In [196…

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sympy import *
from scipy.optimize import fsolve
from scipy.stats import *
from sklearn.preprocessing import StandardScaler,MinMaxScaler
import seaborn as sns
from statsmodels.api import *
import matplotlib.pyplot as plt
from matplotlib.ticker import MaxNLocator,MultipleLocator
##################################################
def checker(date):
    if date.month in [1, 12]:
        return True
    return False

def get_year(date):
    return date.year

def to_float(x):
    x = float(x.replace('.','').replace(',', '.'))
    return x

def to_value(x):
    x = float(x[:-1].replace('.','').replace(',', '.'))*1000
    return x

def to_perc(x):
    return float(x[:-1].replace('.','').replace(',', '.'))/100
```

In [197…

```
databook = pd.read_excel('Ozon_Databook_Q3_24.xlsx', 'Public Databook', header =
databook = databook.drop(list(range(0,4))+list(range(8,12))+list(range(15,18))+[
databook= databook.set_index('Unnamed: 0')

for i in range(4):
    databook.insert(i,f'Q{i+1}_18',databook['FY_2018']*(i+1)/10)

for i in range(4):
    databook.insert(i+4,f'Q{i+1}_19',databook['FY_2019']*(i+1)/10)

databook = databook.drop([i for i in databook.columns if 'F' in i]+['Unnamed: 23

databook = databook.T
databook.columns.name = 'Quarter'
data_cols = databook.columns
databook
```

Out[197…

Quarter	GMV incl. services	Share of Marketplace, as % of GMV incl. services	Number of orders, millions	Number of active buyers, millions	Total revenue	(Loss)/profit for the period	
Q1_18	4188.818931	0.000871	1.528649	0.480000	3722.0	-566.1	
Q2_18	8377.637862	0.001742	3.057297	0.960000	7444.0	-1132.2	-
Q3_18	12566.456792	0.002614	4.585946	1.440000	11166.0	-1698.3	-
Q4_18	16755.275723	0.003485	6.114594	1.920000	14888.0	-2264.4	-
Q1_19	8081.486742	0.017378	3.181056	0.790000	6010.4	-1936.3	-
Q2_19	16162.973484	0.034755	6.362112	1.580000	12020.8	-3872.6	-
Q3_19	24244.460225	0.052133	9.543168	2.370000	18031.2	-5808.9	-
Q4_19	32325.946967	0.069511	12.724224	3.160000	24041.6	-7745.2	-
Q4_20	75847.502151	0.522714	29.610823	13.760362	37751.0	-9407.0	-
Q1_21	74208.000000	0.583574	34.100000	16.000000	33407.0	-6734.0	-
Q2_21	88957.000000	0.621089	40.874333	18.400000	37018.0	-15233.0	-
Q3_21	108290.000000	0.666644	56.166153	21.293626	41492.0	-14018.0	-1
Q4_21	176805.000000	0.676565	92.100000	25.600000	66298.0	-20794.0	-1
Q1_22	177449.000000	0.704078	92.987774	28.700000	63579.0	-19055.0	-
Q2_22	170647.000000	0.760652	90.218180	30.655521	58514.0	-7202.0	
Q3_22	188125.175674	0.781730	107.540000	32.700000	61396.0	-20718.0	
Q4_22	296019.161141	0.783617	174.628703	35.169559	93626.0	-11212.0	
Q1_23	303047.672944	0.795098	179.312166	37.035248	93250.0	10656.0	
Q2_23	372627.819212	0.826189	208.668170	39.463483	94164.0	-13087.0	
Q3_23	450819.466075	0.834000	251.118775	42.443123	108963.0	-22055.0	-
Q4_23	625781.712083	0.848940	326.567897	46.089422	127914.0	-18179.0	
Q1_24	570177.000000	0.852000	305.300000	48.952615	122931.0	-13166.0	
Q2_24	633163.000000	0.861000	334.800000	51.100000	122530.0	-27971.0	
Q3_24	718303.000000	0.859000	371.500000	53.500000	153693.0	-740.0	1

24 rows × 21 columns

In [198…

```
databook.columns = [i if i[len(i)-1]!=' ' else i[:-1] for i in databook.columns]
databook.columns = [i if i[len(i)-1]!=' ' else i[:-1] for i in databook.columns]
databook.columns
```

Out[198…
```
Index(['GMV incl. services',
       'Share of Marketplace, as % of GMV incl. services',
       'Number of orders, millions', 'Number of active buyers, millions',
       'Total revenue', '(Loss)/profit for the period', 'Adjusted EBITDA',
       'Total non-current assets', 'Total current assets',
       'Cash and cash equivalents', 'Total assets', 'Total equity',
       'Total non-current liabilities', 'Total current liabilities',
       'Total liabilities', 'Total equity and liabilities',
       'Movements in working capital1',
       'Net cash (used in) / generated from operating activities1',
       'Capital expenditures',
       'Net cash (used in)/ generated from investing activities',
       'Net cash(used in)/ generated from financing activities2'],
      dtype='object')
```

In [199…
```
exog_cols  = [ i for i in databook.columns if i != 'Number of orders, millions']

# Определяем независимые переменные (X) и зависимую переменную (Y)
exog = databook[exog_cols]


scale = MinMaxScaler()
exog = scale.fit_transform(exog)
# Определяем независимые переменные (X) и зависимую переменную (Y)
Y = databook['Number of orders, millions']

#exog = add_constant(exog)
# Строим модель с помощью метода наименьших квадратов (OLS)
model = OLS(Y,exog[:,[i for i in range(20) if i not in [1,3,4,5,6,11,12,13,17,19

# Выводим результаты модели
sns.heatmap(np.corrcoef(exog[:,[i for i in range(20) if i not in [1,3,4,5,6,11,1

model.summary()
```

Out[199…

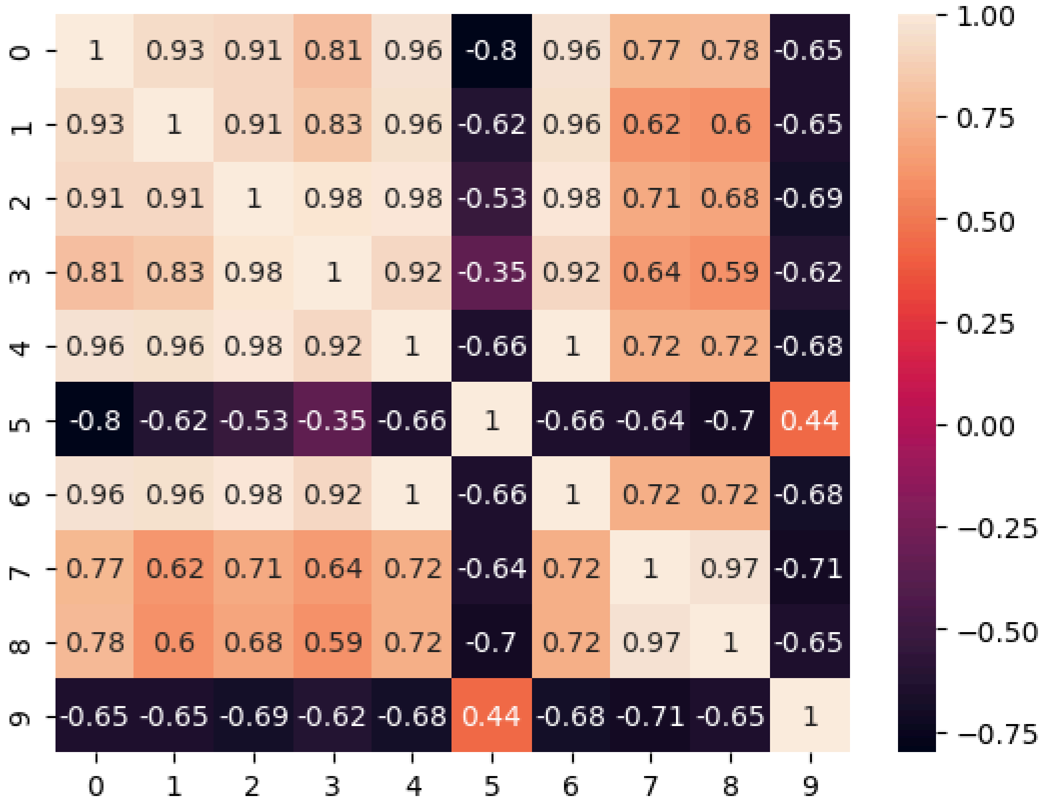
OLS Regression Results

Dep. Variable:	Number of orders, millions	**R-squared (uncentered):**	1.000
Model:	OLS	**Adj. R-squared (uncentered):**	1.000
Method:	Least Squares	**F-statistic:**	1.299e+04
Date:	Вс, 08 дек 2024	**Prob (F-statistic):**	5.57e-26
Time:	23:19:09	**Log-Likelihood:**	-47.150
No. Observations:	24	**AIC:**	114.3
Df Residuals:	14	**BIC:**	126.1
Df Model:	10		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
x1	268.4647	21.973	12.218	0.000	221.338	315.591
x2	107.9043	12.790	8.436	0.000	80.472	135.337
x3	416.9945	55.745	7.480	0.000	297.434	536.555
x4	-181.0222	26.295	-6.884	0.000	-237.418	-124.626
x5	1.067e+07	4.58e+06	2.328	0.035	8.4e+05	2.05e+07
x6	-42.9448	8.465	-5.073	0.000	-61.101	-24.789
x7	-1.067e+07	4.58e+06	-2.328	0.035	-2.05e+07	-8.41e+05
x8	-50.4180	9.938	-5.073	0.000	-71.733	-29.103
x9	72.6458	14.108	5.149	0.000	42.388	102.904
x10	17.7192	4.438	3.993	0.001	8.201	27.238

Omnibus:	0.209	**Durbin-Watson:**	2.228
Prob(Omnibus):	0.901	**Jarque-Bera (JB):**	0.400
Skew:	0.139	**Prob(JB):**	0.819
Kurtosis:	2.432	**Cond. No.**	1.91e+07

Notes:
[1] R^2 is computed without centering (uncentered) since the model does not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[3] The smallest eigenvalue is 1.21e-13. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.



	0	1	2	3	4	5	6	7	8	9
0	1	0.93	0.91	0.81	0.96	-0.8	0.96	0.77	0.78	-0.65
1	0.93	1	0.91	0.83	0.96	-0.62	0.96	0.62	0.6	-0.65
2	0.91	0.91	1	0.98	0.98	-0.53	0.98	0.71	0.68	-0.69
3	0.81	0.83	0.98	1	0.92	-0.35	0.92	0.64	0.59	-0.62
4	0.96	0.96	0.98	0.92	1	-0.66	1	0.72	0.72	-0.68
5	-0.8	-0.62	-0.53	-0.35	-0.66	1	-0.66	-0.64	-0.7	0.44
6	0.96	0.96	0.98	0.92	1	-0.66	1	0.72	0.72	-0.68
7	0.77	0.62	0.71	0.64	0.72	-0.64	0.72	1	0.97	-0.71
8	0.78	0.6	0.68	0.59	0.72	-0.7	0.72	0.97	1	-0.65
9	-0.65	-0.65	-0.69	-0.62	-0.68	0.44	-0.68	-0.71	-0.65	1

In [200…

```
exog_cols = np.array(exog_cols)[[i for i in range(20) if i not in [1,3,4,5,6,11,
```

In [201…

```
exog_cols
```

Out[201…

```
array(['GMV incl. services', 'Number of active buyers, millions',
       'Total current assets', 'Cash and cash equivalents',
       'Total assets', 'Total equity', 'Total equity and liabilities',
       'Movements in working capital1',
       'Net cash (used in) / generated from operating activities1',
       'Net cash (used in)/ generated from investing activities'],
      dtype='<U57')
```

In [202…

```
print(*exog_cols,sep='\n')
```

```
GMV incl. services
Number of active buyers, millions
Total current assets
Cash and cash equivalents
Total assets
Total equity
Total equity and liabilities
Movements in working capital1
Net cash (used in) / generated from operating activities1
Net cash (used in)/ generated from investing activities
```

In [203…

```
stat, p_value = shapiro(model.resid)
print("Статистика Shapiro-Wilk:", stat)
print("p-значение:", p_value)

if p_value > 0.05:
    print("Распределение данных похоже на нормальное")
else:
    print('Распределение данных отличается от нормального')
```

```
Статистика Shapiro-Wilk: 0.9829087422813559
p-значение: 0.9428080635525465
Распределение данных похоже на нормальное
```

In [204…

```
dw_stat = stats.durbin_watson(model.resid)
print(f"Статистика Дарбина-Уотсона: {dw_stat}")

if dw_stat == 2:
    print('Остатки случайны, автокорреляции нет, и модель хорошо описывает данны
elif dw_stat < 2:
    print('Положительная автокорреляция. Это может означать, что модель не учла 
else:
    print('Отрицательная автокорреляция. Это редко встречается в эконометрически
```

```
Статистика Дарбина-Уотсона: 2.227518413145096
Отрицательная автокорреляция. Это редко встречается в эконометрических данных, но
может наблюдаться в некоторых временных рядах, где данные колеблются вокруг средн
его значения.
```

In [205…

```
r = 1 - dw_stat/2

X = model.model.exog
Y = model.model.endog

model1 = OLS(Y, X).fit()
residuals = model1.resid

# Шаг 2: Оценка коэффициента автокорреляции
rho = np.corrcoef(residuals[1:], residuals[:-1])[0, 1]

# Итерационный процесс
for _ in range(10):  # Максимум 10 итераций, можно остановиться раньше, если раз
    Y_transformed = Y[1:] - rho * Y[:-1]
    X_transformed = X[1:] - rho * X[:-1]

    # Переоценка модели на преобразованных данных
    model1 = OLS(Y_transformed, X_transformed).fit()
    residuals = model1.resid

    # Обновляем rho и проверяем условие остановки
    new_rho = np.corrcoef(residuals[1:], residuals[:-1])[0, 1]
    if abs(new_rho - rho) < 1e-5:  # Условие остановки
        break
    rho = new_rho

print("Коэффициент автокорреляции rho:", rho)
print("Модель со схемой AR(1):", model1.summary())
```

```
Коэффициент автокорреляции rho: -0.10323233299995692
Модель со схемой AR(1):                                    OLS Regression Results
================================================================================
======
Dep. Variable:                      y   R-squared (uncentered):
1.000
Model:                            OLS   Adj. R-squared (uncentered):
1.000
Method:                 Least Squares   F-statistic:                          1.4
87e+04
Date:                Вс, 08 дек 2024   Prob (F-statistic):                    1.
11e-24
Time:                        23:19:10   Log-Likelihood:                        -
45.276
No. Observations:                  23   AIC:
110.6
Df Residuals:                      13   BIC:
121.9
Df Model:                          10
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
x1           268.2722     22.929     11.700      0.000     218.737     317.808
x2           108.6922     12.542      8.667      0.000      81.598     135.787
x3           420.8912     56.984      7.386      0.000     297.785     543.998
x4          -181.3935     27.078     -6.699      0.000    -239.891    -122.896
x5          1.055e+07   4.71e+06      2.241      0.043    3.78e+05    2.07e+07
x6           -43.2630      8.747     -4.946      0.000     -62.159     -24.367
x7         -1.055e+07   4.71e+06     -2.241      0.043   -2.07e+07   -3.78e+05
x8           -51.8164     10.066     -5.148      0.000     -73.563     -30.070
x9            74.7095     14.021      5.329      0.000      44.420     104.999
x10           17.5085      4.793      3.653      0.003       7.154      27.863
==============================================================================
Omnibus:                        0.375   Durbin-Watson:                   2.184
Prob(Omnibus):                  0.829   Jarque-Bera (JB):                0.523
Skew:                           0.126   Prob(JB):                        0.770
Kurtosis:                       2.305   Cond. No.                     2.10e+07
==============================================================================

Notes:
[1] R² is computed without centering (uncentered) since the model does not contai
n a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[3] The smallest eigenvalue is 1.2e-13. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

In [206…

```
# Пример кода для форматирования чисел в выводе
formatted_params = model.params.apply(lambda x: f"{x:.2f}")  # Округление до дву
print(f'\nКоэффициенты b при каждом члене равны соответственно:\n{formatted_para
```

```
Коэффициенты b при каждом члене равны соответственно:
x1            268.46
x2            107.90
x3            416.99
x4           -181.02
x5       10673243.42
x6            -42.94
x7      -10673510.31
x8            -50.42
x9             72.65
x10            17.72
dtype: object .
```

In [207…

```
# Пример кода для форматирования чисел в выводе
formatted_params = model.pvalues.apply(lambda x: f"{x:.2f}")  # Округление до дв
print(f'\nP-значения коэффициентов b при каждом члене равны соответственно:\n{fo
```

```
P-значения коэффициентов b при каждом члене равны соответственно:
x1     0.00
x2     0.00
x3     0.00
x4     0.00
x5     0.04
x6     0.00
x7     0.04
x8     0.00
x9     0.00
x10    0.00
dtype: object .
```

In [208…

```
print(f'{model.f_pvalue:.2f}')
```

```
0.00
```

In [209…

```
from scipy.stats import spearmanr

corr, p_value = spearmanr(model.resid, model.model.exog[:, 1])
print(f"Spearman correlation: {corr}")
print(f"p-value: {p_value}")

if p_value < 0.05:
    print("Признаки гетероскедастичности обнаружены.")
else:
    print("Гетероскедастичность не обнаружена.")
```

```
Spearman correlation: -0.05826086956521739
p-value: 0.7868436161547347
Гетероскедастичность не обнаружена.
```

In [211…

```
coefficients = model.params
intercept = 0  # Свободный член, для примера считаем 0

x = pd.DataFrame(exog[:,[i for i in range(20) if i not in [1,3,4,5,6,11,12,13,17

X_mean = x.mean().to_numpy()

n = x.shape[0]
X_new = X_mean*1.1

residuals_std = np.sqrt(model.scale)
k = len(coefficients)  # Число независимых переменных
```

```
alpha = 0.05  # Уровень значимости

# Вычисление прогноза
y_pred = intercept + np.dot(X_new, coefficients)

# Вычисление стандартной ошибки прогноза
X_diff = X_new - X_mean
se = residuals_std * np.sqrt(1 + (1 / n) + np.sum(X_diff**2) / np.sum((X_mean -

# Квантиль распределения Стьюдента
t_critical = t.ppf(1 - alpha / 2, df=n - k - 1)

# Доверительный интервал
ci_lower = y_pred - t_critical * se
ci_upper = y_pred + t_critical * se

ci_lower, ci_upper
```

Out[211…

```
(120.38674611961844, 130.94808042896537)
```

In [212…

```
y_pred
```

Out[212…

```
125.6674132742919
```