

```
In [45]: # CODE FOR EXTRACTING API AND WRITING JSON TO FILE

import requests
import json
import pandas as pd
import numpy as np
response = requests.get("https://data.cityofnewyork.us/resource/jb7j-dtam.json")
#print(response.status_code)
json1 = response.json()
#print(json1)
with open('data.json', 'w') as outfile:
    json.dump(json1, outfile)
df = pd.read_json('data.json')
df.head()
```

Out[45]:

	age_adjusted_death_rate	death_rate	deaths	leading_cause	race_ethnicity	sex	year
0	6.9	6.7	42	Accidents Except Drug Posioning (V01-X39, X43,...	Asian and Pacific Islander	F	2014
1	12.5	11.2	142	Alzheimer's Disease (G30)	Hispanic	F	2014
2	126.4	146.4	1536	All Other Causes	Black Non-Hispanic	F	2014
3	4.2	4.5	28	Intentional Self-Harm (Suicide: X60-X84, Y87.0)	Asian and Pacific Islander	F	2014
4	.	.	.	Alzheimer's Disease (G30)	Not Stated/Unknown	F	2014

```
In [47]: #Connect to MongoDB Cloud Server and Inserting data
from pymongo import MongoClient
client=MongoClient("mongodb+srv://harsh2311:Ireland%4023@cluster0-a5c74.mongodb.net/test?retryWrites=true&w=majority")

db=client.get_database('harsh_db')
db
records=db.death
records.insert_many(df.to_dict('records'))
```

Out[47]: <pymongo.results.InsertManyResult at 0x200942d3ec8>

```
In [49]: #Data Cleaning
df['year'][0:3] = 0
df['age_adjusted_death_rate'] = df['age_adjusted_death_rate'].replace('.', np.NaN)
df['death_rate'] = df['death_rate'].replace('.', np.NaN)
df['deaths'] = df['deaths'].replace('.', np.NaN)
df['race_ethnicity'] = df['race_ethnicity'].replace('Not Stated/Unknown', np.NaN)
df['race_ethnicity'] = df['race_ethnicity'].replace('Other Race Ethnicity', np.NaN)
df['race_ethnicity'] = df['race_ethnicity'].replace('Other Race/ Ethnicity', np.NaN)
df['race_ethnicity'] = df['race_ethnicity'].replace('White Non-Hispanic', 'White Hispanic')
df['race_ethnicity'] = df['race_ethnicity'].replace('Black Non-Hispanic', 'Black Hispanic')
```

C:\Users\Harsh\Anaconda3\lib\site-packages\ipykernel\_launcher.py:2: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: <http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy>  
(<http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy>)

```
In [50]: #Before Cleaning
df.isnull().sum()
```

```
Out[50]: age_adjusted_death_rate    355
death_rate                        355
deaths                           122
leading_cause                     0
race_ethnicity                   355
sex                               0
year                             0
dtype: int64
```

```
In [51]: #Imputing NA values with different methods
df['age_adjusted_death_rate'].fillna(df['age_adjusted_death_rate'].median(),inplace = True)
df['death_rate'].fillna(df['death_rate'].median(),inplace = True)
df['deaths'].fillna(df['deaths'].median(),inplace = True)
df['year'][0:3] = '2014'
df['leading_cause'].fillna( method = 'bfill', inplace = True)
df['race_ethnicity'].fillna( method = 'ffill', inplace = True)
df['race_ethnicity'].fillna( method = 'bfill', inplace = True)
df['sex'].fillna( method = 'backfill', inplace = True)
```

C:\Users\Harsh\Anaconda3\lib\site-packages\ipykernel\_launcher.py:5: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: <http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy> (<http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy>)

```
In [52]: #After Cleaning
df.isnull().sum()
```

```
Out[52]: age_adjusted_death_rate    0
death_rate                        0
deaths                           0
leading_cause                     0
race_ethnicity                    0
sex                               0
year                             0
dtype: int64
```

```
In [12]: #Convert data types
df['age_adjusted_death_rate']=df['age_adjusted_death_rate'].astype(float)
df['death_rate']=df['death_rate'].astype(float)
df['deaths']=df['deaths'].astype(int)
df['year']=df['year'].astype(int)
df.dtypes
```

```
Out[12]: age_adjusted_death_rate    float64
death_rate                        float64
deaths                           int32
leading_cause                     object
race_ethnicity                   object
sex                              object
year                             int32
dtype: object
```

```
In [58]: #Connect to Postgres
import psycopg2
try:
    dbConnection = psycopg2.connect(
        user = "dap",
        password = "dap",
        host = "192.168.56.30",
        port = "5432",
        database = "postgres")
    dbConnection.set_isolation_level(0) # AUTOCOMMIT
    dbCursor = dbConnection.cursor()
    dbCursor.execute('CREATE DATABASE test2;')
    dbCursor.close()
except (Exception , psycopg2.Error) as dbError :
    print ("Error while connecting to PostgreSQL", dbError)
finally:
    if(dbConnection): dbConnection.close()
```

Error while connecting to PostgreSQL database "test2" already exists

```
In [38]: #Create table
import psycopg2
try:
    dbConnection = psycopg2.connect(
        user = "dap",
        password = "dap",
        host = "192.168.56.30",
        port = "5432",
        database = "postgres")
    dbConnection.set_isolation_level(0) # AUTOCOMMIT
    dbCursor = dbConnection.cursor()
    dbCursor.execute("""
        CREATE TABLE test2(
            age_adjusted_death_rate numeric(8,2),
            death_rate numeric(8,2),
            deaths numeric(5,1),
            leading_cause varchar(255),
            race_ethnicity varchar(255),
            sex varchar(255),
            year int
        );
    """)
except (Exception , psycopg2.Error) as dbError :
    print ("Error while connecting to PostgreSQL", dbError)
finally:
    if(dbConnection): dbConnection.close()
```

Error while connecting to PostgreSQL relation "test2" already exists

```
In [41]: #Insert values
import csv
try:
    dbConnection = psycopg2.connect(
        user = "dap",
        password = "dap",
        host = "192.168.56.30",
        port = "5432",
        database = "postgres")
    dbConnection.set_isolation_level(0) # AUTOCOMMIT
    dbCursor = dbConnection.cursor()
    with open('C:/Users/Harsh/Desktop/NCI/Database and Analytics Programming/DAP Project/test_data.csv', 'r') as f:
        reader = csv.reader(f)
        next(reader) # skip the header
        for row in reader:
            dbCursor.execute("INSERT INTO test2 VALUES (%s,%s,%s,%s,%s,%s,%s,%s)", row)
    dbConnection.commit()
    dbCursor.close()
except (Exception , psycopg2.Error) as dbError :
    print ("Error:", dbError)
finally:
    if(dbConnection): dbConnection.close()
```

```

In [46]: #Query Db
import pandas as pd
import pandas.io.sql as sqlio
import psycopg2
sql = "SELECT * from test2;"
try:
    dbConnection = psycopg2.connect(
        user = "dap",
        password = "dap",
        host = "192.168.56.30",
        port = "5432",
        database = "postgres")
    df = sqlio.read_sql_query(sql, dbConnection)
except (Exception , psycopg2.Error) as dbError :
    print ("Error:", dbError)
finally:
    if(dbConnection): dbConnection.close()

```

```

In [53]: df.head()

```

Out[53]:

	age_adjusted_death_rate	death_rate	deaths	leading_cause	race_ethnicity	sex	year
0	6.9	6.7	42	Accidents Except Drug Posioning (V01-X39, X43,...	Asian and Pacific Islander	F	2014
1	12.5	11.2	142	Alzheimer's Disease (G30)	Hispanic	F	2014
2	126.4	146.4	1536	All Other Causes	Black Hispanic	F	2014
3	4.2	4.5	28	Intentional Self-Harm (Suicide: X60-X84, Y87.0)	Asian and Pacific Islander	F	2014
4	20.2	18.5	148	Alzheimer's Disease (G30)	Asian and Pacific Islander	F	2014

```

In [54]: Users\Harsh\Desktop\NCI\Database and Analytics Programming\DA Project\Harsh_data.csv', index = None,encoding="UTF-8", head

```

```

In [ ]:

```

```
In [122]: #Graph for Comparing Deaths by Gender per year from Period 2007 to 2014
import plotly.graph_objects as go

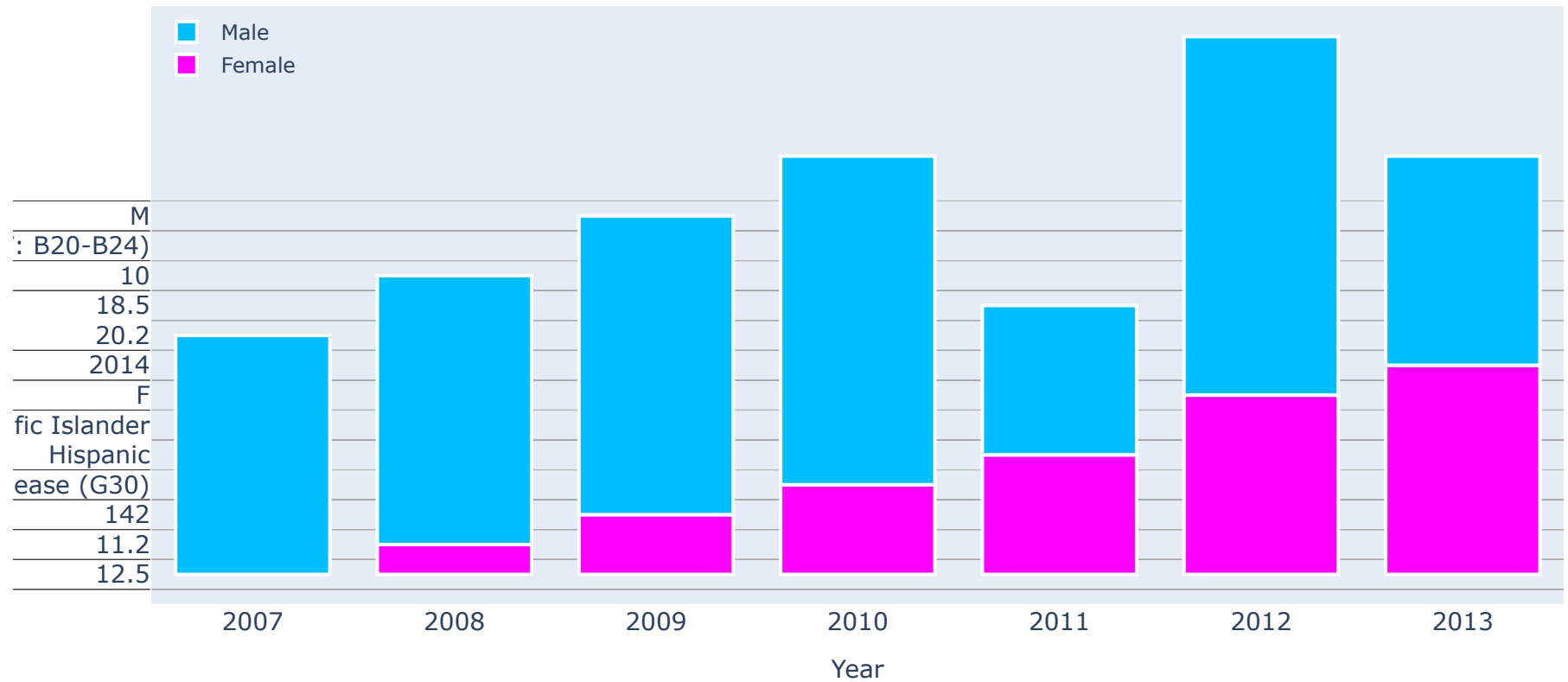
years = [2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014]

fig = go.Figure()
fig.add_trace(go.Bar(x=years,
                    y=df[df['sex']=='F'],
                    name='Female',
                    marker_color='fuchsia'
                    ))
fig.add_trace(go.Bar(x=years,
                    y=df[df['sex']=='M'],
                    name='Male',
                    marker_color='deepskyblue'
                    ))

fig.data[0].marker.line.width = 2
fig.data[0].marker.line.color = "white"
fig.data[1].marker.line.width = 2
fig.data[1].marker.line.color = "white"
fig.update_layout(
    title='Gender Death Ratio Per Year',
    xaxis_tickfont_size=14,
    yaxis=dict(
        title='Gender',
        titlefont_size=16,
        tickfont_size=14,
    ),
    xaxis_title="Year",
    legend=dict(
        x=0,
        y=1.0,
        bgcolor='rgba(245, 255, 255, 0)',
        bordercolor='gold'
    ),
    barmode='stack',
    bargap=0.15, # gap between bars of adjacent location coordinates.
    bargroupgap=0.1 # gap between bars of the same location coordinate.
)
fig.show()
```



## Gender Death Ratio Per Year

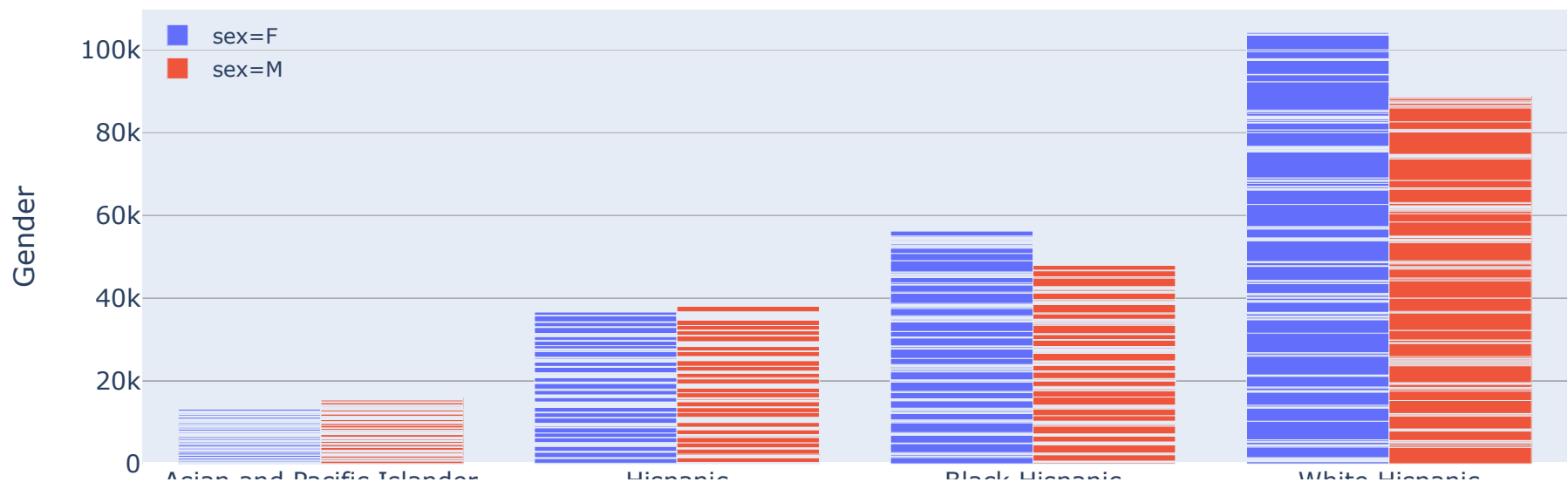


```

In [121]: #Death count as per gender vs Ethnicity
import plotly.express as px
fig = px.bar(df, x="race_ethnicity", y="deaths", color='sex', barmode='group',
             height=400)
fig.update_layout(
    title='Comparision of Deaths occured by Gender as per Ethnicity',
    xaxis_tickfont_size=14,
    yaxis=dict(
        title='Gender',
        titlefont_size=16,
        tickfont_size=14,
    ),
    legend=dict(
        x=0,
        y=1.0,
        bgcolor='rgba(245, 255, 255, 0)',
        bordercolor='gold'
    )
)
fig.show()

```

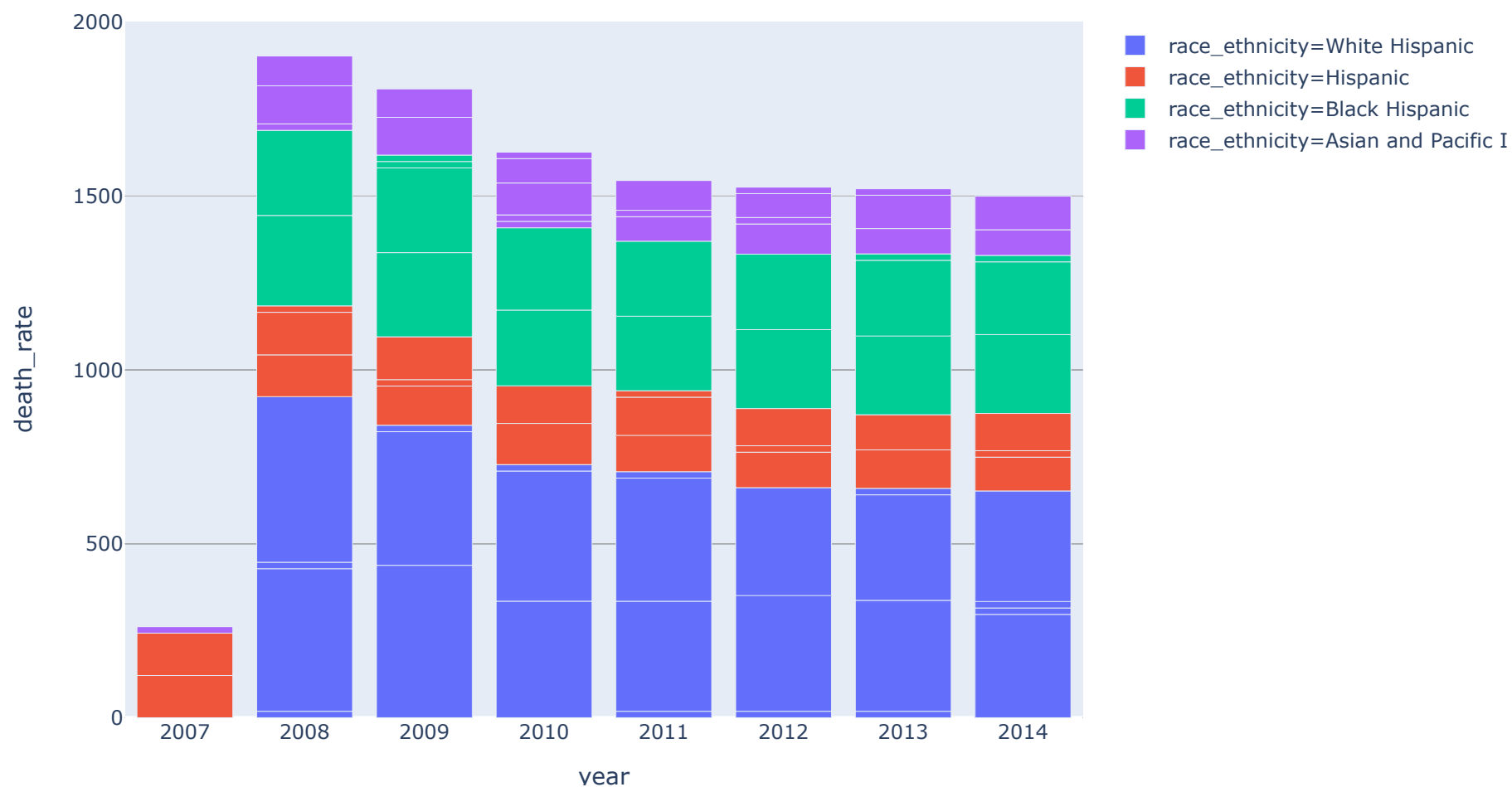
Comparision of Deaths occured by Gender as per Ethnicity



In [ ]:

```
In [120]: of Leading cause death
plotly.express as px
[ df['leading_cause']=='Diseases of Heart (I00-I09, I11, I13, I20-I51)']
x.bar(test, x=test['year'], y=test['death_rate'], color=test['race_ethnicity'],title='Comparision of Death count based on I
v()
```

Comparision of Death count based on Ethnicity each year

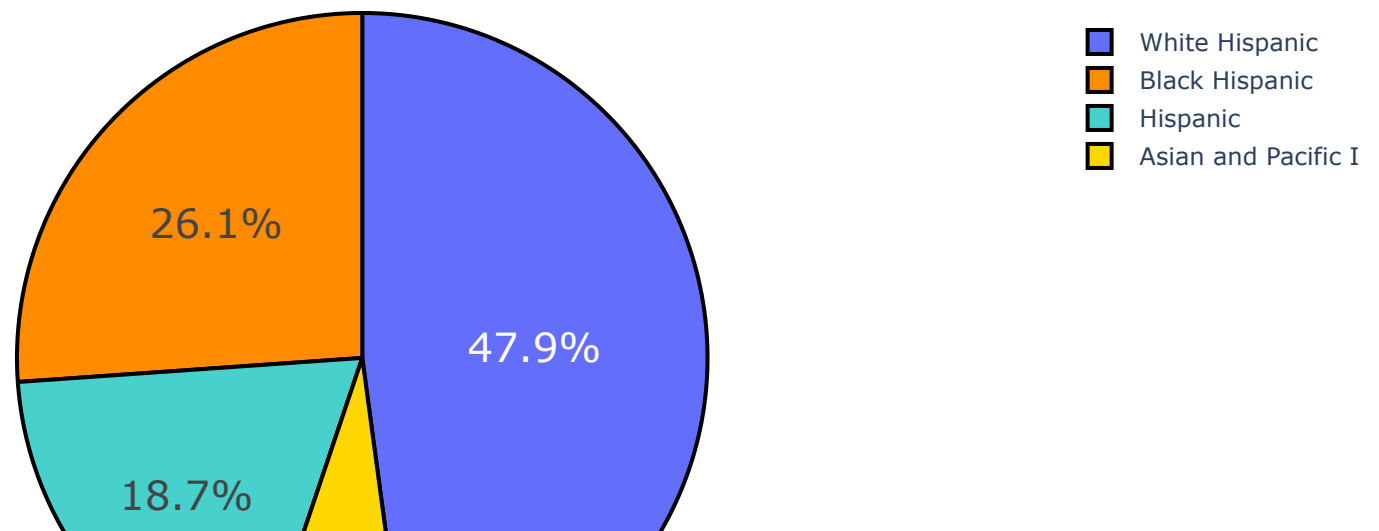


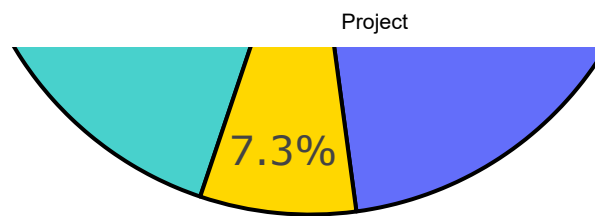


```
In [119]: #Overall Death count as per Ethnicity
import plotly.graph_objects as go

labels = df['race_ethnicity']
values = df['deaths']
colors = ['gold', 'mediumturquoise', 'darkorange', 'lightgreen']
# Use `hole` to create a donut-like pie chart
fig = go.Figure(data=[go.Pie(labels=labels, values=values)])
fig.update_layout(
    title={
        'text': "Overall Death count as per Ethnicity",
        'y':0.9,
        'x':0.4,
        'xanchor': 'center',
        'yanchor': 'top'},
)
fig.update_traces( textfont_size=20,
                    marker=dict(colors=colors, line=dict(color='#000000', width=2)))
fig.show()
```

Overall Death count as per Ethnicity





```
In [127]: #Word Cloud of Leading Death Causes
from wordcloud import WordCloud, STOPWORDS
import matplotlib.pyplot as plt
stopwords = set(STOPWORDS)

def show_wordcloud(data, title = "Word Cloud of Leading Death Causes"):
    wordcloud = WordCloud(include_numbers=True,colormap='cubehelix',
        background_color='white',
        stopwords=stopwords,
        max_words=200,
        max_font_size=40,
        scale=3,
        random_state=1 # chosen at random by flipping a coin; it was heads
    ).generate(str(data))

    fig = plt.figure(1, figsize=(12, 12))
    #plt.axis('off')
    if title:
        fig.suptitle(title, fontsize=20)
        fig.subplots_adjust(top=2.3)

    plt.imshow(wordcloud)
    plt.show()

show_wordcloud(df.leading_cause)
```



