In [538]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import os
import datetime
import re
import time
%matplotlib inline
```

In [539]:
```
df = pd.read_excel('chugik 01-03 2021 simple.xlsx')
```

In [540]:
```
df.shape
```

```
(576, 5)
```

Цену приводим в float

In [541]:
```
df['Price'] = df['Price'].replace(r'\s+','',regex=True)
```

In [542]:
```
df['Price'] = pd.to_numeric(df['Price'])
```

Суммарная выручка за январь - март

In [543]:
```
profit = df.groupby(df.Date.dt.month)['Price'] \
    .sum().sort_values(ascending=False).to_frame().round()
profit
```

Date	Price
3	6093173.0
2	4944146.0
1	4044466.0

In [544]:
```
profit.sum()
```

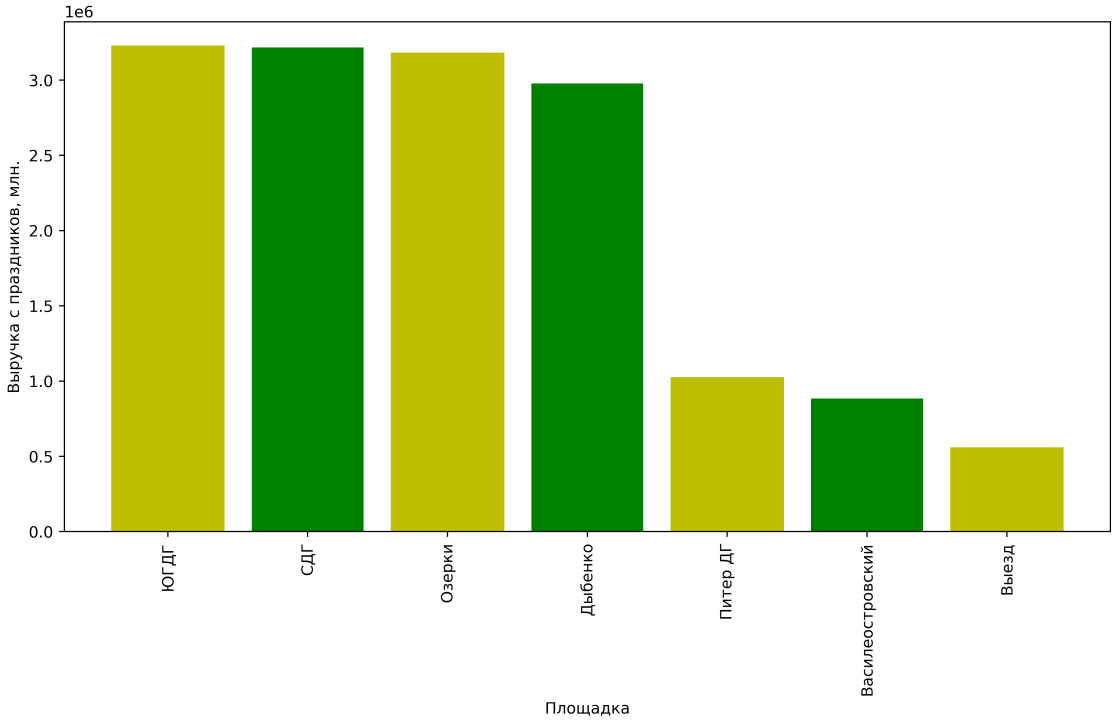
```
Price    15081785.0
dtype: float64
```

Группируем по выручке по площадкам. Убираем Онлайн.

In [545]:
```
df = df[df['Place'] != 'Онлайн']
```

In [546]:
```
df_sum_Place = df.groupby('Place')['Price'].sum().sort_values(ascending=False).to_frame()
```

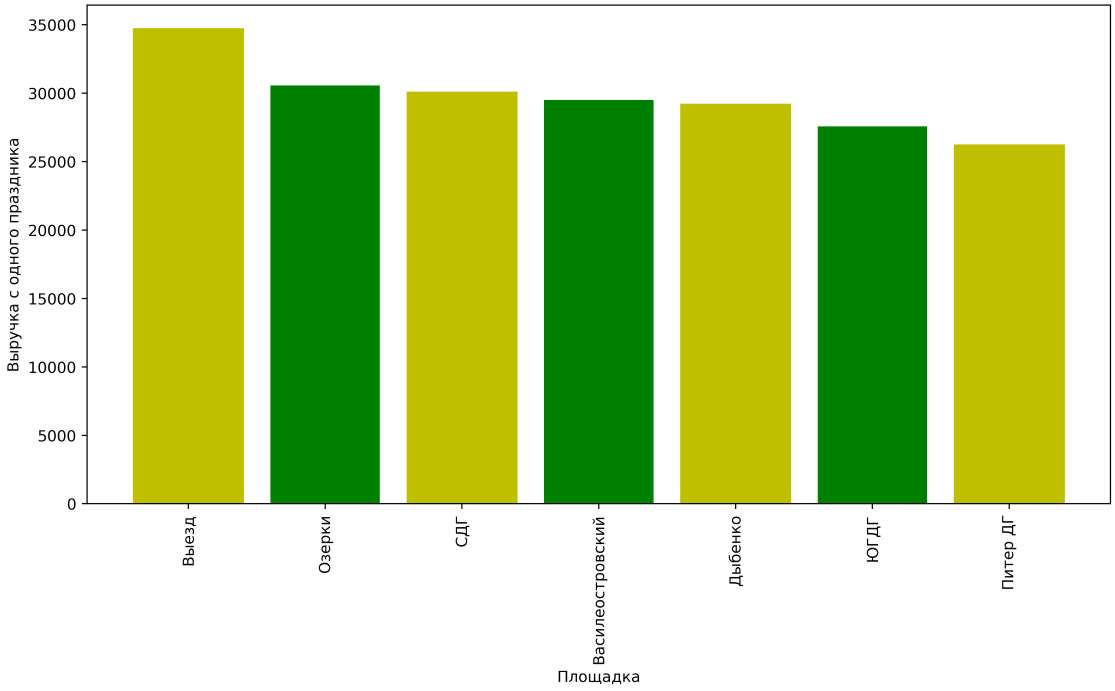
In [547]:
```
plt.figure(figsize=(12,6), dpi=500)
for_color = plt.bar(df_sum_Place.index, df_sum_Place['Price'], color='g')
for n in range(0,7,2):
    for_color[n].set_color('y')
plt.ylabel('Выручка с праздников, млн.')
plt.xlabel('Площадка')
plt.xticks(rotation='vertical', size=10)
plt.show()
```



Группируем по средней стоимости праздника по площадкам

In [548]:
```
df_perHoliday_Place = df.groupby('Place')['Price'].mean() \
    .sort_values(ascending=False).to_frame().round()
```

In [549]:
```
plt.figure(figsize=(12,6), dpi=500)
for_color = plt.bar(df_perHoliday_Place.index, df_perHoliday_Place['Price'], color='g')
for n in range(0,7,2):
    for_color[n].set_color('y')
plt.ylabel('Выручка с одного праздника')
plt.xlabel('Площадка')
plt.xticks(rotation='vertical', size=10)
plt.show()
```



Добавляем столбец Time, убираем лишние данные из Name

In [550]:
```
df['Time'] = df.Name.apply(lambda x: x.split(' ')[1]).to_frame()
```

In [551]:
```
df.Name = df.Name.apply(lambda x: x.split('//')[1])
```

Гистограмма занятности площадок по дням недели и времени

In [552]:
```
df.sort_values('Time', ascending=True)['Time'].unique();
```

In [553]:
```
df_SDG = df[df['Place'] == 'СДГ'].groupby('Date')['Time'].count().to_frame()
```

In [554]:
```
def time_gap(x):
    gap = x.split('-')
    d = datetime.datetime.strptime(gap[0], '%H:%M')
    dd = ''
    while str(dd) != gap[1]:
        d += datetime.timedelta(minutes=30)
        minute = ('00' if str(d.minute) == '0' else '30')
        dd = f'{d.hour}:{minute}'
        if dd != gap[1]:
            gap.append(dd)
    gap.sort()
    return gap
```

In [555]:
```
df['Hours'] = df['Time'].apply(lambda x: time_gap(x))
```

In [556]:
```
df['Weekday'] = df['Date'].dt.weekday
df['Day name'] = df['Date'].dt.day_name()
```

In [557]:
```
df_Hours_inPlace_SDG = df[df['Place'] == 'СДГ'].groupby(['Place', 'Weekday'], as_index=False) \
    .agg({'Hours': 'sum', 'Name': 'count', 'Day_name': 'max'}). \
    rename(columns={'Name': 'Amount of holidays'})
```

In [558]:
```
df_Hours_inPlace_SDG;
```

In [559]:
```
time_df = pd.DataFrame(columns=['12:00', '12:30',
                  '13:00', '13:30', '14:00', '14:30', '15:00', '15:30', '16:00', '16:30',
                  '17:00', '17:30', '18:00', '18:30', '19:00', '19:30', '20:00', '20:30',
                  '21:00', '21:30'])
time_dict = {}
for day in range(0,7):
    time = df_Hours_inPlace_SDG.iloc[day, 2]
    check_time = ['12:00', '12:30',
                  '13:00', '13:30', '14:00', '14:30', '15:00', '15:30', '16:00', '16:30',
                  '17:00', '17:30', '18:00', '18:30', '19:00', '19:30', '20:00', '20:30',
                  '21:00', '21:30']
    for check in check_time:
        time_dict[check] = time.count(check)
    time_df = time_df.append(time_dict, ignore_index=True)

```

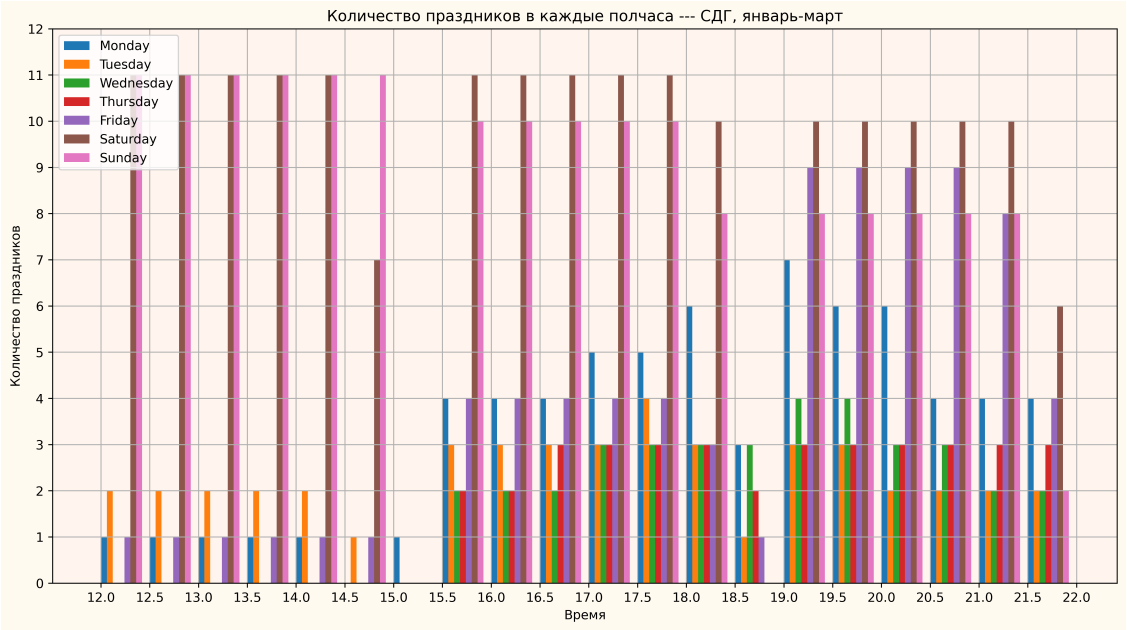
Расчет: Количество праздников в каждые полчаса --- СДГ, январь-март

In [560]:
```
df_SDG_time = df_Hours_inPlace_SDG['Day name'].to_frame().join(time_df)
```

In [561]:
```
X = [np.arange(12.03, 21.53, 0.5, dtype='float')] * 7
for i in range(0,7):
    X[i] = X[i] + 0.06*i
```

In [581]:

```
fig, ax = plt.subplots(dpi=500)

for i in range(0,7):
    x = X[i]
    y = df_SDG_time.drop(['Day_name'], axis=1).iloc[i,:]
    ax.bar(x, y, width=0.06, label=df_SDG_time.Day_name[i])

ax.set_facecolor('seashell')
fig.set_figwidth(15)
fig.set_figheight(8)
fig.set_facecolor('floralwhite')

plt.xticks(np.arange(12,22.5,0.5))
plt.yticks(np.arange(0,13))
plt.legend(loc=2)
plt.title('Количество праздников в каждые полчаса --- СДГ, январь-март')
plt.xlabel('Время')
plt.ylabel('Количество праздников')
plt.grid()
plt.show()
```



Аналогично по будням

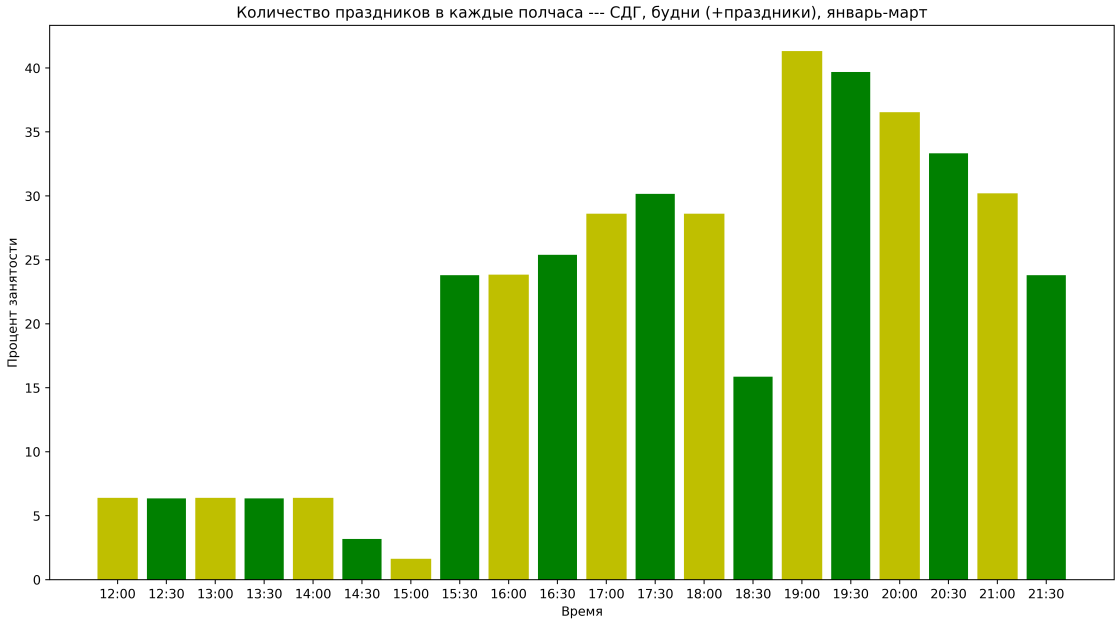
In [563]:

```
time_df = pd.DataFrame(columns=['12:00', '12:30',
                  '13:00', '13:30', '14:00', '14:30', '15:00', '15:30', '16:00', '16:30',
                  '17:00', '17:30', '18:00', '18:30', '19:00', '19:30', '20:00', '20:30',
                  '21:00', '21:30'])
time_dict = {}
for day in range(0,5):
    time = df_Hours_inPlace_SDG.iloc[day, 2]
    check_time = ['12:00', '12:30',
                  '13:00', '13:30', '14:00', '14:30', '15:00', '15:30', '16:00', '16:30',
                  '17:00', '17:30', '18:00', '18:30', '19:00', '19:30', '20:00', '20:30',
                  '21:00', '21:30']
    for check in check_time:
```

```
        if day == 0:
            time_dict[check] = round(time.count(check) / 0.63, 2)
        else:
            time_dict[check] = time_dict[check] + round(time.count(check) / 0.63, 2)
```

In [582]:

```
fig, ax = plt.subplots(dpi=500)

x = time_dict.keys()
y = time_dict.values()
for_color = ax.bar(x, y, color='g')
for n in range(0,len(check_time),2):
    for_color[n].set_color('y')

fig.set_figwidth(15)     #  ширина Figure
fig.set_figheight(8)     #  высота Figure

plt.title('Количество праздников в каждые полчаса --- СДГ, будни (+праздники), январь-март')
plt.xlabel('Время')
plt.ylabel('Процент занятости')
plt.show()
```



Популярность программ

In [565]:

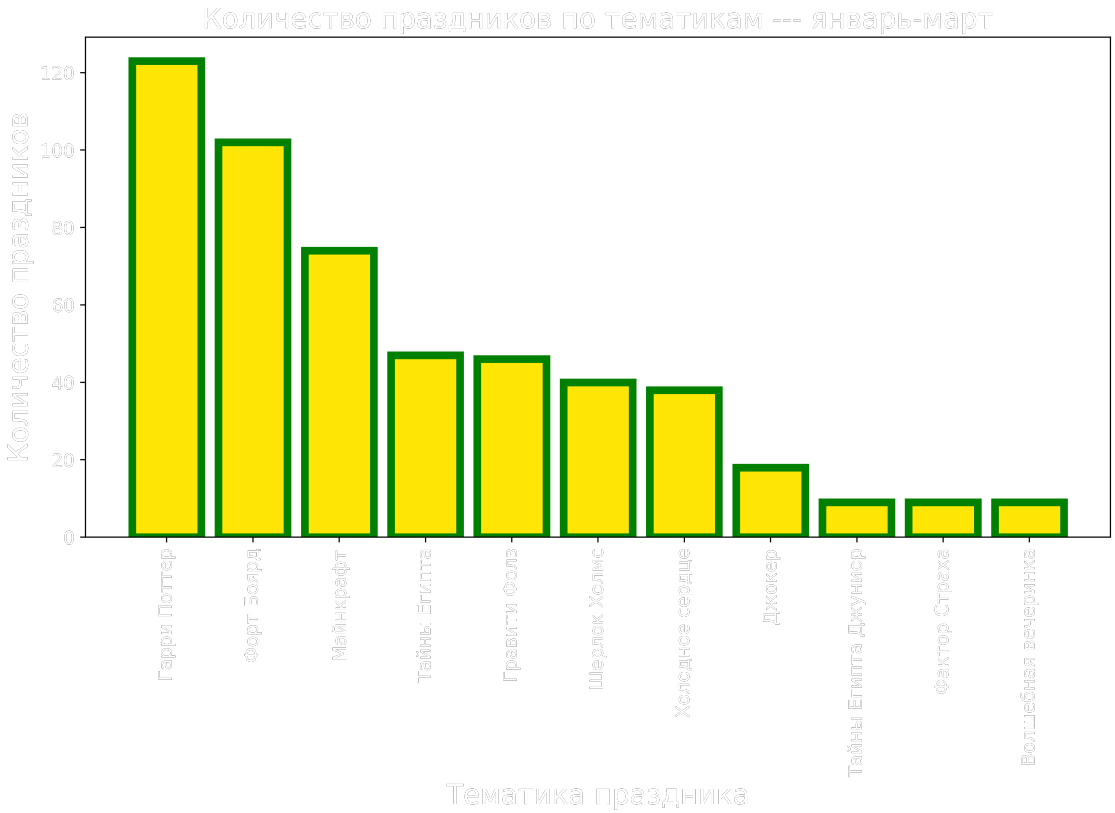
```
programs_list = ['Майнкрафт', 'Гравити Фолз', 'Форт Боярд', 'Тайны Египта Джуниор',
            'Фактор Страха', 'Гарри Поттер', 'Шерлок Холмс', 'Джокер',
            'Холодное сердце', 'Тайны Египта', 'Волшебная вечеринка']

programs = dict.fromkeys(['Майнкрафт', 'Гравити Фолз', 'Форт Боярд', 'Тайны Египта Джуниор',
            'Фактор Страха', 'Гарри Поттер', 'Шерлок Холмс', 'Джокер',
            'Холодное сердце', 'Тайны Египта', 'Волшебная вечеринка'], 0)

def take_prog(x):
    a = 0
    for j in range(len(programs_list)):
        if programs_list[j] in x:
            programs[programs_list[j]] += 1
```

```
            a = 1
    if a == 0:
        raise NameError('Some programs are not defined. CHEEECK IT!')
    return
```

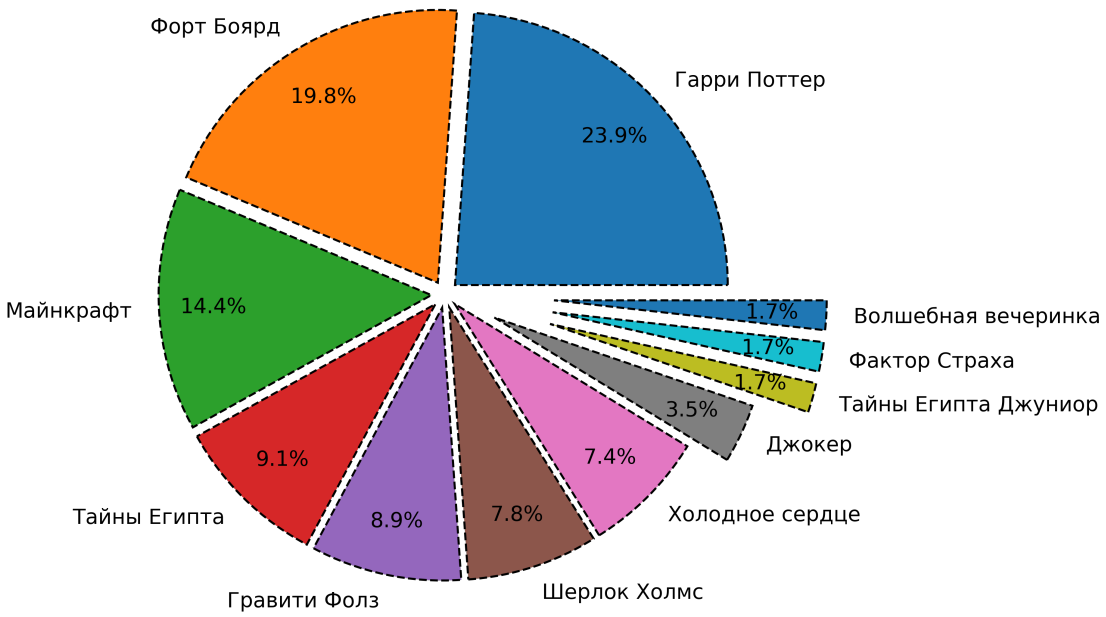
In [566]:
```
df['Name'].apply(lambda x: take_prog(x))
programs['Тайны Египта'] = programs['Тайны Египта'] - programs['Тайны Египта Джуниор']
```

In [567]:
```
list_prog = list(programs.items())
list_prog.sort(key=lambda x: x[1], reverse=True)
y_axis, x_axis = [], []
for i in range(len(list_prog)):
    y_axis.append(list_prog[i][1])
    x_axis.append(list_prog[i][0])
```

In [568]:
```
plt.figure(figsize=(12,6), dpi=500)
for_color = plt.bar(x_axis, y_axis, color=(1, 0.9, 0.02), linewidth=5, edgecolor='g')
plt.title('Количество праздников по тематикам --- январь-март', color='w', size=18)
plt.ylabel('Количество праздников', color='w', size=18)
plt.xlabel('Тематика праздника', color='w', size=18)
plt.xticks(rotation='vertical', size=12, color='w')
plt.yticks(color='w', size=12)
plt.show()
```



In [569]:
```
plt.figure(figsize=(12,6), dpi=500)
exp = (0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.2, 0.4, 0.4, 0.4)
plt.pie(y_axis, labels=x_axis, autopct='%1.1f%%', explode=exp, pctdistance=0.8,
        wedgeprops={'lw':1, 'ls':'--','edgecolor':"k"})
plt.show()
```



Расширенные данные

In [570]:
```
full = pd.read_excel('chugik 01-03 2021 full.xlsx')
```

Избавляемся от дат в quantity и создаем df с данными по product

In [571]:
```
new = full[(full['Product_quantity'] != bad[0]) & (full['Product_quantity'] != bad[1])
           & (full['Product_quantity'] != bad[2])].reset_index(drop=True)
```

In [572]:
```
new['Product_sum'] = new['Product_sum'].replace(r'\s+','',regex=True)

prod = pd.DataFrame(columns=['Product', 'Quantity', 'Cost'])

for index, value in new.Product_name.str.lower().to_frame().iterrows():
    for row in value:
        quantity = float(new.iloc[index, 8])
        cost = float(new.iloc[index, 9])
        prod.loc[len(prod)] = [row, quantity, cost]

```

In [573]:
```
prod_profit = prod.groupby('Product').agg({'Quantity': 'sum', 'Cost': 'sum'}) \
    .sort_values('Cost', ascending=False).reset_index()
```

Объединяем продукты по категориям

In [574]:
```
prod_profit_c = prod_profit.copy()
prod_profit_new = pd.DataFrame(columns=['Product', 'Quantity', 'Cost'])
```

Первый нелогичный вариант

In [575]:
```
'''def zzip(index, product, quantity, cost, re, name):
    for i in range(len(re)):
        if re[i] in product:
            prod_profit_new.loc[len(prod_profit_new)] = [name, quantity, cost]
            prod_profit_c.drop(index=index, axis=0)
    return

re = ['кофе', 'чай', 'капучино']
name = 'напитки'

for index, row in prod_profit_c.iterrows():
    zzip(index, row[0], row[1], row[2], re, name)

prod_profit_new.groupby('Product').agg({'Quantity': 'sum', 'Cost': 'sum'})'''
```

```
"def zzip(index, product, quantity, cost, re, name):\n    for i in range(len(re)):\n        if re[i] in product:\n            prod_profit_new.loc[len(prod_profit_new)] = [name, quantity, cost]\n            prod_profit_c.drop(index=index, axis=0)\n    return    \n\nre = ['кофе', 'чай', 'капучино']\nname = 'напитки'\n\nfor index, row in prod_profit_c.iterrows():\n    zzip(index, row[0], row[1], row[2], re, name)\n\nprod_profit_new.groupby('Product').agg({'Quantity': 'sum', 'Cost': 'sum'})"
```

Актуальный вариант

In [576]:
```
prod_profit_copy = prod_profit.copy()
```

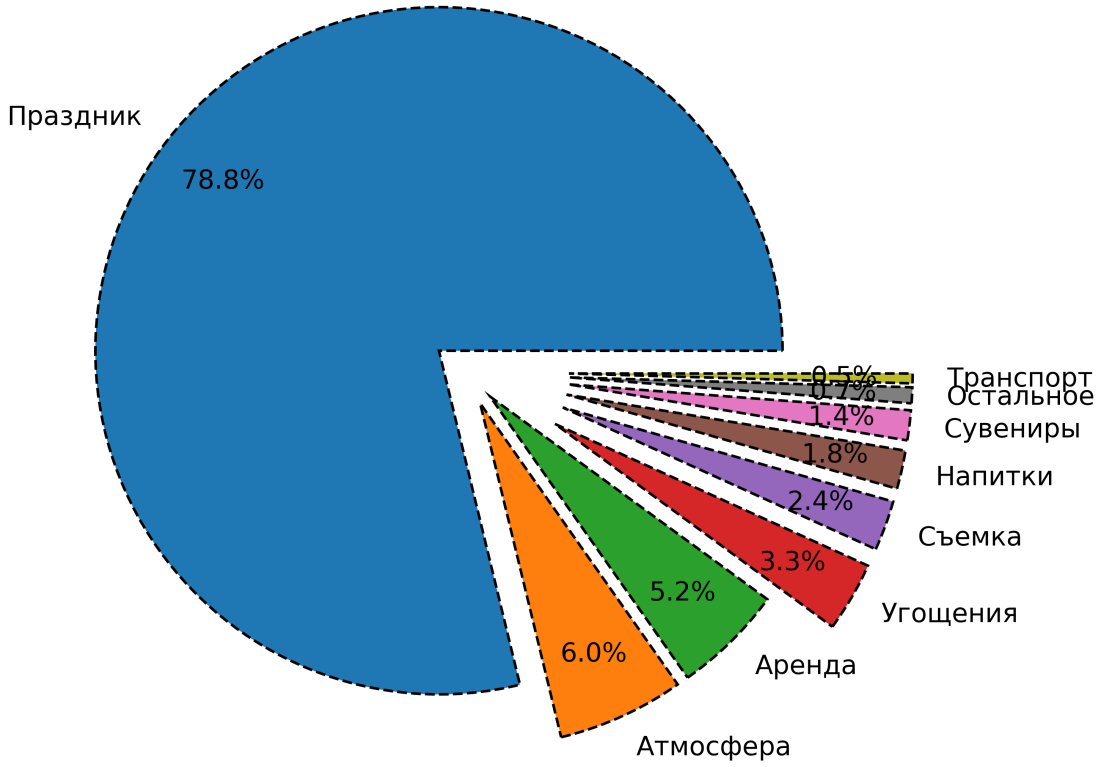
In [577]:
```
drinks = ['кофе', 'чай', 'капучино', 'латте', 'американо']
main = ['квест', 'класс', 'дискотека', 'тесла', 'мафия', 'викторина', 'игры', 'фокус',
        'шоу', 'пати', 'онлайн']
rent = ['аренд']
food = ['кейтеринг', 'торт', 'фонтан', 'официант']
features = ['свечи', 'посуда', 'посуды', 'фейер', 'шар', 'welcome', 'пиньята',
            'слаймы', 'фейр', 'спецэф']
suvs = ['сувен']
photo = ['фото', 'съемка']
transport = ['транспорт']
other = ['доп', 'неуст', 'скидка']

for i in range(len(prod_profit_copy)):
    check = prod_profit_copy.iloc[i,0]
    for ii in range(len(drinks)):
        if drinks[ii] in check:
            prod_profit_copy.iloc[i,0] = 'Напитки'
    for ii in range(len(main)):
        if main[ii] in check:
            prod_profit_copy.iloc[i,0] = 'Праздник'
    for ii in range(len(rent)):
        if rent[ii] in check:
            prod_profit_copy.iloc[i,0] = 'Аренда'
    for ii in range(len(food)):
        if food[ii] in check:
            prod_profit_copy.iloc[i,0] = 'Угощения'
    for ii in range(len(features)):
        if features[ii] in check:
            prod_profit_copy.iloc[i,0] = 'Атмосфера'
    for ii in range(len(suvs)):
        if suvs[ii] in check:
            prod_profit_copy.iloc[i,0] = 'Сувениры'
```

```
    for ii in range(len(photo)):
        if photo[ii] in check:
            prod_profit_copy.iloc[i,0] = 'Съемка'
    for ii in range(len(transport)):
        if transport[ii] in check:
            prod_profit_copy.iloc[i,0] = 'Транспорт'
    for ii in range(len(other)):
        if other[ii] in check:
            prod_profit_copy.iloc[i,0] = 'Остальное'
```

In [578]:
```
prod_profit_copy = prod_profit_copy.groupby('Product') \
    .agg({'Quantity': 'sum', 'Cost': 'sum'}).sort_values('Cost', ascending=False) \
    .reset_index()
```

Строим круговую по всем категориям

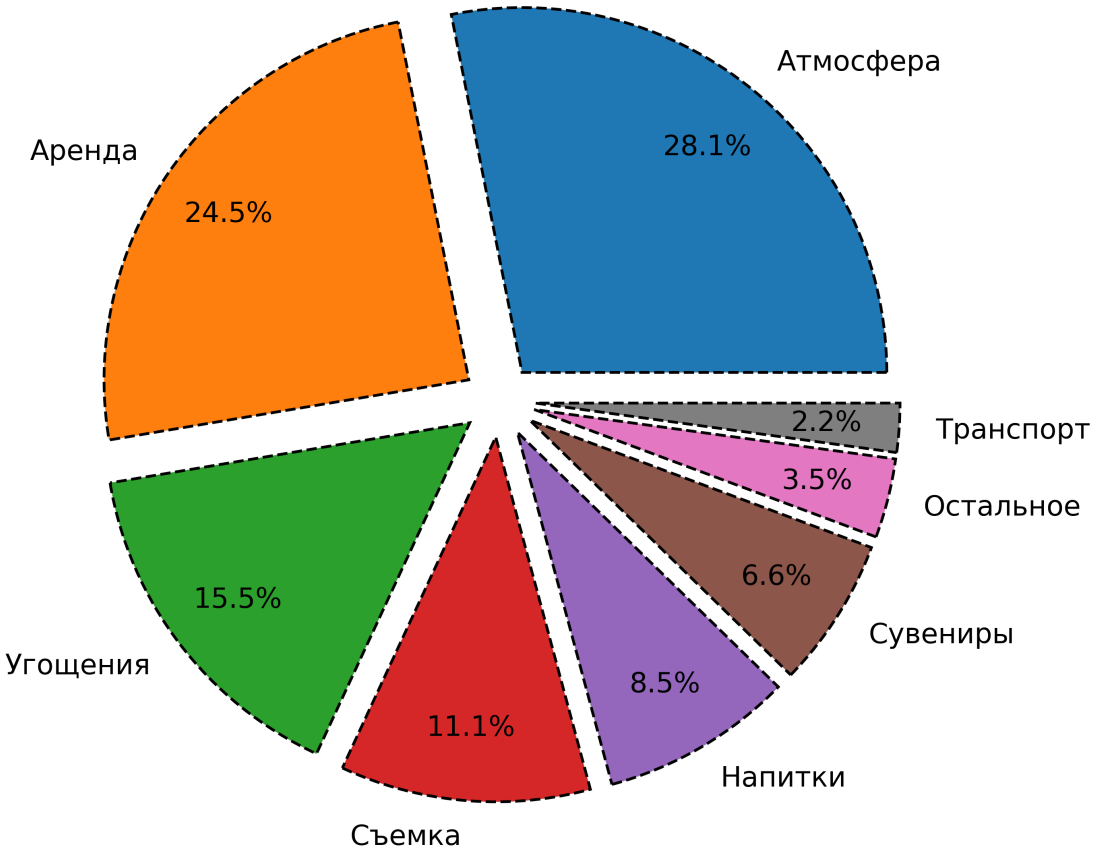
In [579]:
```
plt.figure(figsize=(12,6), dpi=500)
exp = (0.1, 0.1, 0.1, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3)
plt.pie(prod_profit_copy.Cost, labels=prod_profit_copy.Product, autopct='%1.1f%%',
        explode=exp, pctdistance=0.8, wedgeprops={'lw':1, 'ls':'--','edgecolor':"k"})
plt.show()
```



Круговая без "Праздник"

In [580]:
```
prod_profit_copy_2 = prod_profit_copy[prod_profit_copy['Product'] != 'Праздник']
plt.figure(figsize=(12,6), dpi=500)
exp = (0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1)
plt.pie(prod_profit_copy_2.Cost, labels=prod_profit_copy_2.Product, autopct='%1.1f%%',
```

```
        explode=exp, pctdistance=0.8, wedgeprops={'lw':1, 'ls':'--','edgecolor':"k"})
plt.show()
```




```
В [ ]:
1
```