In [1]:

```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
df = pd.read_csv("aviation_accident_data.csv")
```

Q1

In [2]:

```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23967 entries, 0 to 23966
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   date        23408 non-null  object 
 1   type        23933 non-null  object 
 2   operator    23963 non-null  object 
 3   fatalities  20029 non-null  float64
 4   country     23129 non-null  object 
 5   cat         23967 non-null  object 
 6   year        23906 non-null  float64
dtypes: float64(2), object(5)
memory usage: 1.3+ MB
```

In [3]:

```python
## Q1 there are 23967 in the data base
```

In [4]:

```python
df.isna().sum()
```

Out[4]:

```
date           559
type            34
operator         4
fatalities    3938
country        838
cat              0
year            61
dtype: int64
```

In [5]:

```python
Q2 there are 559 records missing in date, 34 in type, 4 in operator, 3938 in fatalities, 838 in country, 0 at cat, and 6
```

In [6]:

```python
gb_type = df.groupby('type')
gb_type_len = gb_type['cat'].apply(len)
filt = (gb_type_len >= 500)
filtered_gb_type = gb_type_len[filt]
filtered_gb_type
```

Out[6]:

```
type
Curtiss C-46A            564
Douglas C-47 (DC-3)      669
Douglas C-47A (DC-3)    1916
Douglas C-47B (DC-3)     592
Name: cat, dtype: int64
```

##Q3 Curtiss C-46A, Douglas C-47 (DC-3), Douglas C-47A (DC-3), Douglas C-47B (DC-3)

In [7]:

```
gb_state = df.groupby('country')
gb_type_ = gb_state['cat'].apply(len)
number_of_accidents_in_USA = gb_type_.loc['USA']
percent = (number_of_accidents_in_USA / 23967) * 100
print(percent)
```

18.26261109024909

##Q4 אחוז התאונות המתועדות שהתרחשו בארה"ב הוא 18.26%

In [8]:

```
gb_category = df.groupby('cat')
gb_fatalities= gb_category['fatalities'].sum()
gb_cat = gb_category.apply(len)
gb_fatalities/ gb_cat
```

Out[8]:

```
cat
A1      6.869089
A2      0.062149
C1      7.583122
C2      0.642857
H1     85.523810
H2      0.172053
I1      0.000000
I2      0.000000
O1      0.072464
O2      0.030769
U1      0.000000
dtype: float64
```

##Q5 קטגוריית תאונה היא הקטלנית ביותר בממוצע היא H

##Q6 part A

In [9]:

```
top_countries_df = df.groupby('country', as_index=False)['cat'].count()
top_countries_df.sort_values(by='cat', ascending=False, inplace=True)
top_countries_df.head()
```

Out[9]:

	country	cat
219	USA	4377
171	Russia	1422
216	U.K.	837
38	Canada	826
96	India	700

here we can see that the countries that expirenced the largest number of accidents are: USA, Russia, U.K., Canada and India

In [10]:

```
desired_types = ['Curtiss C-46A', 'Douglas C-47 (DC-3)', 'Douglas C-47A (DC-3)', 'Douglas C-47B (DC-3)']
desired_countries = ['USA', 'Russia', 'U.K.', 'Canada', 'India']

filt2 = (df['type'].isin(desired_types)) & (df['country'].isin(desired_countries))
filtered_countries_df = df[filt2]

filtered_countries_df
```

Out[10]:

	date	type	operator	fatalities	country	cat	year
1291	1942-08-23	Douglas C-47 (DC-3)	USAAF	12.0	U.K.	A1	1942.0
1293	1942-08-23	Douglas C-47 (DC-3)	USAAF	NaN	USA	A1	1942.0
1296	1942-08-26	Douglas C-47 (DC-3)	USAAF	NaN	USA	A1	1942.0
1309	1942-09-16	Douglas C-47 (DC-3)	USAAF	NaN	USA	O1	1942.0
1311	1942-09-19	Douglas C-47 (DC-3)	USAAF	7.0	USA	A1	1942.0
...	...	...	...	...	...	...	...
19367	2001-07-09	Douglas C-47A (DC-3)	Valiant Air Command	0.0	USA	A2	2001.0
21775	2012-11-09	Curtiss C-46A	Buffalo Airways	0.0	Canada	A2	2012.0
22412	2015-09-25	Curtiss C-46A	Buffalo Airways	0.0	Canada	A1	2015.0
23018	2018-07-21	Douglas C-47B (DC-3)	CAF, Highland Lakes Squadron	0.0	USA	A1	2018.0
23180	2019-05-03	Douglas C-47A (DC-3)	Buffalo Airways	0.0	Canada	A2	2019.0

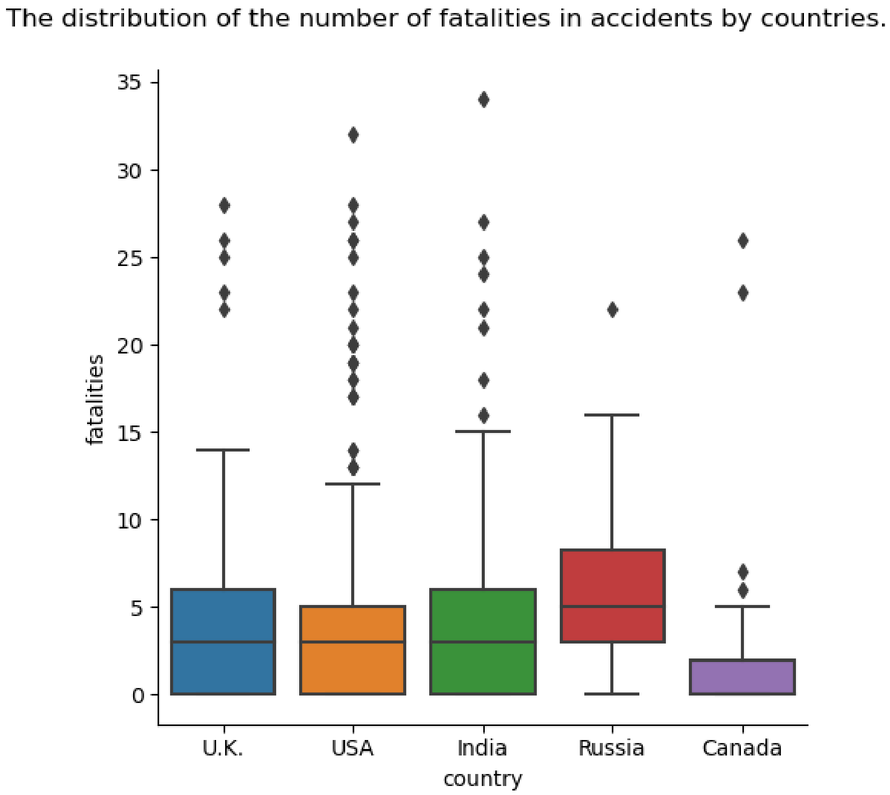
956 rows × 7 columns

In [11]:

```
graph = sns.catplot(x="country", y="fatalities", kind="box", data=filtered_countries_df);
graph.fig.suptitle('The distribution of the number of fatalities in accidents by countries.', y=1.05)
```

Out[11]:

```
Text(0.5, 1.05, 'The distribution of the number of fatalities in accidents by countries.')
```



In [12]:

```
##Q6 part B - looking at the box plot we can see that the country with the highest median of fatalities is Russia.
```

In [13]:

```
##Q7
```

In [14]:

```
df_without_missing_years = df.dropna(subset=['year'])
df_without_missing_years.head()
```

Out[14]:

	date	type	operator	fatalities	country	cat	year
61	1919-08-02	Caproni Ca.48	Caproni	14.0	Italy	A1	1919.0
62	1919-08-11	Felixstowe Fury	RAF	1.0	U.K.	A1	1919.0
63	1920-02-23	Handley Page O/7	Handley Page Transport	0.0	South Africa	A1	1920.0
64	1920-02-25	Handley Page O/400	Handley Page Transport	0.0	Sudan	A1	1920.0
65	1920-06-30	Handley Page O/400	Handley Page Transport	0.0	Sweden	A1	1920.0

In [15]:

```
accidents_by_year_df = df_without_missing_years.groupby('year', as_index=False)['cat'].count()
accidents_by_year_df
```

Out[15]:

	year	cat
0	1919.0	2
1	1920.0	4
2	1921.0	7
3	1922.0	3
4	1923.0	8
...	...	...
100	2019.0	245
101	2020.0	203
102	2021.0	183
103	2022.0	168
104	2023.0	56

105 rows × 2 columns

In [16]:

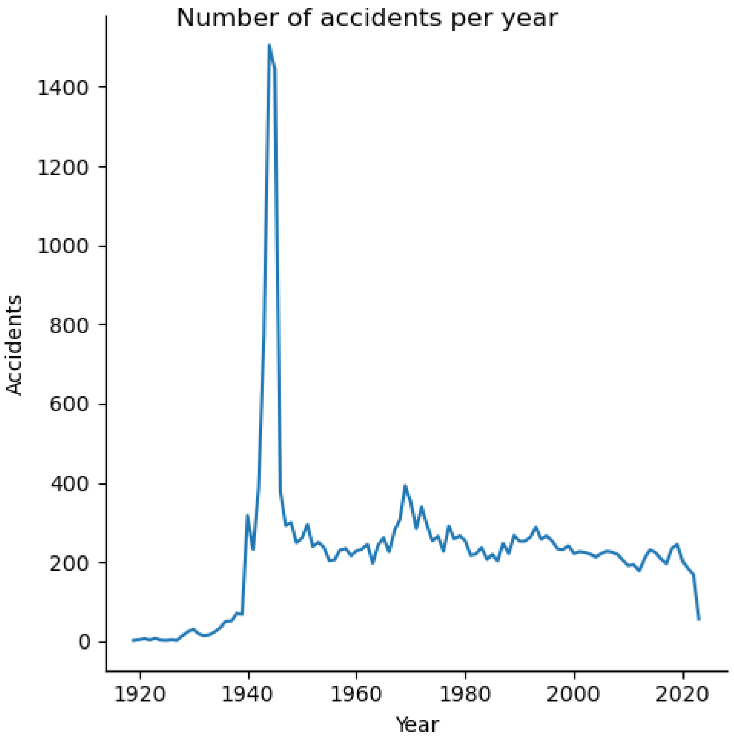
```
accidents_by_year = sns.relplot(x="year", y="cat", kind="line", data=accidents_by_year_df)
accidents_by_year.fig.suptitle('Number of accidents per year')
accidents_by_year.set(xlabel='Year', ylabel='Accidents')
```

Out[16]:

```
<seaborn.axisgrid.FacetGrid at 0x22ce5af6970>
```

In [17]:

```
plt.show()
```



##Q7 there is no correlation between the number of accidents in each year to the year itself. As shown in the graph - as the years go by there isn't a clear upward/ downward trend

##Q8 The 10 most dangerous planes are:

In [18]:

```
dangerous_planes_df = df.groupby('type', as_index=False)['cat'].count()
dangerous_planes_df.sort_values(by='cat', ascending=False, inplace=True)
top_dangerous_planes_df = dangerous_planes_df[:10]
top_dangerous_planes_df
```

Out[18]:

	type	cat
1710	Douglas C-47A (DC-3)	1916
1706	Douglas C-47 (DC-3)	669
1711	Douglas C-47B (DC-3)	592
1485	Curtiss C-46A	564
2433	Junkers Ju-52/3m	471
179	Antonov An-2R	391
1486	Curtiss C-46D	344
1944	Douglas Dakota III (DC-3)	262
1600	DHC-6 Twin Otter 300	258
1301	Cessna 208B Grand Caravan	247

In [19]:

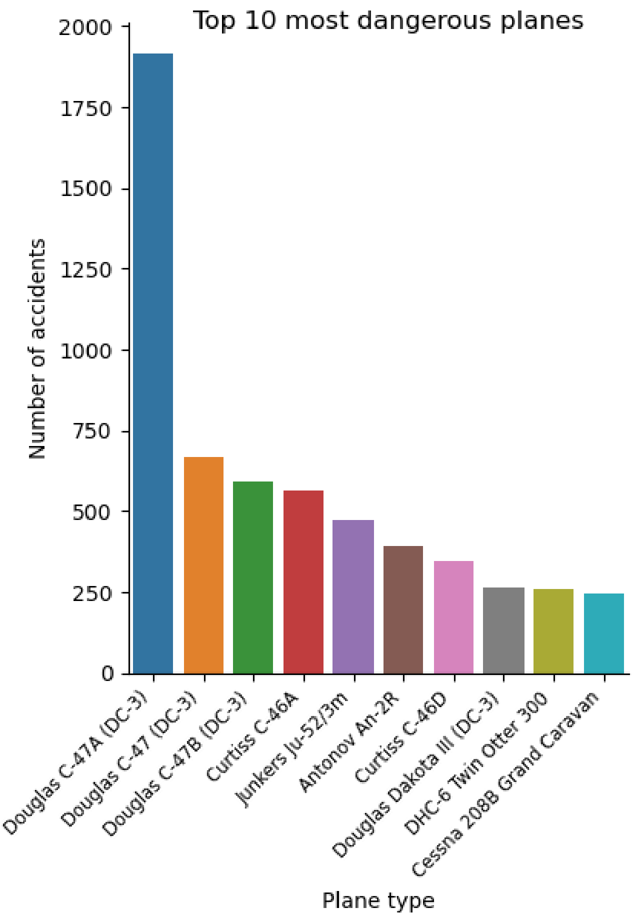
```
facetgrid_obj = sns.catplot(kind='bar', data=top_dangerous_planes_df , x='type', y='cat')
facetgrid_obj.set_xticklabels(rotation=45, ha='right', fontsize='small')
facetgrid_obj.fig.suptitle('Top 10 most dangerous planes')
facetgrid_obj.set(xlabel='Plane type', ylabel='Number of accidents')
```

Out[19]:

```
<seaborn.axisgrid.FacetGrid at 0x22ce5af63d0>
```

In [20]:

```
plt.show()
```



##Q9

In [21]:

```
most_dagerous_planes_lst = ['Douglas C-47A (DC-3)', 'Douglas C-47 (DC-3)', 'Douglas C-47B (DC-3)', 'Curtiss C-46A', 'Ju
                            'Antonov An-2R', 'Curtiss C-46D','Douglas Dakota III (DC-3)', 'DHC-6 Twin Otter 300','Cessna

filt3 = (df['type'].isin(most_dagerous_planes_lst))
# Get the unique operators that have at least one of the planes in the list
operators_with_specific_planes = df.loc[filt3, 'operator'].unique()

# Filter the DataFrame based on the unique operators
filtered_df = df[df['operator'].isin(operators_with_specific_planes)]
dangerous_operators_df = filtered_df.groupby('operator', as_index=False)['cat'].count()
dangerous_operators_df.sort_values(by='cat', ascending=False, inplace=True)
dangerous_operators_df.head(3)
```

Out[21]:

	operator	cat
1043	USAAF	2604
1046	USAF	1120
793	RAF	920

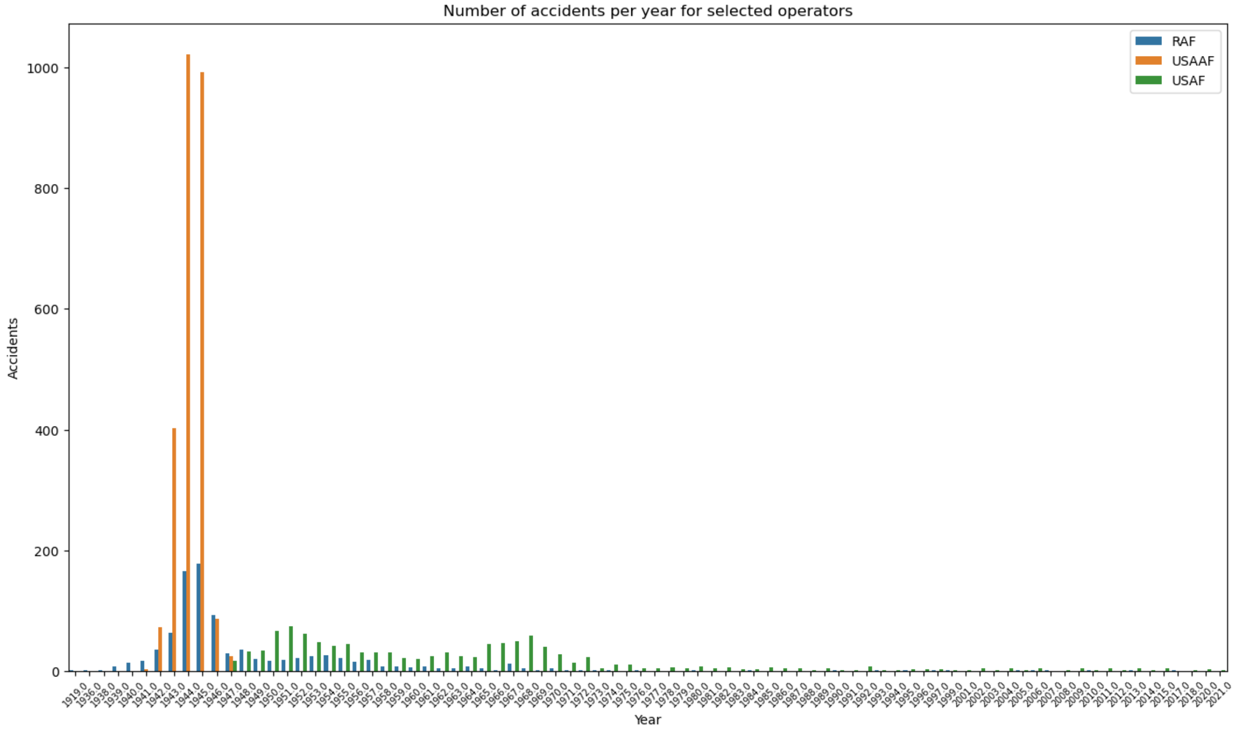
In [22]:

```
##Q9 The 3 most dangerous operators are USAAF with 2604 accidents, USAF with 1120 accidents, and RAF with 920 accidents
```

##Q10 part A

In [93]:

```
most_dangerous_operators = ['USAAF', 'USAF', 'RAF']
mask = df['operator'].isin(most_dangerous_operators)
df_selected_operators = df[mask]

# Count the number of occurrences of each category per year for selected operators
count_per_year = df_selected_operators.groupby(['operator', 'year'])['cat'].count().reset_index(name='count')
plt.figure(figsize=(16, 9))
accidents_by_year_operators = sns.barplot(x='year', y='count', hue='operator', data=count_per_year)
accidents_by_year_operators.set(xlabel='Year', ylabel='Accidents')
accidents_by_year_operators.set_title('Number of accidents per year for selected operators')
plt.xticks(rotation=45)
plt.xticks(fontsize=7)
plt.legend(loc='upper right')
plt.show()
```



Q10 part A

The number of accidents is getting smaller as the years go by for all three planes. One possible explaination could be that as the planes are getting older they are less in use by the operators and replaced by new aircrafts. Another explantion could be a dicrease in criminal activity that includes plane hijacking, sabotage, shoot down etc. which is a cause of some accidents.

Q10 part B

ישנן תצפיות חריגות בין השנים 1942 ל1945. הסבר אפשרי יכול להיות קיומה של מלחמת העולם השניה בה השתתף הן חיל האוויר הבריטי, והן חיל האוויר האמריקאי בהמון קרבות אוויר שגרמו לפגיעתם של מטוסים וקורבנות רבים.

part 2:

##Q1

In [33]:

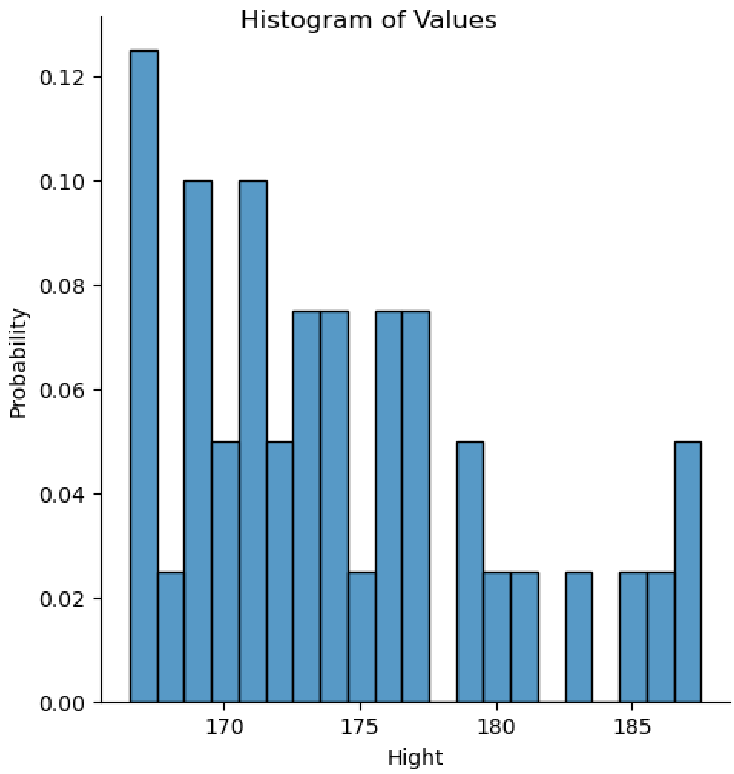
```
mu, sigma = 175, 6 # mean and standard deviation
s = np.random.normal(mu, sigma, 40)
facetgrid_obj = sns.displot(s, bins=np.unique(s), stat='probability', binwidth=1.0)
facetgrid_obj.set(xlabel='Hight', ylabel='Probability')
facetgrid_obj.fig.suptitle('Hight disterbution')
```

Out[33]:

```
Text(0.5, 0.98, 'Histogram of Values')
```

In [34]:

```
plt.show()
```



1A - this is a graph of probability distribution beacause we're given a population's hight median and standard diviation, there for our simulation is based on that data

In [39]:

```
mean_sample = s.mean()
mean_sample
```

Out[39]:

174.15689691010178

1B The value we received above is the mean of hights

1C in our graph we recieved right skewness with a unimodal. for a larger sample, for instance n = 1000, we expect to recieve a normal distribution, similar to the distribution in the population.

2A

H0 = The mean is 175 cm.

H1 = The mean in less than 175 cm.

In [46]:

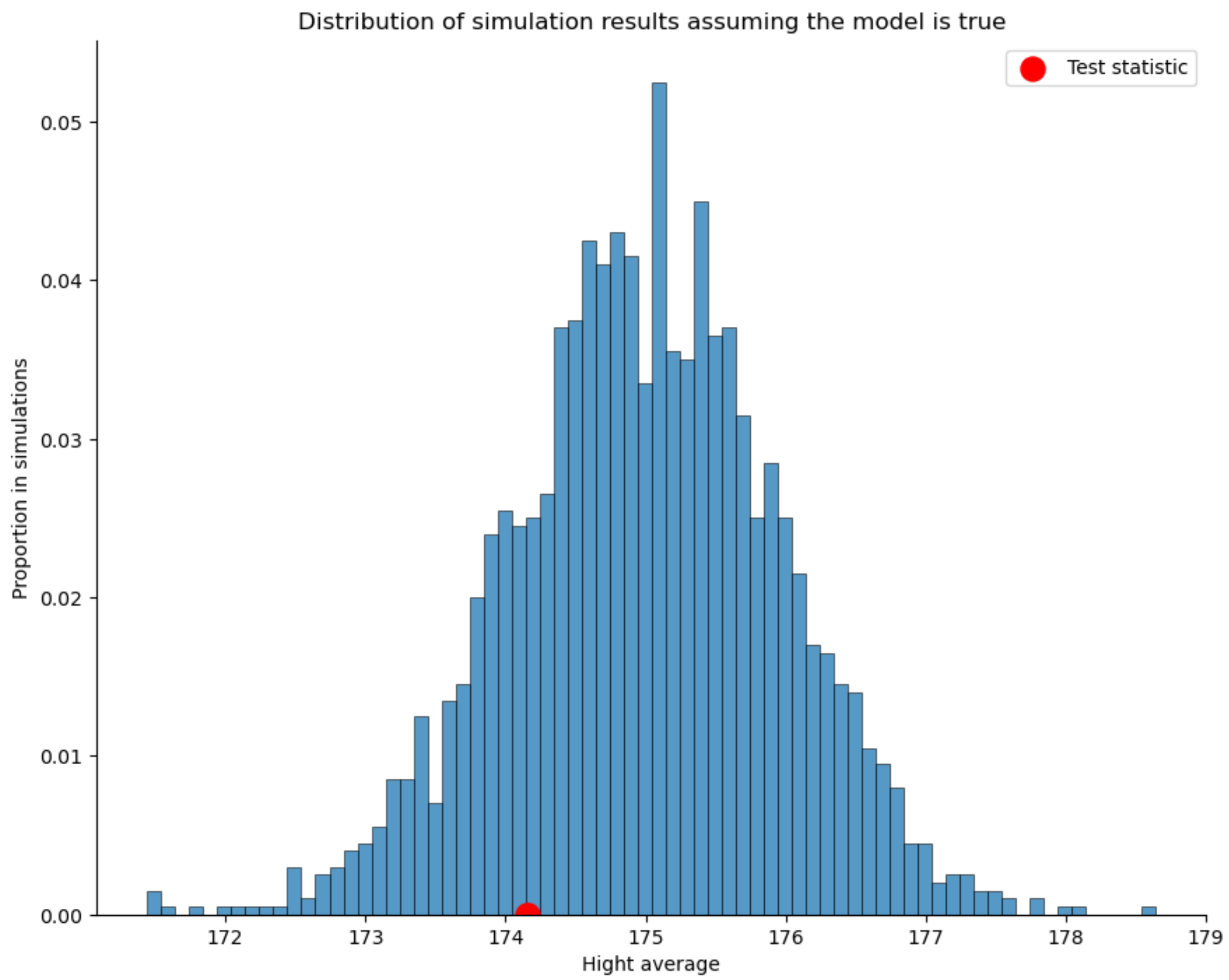
```
# simulate one value
def prob_mean_hight():
    mu, sigma = 175, 6 # mean and standard deviation
    s = np.random.normal(mu, sigma, 40)
    mean = s.mean()
    return mean

# run multiple simulations
num_repetitions = 2000
many_prob_mean_hight = np.array([prob_mean_hight() for i in range(num_repetitions)])
facetgrid_obj = sns.displot(many_prob_mean_hight, bins=np.unique(many_prob_mean_hight), stat='probability', binwidth = (
facetgrid_obj.fig.set_size_inches(10, 7)
facetgrid_obj.set(title='Distribution of simulation results assuming the model is true', xlabel=f'Hight average', ylabe

# Add a red point on the plot marking our data
facetgrid_obj.axes[0, 0].scatter(mean_sample, 0, s=150, color='red')  # draw observed value
facetgrid_obj.axes[0, 0].legend(['Test statistic'])
```

Out[46]:

```
<matplotlib.legend.Legend at 0x22ce72b7a30>
```

In [47]:

```
plt.show()
```



In [60]:

```
num_simulations_like_data_or_more_extreme = np.count_nonzero(many_prob_mean_hight <= mean_sample)
print (f'The p-value is {num_simulations_like_data_or_more_extreme/num_repetitions}')
```

```
The p-value is 0.191
```

```
Q2 Part C - the answer is located in the cell above
```

```
Q2 Part D - For both significance levels: 0.01, 0.1, we will not reject the null hypothesis because the red point is not part of the 0.1%/ 0.01% tail of the distribution.
```

```
##Q3
```

In [76]:

```
def one_mean(mu,sigma,n):
    s = np.random.normal(mu, sigma, n)
    return s.mean()

def get_p_value_heights(sample_heights,n,mean_0):

# run multiple simulations
    num_repetitions = 2000
    many_mean = np.array([one_mean(mean_0, 6 ,n) for i in range(num_repetitions)])
    mean_sample_for_simulation = sample_heights.mean()

    num_simulations_like_data_or_more_extreme = np.count_nonzero(many_mean <= mean_sample_for_simulation )
    return num_simulations_like_data_or_more_extreme/num_repetitions
```

##Q4

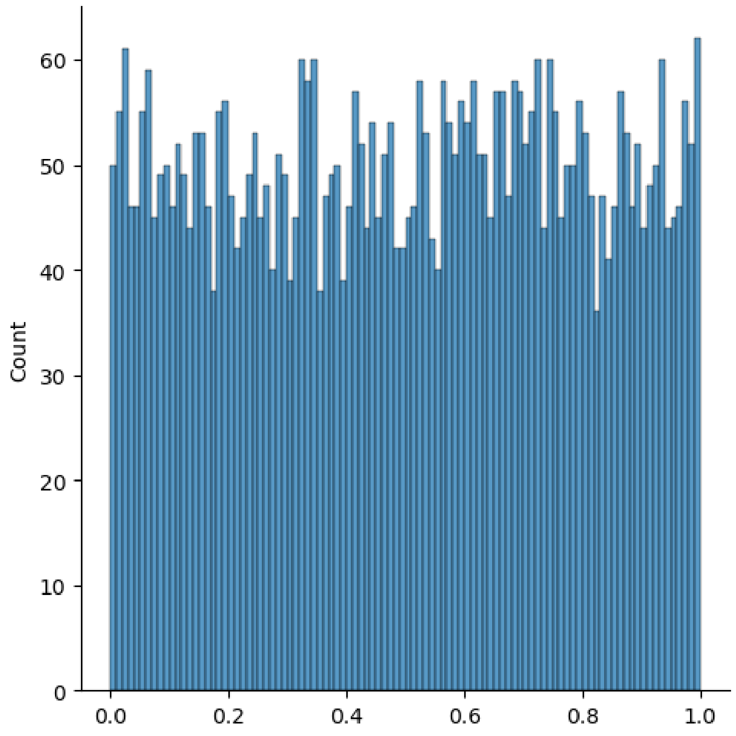
Q4 Part A

מתוך אוכלוסייה, דוגמים 40 אנשים (n) ובודקים מהו הp value של ממוצע הגבהים של האנשים במדגם ביחס לdata כלשהו (במקרה הזה- ממוצע גבהים 175 ס"מ, וסטיית תקן של 6 ס"מ). על התהליך הזה חוזרים 5000 פעמים (בכל פעם דוגמים מדגם אחר).

בהנחה שאנחנו יודעים שעבור האוכלוסייה ממנה אנו דוגמים, גובה אדם בוגר בה אכן מתפלג נורמלית עם ממוצע 175 ס"מ וסטיית תקן של 6 ס"מ, אנו משערות שהרוב הגדול של המדגמים ייצגו בצורה טובה את האוכלוסייה. כלומר, אנחנו צופות שעבור כל מדגם שכזה, ה statistic test ("הנקודה האדומה") יהיה קרוב ל175, ולכן ה p value עבור המדגם הזה לא יהיה מאוד קיצוני (לא מאוד קטן ולא מאוד גדול). לכן, כשנבנה היסטוגרמה של 5000 ערכי p value אלו, אנו צופות ש"הדליים" שמייצגים p values מאוד קטנים ומאוד גדולים יהיו ריקים.

In [77]:

```python
repetitions = 5000
multiple_p_value= np.array([get_p_value_heights((np.random.normal(175, 6, 40)),40,175) for i in range(repetitions)])

histograma = sns.displot(data = multiple_p_value, bins = 100) # Defining the bins
plt.show()
```



Part B Q4 C:

צורת ההתפלגות שנוצרה היא יוניפורמית, היא לא לחלוטין תואמת את ההשארה ההתחלתית שלנו. הופתענו לגלות שהתקבלו ערכי p – value קטנים מאוד (קרובים מאוד ל0) וגדולים מאוד (קרובים מאוד ל1).

In [82]:

```python
simulations_for_p_value = np.count_nonzero(multiple_p_value <= 0.05 )
simulations_for_p_value/repetitions
```

Out[82]:

```
0.052
```

Part B Q4 D

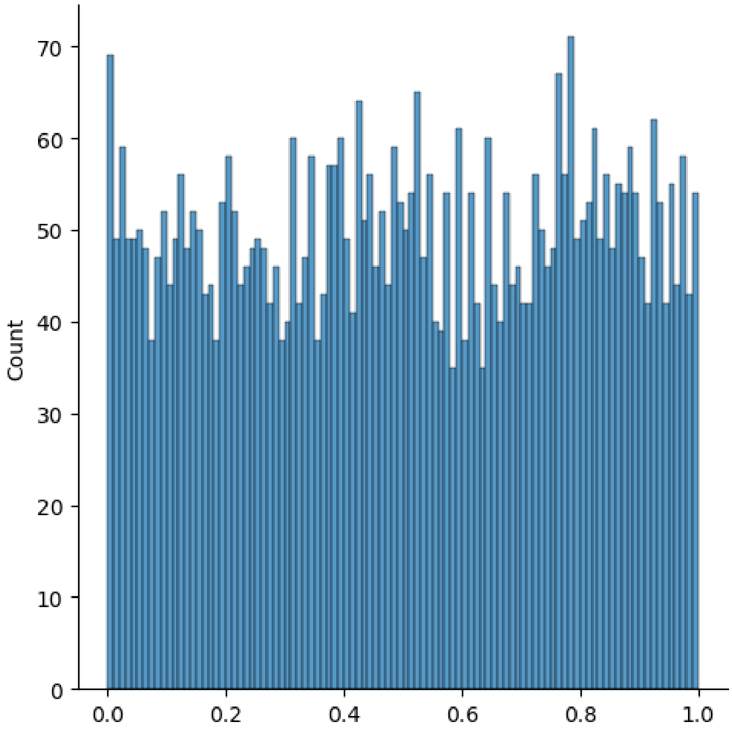
מאחר שקיבלנו בסעיף הקודם היסטוגרמה שמתפלגת יוניפורמית, הגיוני שחמישה אחוזים מערכי הp value קטנים מ0.05. אנו משערות שבפועל ההיסטוגרמה מתפלגת יוניפורמית מאחר שבכל פעם לקחנו מדגם מאוכלוסייה עם סטיית תקן שהיא לא אפס, ולכן המדגם לא בהכרח מייצג את הממוצע של האוכלוסייה, מה שגרם ל"רעש" בסימולציות. מאחר שהמדגם לא מייצג, הגיוני שסטטיסטי המבחן עבור כל מדגם ינוע לרוחב גרף ההתפלגות ולא בהכרח ימורכז באזור 175, ולכן גם ערכי הp value ישתנו בהתאם, מאחר שהp value מייצג את אחוז המקרים ששווים או קטנים מסטטיסטי המבחן מתוך סך המדידות, מה שמוביל להתפלגות היוניפורמית שראינו.

Part B Q5

אנו משערות שמאחר שכעת המדגמים שנלקחים גדולים יותר, לפי חוק המספרים הגדולים, הם יהיו גם

In [83]:

```
repetitions = 5000
multiple_p_value_new= np.array([get_p_value_heights((np.random.normal(175, 6, 200)),200,175) for i in range(repetitions

histograma = sns.displot(data = multiple_p_value_new, bins = 100) # Defining the bins
plt.show()
```



In [ ]: