MuscleHub A/B 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.

In [1]:
```
from codecademySQL import sql_query
```

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.

In [2]:
```
#examine visits
sql_query('''
SELECT *
FROM visits
LIMIT 5
''')
```

Out[2]:

	index	first_name	last_name	email	gender	visit_date
0	0	Karen	Manning	Karen.Manning@gmail.com	female	5-1-17
1	1	Annette	Boone	AB9982@gmail.com	female	5-1-17
2	2	Salvador	Merritt	SalvadorMerritt12@outlook.com	male	5-1-17
3	3	Martha	Maxwell	Martha.Maxwell@gmail.com	female	5-1-17
4	4	Andre	Mayer	AndreMayer90@gmail.com	male	5-1-17

In [3]:
```
#examine fitness_tests
sql_query('''
SELECT *
FROM fitness_tests
LIMIT 5
''')
```

Out[3]:
```
   index first_name last_name                  email  gender fitness_test_date
0      0        Kim    Walter   KimWalter58@gmail.com  female        2017-07-03
1      1        Tom   Webster      TW3857@gmail.com    male        2017-07-02
2      2     Marcus     Bauer  Marcus.Bauer@gmail.com    male        2017-07-01
3      3    Roberta      Best     RB6305@hotmail.com  female        2017-07-02
4      4     Carrie   Francis     CF1896@hotmail.com  female        2017-07-05
```

In [4]:
```
#examine applications
sql_query('''
SELECT *
FROM applications
LIMIT 5
''')
```

Out[4]:
```
   index first_name last_name                     email  gender application_date
0      0        Roy    Abbott     RoyAbbott32@gmail.com    male       2017-08-12
1      1      Agnes   Acevedo  AgnesAcevedo1@gmail.com  female       2017-09-29
2      2    Roberta   Acevedo         RA8063@gmail.com  female       2017-09-15
3      3     Darren    Acosta  DAcosta1996@hotmail.com    male       2017-07-26
4      4     Vernon    Acosta     VAcosta1975@gmail.com    male       2017-07-14
```

In [5]:
```
#examine purchases
sql_query('''
SELECT *
FROM purchases
LIMIT 5
''')
```

Out[5]:
```
   index first_name last_name                    email  gender purchase_date
0      0        Roy    Abbott    RoyAbbott32@gmail.com    male    2017-08-18
1      1    Roberta   Acevedo         RA8063@gmail.com  female    2017-09-16
2      2     Vernon    Acosta    VAcosta1975@gmail.com    male    2017-07-20
3      3     Darren    Acosta DAcosta1996@hotmail.com    male    2017-07-27
4      4       Dawn    Adkins   Dawn.Adkins@gmail.com  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 [6]:
```
df = sql_query('''
SELECT visits.first_name,
       visits.last_name,
       visits.gender,
       visits.email,
       visits.visit_date,
       fitness_tests.fitness_test_date,
       applications.application_date,
       purchases.purchase_date
FROM visits
LEFT JOIN fitness_tests
    ON visits.first_name = fitness_tests.first_name
    AND visits.last_name = fitness_tests.last_name
    AND visits.email = fitness_tests.email
LEFT JOIN applications
    ON visits.first_name = applications.first_name
    AND visits.last_name = applications.last_name
    AND visits.email = applications.email
LEFT JOIN purchases
    ON visits.first_name = purchases.first_name
    AND visits.last_name = purchases.last_name
    AND visits.email = purchases.email
WHERE visits.visit_date >= '7-1-17'
''')
df.tail()
#dataframe has 5004 total rows
```

Out[6]:

	first_name	last_name	gender	email	visit_date	fitness_test_date	applicati
4999	Rachel	Hensley	female	RachelHensley38@gmail.com	9-9-17	None	
5000	Leon	Harmon	male	Leon.Harmon@gmail.com	9-9-17	2017-09-15	

	first_name	last_name	gender	email	visit_date	fitness_test_date	applicati
5001	Andy	Pratt	male	AndyPratt27@gmail.com	9-9-17	2017-09-15	
5002	Ruben	Nielsen	male	RubenNielsen93@hotmail.com	9-9-17	None	20
5003	Charles	Carver	male	CC2490@gmail.com	9-9-17	2017-09-12	

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 [7]:
```
import pandas as pd
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 [8]:
```
df["ab_test_group"] = df.fitness_test_date.apply(lambda x: "A" if pd.notnull(x) else "B
df.head()
```

Out[8]:

	first_name	last_name	gender	email	visit_date	fitness_test_date	application_dat
0	Kim	Walter	female	KimWalter58@gmail.com	7-1-17	2017-07-03	Non
1	Tom	Webster	male	TW3857@gmail.com	7-1-17	2017-07-02	Non
2	Edward	Bowen	male	Edward.Bowen@gmail.com	7-1-17	None	2017-07-0
3	Marcus	Bauer	male	Marcus.Bauer@gmail.com	7-1-17	2017-07-01	2017-07-0
4	Roberta	Best	female	RB6305@hotmail.com	7-1-17	2017-07-02	Non

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 [9]:
```
ab_counts = df.groupby("ab_test_group").first_name.count().reset_index()
ab_counts
```

Out[9]:

	ab_test_group	first_name
0	A	2504

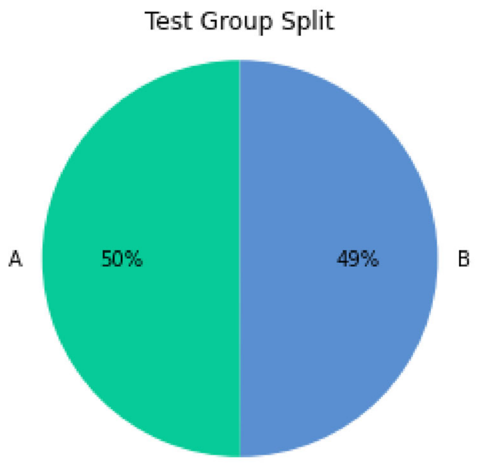
	ab_test_group	first_name
1	B	2500

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 [10]:

```
plt.pie(ab_counts.first_name.values, labels=["A", "B"], autopct="%d%%", startangle=90,
plt.axis("equal")
plt.title("Test Group Split")
plt.show()
plt.savefig("ab_test_pie_chart.png")
```



```
<Figure size 432x288 with 0 Axes>
```

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 [11]:

```
df["is_application"] = df.application_date.apply(lambda x: "Application" if pd.notnull(
```

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 [12]:
```
app_counts = df.groupby(["ab_test_group", "is_application"]).first_name.count().reset_i
app_counts
```

Out[12]:

	ab_test_group	is_application	first_name
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 [13]:
```
app_pivot = app_counts.pivot(index="ab_test_group", columns="is_application", values="f
```

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

In [14]:
```
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 [15]:
```
app_pivot["Percent with Application"] = app_pivot["Application"] / app_pivot["Total"]
app_pivot
```

Out[15]:

is_application	ab_test_group	Application	No Application	Total	Percent with Application
0	A	250	2254	2504	0.09984
1	B	325	2175	2500	0.13000

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 [16]:
```
from scipy.stats import chi2_contingency

app_contingency = ([250, 2254], [325, 2175])
_, app_p, _, _ = chi2_contingency(app_contingency)
if app_p < .05:
    print(round(app_p, 5))
    print("Significant!")
else:
    print(round(app_p, 5))
    print("Not Significant!")
```

```
0.00096
Significant!
```

Step 5: 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 [17]:

```
df["is_member"] = df.purchase_date.apply(lambda x: "Member" if pd.notnull(x) else "Not
```

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

In [18]:

```
just_apps = df[df.is_application == "Application"]
just_apps.head()
```

Out[18]:

	first_name	last_name	gender	email	visit_date	fitness_test_date	application_da
2	Edward	Bowen	male	Edward.Bowen@gmail.com	7-1-17	None	2017-07-
3	Marcus	Bauer	male	Marcus.Bauer@gmail.com	7-1-17	2017-07-01	2017-07-
9	Salvador	Cardenas	male	SCardenas1980@gmail.com	7-1-17	2017-07-07	2017-07-
11	Valerie	Munoz	female	VMunoz1998@gmail.com	7-1-17	2017-07-03	2017-07-
35	Michael	Burks	male	MB9820@gmail.com	7-1-17	None	2017-07-

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 [19]:

```
member_counts = just_apps.groupby(["ab_test_group", "is_member"]).first_name.count().re
#member_count

member_pivot = member_counts.pivot(index="ab_test_group", columns="is_member", values="
member_pivot["Total"] = member_pivot["Member"] + member_pivot["Not Member"]
member_pivot["Percent Purchased"] = member_pivot["Member"] / member_pivot["Total"]
member_pivot
```

Out[19]:

is_member	ab_test_group	Member	Not Member	Total	Percent Purchased
0	A	200	50	250	0.800000
1	B	250	75	325	0.769231

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 [20]:

```
member_contingency = ([200, 50], [250, 75])
_, mem_p, _, _ = chi2_contingency(member_contingency)
if mem_p < .05:
    print(round(mem_p, 5))
    print("Significant!")
else:
    print(round(mem_p, 5))
    print("Not Significant!")
```

```
0.43259
Not Significant!
```

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 [21]:

```
final_member_count = df.groupby(["ab_test_group", "is_member"]).first_name.count().rese
#final_member_count

final_member_pivot = final_member_count.pivot(index="ab_test_group", columns="is_member

final_member_pivot["Total"] = final_member_pivot["Member"] + final_member_pivot["Not Me
final_member_pivot["Percent Purchased"] = final_member_pivot["Member"] / final_member_p
final_member_pivot
```

Out[21]:

is_member	ab_test_group	Member	Not Member	Total	Percent Purchased
0	A	200	2304	2504	0.079872
1	B	250	2250	2500	0.100000

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 [22]:

```
fmem_contingency = ([200, 2304], [250, 2250])
_, fmem_p, _, _ = chi2_contingency(fmem_contingency)
if fmem_p < .05:
    print(round(fmem_p, 5))
    print("Significant!")
else:
    print(round(fmem_p, 5))
    print("Not Significant!")
```

```
0.01472
Significant!
```

Step 6: 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:

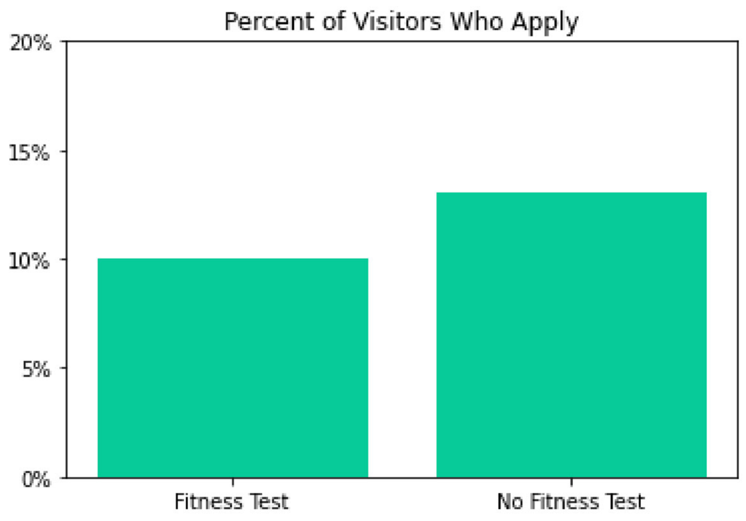
- Percent of visitors who apply
- Percent of applicants who purchase a membership
- Percent of visitors who purchase a membership

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

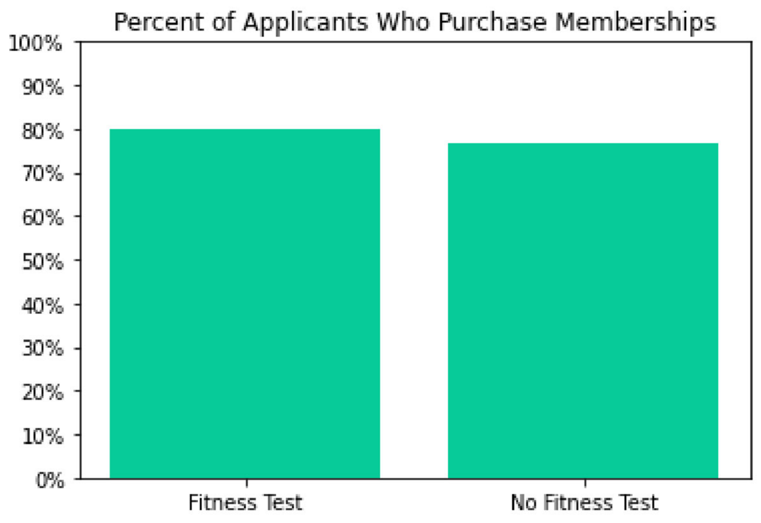
In [23]:

```
#Percent of visitors who apply
ax = plt.subplot()
plt.bar(range(len(app_pivot)), app_pivot["Percent with Application"].values, color="#07
ax.set_xticks(range(len(app_pivot)))
ax.set_xticklabels(["Fitness Test", "No Fitness Test"])
ax.set_yticks([0, 0.05, 0.10, 0.15, 0.20])
ax.set_yticklabels(["0%", "5%", "10%", "15%", "20%"])
plt.title("Percent of Visitors Who Apply")
plt.show()
plt.savefig("pct_visitors_who_apply.png")

app_sig = "The difference of more applicants from group b is significant given p-value:
app_sig
```



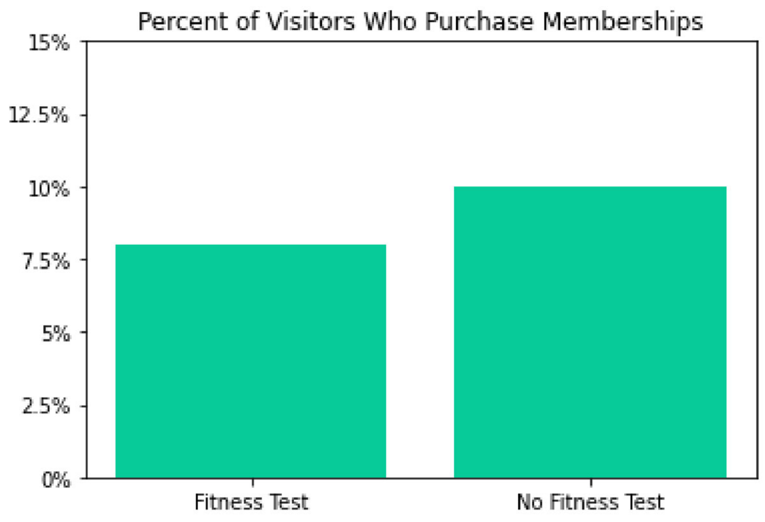
Out[23]:
```
'The difference of more applicants from group b is significant given p-value: 0.00096'
<Figure size 432x288 with 0 Axes>
```

In [24]:
```
#Percent of applicants who purchase a membership
ax = plt.subplot()
plt.bar(range(len(member_pivot)), member_pivot["Percent Purchased"].values, color="#07C
ax.set_xticks(range(len(member_pivot)))
ax.set_xticklabels(["Fitness Test", "No Fitness Test"])
ax.set_yticks([0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1])
ax.set_yticklabels(["0%", "10%", "20%", "30%", "40%", "50%", "60%", "70%", "80%", "90%"
plt.title("Percent of Applicants Who Purchase Memberships")
plt.show()
plt.savefig("pct_applicants_who_purchase_memberships.png")

mem_sig = "The difference of more applicants from group a purchasing memberships is not
mem_sig
```



Out[24]:
```
'The difference of more applicants from group a purchasing memberships is not significan
t given p-value: 0.4326'
<Figure size 432x288 with 0 Axes>
```

In [25]:
```
#Percent of visitors who purchase a membership
ax = plt.subplot()
plt.bar(range(len(final_member_pivot)), final_member_pivot["Percent Purchased"].values,
ax.set_xticks(range(len(final_member_pivot)))
ax.set_xticklabels(["Fitness Test", "No Fitness Test"])
ax.set_yticks([0, 0.025, 0.05, 0.075, 0.1, 0.125, 0.15])
ax.set_yticklabels(["0%", "2.5%", "5%", "7.5%", "10%", "12.5%", "15%"])
plt.title("Percent of Visitors Who Purchase Memberships")
plt.show()
plt.savefig("pct_visitors_who_purchase_memberships.png")

fmem_sig = "The difference of more memberships purchased by group b is significant p-va
fmem_sig
```



```
Out[25]: 'The difference of more memberships purchased by group b is significant p-value: 0.0147'

         <Figure size 432x288 with 0 Axes>
```