Fahreza Apriyoga Arizal

Data Scientist - Universitas Lampung

No. Registrasi : 0161382101-21

Nutrition Fact for McDonald’s Menu

Calories, fat, and sugar for every cheeseburger, fries, and milkshake on menu

In [1]:

```
import pandas as pd
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [8]:

```
ds = pd.read_csv('D:/CSV/menu.csv')
```

In [9]:

```
#menampilkan 5 row teratas dari data set
ds.head()
```

Out[9]:

	Category	Item	Serving Size	Calories	Calories from Fat	Total Fat	Total Fat (% Daily Value)	Saturated Fat	Saturated Fat (% Daily Value)	Trans Fat	...	Carbohydrates	Carbohydrates (% Daily Value)	Dieta Fil
0	Breakfast	Egg McMuffin	4.8 oz (136 g)	300	120	13.0	20	5.0	25	0.0	...	31	10	
1	Breakfast	Egg White Delight	4.8 oz (135 g)	250	70	8.0	12	3.0	15	0.0	...	30	10	
2	Breakfast	Sausage McMuffin	3.9 oz (111 g)	370	200	23.0	35	8.0	42	0.0	...	29	10	
3	Breakfast	Sausage McMuffin with Egg	5.7 oz (161 g)	450	250	28.0	43	10.0	52	0.0	...	30	10	
4	Breakfast	Sausage McMuffin with Egg Whites	5.7 oz (161 g)	400	210	23.0	35	8.0	42	0.0	...	30	10	

5 rows × 24 columns

In [13]:

```

```

In [14]:

```
ds.describe(include="all")
```

Out[14]:

	Category	Item	Serving Size	Calories	Calories from Fat	Total Fat	Total Fat (% Daily Value)	Saturated Fat	Saturated Fat (% Daily Value)	Trans Fat	...	Carb

	Category	Item	Serving Size	Calories	Calories from Fat	Total Fat	Total Fat (% Daily Value)	Saturated Fat	Saturated Fat (% Daily Value)	Trans Fat	...	Carb
count	260	260	260	260.000000	260.000000	260.000000	260.000000	260.000000	260.000000	260.000000	...	2
unique	9	260	109	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	
top	Coffee & Tea	Frappé Chocolate Chip (Small)	16 fl oz cup	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	
freq	95	1	45	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	
mean	NaN	NaN	NaN	368.269231	127.096154	14.165385	21.815385	6.007692	29.965385	0.203846	...	
std	NaN	NaN	NaN	240.269886	127.875914	14.205998	21.885199	5.321873	26.639209	0.429133	...	
min	NaN	NaN	NaN	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	
25%	NaN	NaN	NaN	210.000000	20.000000	2.375000	3.750000	1.000000	4.750000	0.000000	...	
50%	NaN	NaN	NaN	340.000000	100.000000	11.000000	17.000000	5.000000	24.000000	0.000000	...	
75%	NaN	NaN	NaN	500.000000	200.000000	22.250000	35.000000	10.000000	48.000000	0.000000	...	
max	NaN	NaN	NaN	1880.000000	1060.000000	118.000000	182.000000	20.000000	102.000000	2.500000	...	1

11 rows × 24 columns

In [15]:

```
ds.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 260 entries, 0 to 259
Data columns (total 24 columns):
 #   Column                         Non-Null Count  Dtype  
---  ------                         --------------  -----  
 0   Category                       260 non-null    object 
 1   Item                           260 non-null    object 
 2   Serving Size                   260 non-null    object 
 3   Calories                       260 non-null    int64  
 4   Calories from Fat              260 non-null    int64  
 5   Total Fat                      260 non-null    float64
 6   Total Fat (% Daily Value)      260 non-null    int64  
 7   Saturated Fat                  260 non-null    float64
 8   Saturated Fat (% Daily Value)  260 non-null    int64  
 9   Trans Fat                      260 non-null    float64
 10  Cholesterol                    260 non-null    int64  
 11  Cholesterol (% Daily Value)    260 non-null    int64  
 12  Sodium                         260 non-null    int64  
 13  Sodium (% Daily Value)         260 non-null    int64  
 14  Carbohydrates                  260 non-null    int64  
 15  Carbohydrates (% Daily Value)  260 non-null    int64  
 16  Dietary Fiber                  260 non-null    int64  
 17  Dietary Fiber (% Daily Value)  260 non-null    int64  
 18  Sugars                         260 non-null    int64  
 19  Protein                        260 non-null    int64  
 20  Vitamin A (% Daily Value)      260 non-null    int64  
 21  Vitamin C (% Daily Value)      260 non-null    int64  
 22  Calcium (% Daily Value)        260 non-null    int64  
 23  Iron (% Daily Value)           260 non-null    int64  
dtypes: float64(3), int64(18), object(3)
memory usage: 48.9+ KB
```

A. How Many Calories Does The Average McDonald's Value Meal Contain ?

In [38]:

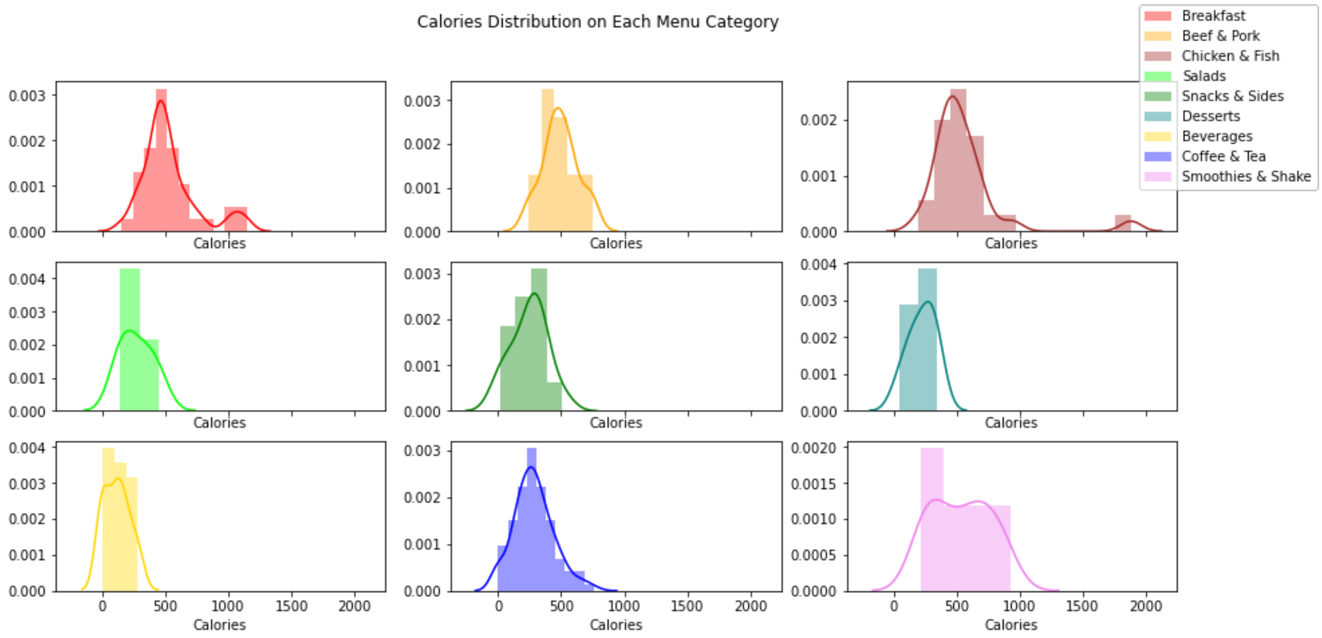
```
#menampilkan tipe makanan
df.Category.unique()
```

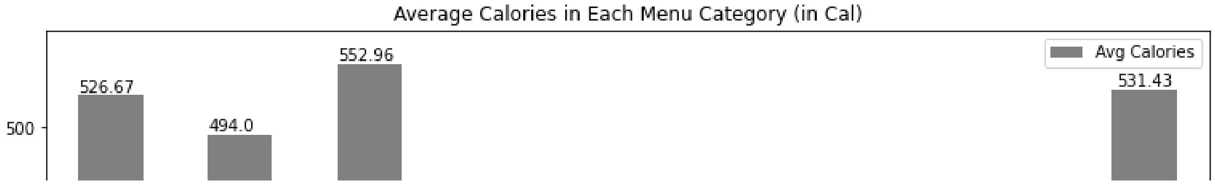
Out[38]:

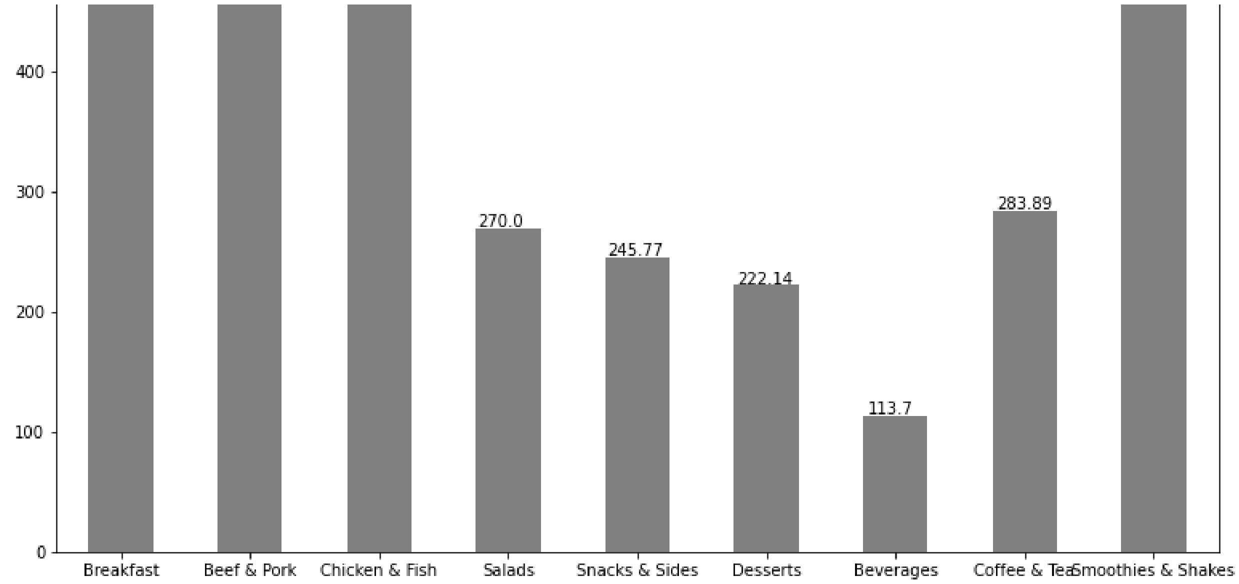
```
array(['Breakfast', 'Beef & Pork', 'Chicken & Fish', 'Salads',
       'Snacks & Sides', 'Desserts', 'Beverages', 'Coffee & Tea',
       'Smoothies & Shakes'], dtype=object)
```

In [39]:

```
#menetapkan dataframe untuk setiap kategori
brkf = df.loc[df.Category == 'Breakfast']
bnp = df.loc[df.Category == 'Beef & Pork']
cnf = df.loc[df.Category == 'Chicken & Fish']
sld = df.loc[df.Category == 'Salads']
snass = df.loc[df.Category == 'Snacks & Sides']
dess = df.loc[df.Category == 'Desserts']
bev = df.loc[df.Category == 'Beverages']
cnt = df.loc[df.Category == 'Coffee & Tea']
ss = df.loc[df.Category == 'Smoothies & Shakes']


# Plot calorie distribution for each category
fig, axes = plt.subplots(3, 3, figsize=(15, 7), sharex=True)
sns.color_palette("tab10")
sns.distplot( brkf["Calories"] , color='red', ax=axes[0, 0], label = "Breakfast")
sns.distplot( bnp["Calories"] , color='orange',ax=axes[0, 1], label = "Beef & Pork")
sns.distplot( cnf["Calories"] , color='brown',ax=axes[0, 2], label = "Chicken & Fish")
sns.distplot( sld["Calories"] , color='lime',ax=axes[1, 0], label = "Salads")
sns.distplot( snass["Calories"] , color='green',ax=axes[1, 1], label = "Snacks & Sides")
sns.distplot( dess["Calories"] ,  color='teal',ax=axes[1, 2], label = "Desserts")
sns.distplot( bev["Calories"] ,  color='gold',ax=axes[2, 0], label = "Beverages")
sns.distplot( cnt["Calories"] ,  color='blue',ax=axes[2, 1], label = "Coffee & Tea")
sns.distplot( ss["Calories"] ,  color='violet',ax=axes[2, 2], label = "Smoothies & Shake")
fig.suptitle("Calories Distribution on Each Menu Category")
fig.legend()
plt.show()


#rata rata kalori disetiap kategori makanan
avg_cat = [round(brkf['Calories'].mean(axis=0), 2), round(bnp['Calories'].mean(axis=0), 2), round(c
nf['Calories'].mean(axis=0), 2),
          round(sld['Calories'].mean(axis=0), 2), round(snass['Calories'].mean(axis=0), 2), round(d
ess['Calories'].mean(axis=0), 2),
          round(bev['Calories'].mean(axis=0), 2), round(cnt['Calories'].mean(axis=0), 2), round(ss[
'Calories'].mean(axis=0), 2)]
index = ['Breakfast', 'Beef & Pork', 'Chicken & Fish', 'Salads', 'Snacks & Sides', 'Desserts', 'Bev
erages', 'Coffee & Tea', 'Smoothies & Shakes']
avg_calat= pd.DataFrame({'Avg Calories': avg_cat}, index=index)
ax = avg_calat.plot.bar(rot=0, color='gray', figsize=(13,8), title='Average Calories in Each Menu C
ategory (in Cal)', legend=True)
for p in ax.patches:
    ax.annotate(str(p.get_height()), (p.get_x() * 1.005, p.get_height() * 1.005))
```







In [41]:

```
print("Average calories of all McD's meals (include drinks) is ", round(df['Calories'].mean(axis=0
), 2), "Cal.") #rata rata kalori semua makanan
print("Average calories of all McD's meals (drinks excluded) is ", round(meals.Calories.mean(axis=
0), 2), "Cal.") #rata rata kalori makanan
```

```
Average calories of all McD's meals (include drinks) is  368.27 Cal.
Average calories of all McD's meals (drinks excluded) is  462.09 Cal.
```

B. How Much Do Beverages, Like Soda or Coffee, Contribute To The Overall Caloricv Intake ?

In [45]:

```
#mengambil data dengan kategori beverages
beverages = ds.loc[ds.Category == 'Beverages']
```

In [86]:

```
kalori_beverages = pd.DataFrame({'Item': beverages.Item, 'Calories': beverages.Calories})
#Menghitung asupan kalori per item minuman bagi laki-laki
kalori_beverages['Laki-laki'] = kalori_beverages.Calories/2500
#Menghitung asupan kalori per item minuman bagi perempuan
kalori_beverages['Perempuan'] = kalori_beverages.Calories/2000
kalori_beverages
```

Out[86]:

	Item	Calories	Laki-laki	Perempuan
110	Coca-Cola Classic (Small)	140	0.056	0.070
111	Coca-Cola Classic (Medium)	200	0.080	0.100
112	Coca-Cola Classic (Large)	280	0.112	0.140
113	Coca-Cola Classic (Child)	100	0.040	0.050
114	Diet Coke (Small)	0	0.000	0.000
115	Diet Coke (Medium)	0	0.000	0.000
116	Diet Coke (Large)	0	0.000	0.000
117	Diet Coke (Child)	0	0.000	0.000
118	Dr Pepper (Small)	140	0.056	0.070
119	Dr Pepper (Medium)	190	0.076	0.095
120	Dr Pepper (Large)	270	0.108	0.135
121	Dr Pepper (Child)	100	0.040	0.050
122	Diet Dr Pepper (Small)	0	0.000	0.000

	Item	Calories	Laki-laki	Perempuan
123	Diet Dr Pepper (Medium)	0	0.000	0.000
124	Diet Dr Pepper (Large)	0	0.000	0.000
125	Diet Dr Pepper (Child)	0	0.000	0.000
126	Sprite (Small)	140	0.056	0.070
127	Sprite (Medium)	200	0.080	0.100
128	Sprite (Large)	280	0.112	0.140
129	Sprite (Child)	100	0.040	0.050
130	1% Low Fat Milk Jug	100	0.040	0.050
131	Fat Free Chocolate Milk Jug	130	0.052	0.065
132	Minute Maid 100% Apple Juice Box	80	0.032	0.040
133	Minute Maid Orange Juice (Small)	150	0.060	0.075
134	Minute Maid Orange Juice (Medium)	190	0.076	0.095
135	Minute Maid Orange Juice (Large)	280	0.112	0.140
136	Dasani Water Bottle	0	0.000	0.000

In [50]:

```
minuman = ds.loc[(ds["Category"].isin(['Beverages', 'Coffee & Tea',
 'Smoothies & Shakes']))]
```

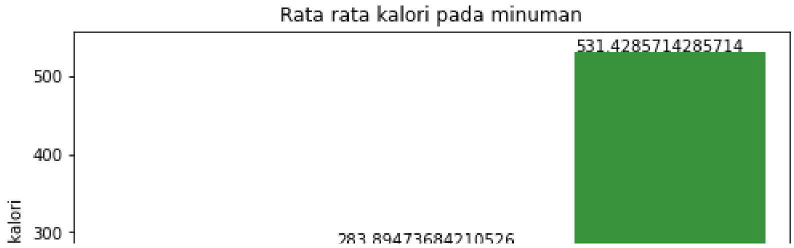
In [90]:

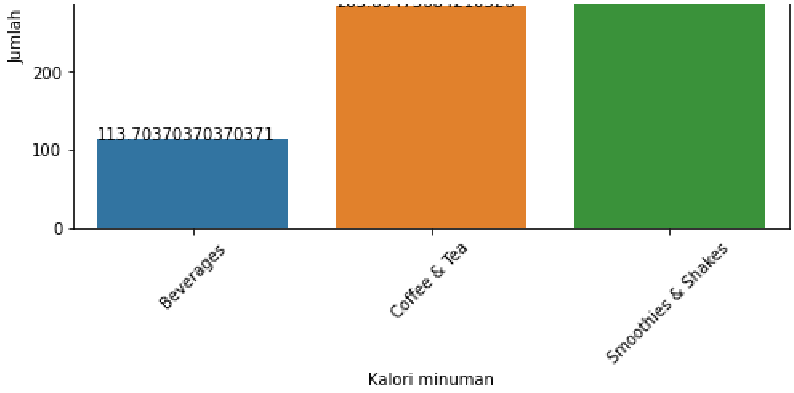
```
avg_minuman = minuman['Calories'].mean(axis=0)

#menampilkan hasil perhitungan
print("Rata rata kalori semua minuman ", round(avg_minuman, 2))
#mengubah menjadi numerik
avg_minuman = pd.to_numeric(avg_minuman)

#asupan kalori laki-laki
drink_men = (avg_minuman/2500)*100
print("Rata rata kalori pada asupan laki-laki ", round(drink_men, 2),"%")
#asupan kalori permpuan
drink_women =(avg_minuman/2000)*100
print("Rata rata kalori pada asupan perempuan ", round(drink_women, 2),"%")
```

```
Rata rata kalori semua minuman  299.47
Rata rata kalori pada asupan laki-laki  11.98 %
Rata rata kalori pada asupan perempuan  14.97 %
```

In [60]:

```
plt.figure(figsize=(8,5))
#membuat barchart dengan nilai kalori
ax = sns.barplot(x=ds.groupby(minuman["Category"])['Calories'].mean().index,
 y=ds.groupby(minuman["Category"])['Calories'].mean().values)
#menampilkan nilai rata rata
for p in ax.patches:
 ax.annotate(str(p.get_height()), (p.get_x(), p.get_height()))
#penamaan chart
plt.title("Rata rata kalori pada minuman")
plt.ylabel("Jumlah kalori")
plt.xlabel("Kalori minuman")
plt.xticks(rotation=45)
plt.show()
```





C. Does ordered grilled chicken instead of crispy increase a sandwich's nutritional value?

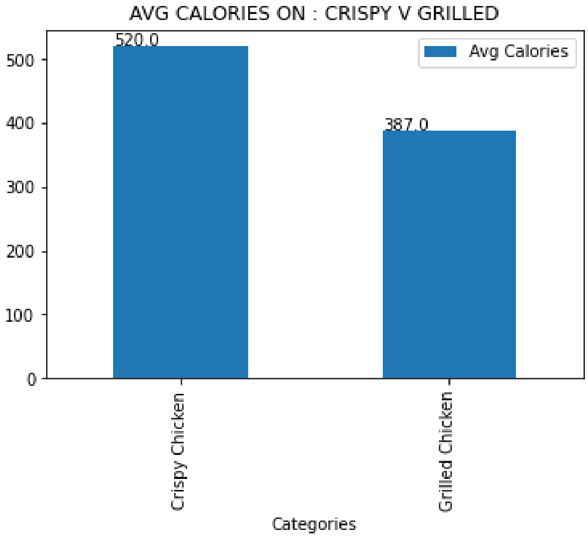
In [61]:

```
# EXPLORASI JUMLAH KALORI PADA CRISPY CHICKEN
crispy = df[df['Item'].str.contains('Crispy Chicken')]
crispy_cal = pd.DataFrame({'Item': crispy.Item, 'Calories': crispy.Calories})

# KALORI PADA CRISPY CHICKEN - RATA-RATA
avg_criscal = crispy.Calories.mean(axis=0)
print("CALORIES ON CRISPY CHICKEN (AVG): ", avg_criscal, "Cal.")

# EXPLORASI JUMLAH KALORI PADA GRILLED CHICKEN
grilled = df[df['Item'].str.contains('Grilled Chicken')]
grilled_cal = pd.DataFrame({'Item': grilled.Item, 'Calories': grilled.Calories})

# KALORI PADA PADA GRILLED CHICKEN - RATA-RATA
avg_grilcal = round(grilled.Calories.mean(axis=0))
print("CALORIES ON GRILLED CHICKEN (AVG): ", avg_grilcal, "Cal.")
```

```
CALORIES ON CRISPY CHICKEN (AVG):  520.0 Cal.
CALORIES ON GRILLED CHICKEN (AVG):  387 Cal.
```

In [67]:

```
#membuat barchart dengan nilai kalori
avg_cal = pd.DataFrame({'Categories':['Crispy Chicken', 'Grilled Chicken'], 'Avg Calories': [avg_c
riscal, avg_grilcal]})
ax = avg_cal.plot.bar(x = 'Categories', y = 'Avg Calories')
ax.set_title("AVG CALORIES ON : CRISPY V GRILLED")
for p in ax.patches:
    ax.annotate(str(p.get_height()), (p.get_x() * 1.005, p.get_height() * 1.005))
```



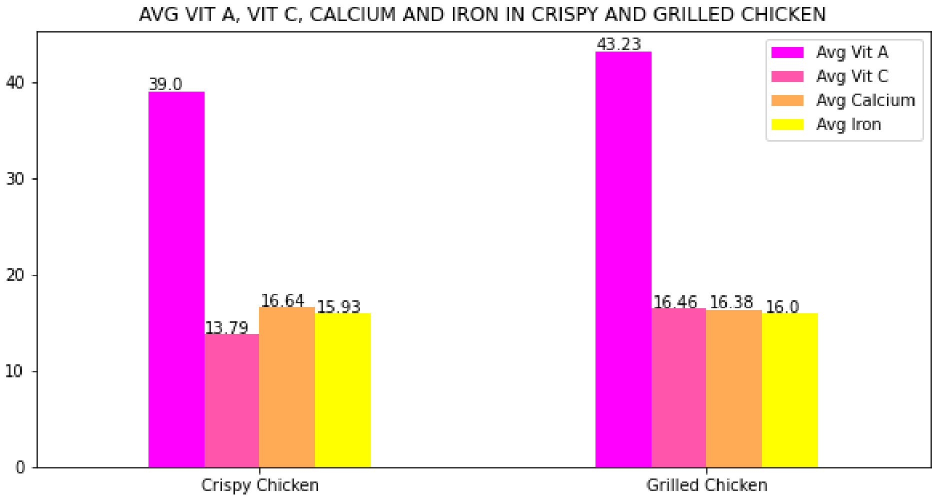
In [69]:

```
# EXPLORASI VIT A, VIT C, CALCIUM, DAN IRON PADA CRISPY CHICKEN
# EXPLORASI VIT A, VIT C, CALCIUM, DAN IRON PADA CRISPY CHICKEN
crispy_vm = pd.DataFrame({'Item': crispy.Item, 'Vit A': crispy['Vitamin A (% Daily Value)'], 'Vit
C': crispy['Vitamin C (% Daily Value)'], 'Calcium': crispy['Calcium (% Daily Value)'], 'Iron': cris
py['Iron (% Daily Value)']})

# EXPLORASI VIT A, VIT C, CALCIUM, DAN IRON PADA CRISPY CHICKEN
grilled_vm = pd.DataFrame({'Item': grilled.Item, 'Vit A': grilled['Vitamin A (% Daily Value)'],
'Vit C': grilled['Vitamin C (% Daily Value)'], 'Calcium': grilled['Calcium (% Daily Value)'],
'Iron': grilled['Iron (% Daily Value)']})

#DV : DAILY VALUE

# RATA-RATA VIT A, VIT C, CALCIUM, DAN IRON PADA CRISPY CHICKEN
avg_crispy_vita = round(crispy_vm['Vit A'].mean(axis=0), 2)
avg_crispy_vitc = round(crispy_vm['Vit C'].mean(axis=0), 2)
avg_crispy_calc = round(crispy_vm['Calcium'].mean(axis=0), 2)
avg_crispy_iron = round(crispy_vm['Iron'].mean(axis=0), 2)
print("AVG OF VIT A, VIT C, CALCIUM, AND IRON IN CRISPY  CHICKEN: ",
      avg_crispy_vita, "%DV,", avg_crispy_vitc, "%DV,", avg_crispy_calc, "%DV, and",
avg_crispy_iron, "%DV.")

# RATA-RATA VIT A, VIT C, CALCIUM, DAN IRON PADA CRISPY CHICKEN
avg_grilled_vita = round(grilled_vm['Vit A'].mean(axis=0), 2)
avg_grilled_vitc = round(grilled_vm['Vit C'].mean(axis=0), 2)
avg_grilled_calc = round(grilled_vm['Calcium'].mean(axis=0), 2)
avg_grilled_iron = round(grilled_vm['Iron'].mean(axis=0), 2)
print("AVG OF VIT A, VIT C, CALCIUM, AND IRON IN GRILLED  CHICKEN: ", avg_grilled_vita, "%DV,", av
g_grilled_vitc, "%DV,", avg_grilled_calc, "%DV, and", avg_grilled_iron, "%DV.")
```

```
AVG OF VIT A, VIT C, CALCIUM, AND IRON IN CRISPY  CHICKEN:  39.0 %DV, 13.79 %DV, 16.64 %DV, and 15
.93 %DV.
AVG OF VIT A, VIT C, CALCIUM, AND IRON IN GRILLED  CHICKEN:  43.23 %DV, 16.46 %DV, 16.38 %DV, and
16.0 %DV.
```

In [80]:

```
#membuat barchart

avg_vita = [avg_crispy_vita, avg_grilled_vita]
avg_vitc = [avg_crispy_vitc, avg_grilled_vitc]
avg_calc = [avg_crispy_calc, avg_grilled_calc]
avg_iron = [avg_crispy_iron, avg_grilled_iron]
index = ['Crispy Chicken', 'Grilled Chicken']
avg_vm = pd.DataFrame({'Avg Vit A': avg_vita,
                   'Avg Vit C': avg_vitc,
                   'Avg Calcium': avg_calc,
                   'Avg Iron': avg_iron}, index=index)
ax = avg_vm.plot.bar(rot=0, colormap='spring', figsize=(10,5))
ax.set_title("AVG VIT A, VIT C, CALCIUM AND IRON IN CRISPY AND GRILLED CHICKEN")
for p in ax.patches:
    ax.annotate(str(p.get_height()), (p.get_x() * 1.005, p.get_height() * 1.005))
```



D. What about ordering egg whites instead of whole eggs?

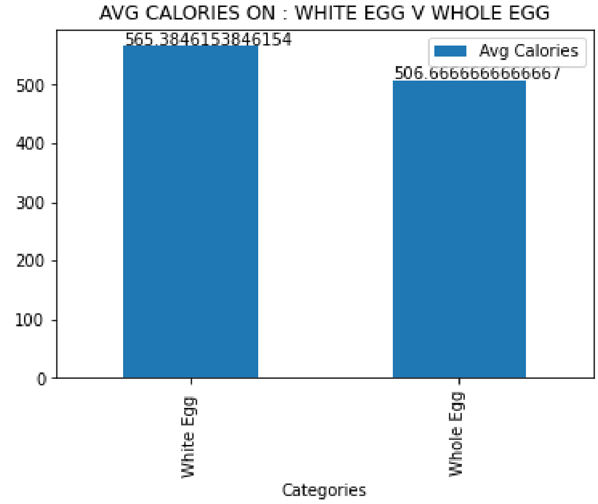
In [78]:

```
#JUMLAH KALORI PADA WHITES EGG
whites = df[df['Item'].str.contains('Egg White')]
whites_cal = pd.DataFrame({'Item': whites.Item, 'Calories': whites.Calories})

#RATA-RATA KALORI PADA WHITES EGG
avg_whites_cal = whites.Calories.mean(axis=0)
print("CALORIES ON WHITE EGGS (AVG): ", avg_whites_cal, "Cal.")

#JUMLAH KALORI PADA WHOLE EGG
whole = df[df['Item'].str.contains('Egg')]
whole = whole[~whole['Item'].str.contains('White')]
whole_cal = pd.DataFrame({'Item': whole.Item, 'Calories': whole.Calories})

#RATA-RATA KALORI PADA PADA WHOLE EGG
avg_whole_cal = whole.Calories.mean(axis=0)
print("CALORIES ON WHOLE EGGS (AVG): ", avg_whole_cal, "Cal.")
```

```
CALORIES ON WHITE EGGS (AVG):  565.3846153846154 Cal.
CALORIES ON WHOLE EGGS (AVG):  506.6666666666667 Cal.
```

In [79]:

```
#membuat barchart

avg_egg_cal = pd.DataFrame({'Categories':['White Egg ', 'Whole Egg'], 'Avg Calories':
[avg_whites_cal, avg_whole_cal]})
ax = avg_egg_cal.plot.bar(x = 'Categories', y = 'Avg Calories')
ax.set_title("AVG CALORIES ON : WHITE EGG V WHOLE EGG")
for p in ax.patches:
    ax.annotate(str(p.get_height()), (p.get_x() * 1.005, p.get_height() * 1.005))
```



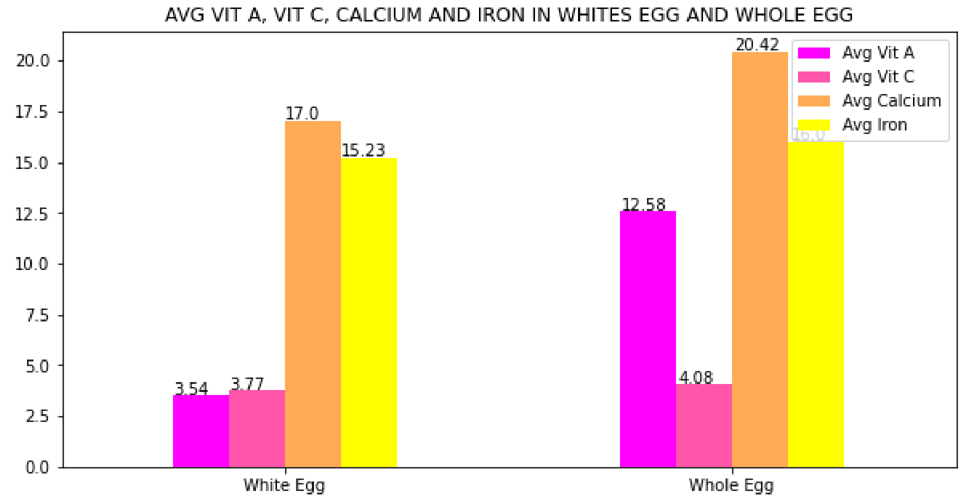
In [85]:

```
#VIT A, VIT C, CALCIUM, DAN IRON PADA WHITE EGG
whites_vm = pd.DataFrame({'Item': whites.Item, 'Vit A': whites['Vitamin A (% Daily Value)'],
                          'Vit C': whites['Vitamin C (% Daily Value)'],
                          'Calcium': whites['Calcium (% Daily Value)'],
                          'Iron': whites['Iron (% Daily Value)']})

#VIT A, VIT C, CALCIUM, DAN IRON PADA WHOLE EGG
whole_vm = pd.DataFrame({'Item': whole.Item, 'Vit A': whole['Vitamin A (% Daily Value)'],
                         'Vit C': whole['Vitamin C (% Daily Value)'],
                         'Calcium': whole['Calcium (% Daily Value)'],
                         'Iron': grilled['Iron (% Daily Value)']})

#DV : DAILY VALUE

#RATA-RATA VIT A, VIT C, CALCIUM, DAN IRON PADA WHITE EGG
avg_whites_vita = round(whites_vm['Vit A'].mean(axis=0), 2)
avg_whites_vitc = round(whites_vm['Vit C'].mean(axis=0), 2)
avg_whites_calc = round(whites_vm['Calcium'].mean(axis=0), 2)
avg_whites_iron = round(whites_vm['Iron'].mean(axis=0), 2)
avg_whites_iron = round(whites_vm['Iron'].mean(axis=0), 2)
print("AVG OF VIT A, VIT C, CALCIUM, AND IRON IN WHITE EGG: ",
      avg_whites_vita, "%,", avg_whites_vitc, "%,", avg_whites_calc, "%, and", avg_whites_iron, "%.
")

#RATA-RATA VIT A, VIT C, CALCIUM, DAN IRON PADA WHOLE EGG
avg_whole_vita = round(whole_vm['Vit A'].mean(axis=0), 2)
avg_whole_vitc = round(whole_vm['Vit C'].mean(axis=0), 2)
avg_whole_calc = round(whole_vm['Calcium'].mean(axis=0), 2)
avg_whole_iron = round(whole_vm['Iron'].mean(axis=0), 2)
print("AVG OF VIT A, VIT C, CALCIUM, AND IRON IN WHOLE EGG: ",
      avg_whole_vita, "%,", avg_whole_vitc, "%,", avg_whole_calc, "%, and", avg_whole_iron, "%.")
```

```
AVG OF VIT A, VIT C, CALCIUM, AND IRON IN WHITE EGG:  3.54 %, 3.77 %, 17.0 %, and 15.23 %.
AVG OF VIT A, VIT C, CALCIUM, AND IRON IN WHOLE EGG:  12.58 %, 4.08 %, 20.42 %, and 16.0 %.
```

In [84]:

```
#membuat barchart

avg_vita = [avg_whites_vita, avg_whole_vita]
avg_vitc = [avg_whites_vitc, avg_whole_vitc]
avg_calc = [avg_whites_calc, avg_whole_calc]
avg_iron = [avg_whites_iron, avg_whole_iron]
index = ['White Egg', 'Whole Egg']
avg_vm = pd.DataFrame({'Avg Vit A': avg_vita,
                       'Avg Vit C': avg_vitc,
                       'Avg Calcium': avg_calc,
                       'Avg Iron': avg_iron}, index=index)
ax = avg_vm.plot.bar(rot=0, colormap='spring', figsize=(10,5))
ax.set_title("AVG VIT A, VIT C, CALCIUM AND IRON IN WHITES EGG AND WHOLE EGG")
for p in ax.patches:
    ax.annotate(str(p.get_height()), (p.get_x() * 1.005, p.get_height() * 1.005))
```



In [ ]: