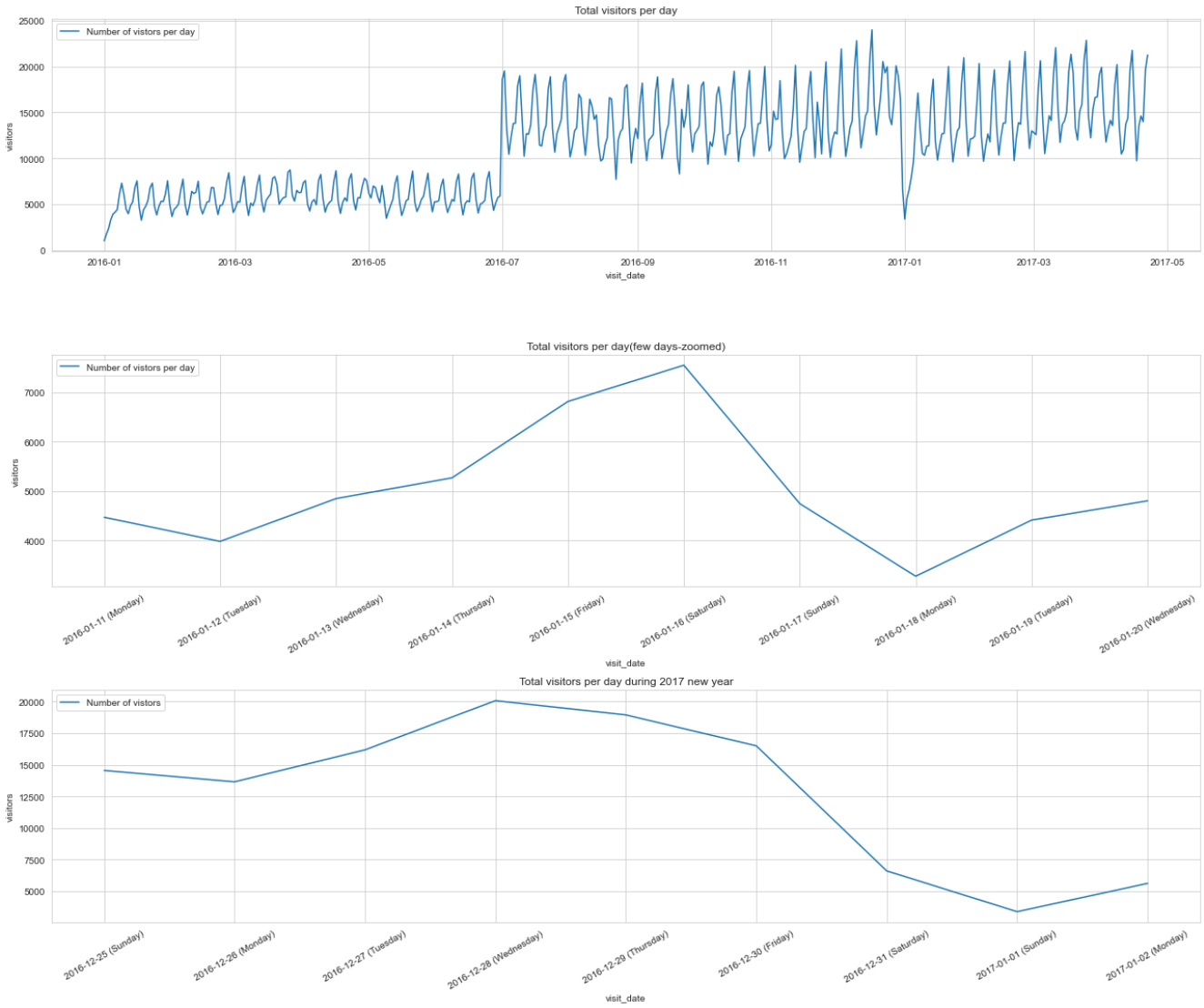
	air_store_id	visit_date	visitors	day_of_week	holiday_flg	air_genre_name	air_area_name	latitude	longitude	reser
0	air_fab092c35776a9b1	2016-01-01	19	Friday	1	Cafe/Sweets	Fukuoka-ken Fukuoka-shi Momochi	33.581941	130.348436	
1	air_f26f36ec4dc5adb0	2016-01-01	64	Friday	1	Izakaya	Tōkyō-to Shinjuku-ku Kabukichō	35.693840	139.703549	
2	air_d97dabf7aae60da5	2016-01-01	102	Friday	1	Cafe/Sweets	Tōkyō-to Shibuya-ku Jingūmae	35.669290	139.707056	
3	air_39dccf7df20b1c6a	2016-01-01	55	Friday	1	Izakaya	Hyōgo-ken Takarazuka-shi Tōyōchō	34.799767	135.360073	
4	air_79f528087f49df06	2016-01-01	42	Friday	1	Western food	Tōkyō-to Suginami-ku Asagayaminami	35.699566	139.636438	

Visitors trend across the days

In [6]:

```
# per day visitors
fig = plt.figure(figsize=(18,15))

ax1 = fig.add_subplot(311)

temp = train_data.groupby("visit_date").visitors.sum()

sns.lineplot(temp.index, temp, ax=ax1, label="Number of vistors per day")
plt.title("Total visitors per day")
plt.legend(loc='upper left')


ax2 = fig.add_subplot(312)
sns.lineplot(temp.index[10:20], temp[10:20], ax=ax2, label="Number of vistors per day")
ax2.set_xticklabels(temp[10:20].index.strftime("%Y-%m-%d (%A)"), rotation=30)
plt.legend(loc='upper left')
plt.title("Total visitors per day(few days-zoomed)")


temp = temp.loc['2016-12-25': '2017-01-02']

ax3 = fig.add_subplot(313)
```

```
sns.lineplot(temp.index, temp, ax=ax3, label="Number of vistors")
ax3.set_xticklabels(temp.index.strftime("%Y-%m-%d (%A)"), rotation=30)
plt.title("Total visitors per day during 2017 new year")
plt.legend(loc='upper left')
plt.tight_layout()
plt.show()
```



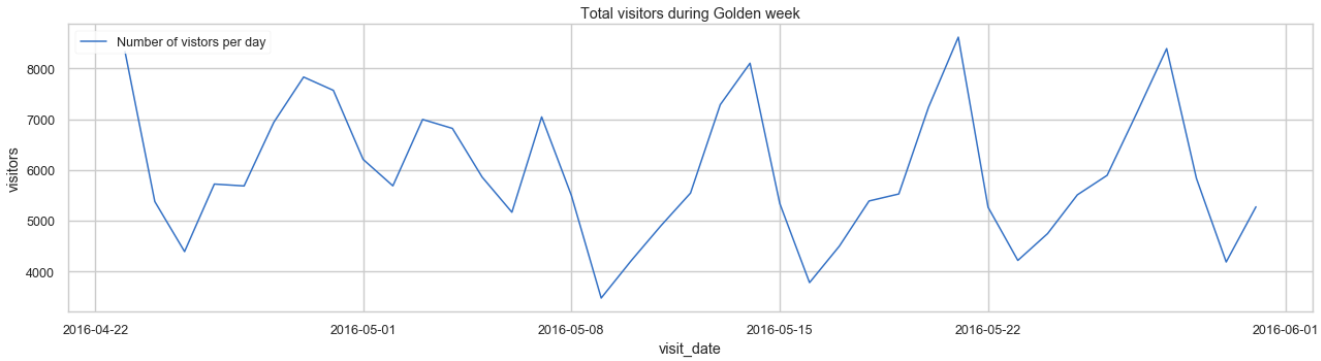
- It can be observed that there is seasonality for visitors variable. Around July 2016, there is a sharp rise is in visitors, this could be due to more restaurants being added to the database.
- The seasonality across training data is corresponding to the weekly cycle. As the weekend nears, there is rise in visitors for AIR restaurants.
- There is also sharp dip in visitors during 2017-01-01, which is due to New year celebration.

Visitors trend during golden week

In [ ]:

```
# per day visitors
fig = plt.figure(figsize=(18,5))

ax1 = fig.add_subplot(111)

temp = train_data.groupby("visit_date").visitors.sum()

# the golden week is between april and may
temp = temp.loc['2016-04-23': '2016-05-31']

sns.lineplot(temp.index, temp, ax=ax1, label="Number of vistors per day")
plt.title("Total visitors during Golden week")
plt.legend(loc='upper left')
```

```
plt.tight_layout()
plt.show()
```



- The golden week(April 29 - May 05) effect can be seen on visitors. There is consistency in visitors for AIR restaurants.
- After the golden week(May 06), there is dip in visitors, even on the weekend. The next week the weekly trend continues.
- Since the test data set spans around the golden week, this exception needs to be handled.

In [ ]:

```
train_data.head()
```

Out[ ]:

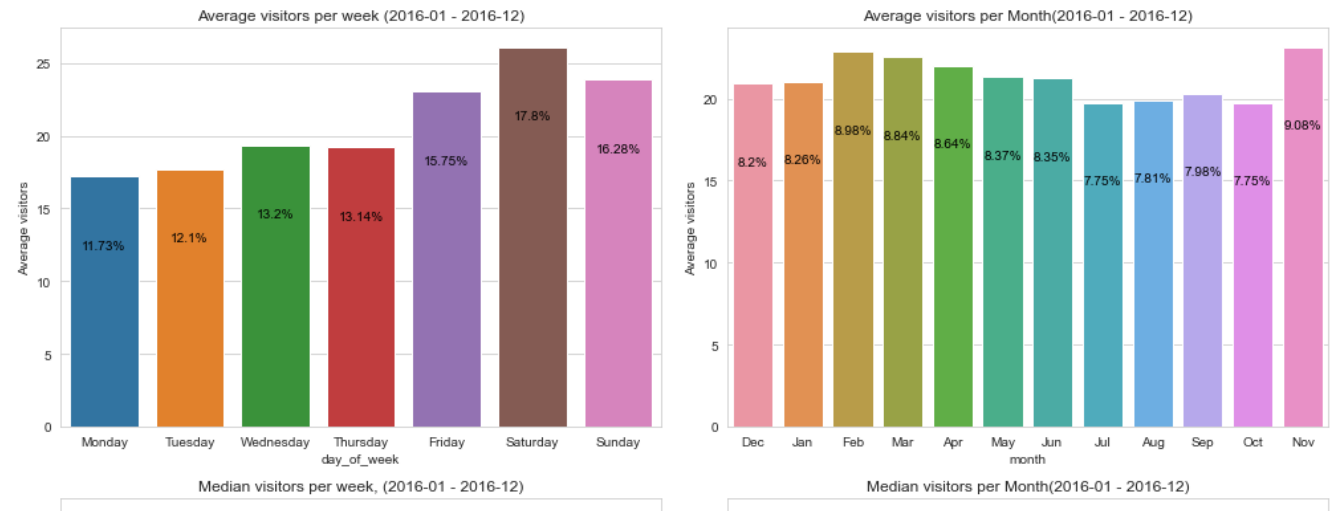
	air_store_id	visit_date	visitors	day_of_week	holiday_flg	air_genre_name	air_area_name	latitude	longitude	reser
0	air_fab092c35776a9b1	2016-01-01	19	Friday	1	Cafe/Sweets	Fukuoka-ken Fukuoka-shi Momochi	33.581941	130.348436	
1	air_f26f36ec4dc5adb0	2016-01-01	64	Friday	1	Izakaya	Tōkyō-to Shinjuku-ku Kabukichō	35.693840	139.703549	
2	air_d97dabf7aae60da5	2016-01-01	102	Friday	1	Cafe/Sweets	Tōkyō-to Shibuya-ku Jingūmae	35.669290	139.707056	
3	air_39dccf7df20b1c6a	2016-01-01	55	Friday	1	Izakaya	Hyōgo-ken Takarazuka-shi Tōyōchō	34.799767	135.360073	
4	air_79f528087f49df06	2016-01-01	42	Friday	1	Western food	Tōkyō-to Suginami-ku Asagayaminami	35.699566	139.636438	

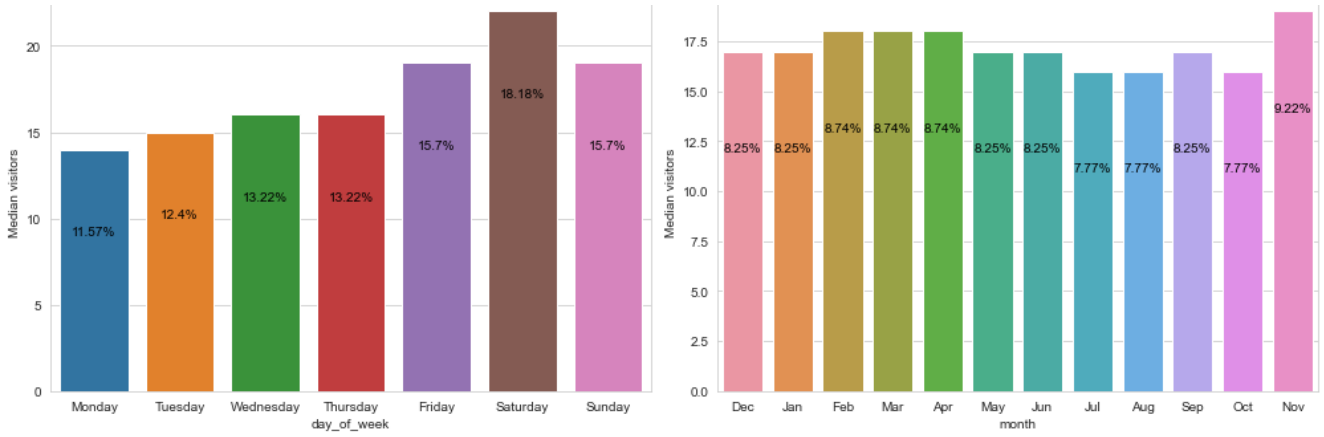
Average visitors by week/month for the year 2016-17

In [8]:

```
# average per week, per month
fig = plt.figure(figsize=(14,10))

ax1 = fig.add_subplot(221)
# order the days
# https://stackoverflow.com/a/35194104/9103175
train_data.day_of_week = pd.Categorical(train_data['day_of_week'], categories=
    ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday'],
    ordered=True)

train_2016_data = train_data.loc[train_data.visit_date.dt.year == 2016]

temp = train_2016_data.groupby("day_of_week", as_index=False).visitors.mean()
temp = temp.sort_index()
temp['percent'] =  (temp.visitors / temp.visitors.sum()) * 100
```

```
g = sns.barplot(temp.day_of_week, temp.visitors, ax=ax1)

for index, row in temp.iterrows():
    g.text(index, row.visitors-5, str(np.round(row.percent,2)) + "%", color='black', ha="center")

plt.title("Average visitors per week (2016-01 - 2016-12)")
plt.ylabel("Average visitors")

train_2016_data.loc[:, 'month'] = train_2016_data.loc[:, 'visit_date'].dt.month
temp = train_2016_data.groupby("month", as_index=False).visitors.mean()
temp['percent'] =  (temp.visitors / temp.visitors.sum()) * 100

ax2 = fig.add_subplot(222)
g = sns.barplot(temp.month, temp.visitors, ax=ax2)
months = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec']

for index, row in temp.iterrows():
    g.text(index, row.visitors-5, str(np.round(row.percent,2)) + "%", color='black', ha="center")

    plt.title("Average visitors per Month(2016-01 - 2016-12)")
x_labels = [ months[i-1] for i in temp.index]
ax2.set_xticklabels(x_labels)
plt.ylabel("Average visitors")


# Median per week, per month
ax3 = fig.add_subplot(223)
temp = train_2016_data.groupby("day_of_week", as_index=False).visitors.median()
temp = temp.sort_index()
temp['percent'] =  (temp.visitors / temp.visitors.sum()) * 100

g = sns.barplot(temp.day_of_week, temp.visitors, ax=ax3)

for index, row in temp.iterrows():
    g.text(index, row.visitors-5, str(np.round(row.percent,2)) + "%", color='black', ha="center")

plt.title("Median visitors per week, (2016-01 - 2016-12)")
plt.ylabel("Median visitors")

train_2016_data.loc[:, 'month'] = train_2016_data.loc[:, 'visit_date'].dt.month
temp = train_2016_data.groupby("month", as_index=False).visitors.median()
temp['percent'] =  (temp.visitors / temp.visitors.sum()) * 100

ax4 = fig.add_subplot(224)
g = sns.barplot(temp.month, temp.visitors, ax=ax4)

for index, row in temp.iterrows():
    g.text(index, row.visitors-5, str(np.round(row.percent,2)) + "%", color='black', ha="center")

months = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec']
plt.title("Median visitors per Month(2016-01 - 2016-12)")
x_labels = [ months[i-1] for i in temp.index]
ax4.set_xticklabels(x_labels)
plt.ylabel("Median visitors")
plt.tight_layout()
plt.show()
```



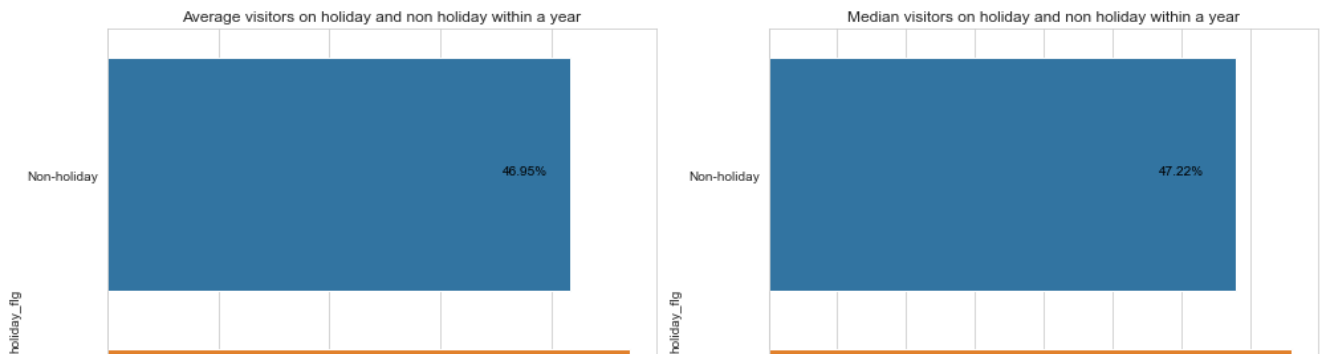


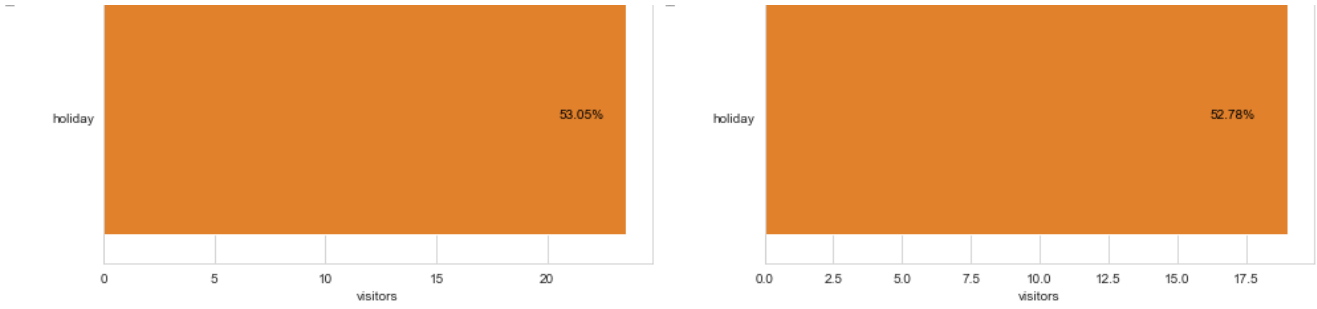
- Saturday seems to be popular choice of day for restaurant dinner. Friday being second. This seems obvious given that Friday is last working day and Saturday, Sunday are holidays
- December gets highest visitors due to New year celebrations. March, April and May have consistent visitors
- There is a significant difference in mean and median values of visitors. There is presence of some outliers which needs to dealt

Vistors vs next day holiday

In [9]:

```
# average vistors on holiday and non holiday
fig = plt.figure(figsize=(14,7))

ax1 = fig.add_subplot(121)

temp = train_2016_data.groupby("holiday_flg", as_index=False).visitors.mean()
temp['percent'] = (temp.visitors / temp.visitors.sum()) * 100

g = sns.barplot(temp.visitors, temp.holiday_flg, ax=ax1, orient='h')

for index, row in temp.iterrows():
    g.text(row.visitors-2, index, str(np.round(row.percent, 2)) + "%", color='black', ha="center")
labels = ['Non-holiday', 'holiday']
ax1.set_yticklabels([labels[i] for i in temp.index])
plt.title("Average visitors on holiday and non holiday within a year")

temp = train_2016_data.groupby("holiday_flg", as_index=False).visitors.median()
temp['percent'] = (temp.visitors / temp.visitors.sum()) * 100
ax2 = fig.add_subplot(122)
g = sns.barplot(temp.visitors, temp.holiday_flg, ax=ax2, orient='h')
for index, row in temp.iterrows():
    g.text(row.visitors-2, index, str(np.round(row.percent, 2)) + "%", color='black', ha="center")

labels = ['Non-holiday', 'holiday']
ax2.set_yticklabels([labels[i] for i in temp.index])
plt.title("Median visitors on holiday and non holiday within a year")

plt.tight_layout()
plt.show()
```





- For obvious reasons, the visitors are high on holidays in comparison to visitors on non-holiday

Visitors trend on weekends vs weekdays

In [11]:

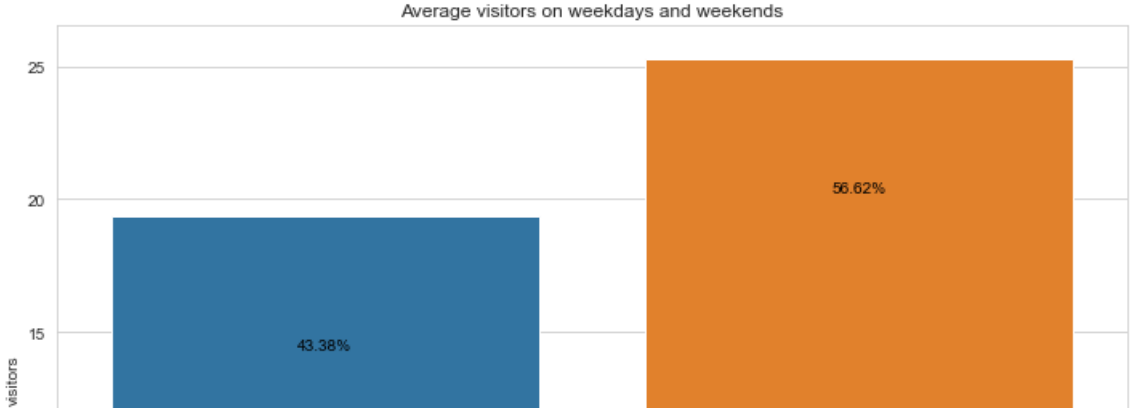
```
# weekend_flag
train_data['weekend_flag'] = train_data.day_of_week.isin(['Saturday','Sunday']).astype(int)
train_data.head(2)
```

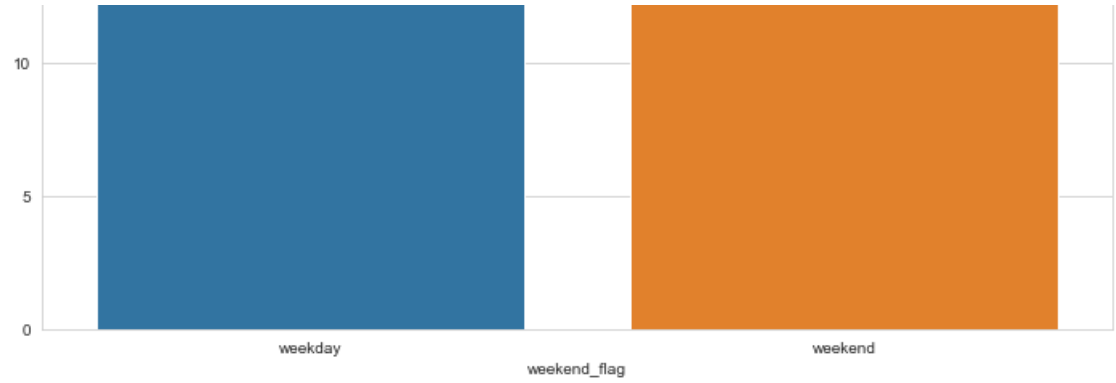
Out[11]:

	air_store_id	visit_date	visitors	day_of_week	holiday_flg	air_genre_name	air_area_name	latitude	longitude	reser
0	air_fab092c35776a9b1	2016-01-01	19	Friday	1	Cafe/Sweets	Fukuoka-ken Fukuoka-shi Momochi	33.581941	130.348436	
1	air_f26f36ec4dc5adb0	2016-01-01	64	Friday	1	Izakaya	Tōkyō-to Shinjuku-ku Kabukichō	35.693840	139.703549	

In [12]:

```
# average vistors on weekend and weekdays
fig = plt.figure(figsize=(10,7))

ax1 = fig.add_subplot(111)

temp = train_data.groupby("weekend_flag", as_index=False).visitors.mean()
temp['percent'] = (temp.visitors / temp.visitors.sum()) * 100
g = sns.barplot(temp.weekend_flag, temp.visitors, ax=ax1)

for index, row in temp.iterrows():
    g.text(index, row.visitors-5, str(np.round(row.percent, 2)) + "%", color='black', ha='center')

labels = ['weekday', 'weekend']
ax1.set_xticklabels([labels[i] for i in temp.index])
plt.title("Average visitors on weekdays and weekends")

plt.tight_layout()
plt.show()
```





- `weekends have more visitors in comparison to weekdays`

Visitors trend if next day is holiday

In [13]:

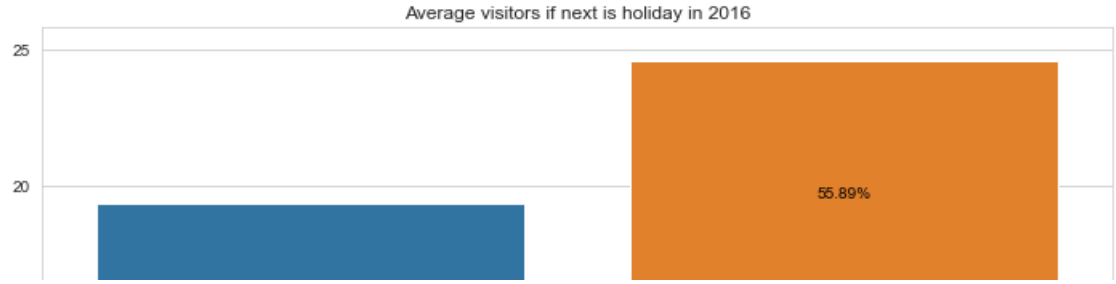
```
# next_day_holiday_flag
train_data['nxt_day_holiday_flg'] = train_data.day_of_week.isin(['Friday','Saturday']).astype(int)
train_data.head(2)
```

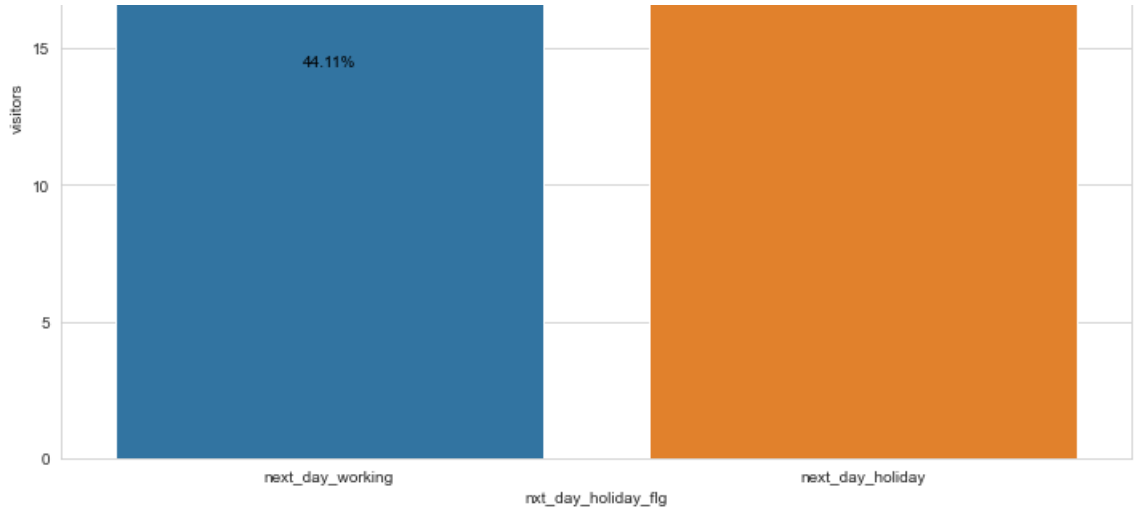
Out[13]:

	air_store_id	visit_date	visitors	day_of_week	holiday_flg	air_genre_name	air_area_name	latitude	longitude	reser
0	air_fab092c35776a9b1	2016-01-01	19	Friday	1	Cafe/Sweets	Fukuoka-ken Fukuoka-shi Momochi	33.581941	130.348436	
1	air_f26f36ec4dc5adb0	2016-01-01	64	Friday	1	Izakaya	Tōkyō-to Shinjuku-ku Kabukichō	35.693840	139.703549	

In [14]:

```
# average vistors if next day is holiday
fig = plt.figure(figsize=(10,7))

ax1 = fig.add_subplot(111)

train_2016_data = train_data.loc[train_data.visit_date.dt.year == 2016]

temp = train_2016_data.groupby("nxt_day_holiday_flg", as_index=False).visitors.mean()
temp['percent'] = (temp.visitors / temp.visitors.sum()) * 100

g = sns.barplot(temp.nxt_day_holiday_flg, temp.visitors, ax=ax1)
for index, row in temp.iterrows():
    g.text(index, row.visitors-5, str(np.round(row.percent, 2)) + "%", color='black', ha='center')

labels = ['next_day_working', 'next_day_holiday']
ax1.set_xticklabels([labels[i] for i in temp.index])
plt.title("Average visitors if next is holiday in 2016")

plt.tight_layout()
plt.show()
```



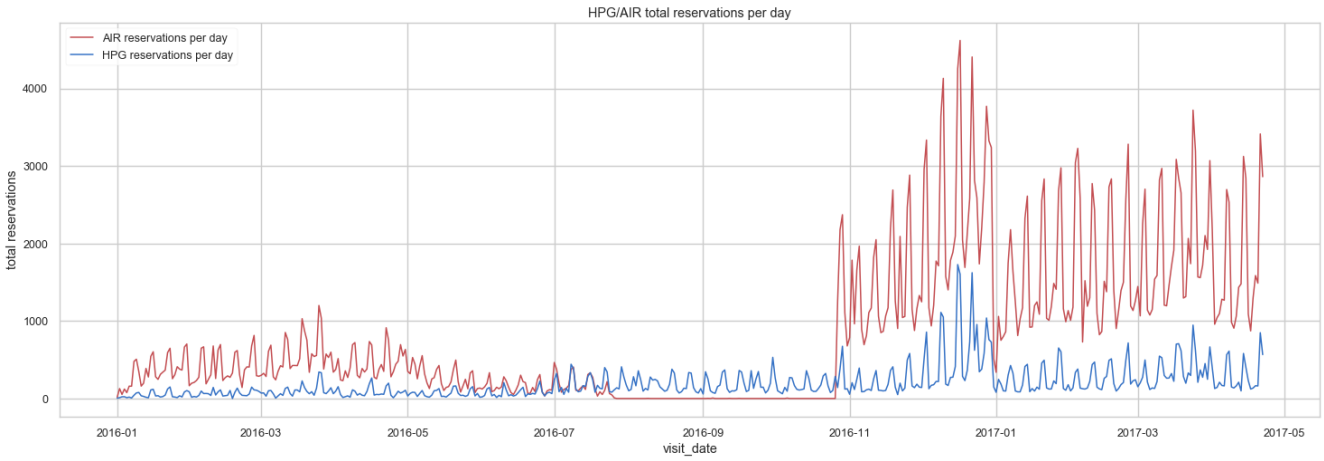


- `Visitors are high not only during holidays, but also if next day is holiday`

reservations done through AIR/HPG system

In [ ]:

```
# reservations
# total reservations per day
temp = train_data.groupby("visit_date").reserve_visitors_air.sum()
temp2 = train_data.groupby("visit_date").reserve_visitors_hpg.sum()

fig = plt.figure(figsize=(25,8))

ax = fig.add_subplot(111)

sns.lineplot(temp.index, temp, label="AIR reservations per day", color='r', ax=ax)
sns.lineplot(temp2.index, temp2, label="HPG reservations per day", color='b', ax=ax)
plt.legend(loc='upper left')
plt.ylabel("total reservations")

plt.title("HPG/AIR total reservations per day")
plt.show()
```

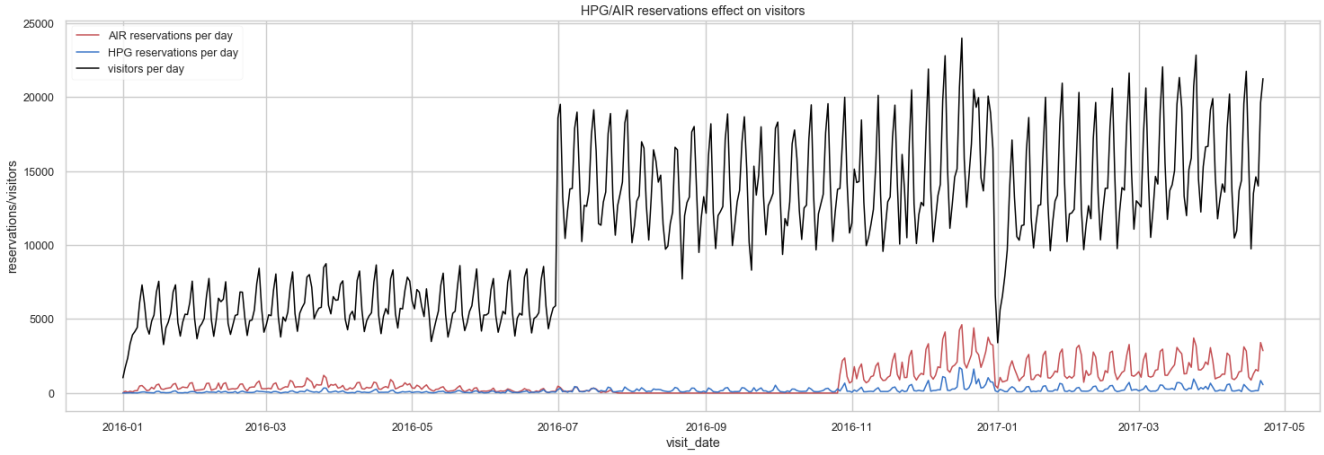


- `AIR reservations have missing data between 2016-07 to 2016-11`
- `Both reservations are high during the December and drop on the 2017-01-01`
- `The reservations in both systems seems to be more in 2017 especially in AIR system compared to 2016`

Does reservations has any effect on visitors?

In [ ]:

```
# reservations
# total reservations per day
temp = train_data.groupby("visit_date").reserve_visitors_air.sum()
temp2 = train_data.groupby("visit_date").reserve_visitors_hpg.sum()
temp3 = train_data.groupby("visit_date").visitors.sum()

fig = plt.figure(figsize=(25,8))

ax = fig.add_subplot(111)

sns.lineplot(temp.index, temp, label="AIR reservations per day", color='r', ax=ax)
sns.lineplot(temp2.index, temp2, label="HPG reservations per day", color='b', ax=ax)
sns.lineplot(temp3.index, temp3, label="visitors per day", color='black', ax=ax)
plt.legend(loc='upper left')
plt.ylabel("reservations/visitors")

plt.title("HPG/AIR reservations effect on visitors")
plt.show()
```



- The reservations do have positive effect on visitors. Around 2016-11, it can be observed that visitors are spiking with reservations
- This observation can also mean that most of the reservations are done for single day
- Also visitor number is higher in comparison with reservations. This is obvious as most prefer to walk in than reservations. Also the reservations are higher in December. This is due to busy restaurants on new years eve and its better to reserve than walking in

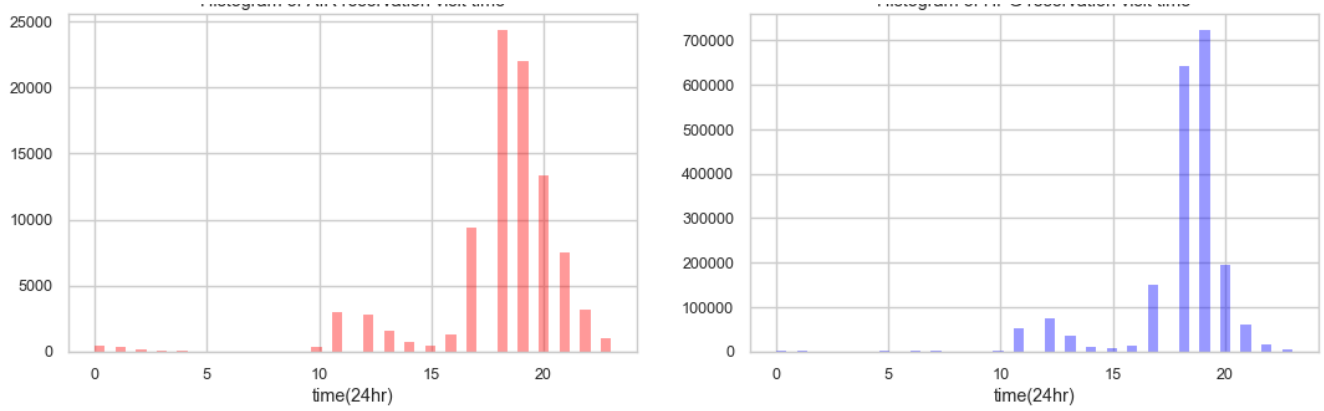
Reservation time

In [ ]:

```
air_reservation_data.visit_datetime = pd.to_datetime(air_reservation_data.visit_datetime)
air_reservation_data.reserve_datetime = pd.to_datetime(air_reservation_data.reserve_datetime)

fig = plt.figure(figsize=(18, 5))
ax1 = fig.add_subplot(121)

sns.distplot(air_reservation_data.visit_datetime.dt.hour, color='red', kde=False, ax=ax1)
plt.title("Histogram of AIR reservation visit time")
plt.xlabel("time(24hr)")

ax2 = fig.add_subplot(122)
hpg_reservation_data.visit_datetime = pd.to_datetime(hpg_reservation_data.visit_datetime)
hpg_reservation_data.reserve_datetime = pd.to_datetime(hpg_reservation_data.reserve_datetime)
sns.distplot(hpg_reservation_data.visit_datetime.dt.hour, color='blue', kde=False, ax=ax2)
plt.title("Histogram of HPG reservation visit time")
plt.xlabel("time(24hr)")
plt.show()

plt.show()
```

Histogram of AIR reservation visit time

Histogram of HPG reservation visit time

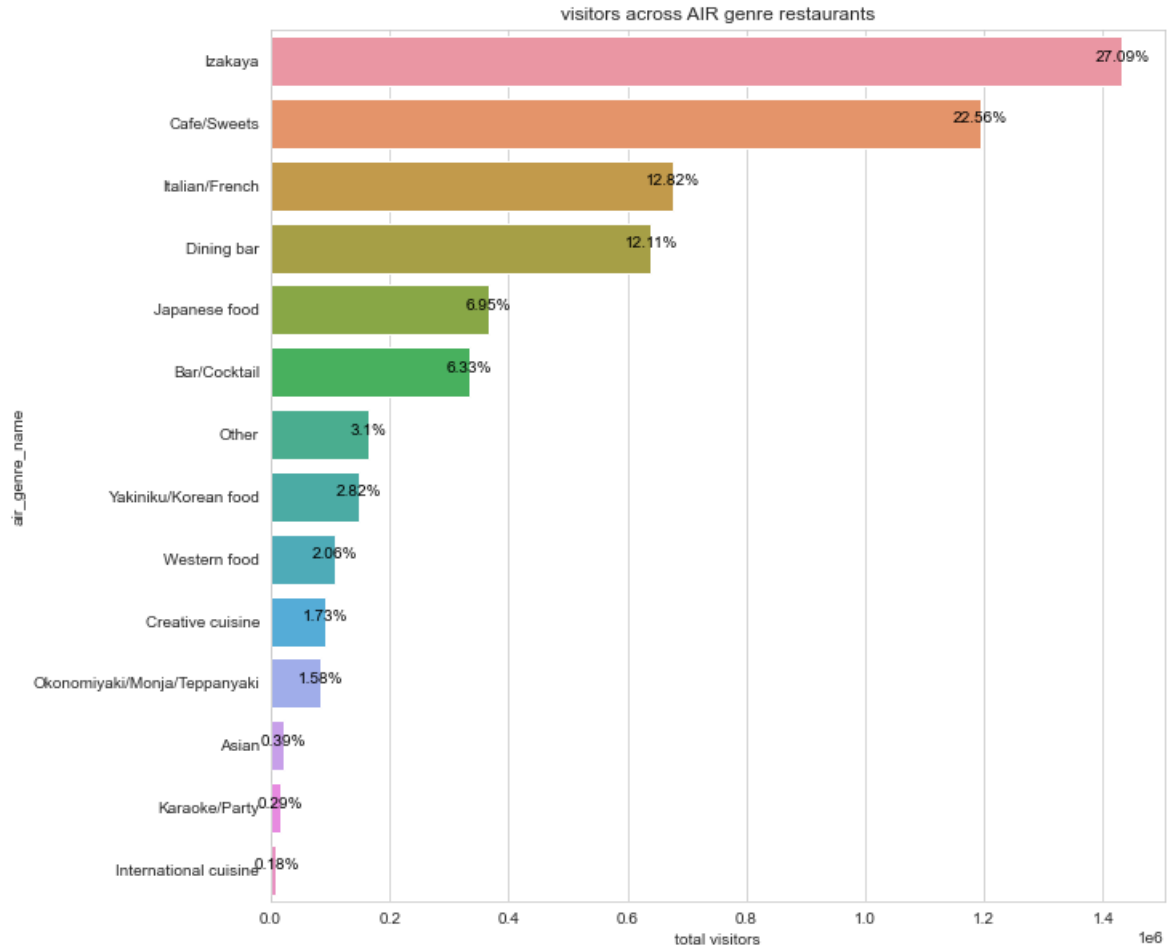


- Both in AIR and HPG, reservations are high for dinner time
- reservations on HPG system are lot higher in comparison with AIR

Which genre is popular among visitors?

In [15]:

```
fig, ax = plt.subplots(figsize=(10,10))

temp = train_data.groupby("air_genre_name", as_index=False).visitors.sum()
temp.sort_values(by='visitors', inplace=True, ascending=False)
temp = temp.reset_index()
temp['percent'] = (temp.visitors / temp.visitors.sum()) * 100

g = sns.barplot(x=temp.visitors, y=temp.air_genre_name, ax=ax)
for index, row in temp.iterrows():
    g.text(row.visitors - 7, index, str(np.round(row.percent, 2)) + '%', ha='center', color='black')
plt.title("visitors across AIR genre restaurants")
plt.xlabel("total visitors")
plt.show()
```

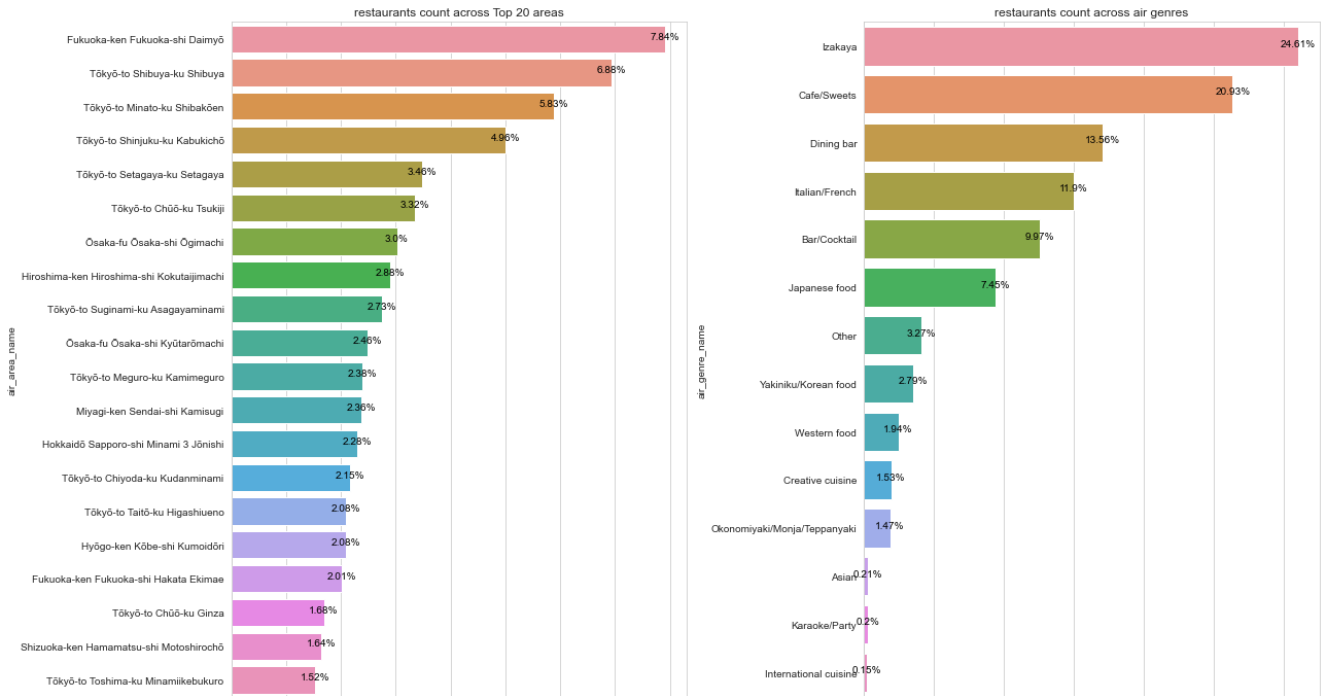


- 'IZAKAYA' seem genre is popular among visitors. A google search reveals that 'IZAKAYA' are japanese bars which serve alcohol and snacks and is common choice for after work drinking
- after work or morning cafes/sweets has second highest visitors
- Though reservation is highest for Dinner time, the visitor count for food restaurants is less
- International cuisine has the least visitors

area wise, genre wise restaurant count

In [16]:

```
fig = plt.figure(figsize=(18,10))

# top 20
temp = train_data.groupby("air_area_name", as_index=False).air_store_id.count()
temp.sort_values(by='air_store_id', inplace=True, ascending=False)
temp.reset_index(inplace=True)
temp['percent'] = (temp.air_store_id / temp.air_store_id.sum()) * 100

ax1 = fig.add_subplot(121)
g = sns.barplot(x=temp[:20].air_store_id, y=temp.air_area_name[:20], ax=ax1)
for index, row in temp[:20].iterrows():
    g.text(row.air_store_id - 7, index, str(np.round(row.percent, 2)) + '%', ha='center', color='black'
)
plt.title("restaurants count across Top 20 areas")
plt.xlabel("total restaurants")

temp = train_data.groupby("air_genre_name", as_index=False).air_store_id.count()
temp.sort_values(by='air_store_id', inplace=True, ascending=False)
temp.reset_index(inplace=True)
temp['percent'] = (temp.air_store_id / temp.air_store_id.sum()) * 100

ax2 = fig.add_subplot(122)
g = sns.barplot(x=temp.air_store_id, y=temp.air_genre_name, ax=ax2)

for index, row in temp[:20].iterrows():
    g.text(row.air_store_id - 7, index, str(np.round(row.percent, 2)) + '%', ha='center', color='black'
)

plt.title("restaurants count across air genres")
plt.xlabel("total restaurants")

plt.tight_layout()
plt.show()
```



- `Fukouka has the most restaurants followed by many areas around tokyo`
- `Izakaya, Cafe/sweets types of restaurants are high in numbers. No wonder the visitors count are high among these types`

Diff genre restaurants area wise

In [54]:

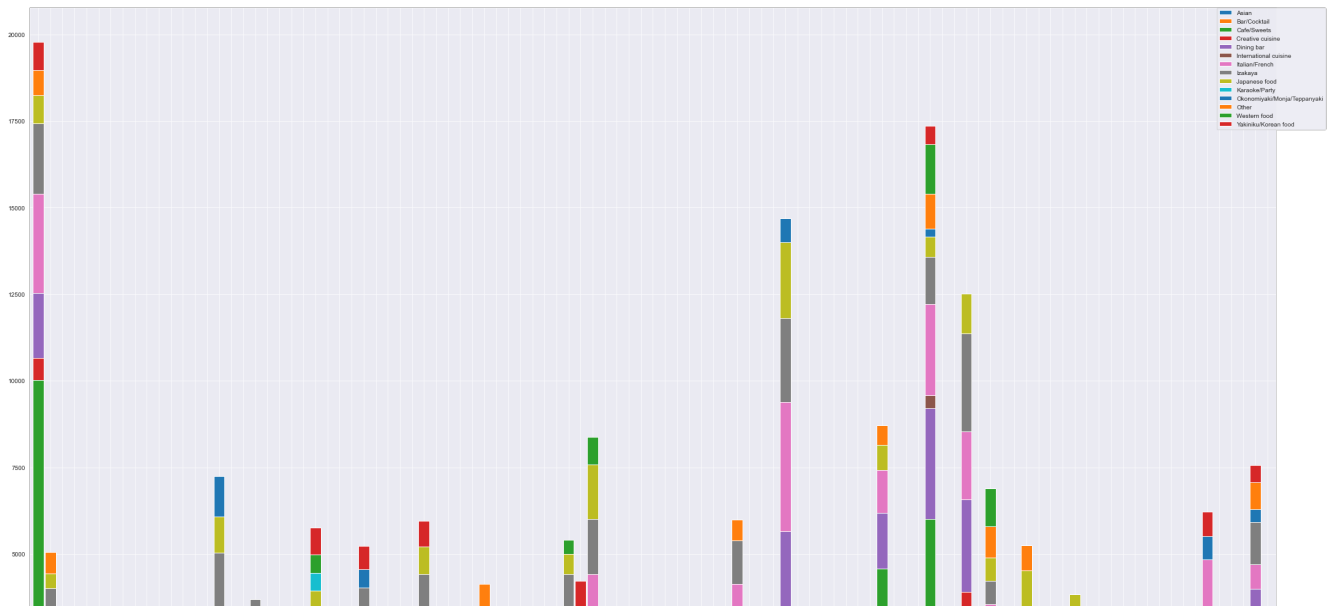
```
temp = train_data.groupby(['air_area_name', 'air_genre_name'], as_index=True).air_store_id.count()
temp = temp.unstack(fill_value=0)
# temp = (temp.T / temp.T.sum()).T
temp.head()
```

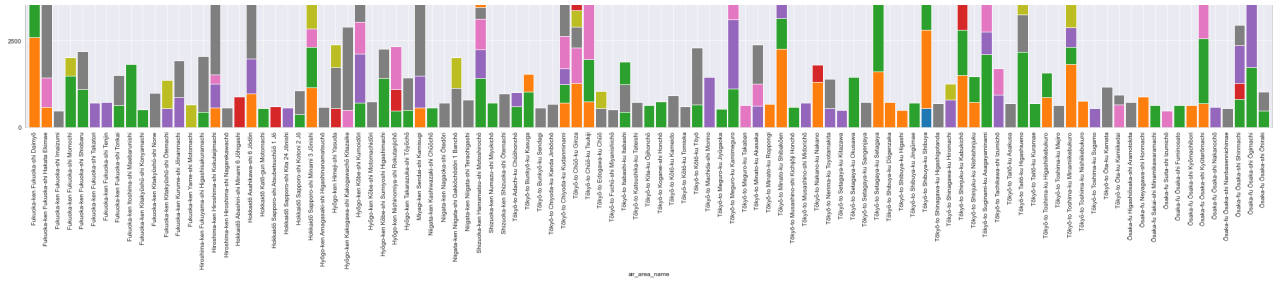
Out[54]:

air_area_name	air_genre_name	Asian	Bar/Cocktail	Cafe/Sweets	Creative cuisine	Dining bar	International cuisine	Italian/French	Izakaya	Japanese food	Karaoke/Party	Oki
Fukuoka-ken Fukuoka-shi Daimyō		0	2594	7421	648	1861	0	2870	2037	808	0	
Fukuoka-ken Fukuoka-shi Hakata Ekimae		0	574	0	0	0	0	844	2591	422	0	
Fukuoka-ken Fukuoka-shi Imaizumi		0	0	0	0	0	0	0	480	0	0	
Fukuoka-ken Fukuoka-shi Momochi		0	0	1477	0	0	0	0	0	538	0	
Fukuoka-ken Fukuoka-shi Shiobaru		0	0	1097	0	0	0	0	1082	0	0	

In [46]:

```
temp.plot(kind='bar', stacked=True, figsize=(30, 20), width=0.9)
# https://stackoverflow.com/a/43439132/9103175
plt.legend(bbox_to_anchor=(1.04,1), borderaxespad=0)
plt.tight_layout()
plt.show()
```





- Daimyo is the most famous place with 19k restaurants. Interestingly Cafe/sweets genre(second fav among visitors) dominate this area in comparison to Izakaya.
- Cafe/sweets and Izakaya seems to be most common in almost all the areas followed by Italian/French

area wise restaurant cluster

In [23]:

```
import folium
from folium.plugins import MarkerCluster

japan_map = folium.Map(location=[36.2048, 138.2529], tiles="CartoDBpositron", zoom_start=6, min_zoom=4)
marker_cluster = MarkerCluster().add_to(japan_map)

for row in air_store_info.values:

    folium.Marker(location=[row[-2], row[-1]], popup=f"{row[0]} - {row[1]} - {row[2]}").add_to(marker_c
luster)

japan_map
```

Out[23]:

Make this Notebook Trusted to load map: File -> Trust Notebook

- From the map, the Fukuoka has highest restaurants around one single place. Hence the high number of visitors.
- while the Tokyo area has overall high number of restaurants, they form different clusters spread in different areas of Tokyo. Hence the visitors are spread out.
- Area name could be very important information is determining the visitors.
- Osaka is second most popular and reasturants are spread out

- Osaka is second most popular and restaurants are spread out
- Also we can see that more the restaurants in the area more the visitors count. Area wise resturant count could be useful

In [24]:

```
japan_map.save("japan.html")
```

visitors trend across genre

In [17]:

```
temp = train_data.groupby(["visit_date", 'air_genre_name']).visitors.mean()
temp = temp.unstack(-1, fill_value=0)

fig = plt.figure(figsize=(30,10))

columns = temp.columns
for index, genre in enumerate(columns):

    ax = fig.add_subplot(3,5, index+1)
    sns.lineplot(temp.index, temp[genre], ax=ax)
    ax.set_xticklabels(labels=[])
    ax.set_yticklabels(labels=[])
    ax.set_title(genre, fontsize=25)
    plt.xlabel("")
    plt.ylabel("")

fig.suptitle("AVG visitors trend accross AIR genre restaurants", fontsize=30, ha='center', va='baseline
')
plt.tight_layout()
plt.show()
```

- Most genre have consistent visitors. Some of them, 'Karoke/Party', 'International Cuisine' have no visits data upto certain date
- Korean Food, Western Food, Asian have variable visits
- Japanese food seems to have the most consistent visits

Conclusion

- Visitors variable data is periodic in nature corresponding to a week pattern where there is a rise in visitors on weekends.
- The visit database has spike in data during July 2016, which might be the effect of additional restaurants being added to database.
- The Golden Week Season data can be seen in visitors variable, where the visitors have been consistent. This occurs during April 29 2016 to May 05 2016.
- We also see sharp dip in visitors on 2017-01-01. This might be due to most restaurants are closed on Jan 1st.
- On weekly basis, Visitors are high on Saturday and equally on Friday and Sunday.
- Month wise, December has more visitors compared to rest owing to festivals and New year's Eve.
- For obvious reasons, the visitors are high if its a public holiday

- For obvious reasons, the visitors are high if its a public holiday.
- Restaurants see a rise in visitors if the next day is holiday, compared to next day working. This trend is also seen on weekly basis where Friday has more visitors compared to other working days, since Saturday is weekend.
- The reservations has similar trend to visitors. However, many prefer to walk in. This is evident from the huge difference between reservations and visitors.
- The AIR reservation system has missing data between 2016-07 to 2016-11. No explanation is given for this missing data.
- Both in AIR and HPG reservation system, the reservations are less before 2016-11, but there is sudden rise in reservations after this.
- Majority of the reservations are done for dinner time.
- Bars and snacks genre, also known as 'Izakaya' in Japan is most popular among customers. Also this is highly spread across locations.
- Next to Izakaya, Cafe/Sweets seems to be most popular. Most office workers are common at cafe for morning and after coffee.
- International Cusine/Asian/Karoke Party are least popular. Interestingly Party is least popular. Japanese is popular for its longest working hours, which might be one of the reasons.
- 'Fukuoka' is the most popular area for restaurants though it has marginally less restaurants than tokyo. However Tokyo has restaurants spread in clusters which in turn makes the visitors to be spread. Tokyo is the next popular area among consumers
- Osaka though contains second most restaurants, the visitors is marginally less compared to other two.
- 'Izakaya' is present in majority of locations followed by Cafe/sweets.
- Majority of genres have consistent customers, except western food, International Cuisine, Asian and Karaoke genres.

1.6 Feature Engineering

In [4]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [3]:

```
train_data = pd.read_csv(processed_data_path + "/processed_train_data.csv")
submission = pd.read_csv(processed_data_path + "/processed_test_data.csv")
train_data.visit_date = pd.to_datetime(train_data.visit_date)
submission.visit_date = pd.to_datetime(submission.visit_date)
train_data.head()
```

Out[3]:

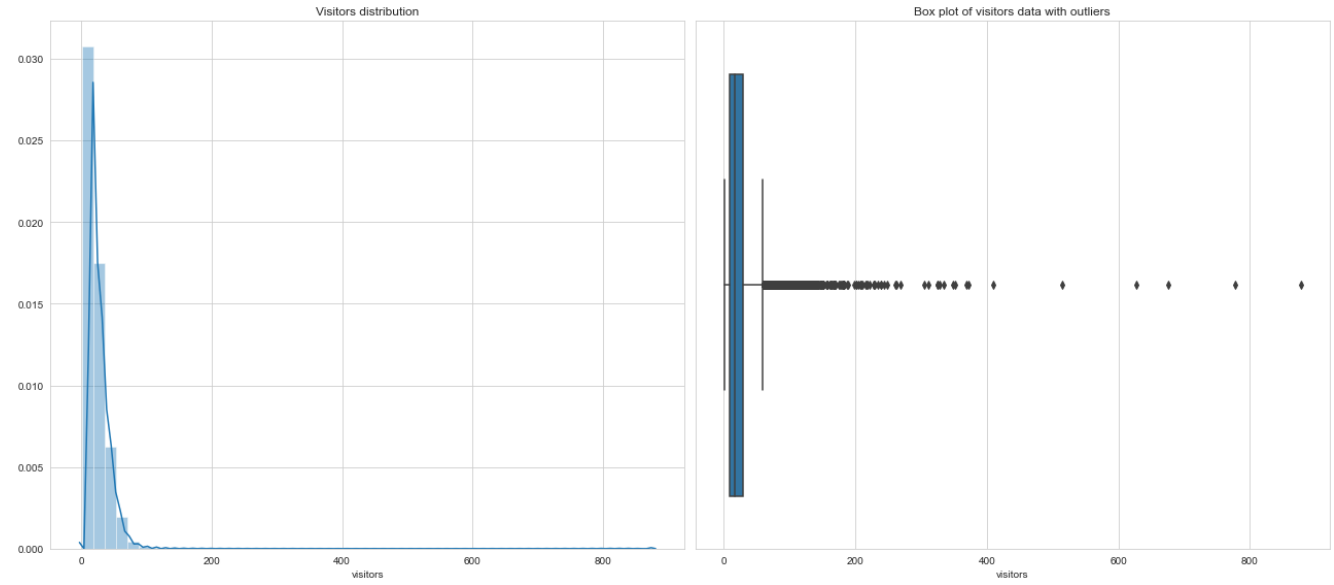
	air_store_id	visit_date	visitors	day_of_week	holiday_flg	air_genre_name	air_area_name	latitude	longitude	reser
0	air_fab092c35776a9b1	2016-01-01	19	Friday	1	Cafe/Sweets	Fukuoka-ken Fukuoka-shi Momochi	33.581941	130.348436	
1	air_f26f36ec4dc5adb0	2016-01-01	64	Friday	1	Izakaya	Tōkyō-to Shinjuku-ku Kabukichō	35.693840	139.703549	
2	air_d97dabf7aae60da5	2016-01-01	102	Friday	1	Cafe/Sweets	Tōkyō-to Shibuya-ku Jingūmae	35.669290	139.707056	
3	air_39dccf7df20b1c6a	2016-01-01	55	Friday	1	Izakaya	Hyōgo-ken Takarazuka-shi Tōyōchō	34.799767	135.360073	
4	air_79f528087f49df06	2016-01-01	42	Friday	1	Western food	Tōkyō-to Suginami-ku Asagayaminami	35.699566	139.636438	

remove outliers in visitors variable

- Some restaurants have visitors above 80. However most restaurants in dataset are small and have visitors in range of 15 to 40.
- Remove the rest using IQR

In [18]:

```
fig = plt.figure(figsize=(18, 8))

ax1 = fig.add_subplot(121)

sns.distplot(train_data.visitors, ax=ax1)
plt.title("Visitors distribution")

ax2 = fig.add_subplot(122)

sns.boxplot(train_data.visitors, ax=ax2)
plt.title("Box plot of visitors data with outliers")

plt.tight_layout()
plt.show()
```



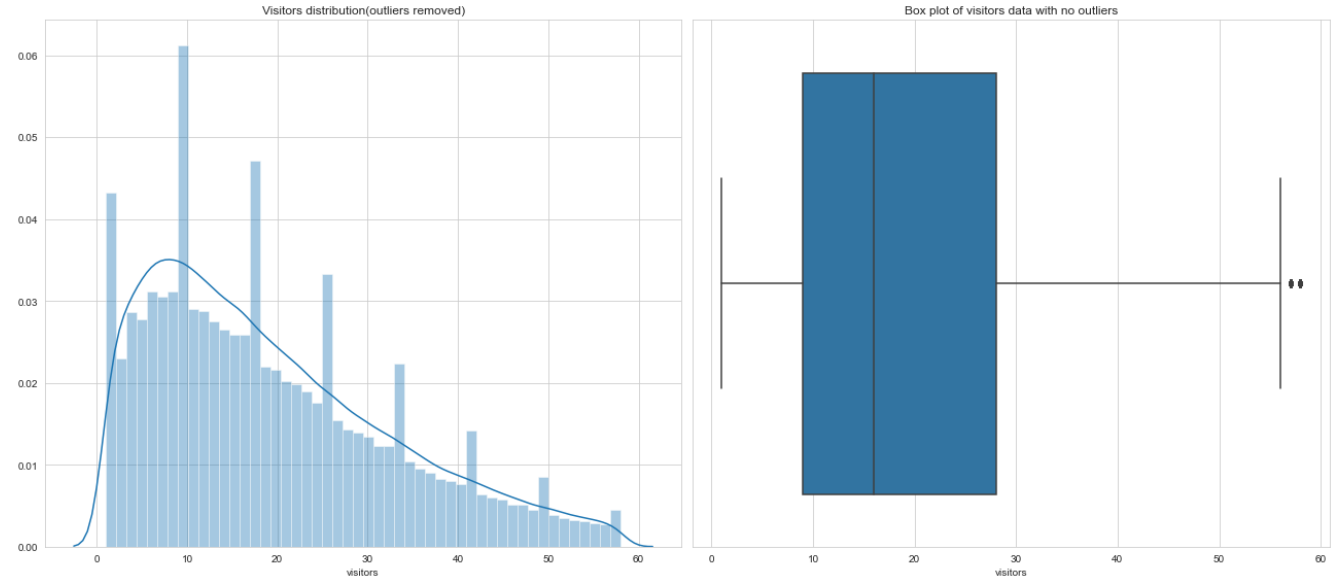
In [19]:

```
# remove the outliers > Q3 + 1.5*IQR

def calulate_IQR_outlier_range(data):

    q1, q3 = np.quantile(data.values, [0.25, 0.75])
    lower_IQR = q1 - (1.5 * (q3 - q1))
    higher_IQR = q3 + (1.5 * (q3 - q1))

    return lower_IQR, higher_IQR


visitors_lower, visitors_higher = calulate_IQR_outlier_range(train_data.visitors)

train_data = train_data[train_data.visitors < visitors_higher]
```

In [20]:

```
fig = plt.figure(figsize=(18, 8))

ax1 = fig.add_subplot(121)

sns.distplot(train_data.visitors, ax=ax1)
plt.title("Visitors distribution(outliers removed)")

ax2 = fig.add_subplot(122)

sns.boxplot(train_data.visitors, ax=ax2)
plt.title("Box plot of visitors data with no outliers")

plt.tight_layout()
```

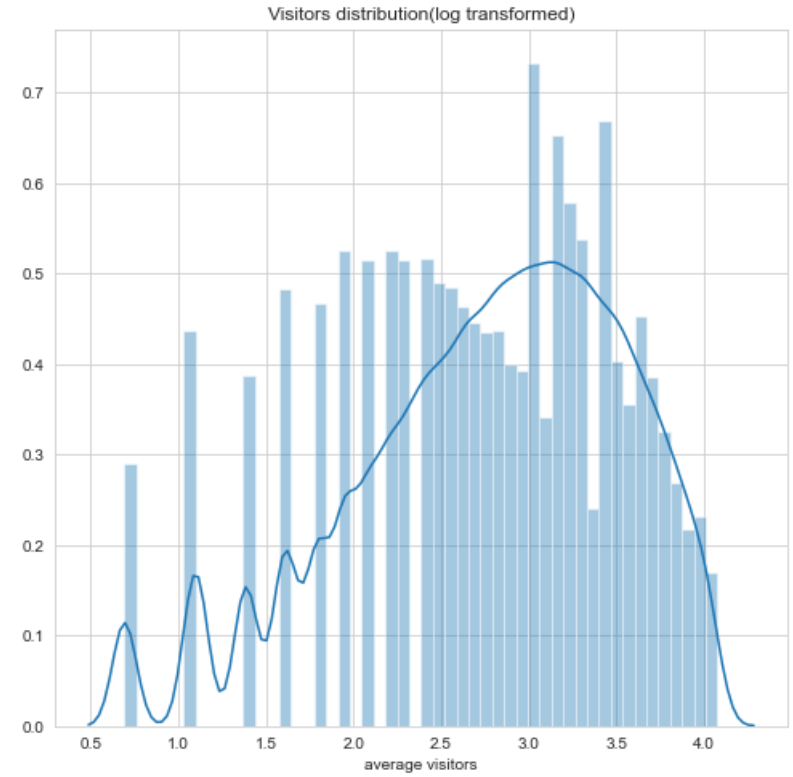
```
plt.show()
```



- **The above distribution is skewed as most of restaurnts have visitors within 20 on average**
- **we will use log transform to make it weakly stationary or weakly gaussian**

In [21]:

```
fig = plt.figure(figsize=(18, 8))

ax1 = fig.add_subplot(121)

sns.distplot(np.log1p(train_data.visitors), ax=ax1)
plt.title("Visitors distribution(log transformed)")
plt.xlabel("average visitors")
plt.show()
```



area wise restaurant count

- area wise restaurant count can be seen as a competition in the area

In [8]:

```
temp = train_data.groupby('air_area_name').air_store_id.count()
train_data = pd.merge(train_data, temp, how='left', on='air_area_name')
train_data.rename(columns={'air_store_id_x': 'air_store_id', 'air_store_id_y': 'area_competition'}, inp
lace=True)
```

In [9]:

```
train_data.head(3)
```

Out[9]:

	air_store_id	visit_date	visitors	day_of_week	holiday_flg	air_genre_name	air_area_name	latitude	longitude	reser
0	air_fab092c35776a9b1	2016-01-01	19	Friday	1	Cafe/Sweets	Fukuoka-ken Fukuoka-shi Momochi	33.581941	130.348436	
1	air_39dccf7df20b1c6a	2016-01-01	55	Friday	1	Izakaya	Hyōgo-ken Takarazuka-shi Tōyōchō	34.799767	135.360073	
2	air_79f528087f49df06	2016-01-01	42	Friday	1	Western food	Tōkyō-to Suginami-ku Asagayaminami	35.699566	139.636438	

area wise same genre restaurant count

- area wise same genre restaurant count can be seen as a competition in the area among same genre

In [9]:

```
temp = train_data.groupby(['air_area_name', 'air_genre_name'], as_index=False).air_store_id.count()
train_data = pd.merge(train_data, temp, how='left', on=['air_area_name', 'air_genre_name'])
train_data.rename(columns={'air_store_id_x': 'air_store_id', 'air_store_id_y': 'area_genre_competition'
}, inplace=True)
```

In [11]:

```
train_data.head(2)
```

Out[11]:

	air_store_id	visit_date	visitors	day_of_week	holiday_flg	air_genre_name	air_area_name	latitude	longitude	reser
0	air_fab092c35776a9b1	2016-01-01	19	Friday	1	Cafe/Sweets	Fukuoka-ken Fukuoka-shi Momochi	33.581941	130.348436	
1	air_39dccf7df20b1c6a	2016-01-01	55	Friday	1	Izakaya	Hyōgo-ken Takarazuka-shi Tōyōchō	34.799767	135.360073	

reservation gap visitor features

In [10]:

```
# get the reservations done on HPG
hpg_reserve = pd.merge(store_id_relation, hpg_reservation_data, how='left', on='hpg_store_id')
hpg_reserve.drop("hpg_store_id", axis=1, inplace=True)

# concat both AIR and HPG reservations
reservation_data = pd.concat([air_reservation_data, hpg_reserve])

# add the hour gap diff between reseravtion time as a feature
```

```
# add the hour gap data between reservation time as a feature
reservation_data.visit_datetime = pd.to_datetime(reservation_data.visit_datetime)
reservation_data.reserve_datetime = pd.to_datetime(reservation_data.reserve_datetime)
reservation_data['reservation_gap'] = reservation_data.visit_datetime - reservation_data.reserve_dateti
me
reservation_data['reservation_gap'] = reservation_data.reservation_gap / np.timedelta64(1,'h')


# encode the visitors based on the reservation gap
# 6th place solutuon features
# if reservation gap is under 12 hr
reservation_data['reserve_visitor_lt_12hr'] = np.where(reservation_data.reservation_gap < 12, reservati
on_data.reserve_visitors, 0)

# if reservation gap is between 12-36 hr
reservation_data['reserve_visitor_bt_12_36'] = np.where((reservation_data.reservation_gap >= 12) & (res
ervation_data.reservation_gap < 36),
                                                        reservation_data.reserve_visitors, 0)

# if reservation gap is between 37-59 hr
reservation_data['reserve_visitor_bt_36_59'] = np.where((reservation_data.reservation_gap >= 36) & (res
ervation_data.reservation_gap < 59),
                                                        reservation_data.reserve_visitors, 0)

# if reservation gap is between 59-85 hr
reservation_data['reserve_visitor_bt_59_85'] = np.where((reservation_data.reservation_gap >= 59) & (res
ervation_data.reservation_gap < 85),
                                                        reservation_data.reserve_visitors, 0)

# if reservation gap is greater 85 hr
reservation_data['reserve_visitor_gt_85'] = np.where(reservation_data.reservation_gap >= 85, reservatio
n_data.reserve_visitors, 0)
reservation_data['visit_date'] = reservation_data.visit_datetime.dt.date

# group by per store on visit date = reservation visitors
reservation_features = reservation_data.groupby(['air_store_id', 'visit_date'], as_index=False)[['reser
ve_visitors', 'reserve_visitor_lt_12hr',
                                                                                              'reserv
e_visitor_bt_12_36', 'reserve_visitor_bt_36_59',
                                                                                              'reserv
e_visitor_bt_59_85', 'reserve_visitor_gt_85']].sum()


# merge the reservation features with train data per store on visit date
train_data.drop(['reserve_visitors_air', 'reserve_visitors_hpg', 'reservation_gap', 'hpg_reserve_gap'],
axis=1, inplace=True)
reservation_features.visit_date = pd.to_datetime(reservation_features.visit_date)
train_data = pd.merge(train_data, reservation_features, how='left', on=['air_store_id', 'visit_date'])
train_data.fillna(0, inplace=True)
train_data.head()
```

Out[10]:

	air_store_id	visit_date	visitors	day_of_week	holiday_flg	air_genre_name	air_area_name	latitude	longitude	area
0	air_fab092c35776a9b1	2016-01-01	19	Friday	1	Cafe/Sweets	Fukuoka-ken Fukuoka-shi Momochi	33.581941	130.348436	
1	air_39dccf7df20b1c6a	2016-01-01	55	Friday	1	Izakaya	Hyōgo-ken Takarazuka-shi Tōyōchō	34.799767	135.360073	
2	air_79f528087f49df06	2016-01-01	42	Friday	1	Western food	Tōkyō-to Suginami-ku Asagayaminami	35.699566	139.636438	
3	air_08ba8cd01b3ba010	2016-01-01	11	Friday	1	Izakaya	Miyagi-ken Sendai-shi Kamisugi	38.269076	140.870403	
4	air_81c5dff692063446	2016-01-01	7	Friday	1	Dining bar	Ōsaka-fu Ōsaka-shi Ōgimachi	34.705362	135.510025	

In [16]:

```
# log transform the reservation featues
```

```
reservation_columns = reservation_features.columns[2:]
for column in reservation_columns:
    train_data[column] = train_data[column].apply(lambda a: np.log1p(a))

train_data.head()
```

Out[16]:

	air_store_id	visit_date	visitors	day_of_week	holiday_flg	air_genre_name	air_area_name	latitude	longitude	area
0	air_fab092c35776a9b1	2016-01-01	19	Friday	1	Cafe/Sweets	Fukuoka-ken Fukuoka-shi Momochi	33.581941	130.348436	
1	air_39dccf7df20b1c6a	2016-01-01	55	Friday	1	Izakaya	Hyōgo-ken Takarazuka-shi Tōyōchō	34.799767	135.360073	
2	air_79f528087f49df06	2016-01-01	42	Friday	1	Western food	Tōkyō-to Suginami-ku Asagayaminami	35.699566	139.636438	
3	air_08ba8cd01b3ba010	2016-01-01	11	Friday	1	Izakaya	Miyagi-ken Sendai-shi Kamisugi	38.269076	140.870403	
4	air_81c5dff692063446	2016-01-01	7	Friday	1	Dining bar	Ōsaka-fu Ōsaka-shi Ōgimachi	34.705362	135.510025	

Basic statistics features on visitors

In [18]:

```
# log transform the visitors variable:
train_data.visitors = train_data.visitors.apply(lambda x: np.log1p(x))

# average visitors per store on holiday, weekends, and working
# if day is working then 1,  else 0
train_data['working'] = np.where((train_data.holiday_flg == 0) & (train_data.day_of_week.isin(['Monday'
, 'Tuesday', 'Wednesday', 'Thursday', 'Friday'])), 1, 0)

# mean visitor to the restaurant based on day and if its working
mean_visitor_working = train_data.groupby(['air_store_id', 'day_of_week', 'working'], as_index=False).v
isitors.mean()
train_data = pd.merge(train_data, mean_visitor_working, how='left', on=['air_store_id', 'day_of_week',
'working'])
train_data.rename(columns={'visitors_x': 'visitors', 'visitors_y': 'mean_visitors'}, inplace=True)

# Median visitor to the restaurant based on day and if its working
median_visitor_working = train_data.groupby(['air_store_id', 'day_of_week', 'working'], as_index=False)
.visitors.median()
train_data = pd.merge(train_data, median_visitor_working, how='left', on=['air_store_id', 'day_of_week'
, 'working'])
train_data.rename(columns={'visitors_x': 'visitors', 'visitors_y': 'median_visitors'}, inplace=True)

# minimum visitors on restaurant based on day and if its working
min_visitor_working = train_data.groupby(['air_store_id', 'day_of_week', 'working'], as_index=False).vi
sitors.min()
train_data = pd.merge(train_data, min_visitor_working, how='left', on=['air_store_id', 'day_of_week', '
working'])
train_data.rename(columns={'visitors_x': 'visitors', 'visitors_y': 'min_visitors'}, inplace=True)

# maximum visitors on restaurant based on day and if its working
max_visitor_working = train_data.groupby(['air_store_id', 'day_of_week', 'working'], as_index=False).vi
sitors.max()
train_data = pd.merge(train_data, max_visitor_working, how='left', on=['air_store_id', 'day_of_week', '
working'])
train_data.rename(columns={'visitors_x': 'visitors', 'visitors_y': 'max_visitors'}, inplace=True)
```

In [19]:

```
train_data.head(2)
```

Out[19]:

	air_store_id	visit_date	visitors	day_of_week	holiday_flg	air_genre_name	air_area_name	latitude	longitude	area
0	air_fab092c35776a9b1	2016-01-01	2.995732	Friday	1	Cafe/Sweets	Fukuoka-ken Fukuoka-shi Momochi	33.581941	130.348436	
1	air_39dccf7df20b1c6a	2016-01-01	4.025352	Friday	1	Izakaya	Hyōgo-ken Takarazuka-shi Tōyōchō	34.799767	135.360073	

2 rows × 22 columns

expected visitors based on the genre in the area

In [20]:

```
# get total visitors in the area per genre
area_genre_total_visitors = train_data.groupby(['air_area_name', 'air_genre_name'], as_index=False).visitors.sum()
train_data = pd.merge(train_data, area_genre_total_visitors, how='left', on=['air_area_name', 'air_genre_name'])
train_data.rename(columns={'visitors_x': 'visitors', 'visitors_y': 'total_area_genre_visitors'}, inplace=True)

# expected visitors in the area of a genre
train_data['area_genre_expected_visitors'] = train_data.total_area_genre_visitors / train_data.area_genre_competition
train_data.drop('total_area_genre_visitors', axis=1, inplace=True)
train_data.head()
```

Out[20]:

	air_store_id	visit_date	visitors	day_of_week	holiday_flg	air_genre_name	air_area_name	latitude	longitude	are
0	air_fab092c35776a9b1	2016-01-01	2.995732	Friday	1	Cafe/Sweets	Fukuoka-ken Fukuoka-shi Momochi	33.581941	130.348436	
1	air_39dccf7df20b1c6a	2016-01-01	4.025352	Friday	1	Izakaya	Hyōgo-ken Takarazuka-shi Tōyōchō	34.799767	135.360073	
2	air_79f528087f49df06	2016-01-01	3.761200	Friday	1	Western food	Tōkyō-to Suginami-ku Asagayaminami	35.699566	139.636438	
3	air_08ba8cd01b3ba010	2016-01-01	2.484907	Friday	1	Izakaya	Miyagi-ken Sendai-shi Kamisugi	38.269076	140.870403	
4	air_81c5dff692063446	2016-01-01	2.079442	Friday	1	Dining bar	Ōsaka-fu Ōsaka-shi Ōgimachi	34.705362	135.510025	

5 rows × 23 columns

expected visitors based on the area

In [21]:

```
# get total visitors in the area
area_total_visitors = train_data.groupby(['air_area_name'], as_index=False).visitors.sum()
train_data = pd.merge(train_data, area_total_visitors, how='left', on=['air_area_name'])
train_data.rename(columns={'visitors_x': 'visitors', 'visitors_y': 'total_area_visitors'}, inplace=True
)

# expected visitors in the area
train_data['area_expected_visitors'] = train_data.total_area_visitors / train_data.area_competition
train_data.drop('total_area_visitors', axis=1, inplace=True)
train_data.head(3)
```

Out[21]:

	air_store_id	visit_date	visitors	day_of_week	holiday_flg	air_genre_name	air_area_name	latitude	longitude	are
0	air_fab092c35776a9b1	2016-01-01	2.995732	Friday	1	Cafe/Sweets	Fukuoka-ken Fukuoka-shi Momochi	33.581941	130.348436	
1	air_39dccf7df20b1c6a	2016-01-01	4.025352	Friday	1	Izakaya	Hyōgo-ken Takarazuka-shi Tōyōchō	34.799767	135.360073	
2	air_79f528087f49df06	2016-01-01	3.761200	Friday	1	Western food	Tōkyō-to Suginami-ku Asagayaminami	35.699566	139.636438	

3 rows × 24 columns

Date features

In [22]:

```
train_data['year'] = train_data.visit_date.dt.year
train_data['month'] = train_data.visit_date.dt.month
train_data['day_of_month'] = train_data.visit_date.dt.day
```

In [18]:

```
train_data.head(2)
```

Out[18]:

	air_store_id	visit_date	visitors	day_of_week	holiday_flg	air_genre_name	air_area_name	latitude	longitude	area_
0	air_fab092c35776a9b1	2016-01-01	19	Friday	1	Cafe/Sweets	Fukuoka-ken Fukuoka-shi Momochi	33.581941	130.348436	
1	air_39dccf7df20b1c6a	2016-01-01	55	Friday	1	Izakaya	Hyōgo-ken Takarazuka-shi Tōyōchō	34.799767	135.360073	

2 rows × 27 columns

No. of reservations on the visit date

In [19]:

```
reservation_count_feature = reservation_data.groupby(['air_store_id', 'visit_date'], as_index=False).visit_datetime.count()
reservation_count_feature.visit_date = pd.to_datetime(reservation_count_feature.visit_date)
train_data = pd.merge(train_data, reservation_count_feature, how='left', on=['air_store_id', 'visit_date'])
train_data.rename(columns={'visit_datetime': 'reservation_count'}, inplace=True)
train_data.fillna(0, inplace=True)
```

In [20]:

```
train_data.head(2)
```

Out[20]:

	air_store_id	visit_date	visitors	day_of_week	holiday_flg	air_genre_name	air_area_name	latitude	longitude	area_
0	air_fab092c35776a9b1	2016-01-01	19	Friday	1	Cafe/Sweets	Fukuoka-ken Fukuoka-shi Momochi	33.581941	130.348436	

	air_store_id	visit_date	visitors	day_of_week	holiday_flg	air_genre_name	air_area_name	latitude	longitude	area_
1	air_39dccf7df20b1c6a	2016-01-01	55	Friday	1	Izakaya	Hyōgo-ken Takarazuka-shi Tōyōchō	34.799767	135.360073	

2 rows × 28 columns

prefecture: Japan has 47 prefectures (hierarchy in area names)

In [23]:

```
#prefecture
train_data['area_prefecture'] = [area.split()[0] for area in train_data['air_area_name'].values]

# sub prefecture
train_data['area_sub_pref'] = [area.split()[1] for area in train_data['air_area_name'].values]

train_data.head()
```

Out[23]:

	air_store_id	visit_date	visitors	day_of_week	holiday_flg	air_genre_name	air_area_name	latitude	longitude	are
0	air_fab092c35776a9b1	2016-01-01	2.995732	Friday	1	Cafe/Sweets	Fukuoka-ken Fukuoka-shi Momochi	33.581941	130.348436	
1	air_39dccf7df20b1c6a	2016-01-01	4.025352	Friday	1	Izakaya	Hyōgo-ken Takarazuka-shi Tōyōchō	34.799767	135.360073	
2	air_79f528087f49df06	2016-01-01	3.761200	Friday	1	Western food	Tōkyō-to Suginami-ku Asagayaminami	35.699566	139.636438	
3	air_08ba8cd01b3ba010	2016-01-01	2.484907	Friday	1	Izakaya	Miyagi-ken Sendai-shi Kamisugi	38.269076	140.870403	
4	air_81c5dff692063446	2016-01-01	2.079442	Friday	1	Dining bar	Ōsaka-fu Ōsaka-shi Ōgimachi	34.705362	135.510025	

5 rows × 29 columns

window statistics features: restaurant wise monthly visitors

In [24]:

```
# get the month wise mean visitors to restaurant
temp = train_data.groupby(['air_store_id', 'month'], as_index=False).visitors.mean()
train_data = pd.merge(train_data, temp, how='left', on=['air_store_id', 'month'])
train_data.rename(columns={'visitors_x': 'visitors', 'visitors_y': 'monthly_mean_visitors'}, inplace=Tr
ue)

# get the month wise median visitors to restaurant
temp = train_data.groupby(['air_store_id', 'month'], as_index=False).visitors.median()
train_data = pd.merge(train_data, temp, how='left', on=['air_store_id', 'month'])
train_data.rename(columns={'visitors_x': 'visitors', 'visitors_y': 'monthly_median_visitor'}, inplace=T
rue)

# get the month wise min visitors to restaurant
temp = train_data.groupby(['air_store_id', 'month'], as_index=False).visitors.min()
train_data = pd.merge(train_data, temp, how='left', on=['air_store_id', 'month'])
train_data.rename(columns={'visitors_x': 'visitors', 'visitors_y': 'monthly_minimum_visitor'}, inplace=
True)

# get the month wise max visitors to restaurant
temp = train_data.groupby(['air_store_id', 'month'], as_index=False).visitors.max()
train_data = pd.merge(train_data, temp, how='left', on=['air_store_id', 'month'])
train_data.rename(columns={'visitors_x': 'visitors', 'visitors_y': 'monthly_max_visitor'}, inplace=True
)
```

window statistics features: restaurant genre wise monthly statistics

In [25]:

```
# get the month wise mean visitors per genre
temp = train_data.groupby(['air_genre_name', 'month'], as_index=False).visitors.mean()
train_data = pd.merge(train_data, temp, how='left', on=['air_genre_name', 'month'])
train_data.rename(columns={'visitors_x': 'visitors', 'visitors_y': 'monthly_genre_mean_visitors'}, inpl
ace=True)

# get the month wise median visitors per genre
temp = train_data.groupby(['air_genre_name', 'month'], as_index=False).visitors.median()
train_data = pd.merge(train_data, temp, how='left', on=['air_genre_name', 'month'])
train_data.rename(columns={'visitors_x': 'visitors', 'visitors_y': 'monthly_genre_median_visitor'}, inp
lace=True)

# get the month wise min visitors per genre
temp = train_data.groupby(['air_genre_name', 'month'], as_index=False).visitors.min()
train_data = pd.merge(train_data, temp, how='left', on=['air_genre_name', 'month'])
train_data.rename(columns={'visitors_x': 'visitors', 'visitors_y': 'monthly_genre_minimum_visitor'}, in
place=True)

# get the month wise max visitors per genre
temp = train_data.groupby(['air_genre_name', 'month'], as_index=False).visitors.max()
train_data = pd.merge(train_data, temp, how='left', on=['air_genre_name', 'month'])
train_data.rename(columns={'visitors_x': 'visitors', 'visitors_y': 'monthly_genre_max_visitor'}, inplac
e=True)
```

weather data for the restaurants:

- The local weather has also affect on the number of visitors, Hence we will get the local japanese weather stations data for the restaurants from Weather Data for Recruit Restaurant Competition
- we will use the temperature and precipatation, since the rest contain nan values exceedingly

In [26]:

```
# read weather data
weather_data = os.path.join(data_path, "weather_data")
weather_station_data = os.path.join(weather_data, "1-1-16_5-31-17_Weather")

# check if processed file exists
if not os.path.isfile(processed_data_path + "/processed_air_store_weather_info.csv"):

    # prepare weather_data
    air_store_near_station = pd.read_csv(weather_data + "/air_store_info_with_nearest_active_station.cs
v")
    air_store_near_station = air_store_near_station.loc[:,['air_store_id', 'station_id']]
    air_store_near_station = air_store_near_station.set_index('air_store_id')

    # fetch weather data from each station for air_store
    air_store_weather_info = pd.DataFrame(columns=['calendar_date', 'avg_temperature',
                                                   'low_temperature', 'high_temperature',
                                                   'precipitation'])
    for index, row in air_store_near_station.iterrows():
        weather_file = os.path.join(weather_station_data, row.station_id + ".csv")
        temp_weather = pd.read_csv(weather_file).loc[:, ['calendar_date', 'avg_temperature',
                                                   'low_temperature', 'high_temperature',
                                                   'precipitation']]
        temp_weather.fillna(0, inplace=True)
        temp_weather['air_store_id'] = index

        air_store_weather_info = pd.concat([air_store_weather_info, temp_weather])

    # rearrage the columns
    air_store_weather_info = air_store_weather_info.loc[:,['air_store_id','calendar_date', 'avg_tempera
ture',
                                                   'low_temperature', 'high_temperature',
                                                   'precipitation']].reset_index(drop=True)
    # save the processed file
    air_store_weather_info.to_csv(processed_data_path + "/processed_air_store_weather_info.csv", index=
False)
```

```
else:

    air_store_weather_info = pd.read_csv(processed_data_path + "/processed_air_store_weather_info.csv")
```

In [27]:

```
air_store_weather_info.calendar_date = pd.to_datetime(air_store_weather_info.calendar_date)
air_store_weather_info.rename(columns={'calendar_date': 'visit_date'}, inplace=True)
train_data = pd.merge(train_data, air_store_weather_info, how='left', on=['air_store_id', 'visit_date']
)
```

In [28]:

```
train_data.head()
```

Out[28]:

	air_store_id	visit_date	visitors	day_of_week	holiday_flg	air_genre_name	air_area_name	latitude	longitude	are
0	air_fab092c35776a9b1	2016-01-01	2.995732	Friday	1	Cafe/Sweets	Fukuoka-ken Fukuoka-shi Momochi	33.581941	130.348436	
1	air_39dccf7df20b1c6a	2016-01-01	4.025352	Friday	1	Izakaya	Hyōgo-ken Takarazuka-shi Tōyōchō	34.799767	135.360073	
2	air_79f528087f49df06	2016-01-01	3.761200	Friday	1	Western food	Tōkyō-to Suginami-ku Asagayaminami	35.699566	139.636438	
3	air_08ba8cd01b3ba010	2016-01-01	2.484907	Friday	1	Izakaya	Miyagi-ken Sendai-shi Kamisugi	38.269076	140.870403	
4	air_81c5dff692063446	2016-01-01	2.079442	Friday	1	Dining bar	Ōsaka-fu Ōsaka-shi Ōgimachi	34.705362	135.510025	

5 rows × 41 columns

In [29]:

```
train_data.to_csv(processed_data_path + "/train_data_FE_v1.csv", index=False)
```

Feature Engineering - Summary:

1. **area wise restaurant count:** *This features gives the total resturants in the restaurant area.*
2. **area wise same genre restaurant count:** *This feature gives the total same genre count in the restaurant area. This describes the peer competition in the area*
3. **reserve_visitors:** *This feature gives nb of visitors arriving to the restaurant through reservation*
4. **working:** *This feature describes if its a working day or holiday(holiday flg or weekend).*
5. **mean_visitors:** *The average visitors expected for the restaurant on the day of week (working and Non-working)*
6. **median_visitors:** *The median visitors expected for the restaurant on the day of week (working and Non-working)*
7. **min_visitors:** *The minimum visitors expected for the restaurant on the day of week (working and Non-working)*
8. **max_visitors:** *The maximum visitors expected for the restaurant on the day of week (working and Non-working)*
9. **area wise expected visitors:** *This feature describes the nb of expected visitors share given total nb of resturants in the area*
10. **area wise genre expected visitors:** *This feature describes the nb of expected visitors share given total nb of resturants of the same genre in the area*
11. **area_prefecture**: *This are main Japan area_prefecture. Japan has 47 prefectures. "Fukuoka-ken Fukuoka-shi Momochi" where Fukuoka-ken is main prefecture*
12. **area sub prefecture:** *These are sub prefecture under main prefectures*
13. **monthly mean visitors:** *This feature descibes the mean visitors to the restaurant in time window of every month*
14. **monthly median visitors:** *This feature descibes the median visitors to the restaurant in time window of every month*
15. **monthly min visitors:** *This feature descibes the minimum visitors to the restaurant in time window of every month*
16. **monthly max visitors:** *This feature descibes the maximum visitors to the restaurant in time window of every month*
17. **average temperature**: *This feature describes the average temperature in the restaurant area from the nearest weather station*

18. **low temperature**: *This feature describes the lowest temperature in the restaurant area from the nearest weather station*
19. **high temperature**: *This feature describes the highest temperature in the restaurant area from the nearest weather station*
20. **precipitation:** *This feature descibes the possibility of rain in the restaurant area from the nearest weather station*