NORTHWIND'S SALES

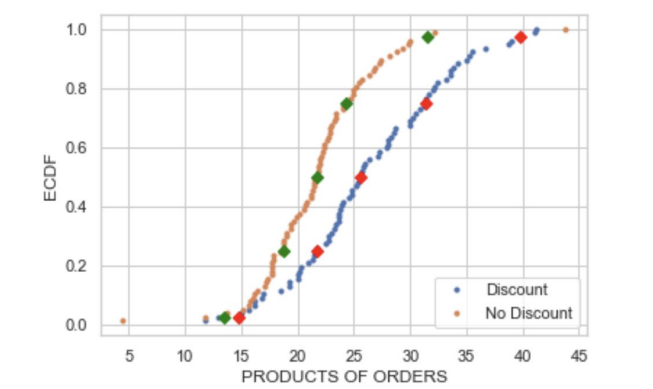
INVESTIGATION & SOLUTIONS

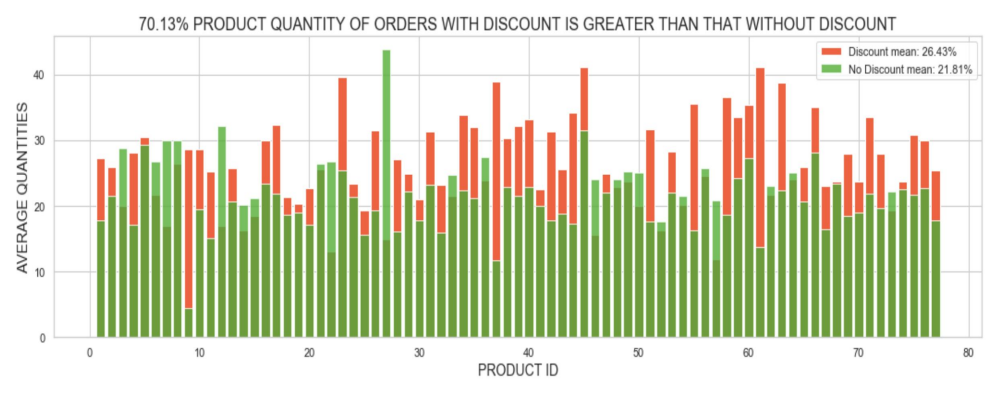
ONLINE-DS-PT-051319
Maia Ngo

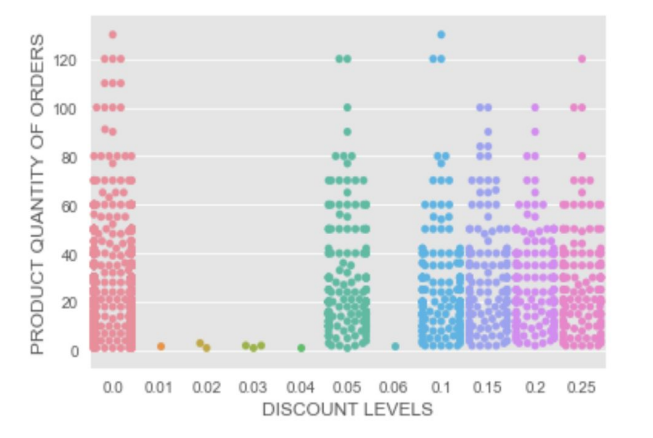
OVERVIEW

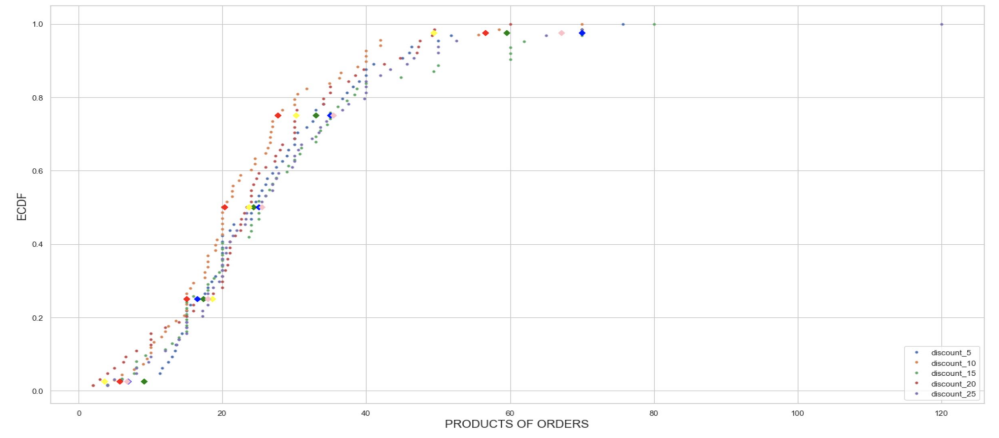
- IMPORTS & EXPORTS SPECIALTY FOODS FROM AROUND THE WORLD
  - 2 BACK OFFICES: UK & USA
  - 9 EMPLOYEES IN SALES DEPARTMENT
  - 77 PRODUCTS ARE DIVIDED INTO 8 CATEGORIES
  - PRODUCTS ARE SOLD TO 21 COUNTRIES.
  - 10 LEVELS OF DISCOUNT : 1%, 2%, 3%, 4%, 5%, 6%, 10%, 15%, 20%, 25%
  - SHIPPER COMPANIES: 3
  - SUPPLIERS: 29 COMPANIES FROM 16 COUNTRIES
- OVERVIEW OF CURRENT SITUATION IN SALES
- HOW TO USE CURRENT SALE DATA TO IMPROVE FUTURE REVENUE

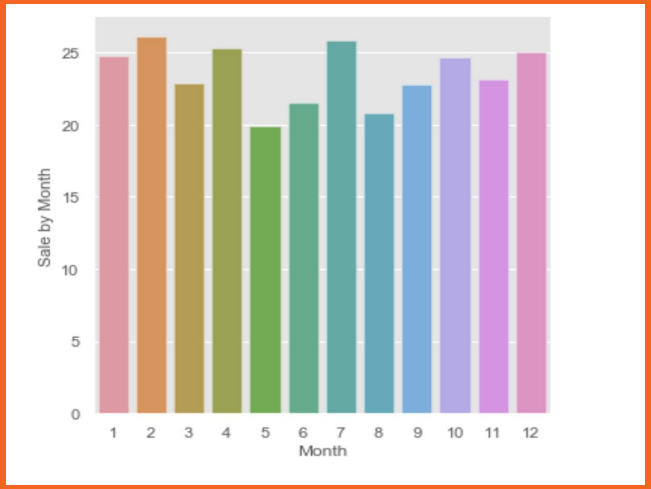
OBSERVATION

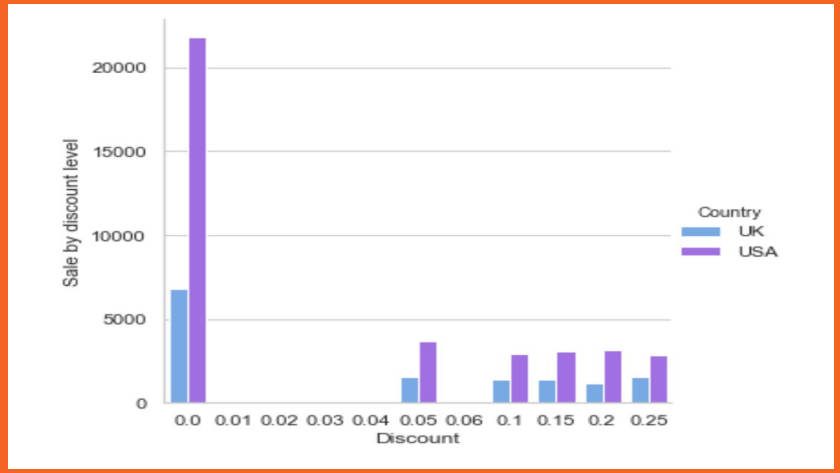


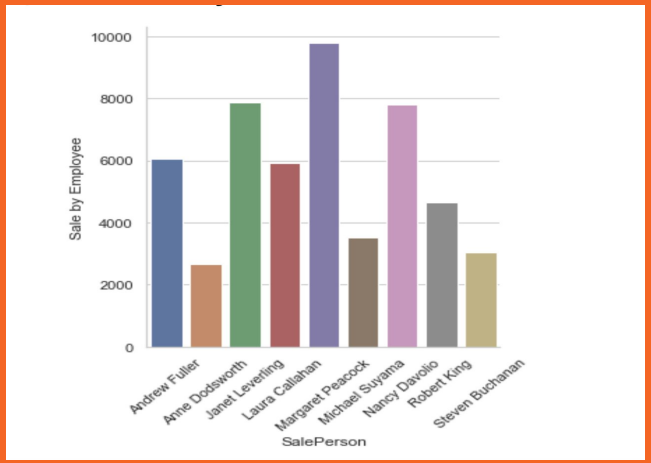


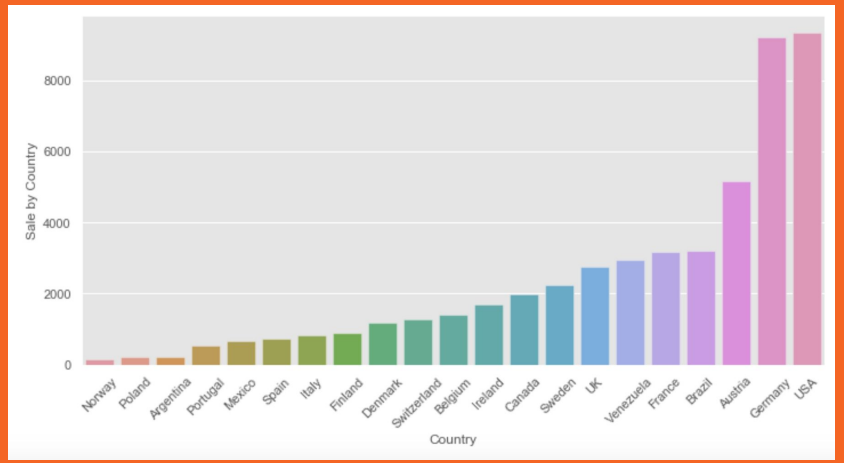





Sale by Month
Month
0
5
10
15
20
25
1
2
3
4
5
6
7
8
9
10
11
12


Sale by discount level
Discount
Country
UK
USA
0
5000
10000
15000
20000
0.0
0.01
0.02
0.03
0.04
0.05
0.06
0.1
0.15
0.2
0.25


Sale by Employee
SalePerson
0
2000
4000
6000
8000
10000
Andrew Fuller
Anne Dodsworth
Janet Leverling
Laura Callahan
Margaret Peacock
Michael Suyama
Nancy Davolio
Robert King
Steven Buchanan


Sale by Country
Country
0
2000
4000
6000
8000
Norway
Poland
Argentina
Portugal
Mexico
Spain
Italy
Finland
Denmark
Switzerland
Belgium
Ireland
Canada
Sweden
UK
Venezuela
France
Brazil
Austria
Germany
USA

EXPERIMENTS & TESTING

SALES BY DISCOUNT

-The discount program increases the product of orders.

- The discount levels effect on the product of order not significantly difference.

SALES BY COUNTRY

-The sales to 21 customer's countries are totally different in the product of orders.

- Need to check on small sales to see how we can increase it, may be cultures, consumer habits, or shipping, customer demand

SALES BY MONTH

-There is difference in product of orders between every month.

- we could take more advantages of seasonal events or holiday for sales, we need more effective promotion, right time, right amount.

SALES BY EMPLOYEES

-There is difference in product of orders between employees.

- Employees need motivation & more challenge

- Besides, marketing training is a need.

FURTHER STEPS

- Analyze on how effective of discount to products, categories of products
- Check how discount affect on sales by customer-countries
- Investigate for the optimal discount levels.
- Compare sales performance between UK & USA employees.
- Check on shipping time, processing time affect to product of orders
- Compare sales of products difference from different supplier.

SUGGESTION

Product demands

Learn customer's product desire; Expand sale to other countries; new promotion strategy: bonus products on orders

Optimizing SCM

Process time, shipping time, inventory.

Motivation & Improvement

Increase sales commission, Motivation by challenge & rewards.

MARKETING | SUPPLY CHAIN | EMPLOYEES

Change

Launching new product, own brand products, new designs especially new package, promotion for holidays or convenient gift

Purchasing, QA

Purchasing, expand sources & quality control

Set new target

Set sales targets, sale & marketing training.

THANK YOU FOR YOUR ATTENTION !

SUPPLEMENT FOR TECHNICAL

SUPPLEMENTARY

```python
# Computing Cohen's d function
def Cohen_d(experimental, control):

    diff = experimental.mean() - control.mean()
    n1, n2 = len(experimental), len(control)
    var1 = experimental.var()
    var2 = control.var()
    d = diff / np.sqrt((n1 * var1 + n2 * var2) / (n1 + n2))

    return abs(d)

```

```python
#%%time
mean_diff = OrderDetail_df[OrderDetail_df['Discount']!=0]['Quantity'].mean() - \
                        OrderDetail_df[OrderDetail_df['Discount']==0]['Quantity'].mean()
sample_diffs = []
counter = 0
for x in range(10000):
    discount_sample = OrderDetail_df.sample(replace = False, \
                                    n = len(OrderDetail_df[OrderDetail_df['Discount']!=0]))
    no_discount_sample = OrderDetail_df.drop(discount_sample.index, axis = 0)
    sample_diff = discount_sample['Quantity'].mean() - \
                                                no_discount_sample['Quantity'].mean()
    sample_diffs.append(sample_diff)
    if sample_diff > mean_diff:
        counter += 1
plt.hist(sample_diffs)
plt.axvline(mean_diff, color = 'b')
p = round(counter / 10000, 2)
print(p)
plt.title(f'p-value: {p}')

```

```
0.0
```

ANOVA TEST

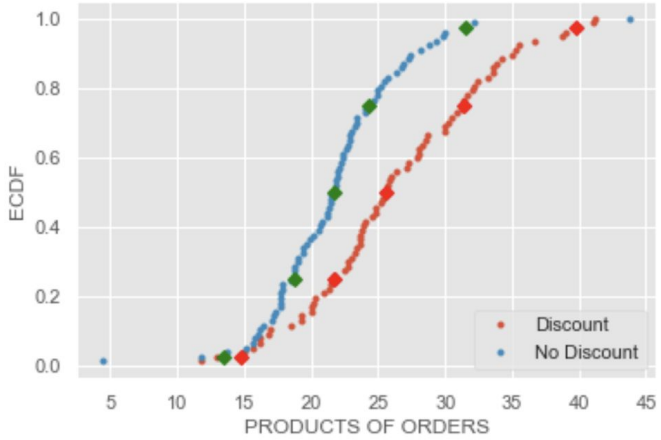
```python
formula = 'Quantity ~ C(Month)'
lm = ols(formula, sale_month).fit()
table = sm.stats.anova_lm(lm, typ=2)
print(table)
```

	sum_sq	df	F	PR(>F)
C(Month)	7395.980026	11.0	1.866405	0.039229
Residual	772003.656168	2143.0	NaN	NaN

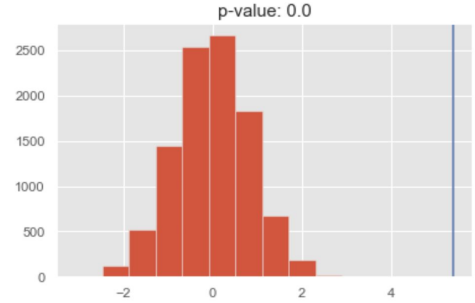
ANOVA TEST

```python
formula = 'Quantity ~ C(SalePersonID)'
lm = ols(formula, empl_sale_df).fit()
table = sm.stats.anova_lm(lm, typ=2)
print(table)
```

	sum_sq	df	F	PR(>F)
C(SalePersonID)	4643.183282	8.0	1.607646	0.11745
Residual	774756.452913	2146.0	NaN	NaN




```
Text(0.5, 1.0, 'p-value: 0.0')
```




```
d = Cohen_d(discount_sample['Quantity'], no_discount_sample['Quantity'])
print('Cohen_d = ', d)
```

```
Cohen_d =  0.07047343170013781
```

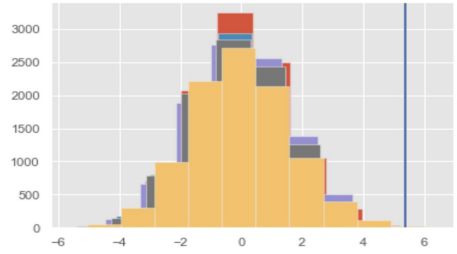
```
def ecdf (data):
    n = len(data)
    x = np.sort(data)
    y = np.arange(1,(n+1))/n
    return x, y
```

```
discount = OrderDetail_df[OrderDetail_df['Discount']!=0].\
                                                groupby('ProductId')['Quantity'].mean()
no_discount = OrderDetail_df[OrderDetail_df['Discount']==0].\
                                                groupby('ProductId')['Quantity'].mean()
discount_a = np.array(discount)
no_discount_a = np.array(no_discount)

x_discount, y_discount = ecdf(discount_a)
x_no_discount, y_no_discount = ecdf(no_discount_a)

# Specify array of percentiles
percentiles = np.array([2.5, 25, 50, 75, 97.5])
ptiles_discount = np.percentile(discount_a, percentiles)
ptiles_no_discount = np.percentile(no_discount_a, percentiles)
# Plot all ECDFs on the same plot:
_=plt.plot(x_discount, y_discount, marker = '.', linestyle = 'none')
_=plt.plot(x_no_discount, y_no_discount, marker = '.', linestyle = 'none')
_=plt.plot(ptiles_discount, percentiles/100, marker = 'D', \
                                        color = 'red', linestyle = 'none')
_=plt.plot(ptiles_no_discount, percentiles/100, marker = 'D', \
                                          color = 'green', linestyle = 'none')
# Annotate the plot:
_=plt.legend(('Discount', 'No Discount'), loc = 'lower right')
_=plt.xlabel('PRODUCTS OF ORDERS')
_=plt.ylabel('ECDF')
plt.show()
```

In [441]:

```
# sample and test for some level of discounts versus no discount
discount_levels = np.array([0.05, 0.1, 0.15, 0.2, 0.25])
for level in discount_levels:
    sample_diffs, p = sample_and_test(level)
    print(f'p-value: {p}')
    plt.hist(sample_diffs)
    plt.axvline(mean_diff, color = 'b')
```

```
p-value: 0.0
p-value: 0.01
p-value: 0.0
p-value: 0.0
p-value: 0.0
```



FUNCTIONS

```
discounts_sig_df = pd.DataFrame(columns=['Discount %', 'p value','Null Hypothesis', 'Cohens d'], index=None)
for level in discount_levels:
    result = sample_one_and_ttest(level, significant_level=0.5)
    discounts_sig_df = discounts_sig_df.append({'Discount %': str(level*100)+'%',
                                                'p value': result[1],
                                                'Null Hypothesis': 'Reject' if result[0] else 'Fail to reject'
                                                'Cohens d': result[2] if result[0] else np.nan},
                                                ignore_index=True)

discounts_sig_df
```

	Discount %	p value	Null Hypothesis	Cohens d
0	5.0%	0.426654	Reject	0.062433
1	10.0%	0.306263	Reject	0.082769
2	15.0%	0.899719	Fail to reject	NaN
3	20.0%	0.122425	Reject	0.129033
4	25.0%	0.392527	Reject	0.072841

```
discount_levels = np.array([0.05, 0.1, 0.15, 0.2, 0.25])
comb = itertools.combinations(discount_levels, 2)
discount_levels_df = pd.DataFrame(columns=['Discount %', 'p value','Null Hypothesis','Cohens d'], index=None)

for i in comb:
    result = sample_one_and_ttest(i[0], i[1], 0.5)
    discount_levels_df = discount_levels_df.append({'Discount %': str(i[0]*100)+'% vs '+str(i[1]*100)+'%',
                                                    'p value': result[1],
                                                    'Null Hypothesis': 'Reject' if result[0] else 'Fail to rej
                                                    'Cohens d': result[2] if result[0] else np.nan},
                                                    ignore_index=True)

discount_levels_df.sort_values('Cohens d', ascending=False)
```

	Discount %	p value	Null Hypothesis	Cohens d
8	15.0% vs 25.0%	0.041000	Reject	0.232738
6	10.0% vs 25.0%	0.060520	Reject	0.208672
7	15.0% vs 20.0%	0.104535	Reject	0.182595
2	5.0% vs 20.0%	0.110155	Reject	0.172633
4	10.0% vs 15.0%	0.202932	Reject	0.140621
0	5.0% vs 10.0%	0.472261	Reject	0.076104
1	5.0% vs 15.0%	0.833171	Fail to reject	NaN
3	5.0% vs 25.0%	0.708094	Fail to reject	NaN
5	10.0% vs 20.0%	0.510425	Fail to reject	NaN
9	20.0% vs 25.0%	0.895094	Fail to reject	NaN

```
# function to perform t-test on two samples
# and compare p-value with provided significant level to reject or not
def perform_ttest(sample1, sample2, significant_level = 0.025):
    # perform t-test to calculate p-value
    st, p = stats.ttest_ind(sample1, sample2)

    # calculate Cohen_d
    d = Cohen_d(sample1, sample2)

    # compare p-value to reject
    reject = True
    if p >= significant_level:
        reject = False

    # return result
    return reject, p, d


# using for calculate cohen_d:
def sample_one_and_ttest(discount_level_1, discount_level_2 = 0.0, significant_level = 0.025):
    # generate sample for discount level 1
    size = len(OrderDetail_df[OrderDetail_df['Discount']==discount_level_1])
    discount_level_1_sample = OrderDetail_df.sample(replace = False, n = size)

    # sample for discount level 2
    size = len(OrderDetail_df[OrderDetail_df['Discount']==discount_level_2])
    discount_level_2_sample = OrderDetail_df.sample(replace = False, n = size)

    # perform t-test and calculate p-value, Cohen_d
    result = perform_ttest(discount_level_1_sample['Quantity'], discount_level_2_sample['Quantity'], \
                                                                                    significant_level)

    # return result - (reject, p, d)
    return result
```

```
# function to sample data and test the difference of mean to calculate p-value for each pair of discounts
def sample_and_test(discount_level_1, discount_level_2 = 0.0, reps = 10000):
    # calculate the difference of means of each level
    mean_diff = OrderDetail_df[OrderDetail_df['Discount']==discount_level_1]['Quantity'].mean() - \
                OrderDetail_df[OrderDetail_df['Discount']==discount_level_2]['Quantity'].mean()

    # list to store all difference means of all samples
    sample_diffs = []

    # count when sample mean difference > original mean difference
    counter = 0

    # loop reps times to generate samples
    for x in range(reps):
        # generate sample for discount level 1
        size = len(OrderDetail_df[OrderDetail_df['Discount']==discount_level_1])
        discount_level_1_sample = OrderDetail_df.sample(replace = False, n = size)

        # generate supplement sample of sample for discount level 1
        supplement_sample = OrderDetail_df.drop(discount_level_1_sample.index, axis = 0)

        # calculate the mean difference of the two samples, and add it to the list
        sample_diff = discount_level_1_sample['Quantity'].mean() - supplement_sample['Quantity'].mean()
        sample_diffs.append(sample_diff)

        # compare same mean difference with original mean difference
        if sample_diff > mean_diff:
            counter += 1

    # calculate p-value
    p = round(counter / reps, 2)

    # return sample mean difference list and p-value
    return sample_diffs, p
```