Capstone Project 1: MuscleHub AB Test

Step 1: Get started with SQL

Like most businesses, Janet keeps her data in a SQL database. Normally, you'd download the data from her database to a csv file, and then load it into a Jupyter Notebook using Pandas.

For this project, you'll have to access SQL in a slightly different way. You'll be using a special Codecademy library that lets you type SQL queries directly into this Jupyter notebook. You'll have pass each SQL query as an argument to a function called `sql_query`. Each query will return a Pandas DataFrame. Here's an example:

In [2]:
```
# This import only needs to happen once, at the beginning of the notebook
from codecademySQL import sql_query
```

In [13]:
```
# Here's an example of a query that just displays some data
sql_query('''
SELECT *
FROM visits
LIMIT 5
''')
```

Out[13]:

	index	first_name	last_name	email	gender	visit_date
0	0				female	5-1-17
1	1				female	5-1-17
2	2				male	5-1-17
3	3				female	5-1-17
4	4				male	5-1-17

In [14]:
```
# Here's an example where we save the data to a DataFrame
df = sql_query('''
SELECT *
FROM applications
LIMIT 5
''')
```

Step 2: Get your dataset

Let's get started!

Janet of MuscleHub has a SQLite database, which contains several tables that will be helpful to you in this investigation:

- `visits` contains information about potential gym customers who have visited MuscleHub
- `fitness_tests` contains information about potential customers in "Group A", who were given a fitness test
- `applications` contains information about any potential customers (both "Group A" and "Group B") who filled out an application. Not everyone in `visits` will have filled out an application.
- `purchases` contains information about customers who purchased a membership to MuscleHub.

Use the space below to examine each table.

In [17]:
```
# Examine visits here

sql_query('SELECT * FROM visits LIMIT 5 ')
```

Out[17]:

	index	first_name	last_name	email	gender	visit_date
0	0				female	5-1-17
1	1				female	5-1-17
2	2				male	5-1-17
3	3				female	5-1-17
4	4				male	5-1-17

In [18]:
```
# Examine fitness_tests here

sql_query('SELECT * FROM fitness_tests LIMIT 5 ')
```

Out[18]:

	index	first_name	last_name	email	gender	fitness_test_date
0	0				female	2017-07-03
1	1				male	2017-07-02
2	2				male	2017-07-01
3	3				female	2017-07-02
4	4				female	2017-07-05

In [19]:
```
# Examine applications here

sql_query('SELECT * FROM applications LIMIT 5 ')
```

Out[19]:

	index	first_name	last_name	email	gender	application_date
0	0				male	2017-08-12
1	1				female	2017-09-29
2	2				female	2017-09-15
3	3				male	2017-07-26
4	4				male	2017-07-14

In [20]:
```
# Examine purchases here

sql_query('SELECT * FROM purchases LIMIT 5 ')
```

Out[20]:

	index	first_name	last_name	email	gender	purchase_date
0	0				male	2017-08-18
1	1				female	2017-09-16
2	2				male	2017-07-20
3	3				male	2017-07-27
4	4				female	2017-08-24

We'd like to download a giant DataFrame containing all of this data. You'll need to write a query that does the following things:

1. Not all visits in `visits` occurred during the A/B test. You'll only want to pull data where `visit_date` is on or after `7-1-17` .
2. You'll want to perform a series of `LEFT JOIN` commands to combine the four tables that we care about. You'll need to perform the joins on `first_name` , `last_name` , and `email` . Pull the following columns:

- `visits.first_name`
- `visits.last_name`
- `visits.gender`
- `visits.email`
- `visits.visit_date`
- `fitness_tests.fitness_test_date`
- `applications.application_date`
- `purchases.purchase_date`

Save the result of this query to a variable called `df` .

Hint: your result should have 5004 rows. Does it?

In [3]:
```
df = sql_query('''WITH temp1 AS (SELECT visits.first_name, visits.last_name, v
isits.gender, visits.email, visits.visit_date,
                        fitness_tests.fitness_test_date FROM visits LEFT OUTER
JOIN fitness_tests ON
                        visits.first_name = fitness_tests.first_name AND visit
s.last_name = fitness_tests.last_name
                        AND visits.email = fitness_tests.email WHERE visits.vi
sit_date >= '7-1-17'),
                       temp2 AS (SELECT temp1.first_name, temp1.last_name, tem
p1.gender, temp1.email, temp1.visit_date,
                        temp1.fitness_test_date , applications.application_dat
e FROM temp1 LEFT OUTER JOIN applications
                        ON temp1.first_name = applications.first_name AND temp
1.last_name = applications.last_name
                        AND temp1.email = applications.email)
                  SELECT temp2.first_name, temp2.last_name, temp2.gender, temp
2.email, temp2.visit_date,
                         temp2.fitness_test_date, temp2.application_date, purc
hases.purchase_date
                         FROM temp2 LEFT OUTER JOIN purchases
                         ON temp2.first_name = purchases.first_name AND temp2.
last_name = purchases.last_name
                         AND temp2.email = purchases.email

''' )
```

Step 3: Investigate the A and B groups

We have some data to work with! Import the following modules so that we can start doing analysis:

- `import pandas as pd`
- `from matplotlib import pyplot as plt`

In [4]:
```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
```

We're going to add some columns to `df` to help us with our analysis.

Start by adding a column called `ab_test_group` . It should be `A` if `fitness_test_date` is not `None` , and `B` if `fitness_test_date` is `None` .

In [5]:
```
df['ab_test_group'] = df.fitness_test_date.apply(lambda x: 'A' if x != None el
se 'B')

#print(df.head(10))
```

Let's do a quick sanity check that Janet split her visitors such that about half are in A and half are in B.

Start by using `groupby` to count how many users are in each `ab_test_group` . Save the results to `ab_counts` .

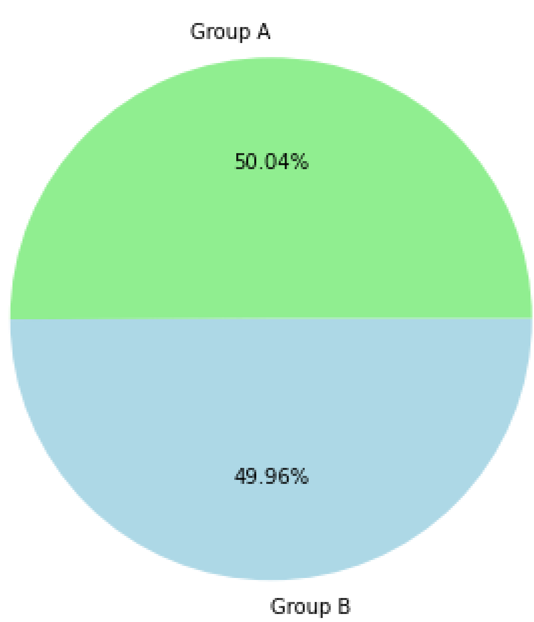
In [6]:
```
ab_counts = df.groupby('ab_test_group').email.count().reset_index()
ab_counts.rename(columns = {'ab_test_group':'Test_Group', 'email':'Count'}, in
place = True)

print(ab_counts)
```

```
  Test_Group  Count
0          A   2504
1          B   2500
```

We'll want to include this information in our presentation. Let's create a pie cart using `plt.pie` . Make sure to include:

- Use `plt.axis('equal')` so that your pie chart looks nice
- Add a legend labeling `A` and `B`
- Use `autopct` to label the percentage of each group
- Save your figure as `ab_test_pie_chart.png`

In [7]:
```
plt.figure(figsize=(6,6))
plt.pie(ab_counts.Count, labels = ['Group A', 'Group B'], autopct = '%0.2f%%',
colors = ['lightgreen', 'lightblue'])
plt.savefig('ab_test_pie_chart.png')
plt.show()
```



Step 4: Who picks up an application?

Recall that the sign-up process for MuscleHub has several steps:

1. Take a fitness test with a personal trainer (only Group A)
2. Fill out an application for the gym
3. Send in their payment for their first month's membership

Let's examine how many people make it to Step 2, filling out an application.

Start by creating a new column in `df` called `is_application` which is `Application` if `application_date` is not `None` and `No Application` , otherwise.

In [8]:
```
df['is_application'] = df.application_date.apply(lambda app: 'Application' if
app != None else 'No Application')
```

Now, using `groupby` , count how many people from Group A and Group B either do or don't pick up an application. You'll want to group by `ab_test_group` and `is_application` . Save this new DataFrame as `app_counts`

In [9]:
```
app_counts = df.groupby(['ab_test_group','is_application']).email.count().rese
t_index()

print(app_counts)
```

```
  ab_test_group  is_application  email
0             A     Application    250
1             A  No Application   2254
2             B     Application    325
3             B  No Application   2175
```

We're going to want to calculate the percent of people in each group who complete an application. It's going to be much easier to do this if we pivot `app_counts` such that:

- The `index` is `ab_test_group`
- The `columns` are `is_application` Perform this pivot and save it to the variable `app_pivot` . Remember to call `reset_index()` at the end of the pivot!

In [16]:
```
app_pivot = app_counts.pivot(columns = 'is_application', index = 'ab_test_grou
p', values = 'email').reset_index()
app_pivot.rename(columns = {'ab_test_group':'Test_Group'}, inplace = True)

print(app_pivot)
```

```
is_application Test_Group  Application  No Application
0                       A          250            2254
1                       B          325            2175
```

Define a new column called `Total` , which is the sum of `Application` and `No Application` .

In [17]:
```
app_pivot['Total'] = app_pivot.Application + app_pivot['No Application']
```

Calculate another column called `Percent with Application` , which is equal to `Application` divided by `Total` .

In [45]:
```
app_pivot['Percent with Application'] = round(100.0* app_pivot.Application / a
pp_pivot.Total,2)

print(app_pivot)
```

```
is_application Test_Group  Application  No Application  Total  \
0                       A          250            2254   2504
1                       B          325            2175   2500

is_application  Percent with Application
0                                   9.98
1                                  13.00
```

It looks like more people from Group B turned in an application. Why might that be?

We need to know if this difference is statistically significant.

Choose a hypothesis tests, import it from `scipy` and perform it. Be sure to note the p-value. Is this result significant?

In [48]:
```
contingency1 = [[250,2253],[325,2175]]

from scipy.stats import chi2_contingency

chi2_1, pvalue1, dof_1, expected_1 = chi2_contingency(contingency1)

print(pvalue1)
```

```
0.000982285498767234
```

Step 4: Who purchases a membership?

Of those who picked up an application, how many purchased a membership?

Let's begin by adding a column to `df` called `is_member` which is `Member` if `purchase_date` is not `None` , and `Not Member` otherwise.

In [26]:
```
df['is_member'] = df.purchase_date.apply(lambda pday: 'Member' if pday != None
else 'Not Member')
```

Now, let's create a DataFrame called `just_apps` the contains only people who picked up an application.

In [32]:
```
just_apps = df[df.is_application == 'Application']
just_apps.reset_index(drop = True, inplace = True)

#print(just_apps.head())
```

Great! Now, let's do a `groupby` to find out how many people in `just_apps` are and aren't members from each group. Follow the same process that we did in Step 4, including pivoting the data. You should end up with a DataFrame that looks like this:

is_member	ab_test_group	Member	Not Member	Total	Percent Purchase
0	A	?	?	?	?
1	B	?	?	?	?

Save your final DataFrame as `member_pivot` .

In [47]:
```
member_counts = just_apps.groupby(['ab_test_group','is_member']).email.count()
.reset_index()

member_pivot = member_counts.pivot(columns = 'is_member', index = 'ab_test_gro
up', values = 'email').reset_index()
member_pivot.rename(columns = {'ab_test_group':'Test_Group'}, inplace = True)

member_pivot['Total'] = member_pivot.Member + member_pivot['Not Member']
member_pivot['Percent Purchase'] = round(100.0*member_pivot.Member / member_pi
vot.Total,2)

print(member_pivot)
```

```
is_member Test_Group  Member  Not Member  Total  Percent Purchase
0                  A     200          50    250             80.00
1                  B     250          75    325             76.92
```

It looks like people who took the fitness test were more likely to purchase a membership **if** they picked up an application. Why might that be?

Just like before, we need to know if this difference is statistically significant. Choose a hypothesis tests, import it from `scipy` and perform it. Be sure to note the p-value. Is this result significant?

In [49]:
```
contingency2 = [[200,50],[250,75]]

chi2_2, pvalue2, dof_2, expected_2 = chi2_contingency(contingency2)

print(pvalue2)
```

```
0.43258646051083327
```

Previously, we looked at what percent of people **who picked up applications** purchased memberships. What we really care about is what percentage of **all visitors** purchased memberships. Return to `df` and do a `groupby` to find out how many people in `df` are and aren't members from each group. Follow the same process that we did in Step 4, including pivoting the data. You should end up with a DataFrame that looks like this:

is_member	ab_test_group	Member	Not Member	Total	Percent Purchase
0	A	?	?	?	?
1	B	?	?	?	?

Save your final DataFrame as `final_member_pivot` .

In [51]:
```
final_counts = df.groupby(['ab_test_group','is_member']).email.count().reset_i
ndex()

final_pivot = final_counts.pivot(columns = 'is_member', index = 'ab_test_grou
p', values = 'email').reset_index()
final_pivot.rename(columns = {'ab_test_group':'Test_Group'}, inplace = True)

final_pivot['Total'] = final_pivot.Member + final_pivot['Not Member']
final_pivot['Percent Purchase'] = round(100.0*final_pivot.Member / final_pivot
.Total,2)

print(final_pivot)
```

```
is_member Test_Group  Member  Not Member  Total  Percent Purchase
0                  A     200        2304   2504              7.99
1                  B     250        2250   2500             10.00
```

Previously, when we only considered people who had **already picked up an application**, we saw that there was no significant difference in membership between Group A and Group B.

Now, when we consider all people who **visit MuscleHub**, we see that there might be a significant different in memberships between Group A and Group B. Perform a significance test and check.

In [53]:
```
contingency3 = [[200,2304],[250,2250]]

chi2_3, pvalue3, dof_3, expected_3 = chi2_contingency(contingency3)

print(pvalue3)
```

```
0.014724114645783203
```

Step 5: Summarize the acquisition funel with a chart

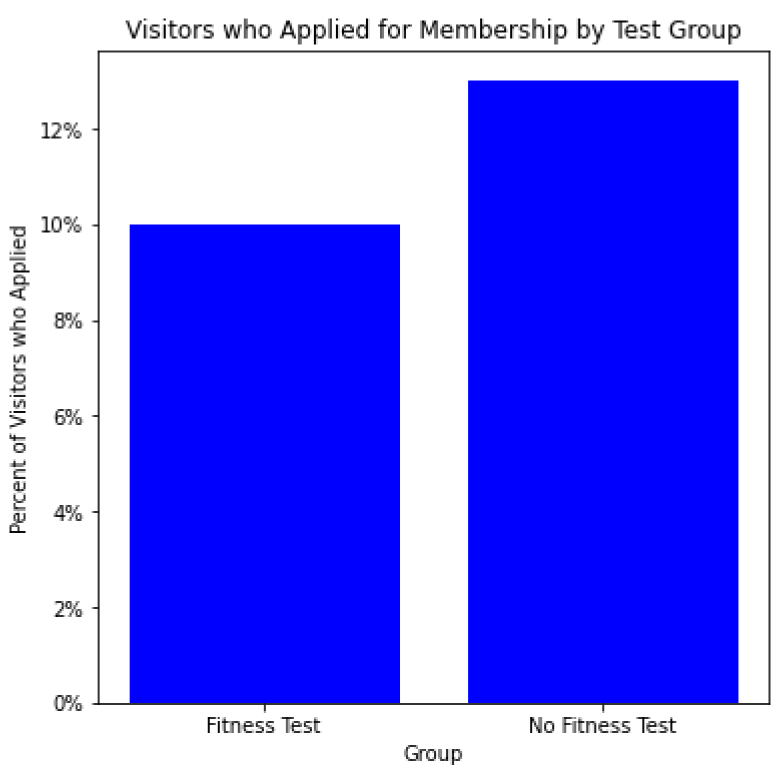
We'd like to make a bar chart for Janet that shows the difference between Group A (people who were given the fitness test) and Group B (people who were not given the fitness test) at each state of the process:

- Percent of visitors who apply
- Percent of applicants who purchase a membership
- Percent of visitors who purchase a membership

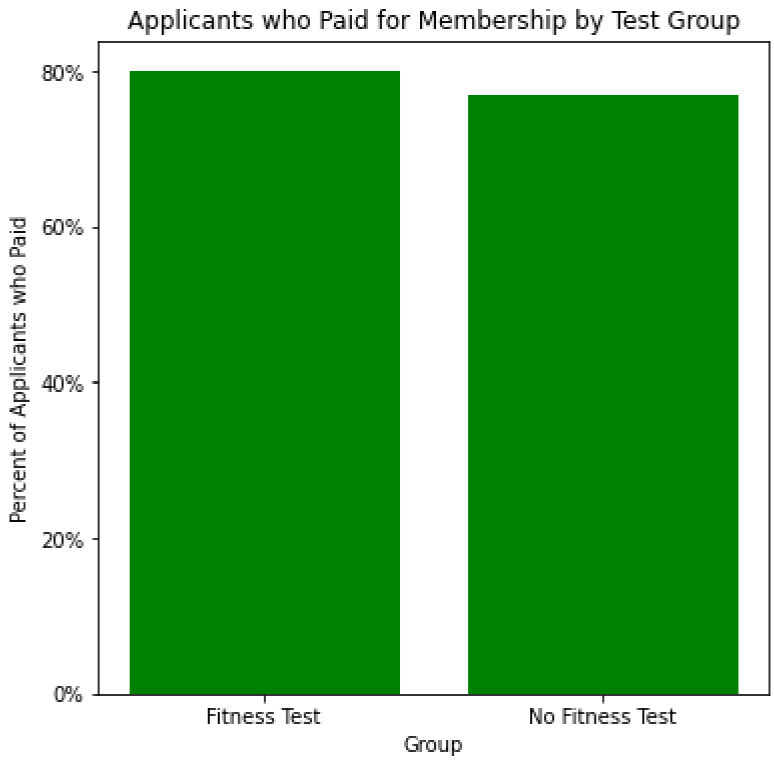
Create one plot for **each** of the three sets of percentages that you calculated in `app_pivot` , `member_pivot` and `final_member_pivot` . Each plot should:

- Label the two bars as `Fitness Test` and `No Fitness Test`
- Make sure that the y-axis ticks are expressed as percents (i.e., `5%`)
- Have a title

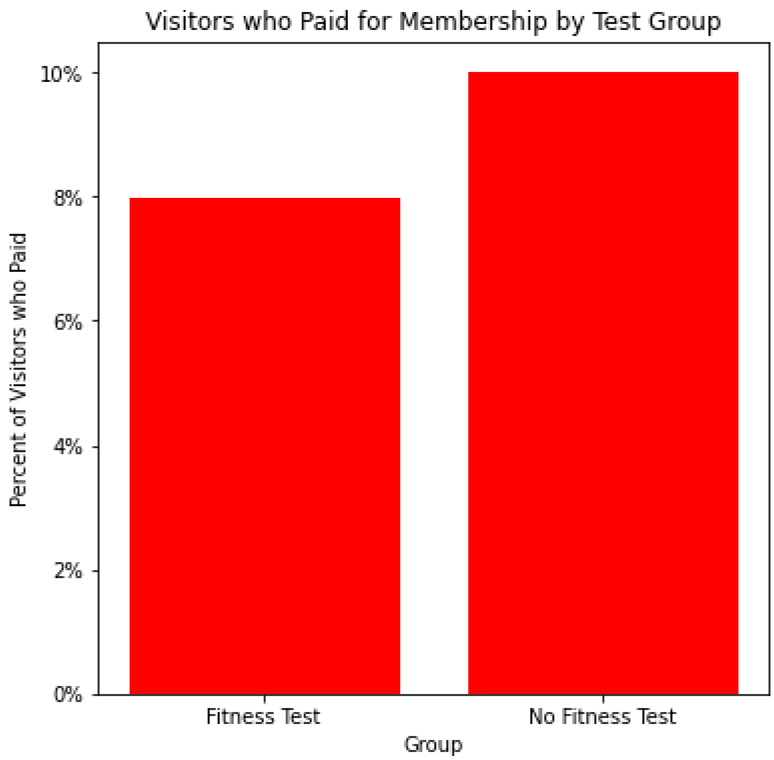
In [64]:
```
plt.figure(figsize=(6,6))
ax1=plt.subplot()
numofgroups=range(2)
plt.bar(numofgroups, app_pivot['Percent with Application'], color='blue')
ax1.set_xticks(numofgroups)
ax1.set_xticklabels(['Fitness Test','No Fitness Test'])
ax1.set_yticks([0,2,4,6,8,10,12])
ax1.set_yticklabels(['0%','2%','4%','6%','8%','10%','12%'])
plt.title('Visitors who Applied for Membership by Test Group')
plt.xlabel('Group')
plt.ylabel('Percent of Visitors who Applied')
plt.savefig('percent_application_bar_graph.png')
plt.show()
```



In [70]:
```
plt.figure(figsize=(6,6))
ax2=plt.subplot()
numofgroups=range(2)
plt.bar(numofgroups, member_pivot['Percent Purchase'], color='green')
ax2.set_xticks(numofgroups)
ax2.set_xticklabels(['Fitness Test','No Fitness Test'])
ax2.set_yticks([0,20,40,60,80])
ax2.set_yticklabels(['0%','20%','40%','60%','80%'])
plt.title('Applicants who Paid for Membership by Test Group')
plt.xlabel('Group')
plt.ylabel('Percent of Applicants who Paid')
plt.savefig('percent_purchase_given_app_bar_graph.png')
plt.show()
```



In [74]:
```
plt.figure(figsize=(6,6))
ax3=plt.subplot()
numofgroups=range(2)
plt.bar(numofgroups, final_pivot['Percent Purchase'], color='red')
ax3.set_xticks(numofgroups)
ax3.set_xticklabels(['Fitness Test','No Fitness Test'])
ax3.set_yticks([0,2,4,6,8,10])
ax3.set_yticklabels(['0%','2%','4%','6%','8%','10%'])
plt.title('Visitors who Paid for Membership by Test Group')
plt.xlabel('Group')
plt.ylabel('Percent of Visitors who Paid')
plt.savefig('percent_purchase_given_visit_bar_graph.png')
plt.show()
```



In [ ]: