UK Government Departmental Spending Analysis

Project Overview

This project analyzes real public spending data from the **Department for Science, Innovation and Technology (DSIT)** — a UK government department responsible for science, research, and digital technology funding.

The dataset covers **monthly transactions over £25,000** from **2024 to 2025**, giving us a transparent view of how taxpayer money is being spent.

What has been done

started by collecting **18 monthly CSV files** spread across two years and combined them into one clean, unified dataset. The raw data was messy it had inconsistent date formats, currency symbols in numbers, missing values, and duplicate records. We cleaned all of that systematically before doing any analysis.

Once the data was clean, I asked real business questions:

- Where is the money going each month?
- Who are the biggest suppliers receiving government payments?
- Which expense categories consume the most budget?
- Are there unusually large transactions that need attention?
- How did spending change between 2024 and 2025?

What has has been Found

The analysis revealed clear patterns in government spending — which suppliers dominate the budget, which expense categories are growing, and where the biggest individual payments are being made. These insights help stakeholders, auditors, and the public understand how public money is being managed.

In [1]:
```
import pandas as pd
from pathlib import Path

# Define the path
# I'm in 'processed data', so go up one level (..) to reach 'data',
# then into 'raw data'
```

```
raw_data_path = Path('..') / 'data' / 'raw data'

# Get CSV files from both folders
csv_files_2024 = list((raw_data_path / '2024').glob('*.csv'))
csv_files_2025 = list((raw_data_path / '2025').glob('*.csv'))

# Combine the lists
all_files = csv_files_2024 + csv_files_2025

# CHECK HOW MANY FILES WERE FOUND
print(f"Files in 2024: {len(csv_files_2024)}")
print(f"Files in 2025: {len(csv_files_2025)}")
print(f"Total files: {len(all_files)}")

if len(all_files) == 0:
    print("No files found! Current working directory:")
    print(Path.cwd())
    print(f"Path exists: {raw_data_path.exists()}")
else:
    # Combine into DataFrame - simple one-liner with encoding fix
    df = pd.concat(
        [pd.read_csv(f, encoding='windows-1252') for f in all_files],
        ignore_index=True
    )
```

```
Files in 2024: 12
Files in 2025: 6
Total files: 18
```

Setting dates as an index and sorting them.

In [2]:
```
# df['Date of Payment'] = pd.to_datetime(df['Date of Payment'], format='%d-%m-%Y')

newdf = df.sort_values('Date of Payment').reset_index(drop= True).copy()
newdf
```

Out[2]:

	Date of Payment	Expense Type	Expense Area	Supplier	Transaction Number	Amount	Descri
0	01/02/2024	Other It Consultancy	Dsit - Science, Innovation And Growth - Dsit -...	Atkinsrealis Uk Ltd	567929	134394.66	Na Undergr Reg
1	01/02/2024	Grant-in-aid To Arms Length Bodies	Dsit - Science, Innovation And Growth - Dsit -...	Ukri - Engineering And Physical Sciences Resea...	566724	90000000.0	Dsit - Finar grant-i To Arm
2	01/02/2024	Grant-in-aid To Arms Length Bodies	Dsit - Science, Innovation And Growth - Dsit -...	Ukri - Medical Research Council	566725	36000000.0	Dsit Finar grant-i To L
3	01/02/2024	Grant-in-aid To Arms Length Bodies	Dsit - Science, Innovation And Growth - Dsit -...	Ukri - Biotechnology And Biological Science Re...	566726	10000000.0	Res Co Pe Sc (B
4	01/02/2024	Grant-in-aid To Arms Length Bodies	Dsit - Science, Innovation And Growth - Dsit -...	Ukri - Biotechnology And Biological Science Re...	566730	22000000.0	Dsit - Finar grant-i To Arm
...	...	...	...	...	...	...	
3491	31/12/2024	Current Grants To Private Sector - Npish	Dsit - Digital And Technology Group - Dsit - C...	The Uk Cyber Security Council	647161	75940.76	Dsit - (Se Co cu Grar
3492	31/12/2024	Capital Grants To Private Sector - Companies	Dsit - Digital And Technology Group - Dsit - D...	University Of Surrey	647122	1419423.64	Dsit- fonrc-f Ne Researc

	Date of Payment	Expense Type	Expense Area	Supplier	Transaction Number	Amount	Descri
3493	31/12/2024	R&D Current Grants To Public Corporations	Dsit - Science, Innovation And Growth - Dsit -...	Npl Management Ltd	647057	261483.81	Dsit Na T Centre Curr
3494	31/12/2024	Faststream - Full Cost	Dsit - Corporate Services - Dsit - Human Resou...	Cabinet Office	647049	33568.0	Dev Ac faststr Ful
3495	31/12/2024	Faststream - Full Cost	Dsit - Corporate Services - Dsit - Human Resou...	Cabinet Office	647065	92831.0	Dev Ac faststr Ful

3496 rows × 9 columns

Cleaning and checking quality data through 4 steps

- Remove / check duplicates data
- Handel null values
- Standradize Data
- Remove unnecessary coulmns or rows

- checking missing data

In [3]:
```
missing_data = newdf.isna().sum()
missing_data[missing_data > 0 ].sort_values(ascending=False)
```

Out[3]:
```
Supplier Post Code    16
Description            3
dtype: int64
```

- Show all duplicate rows, with full details

In [4]:
```
# Show all duplicate rows, with full details
duplicates = newdf[newdf.duplicated(subset=['Transaction Number'], keep= False)]
duplicates.sort_values('Transaction Number')
```

Out[4]:

	Date of Payment	Expense Type	Expense Area	Supplier	Transaction Number	Amount	Descrip
1197	08/01/2024	R & D Current Grants To Private Sector - Npish	Dsit - Science, Innovation And Growth - Dsit -...	The British Academy	561995	1925839.0	Dsit - B Acade & D Cu Grants
1195	08/01/2024	R & D Current Grants To Private Sector - Npish	Dsit - Science, Innovation And Growth - Dsit -...	The British Academy	561995	127881.0	Dsit - B Acad Transit Meas
1201	08/01/2024	R & D Current Grants To Private Sector - Npish	Dsit - Science, Innovation And Growth - Dsit -...	Royal Academy Of Engineering Rae	561997	3190723.32	Dsit - F Academ Enginee r & D C
1198	08/01/2024	R & D Current Grants To Private Sector - Npish	Dsit - Science, Innovation And Growth - Dsit -...	Royal Academy Of Engineering Rae	561997	68221.22	Dsit - F Academ Engine Transit
1193	08/01/2024	R & D Current Grants To Private Sector - Npish	Dsit - Science, Innovation And Growth - Dsit -...	Academy Of Medical Sciences	561998	540498.0	D Academ Me Science D Cu
...	...	...	...	...	...	...	
2331	20/06/2025	R&D Current Grants To Public Corporations	Dsit - Science, Innovation And Growth - Dsit -...	Met Office	696974	115960.86	Dsit - Of Stra Prio Fu
2832	25/06/2025	R&D Current Grants To Public Corporations	Dsit - Science, Innovation And Growth - Dsit -...	Met Office	698016	120012.55	Dsit - Of Clear Analy S
2833	25/06/2025	R&D Current Grants To	Dsit - Science, Innovation	Met Office	698016	39432.25	Dsit - Of Clear

	Date of Payment	Expense Type	Expense Area	Supplier	Transaction Number	Amount	Descrip
		Public Corporations	And Growth - Dsit -...				Analy S
3358	30/06/2025	Cl - Cash Cfers Paid Over To Hmt	Dsit - Digital And Technology Group - Dsit - D...	Consolidated Fund Account 6622	699919	48732174.44	Dsit- O Recei Centra Ca
3360	30/06/2025	Cl - Cash Cfers Paid Over To Hmt	Dsit - Digital And Technology Group - Dsit - D...	Consolidated Fund Account 6622	699919	620966.24	Dsit- Recei Ce (O S

417 rows × 9 columns

- Get Transaction Numbers that appear more than once
- How many each Transaction occures

In [5]:
```
duplicate_counts = newdf['Transaction Number'].value_counts()
duplicate_counts = duplicate_counts[duplicate_counts > 1]
duplicate_counts
```

Out[5]:
```
Transaction Number
618334    7
593836    6
611418    6
593837    6
672067    6
         ..
632135    2
575233    2
575235    2
575223    2
661677    2
Name: count, Length: 173, dtype: int64
```

- Number of duplicates

In [6]:
```
duplicates = newdf[newdf.duplicated(subset=['Transaction Number'], keep=False)]
len(duplicates)
```

Out[6]:
```
417
```

Validate date ranges and spot any anomalous formats.

In [7]:
```
# Convert to datetime and catch format errors
newdf['Date of Payment'] = pd.to_datetime(newdf['Date of Payment'], errors='coerce'

# Find invalid formats (became NaT after conversion)
print(f"Invalid date formats: {newdf['Date of Payment'].isna().sum()}")

# Check date range
print(f"Date range: {newdf['Date of Payment'].min()} to {newdf['Date of Payment'].m

# Find dates outside expected range
anomalies = newdf[
    (newdf['Date of Payment'] < '2024-01-01') |
    (newdf['Date of Payment'] > '2025-12-31')
]
print(f"Dates outside 2024-2025: {len(anomalies)}")

# Show any anomalies found
if len(anomalies) > 0:
    print(anomalies[['Date of Payment', 'Supplier', 'Amount']])
```

```
Invalid date formats: 1856
Date range: 2024-01-02 00:00:00 to 2025-12-06 00:00:00
Dates outside 2024-2025: 0
```

Removing null values in (dates of payment) column

In [8]:
```
newdf  = newdf[newdf['Date of Payment'].notna()].copy()
```

- Display invalid dates

In [9]:
```
invalid_dates = newdf[newdf['Date of Payment'].isna()]
print(f"Invalid dates found: {len(invalid_dates)}")
```

```
Invalid dates found: 0
```

In [10]:
```
# After removing
print(f"Rows after: {len(newdf)}")
print(f"Missing dates now: {newdf['Date of Payment'].isna().sum()}")
```

```
Rows after: 1640
Missing dates now: 0
```

- Duplicated Transaction Numbers after removing duplicates

In [11]:
```
duplicates = newdf[newdf.duplicated(subset=['Transaction Number'], keep=False)]
print(f'Duplicated Transaction Numbers after : {len(duplicates)}')
```

```
Duplicated Transaction Numbers after : 160
```

In [12]:
```
duplicate_counts = newdf['Transaction Number'].value_counts()
duplicate_counts = duplicate_counts[duplicate_counts > 1]
duplicate_counts
```

Out[12]:
```
Transaction Number
593836    6
672067    6
694164    5
575225    4
672550    3
          ..
575233    2
575235    2
575655    2
586609    2
587725    2
Name: count, Length: 72, dtype: int64
```

This code takes messy money values like "£134,394.66" and cleans them up into pure numbers like 134394.66.

In [13]:
```
newdf['Amount'] = newdf['Amount'].astype(str).str.replace('£', '').str.replace(',',
# newdf['Amount'] = pd.to_numeric(newdf['Amount'], errors='coerce')
newdf['Amount'].head()
```

Out[13]:
```
0      134394.66
1    90000000.00
2    36000000.00
3    10000000.00
4    22000000.00
Name: Amount, dtype: float64
```

- Getting a copy of the cleaned datasest

In [14]:
```
newdf.to_csv('master_spend_cleand_data.csv', index=False, encoding='utf-8')
```

In [4]:
```
import pandas as pd
import matplotlib.pyplot as plt


df = pd.read_csv('../data/processed data/master_spend_cleand_data.csv')
```

What are the main spending categories ?

In [6]:
```
# Group by Expense Type and sum the amount
expense_categories = df.groupby('Expense Type')['Amount'].sum().reset_index()

# Sort by amount descending
expense_categories = expense_categories.sort_values('Amount', ascending=False)

# Add percentage column
expense_categories['Percentage'] = (expense_categories['Amount'] / expense_categori

expense_categories
```

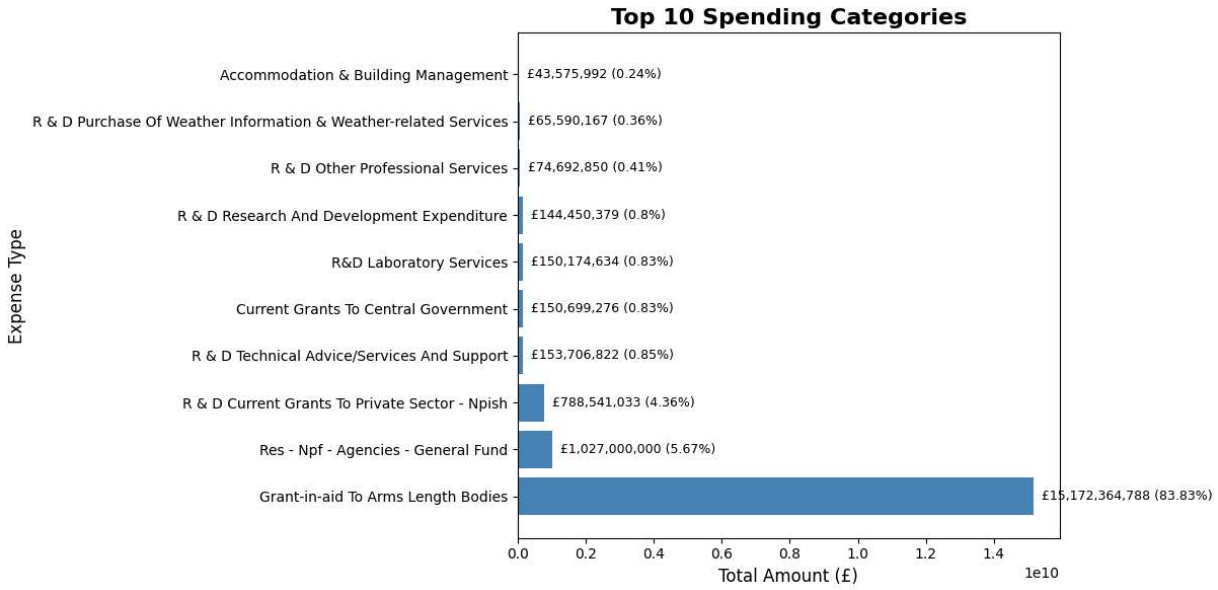
Out[6]:

	Expense Type	Amount	Percentage
22	Grant-in-aid To Arms Length Bodies	1.517236e+10	83.83
84	Res - Npf - Agencies - General Fund	1.027000e+09	5.67
66	R & D Current Grants To Private Sector - Npish	7.885410e+08	4.36
77	R & D Technical Advice/Services And Support	1.537068e+08	0.85
13	Current Grants To Central Government	1.506993e+08	0.83
...	...	...	...
65	R & D Current Grants To Overseas Bodies	3.459001e+04	0.00
26	Independent Experts	3.321700e+04	0.00
54	Pr & Marketing Advice & Services	3.096931e+04	0.00
39	Misc. Non Procurement Spend	3.000000e+04	0.00
14	Current Grants To Local Government	2.980111e+04	0.00

94 rows × 3 columns

In [8]:
```
# Get top 10 only
top10_categories = expense_categories.head(10)

# Plot horizontal bar chart
plt.figure(figsize=(12, 6))
plt.barh(top10_categories['Expense Type'], top10_categories['Amount'], color='steel

# Add labels and title
```

```
plt.title('Top 10 Spending Categories', fontsize=16, fontweight='bold')
plt.xlabel('Total Amount (£)', fontsize=12)
plt.ylabel('Expense Type', fontsize=12)

# Add value labels on bars
for i, (amount, pct) in enumerate(zip(top10_categories['Amount'], top10_categories[
    plt.text(amount, i, f'  £{amount:,.0f} ({pct}%)', va='center', fontsize=9)

plt.tight_layout()
plt.show()
```



Which expense type costs the most?

In [3]:
```
# Find the most expensive expense type
most_expensive = expense_categories.iloc[0]

most_expensive
```

Out[3]:
```
Expense Type    Grant-in-aid To Arms Length Bodies
Amount                         15172364788.389999
Percentage                                  83.83
Name: 22, dtype: object
```

How is the budget distributed across categories ?

In [9]:
```
# Show budget distribution across all categories
total_budget = df['Amount'].sum()

print(f"Total Budget: £{total_budget:,.2f}")
print(f"Total Categories: {len(expense_categories)}")
print("\nBudget Distribution:")
print(expense_categories[['Expense Type', 'Amount', 'Percentage']])
```

```python
# Is it concentrated or distributed?
top3_percentage = expense_categories['Percentage'].head(3).sum()

print(f"\nTop 3 categories represent: {top3_percentage:.1f}% of total budget")

if top3_percentage > 70:
    print("✅ Budget is CONCENTRATED in few categories")
else:
    print("✅ Budget is DISTRIBUTED across many categories")


# ChatGPT is used to quick and write perfect prints and  write if it is concentrate
```

```
Total Budget: £18,097,912,376.53
Total Categories: 94

Budget Distribution:
                                    Expense Type        Amount  Percentage
22              Grant-in-aid To Arms Length Bodies  1.517236e+10       83.83
84             Res - Npf - Agencies - General Fund  1.027000e+09        5.67
66  R & D Current Grants To Private Sector - Npish  7.885410e+08        4.36
77     R & D Technical Advice/Services And Support  1.537068e+08        0.85
13            Current Grants To Central Government  1.506993e+08        0.83
..                                             ...           ...         ...
65         R & D Current Grants To Overseas Bodies  3.459001e+04        0.00
26                             Independent Experts  3.321700e+04        0.00
54               Pr & Marketing Advice & Services  3.096931e+04        0.00
39                    Misc. Non Procurement Spend  3.000000e+04        0.00
14              Current Grants To Local Government  2.980111e+04        0.00

[94 rows x 3 columns]

Top 3 categories represent: 93.9% of total budget
✅ Budget is CONCENTRATED in few categories
```

In [8]:
```
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv('../data/processed data/master_spend_cleand_data.csv')
```

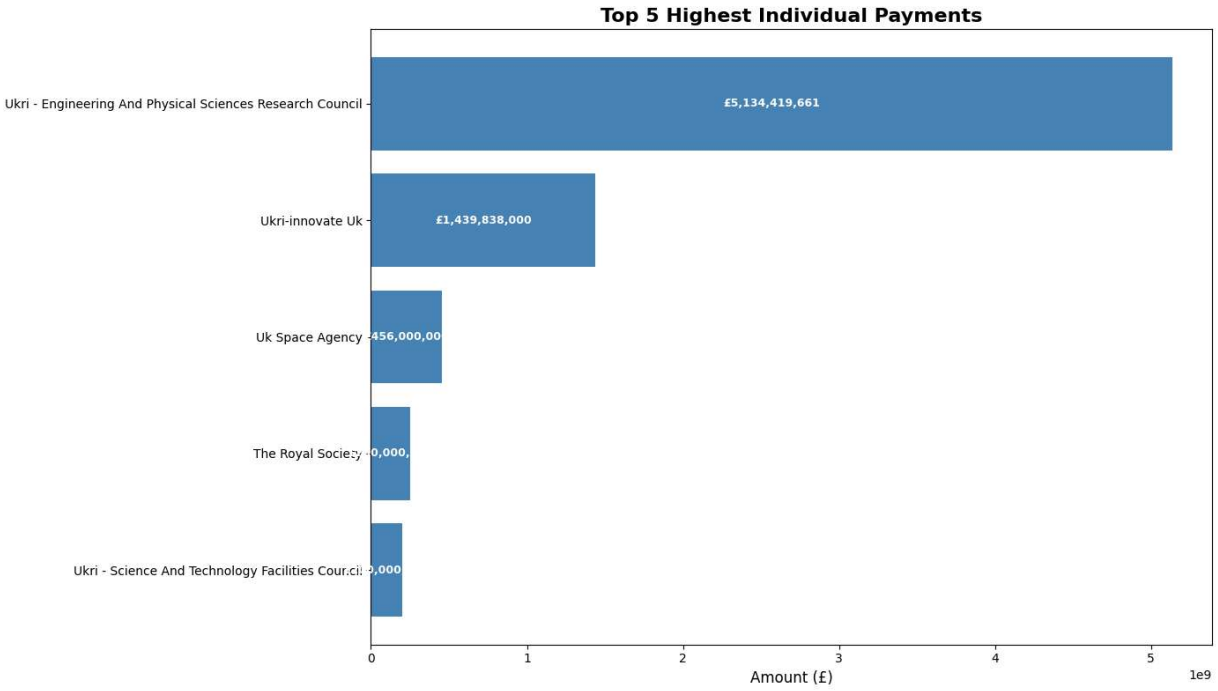
What are the largest individual payments (top 100 and top 20)

In [5]:
```
top_100 = df.nlargest(100, 'Amount')

top_20 =  top_100.sort_values('Amount' , ascending=False).head(20)


print(f"Top 100 transactions: {len(top_100)}")
print(top_20[['Date of Payment', 'Amount']])
```

```
Top 100 transactions: 100
    Date of Payment       Amount
316      2024-02-09  736000000.0
27       2024-01-03  689838000.0
377      2025-03-03  687419661.0
341      2024-02-12  580000000.0
201      2025-02-01  455000000.0
456      2024-03-06  449000000.0
159      2024-01-11  400000000.0
137      2024-01-08  387000000.0
123      2024-01-07  378000000.0
108      2025-01-05  300000000.0
109      2025-01-05  290000000.0
171      2024-02-01  290000000.0
153      2024-01-10  277000000.0
539      2024-04-03  250000000.0
172      2024-02-01  246000000.0
135      2024-01-08  225000000.0
376      2025-03-03  225000000.0
632      2025-04-06  210000000.0
74       2025-01-04  205000000.0
173      2024-02-01  200000000.0
```

In [14]:
```
# Group by supplier and sum amounts first
top_20_grouped = top_20.groupby('Supplier')['Amount'].sum().reset_index()
top_20_grouped = top_20_grouped.sort_values('Amount', ascending=True).tail(20)

# Plot
fig, ax = plt.subplots(figsize=(14, 8))
bars = ax.barh(top_20_grouped['Supplier'], top_20_grouped['Amount'], color='steelbl

# Add labels and title
ax.set_title('Top 5 Highest Individual Payments', fontsize=16, fontweight='bold')
ax.set_xlabel('Amount (£)', fontsize=12)

# Add single clean label per bar
for bar, amount in zip(bars, top_20_grouped['Amount']):
```

```
    ax.text(bar.get_width() * 0.5, bar.get_y() + bar.get_height()/2,
            f'£{amount:,.0f}', va='center', ha='center',
            fontsize=9, color='white', fontweight='bold')

plt.tight_layout()
plt.show()
```



Who received them and for what?

In [6]:
```
(top_20[['Supplier', 'Amount', 'Description', 'Expense Type']])
```

Out[6]:

	Supplier	Amount	Description	Expense Type
316	Ukri - Engineering And Physical Sciences Resea...	736000000.0	Dsit - Epsrc Financing-grant-in-aid To Arms Le...	Grant-in-aid To Arms Length Bodies
27	Ukri-innovate Uk	689838000.0	Dsit - Tsb Financing-grant-in-aid To Arms Leng...	Grant-in-aid To Arms Length Bodies
377	Ukri - Engineering And Physical Sciences Resea...	687419661.0	Dsit - Epsrc Financing-grant-in-aid To Arms Le...	Grant-in-aid To Arms Length Bodies
341	Ukri - Engineering And Physical Sciences Resea...	580000000.0	Dsit - Epsrc Financing-grant-in-aid To Arms Le...	Grant-in-aid To Arms Length Bodies
201	Ukri - Engineering And Physical Sciences Resea...	455000000.0	Dsit - Epsrc Financing-grant-in-aid To Arms Le...	Grant-in-aid To Arms Length Bodies
456	Ukri - Engineering And Physical Sciences Resea...	449000000.0	Dsit - Epsrc Financing-grant-in-aid To Arms Le...	Grant-in-aid To Arms Length Bodies
159	Ukri - Engineering And Physical Sciences Resea...	400000000.0	Dsit - Epsrc Financing-grant-in-aid To Arms Le...	Grant-in-aid To Arms Length Bodies
137	Ukri - Engineering And Physical Sciences Resea...	387000000.0	Dsit - Epsrc Financing-grant-in-aid To Arms Le...	Grant-in-aid To Arms Length Bodies
123	Ukri - Engineering And Physical Sciences Resea...	378000000.0	Dsit - Epsrc Financing-grant-in-aid To Arms Le...	Grant-in-aid To Arms Length Bodies
108	Ukri-innovate Uk	300000000.0	Dsit - Tsb Financing-grant-in-aid To Arms Leng...	Grant-in-aid To Arms Length Bodies
109	Ukri - Engineering And Physical Sciences Resea...	290000000.0	Dsit - Epsrc Financing-grant-in-aid To Arms Le...	Grant-in-aid To Arms Length Bodies
171	Ukri - Engineering And Physical Sciences Resea...	290000000.0	Dsit - Epsrc Financing-grant-in-aid To Arms Le...	Grant-in-aid To Arms Length Bodies
153	Ukri - Engineering And Physical Sciences Resea...	277000000.0	Dsit - Epsrc Financing-grant-in-aid To Arms Le...	Grant-in-aid To Arms Length Bodies
539	The Royal Society	250000000.0	Dsit - Discovery Fellowships Endowment-r & D C...	R & D Current Grants To Private Sector - Npish

	Supplier	Amount	Description	Expense Type
172	Uk Space Agency	246000000.0	Dsit - Uk Space Agency - Supply Funding-res - ...	Res - Npf - Agencies - General Fund
135	Ukri-innovate Uk	225000000.0	Dsit - Tsb Financing-grant-in-aid To Arms Leng...	Grant-in-aid To Arms Length Bodies
376	Ukri-innovate Uk	225000000.0	Dsit - Tsb Financing-grant-in-aid To Arms Leng...	Grant-in-aid To Arms Length Bodies
632	Uk Space Agency	210000000.0	Dsit - Uk Space Agency - Supply Funding-res - ...	Res - Npf - Agencies - General Fund
74	Ukri - Engineering And Physical Sciences Resea...	205000000.0	Dsit - Epsrc Financing-grant-in-aid To Arms Le...	Grant-in-aid To Arms Length Bodies
173	Ukri - Science And Technology Facilities Council	200000000.0	Dsit - Stfc Financing-grant-in-aid To Arms Len...	Grant-in-aid To Arms Length Bodies

What patterns exist in high-value spending?

In [18]:
```
print(f"Total high-value spend: £{top_20['Amount'].sum():,.2f}")
print(f"Average transaction: £{top_20['Amount'].mean():,.2f}")

print("\nBy Expense Type:")
print(top_20.groupby('Expense Type')['Amount'].sum().sort_values(ascending=False))

print("\nBy Supplier Type:")
print(top_20.groupby('Supplier Type')['Amount'].sum().sort_values(ascending=False))

# AI used to print the results.
```

```
Total high-value spend: £7,480,257,661.00
Average transaction: £374,012,883.05

By Expense Type:
Expense Type
Grant-in-aid To Arms Length Bodies                6.774258e+09
Res - Npf - Agencies - General Fund               4.560000e+08
R & D Current Grants To Private Sector - Npish    2.500000e+08
Name: Amount, dtype: float64

By Supplier Type:
Supplier Type
Vendor      5.134420e+09
Wga Only    2.095838e+09
Grant       2.500000e+08
Name: Amount, dtype: float64
```

In [ ]:
```
import pandas as pd

df = pd.read_csv('../data/processed data/master_spend_cleand_data.csv')

# Convert date back to datetime
df['Date of Payment'] = pd.to_datetime(df['Date of Payment'])
```

How does spending change month by month?

In [2]:
```
# Group by month and sum spending
df['Year-Month'] = df['Date of Payment'].dt.to_period('M')
monthly_spend = df.groupby('Year-Month')['Amount'].sum().reset_index()


print(f"\nHighest: {monthly_spend['Amount'].max():,.2f}")
print(f"Lowest: {monthly_spend['Amount'].min():,.2f}")
print(f"Average: {monthly_spend['Amount'].mean():,.2f}")


print("Monthly Spending:")

# ChatGPT used to add commas to these numbers.

monthly_spend
```

```
Highest: 4,896,775,249.64
Lowest: 7,813,383.10
Average: 754,079,682.36
Monthly Spending:
```

Out[2]:

	Year-Month	Amount
0	2024-01	4.896775e+09
1	2024-02	4.270621e+09
2	2024-03	1.274519e+09
3	2024-04	8.736380e+08
4	2024-05	6.897208e+08
5	2024-06	1.060150e+08
6	2024-07	4.241811e+07
7	2024-08	3.826718e+07
8	2024-09	1.635173e+07
9	2024-10	1.872280e+07
10	2024-11	1.914611e+07
11	2024-12	6.240248e+07
12	2025-01	1.821038e+09
13	2025-02	1.329143e+09
14	2025-03	1.965283e+09
15	2025-04	2.545400e+08
16	2025-05	1.051972e+08
17	2025-06	1.649399e+08
18	2025-07	2.698955e+07
19	2025-08	2.984112e+07
20	2025-09	7.813383e+06
21	2025-10	1.063271e+07
22	2025-11	9.757031e+06
23	2025-12	6.413953e+07

Make analysis to identify if there is any unusual spikes or drops?

In [ ]:

```
# Calculate mean and standard deviation
mean = monthly_spend['Amount'].mean()
std = monthly_spend['Amount'].std()
```

```python
# Flag unusual spikes or drops (anything beyond 2 standard deviations)
monthly_spend['Status'] = 'Normal'
monthly_spend.loc[monthly_spend['Amount'] > mean + 2 * std, 'Status'] = '🔴 Spike'
monthly_spend.loc[monthly_spend['Amount'] < mean - 2 * std, 'Status'] = '🔵 Drop'

print(monthly_spend)
print(f"\nSpikes found: {(monthly_spend['Status'] == '🔴 Spike').sum()}")
print(f"Drops found: {(monthly_spend['Status'] == '🔵 Drop').sum()}")

# ChatGPT was used to assist with the calculations to identify spikes and drops.
```

	Year-Month	Amount	Status
0	2024-01	4.896775e+09	🔴 Spike
1	2024-02	4.270621e+09	🔴 Spike
2	2024-03	1.274519e+09	Normal
3	2024-04	8.736380e+08	Normal
4	2024-05	6.897208e+08	Normal
5	2024-06	1.060150e+08	Normal
6	2024-07	4.241811e+07	Normal
7	2024-08	3.826718e+07	Normal
8	2024-09	1.635173e+07	Normal
9	2024-10	1.872280e+07	Normal
10	2024-11	1.914611e+07	Normal
11	2024-12	6.240248e+07	Normal
12	2025-01	1.821038e+09	Normal
13	2025-02	1.329143e+09	Normal
14	2025-03	1.965283e+09	Normal
15	2025-04	2.545400e+08	Normal
16	2025-05	1.051972e+08	Normal
17	2025-06	1.649399e+08	Normal
18	2025-07	2.698955e+07	Normal
19	2025-08	2.984112e+07	Normal
20	2025-09	7.813383e+06	Normal
21	2025-10	1.063271e+07	Normal
22	2025-11	9.757031e+06	Normal
23	2025-12	6.413953e+07	Normal

```
Spikes found: 2
Drops found: 0
```

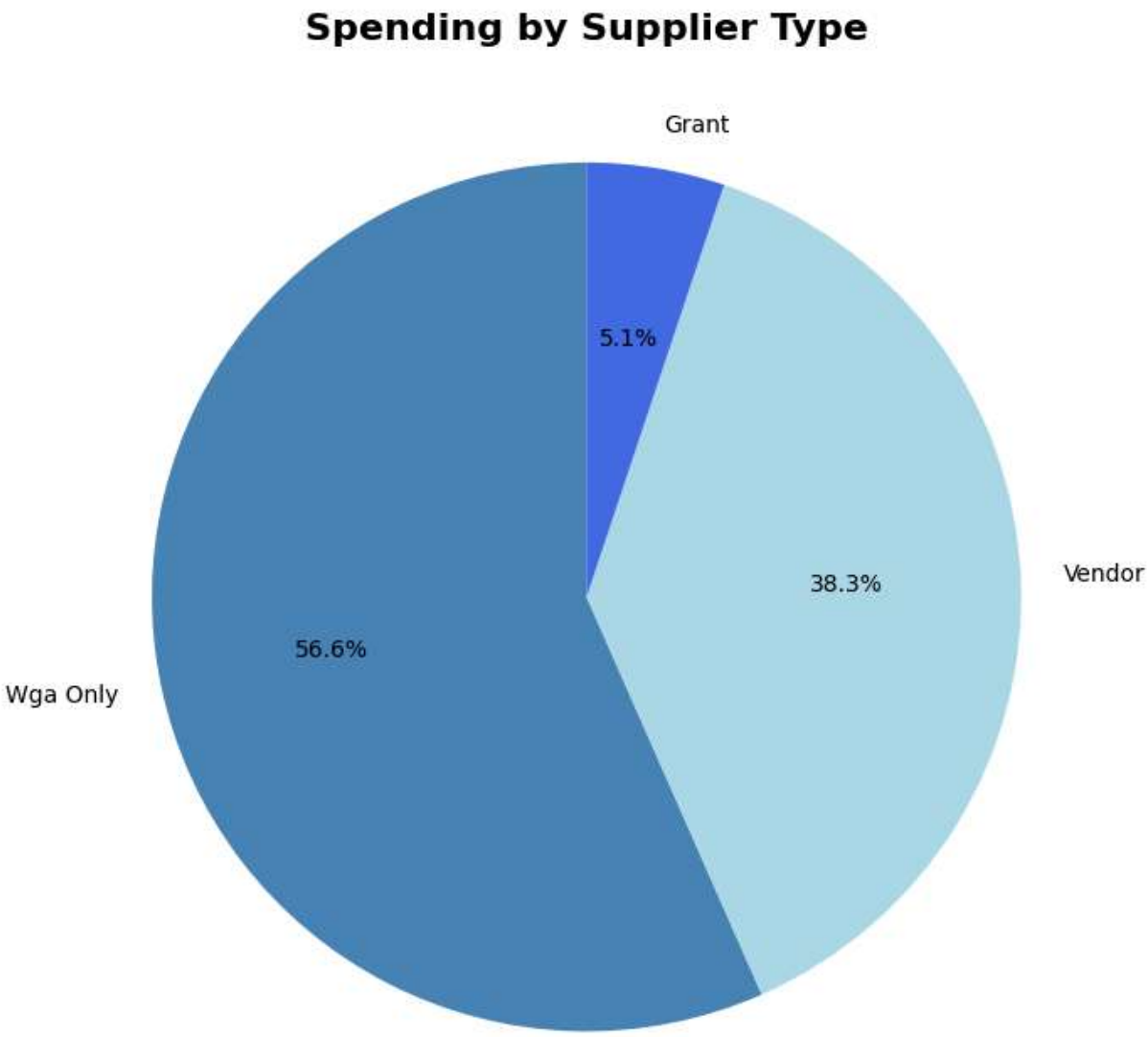
In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv('../data/processed data/master_spend_cleand_data.csv')
```

How much is spent on Vendors vs. Grants vs. other types?

In [3]:
```
supplier_type_spend = df.groupby('Supplier Type')['Amount'].sum().reset_index()
supplier_type_spend = supplier_type_spend.sort_values('Amount', ascending=False)

# Adding percentage column
supplier_type_spend['Percentage'] = (supplier_type_spend['Amount'] / supplier_type_

print("Total Spend by Supplier Type:")
print(supplier_type_spend)
```

```
Total Spend by Supplier Type:
  Supplier Type        Amount  Percentage
2      Wga Only  1.024423e+10       56.60
1        Vendor  6.926358e+09       38.27
0         Grant  9.273213e+08        5.12
```

In [4]:
```
# Plot pie chart
fig, ax = plt.subplots(figsize=(10, 7))

ax.pie(supplier_type_spend['Amount'],
       labels=supplier_type_spend['Supplier Type'],
       autopct='%1.1f%%',
       colors=['steelblue', 'lightblue', 'royalblue', 'skyblue'],
       startangle=90)

ax.set_title('Spending by Supplier Type', fontsize=16, fontweight='bold')

plt.tight_layout()
plt.show()
```

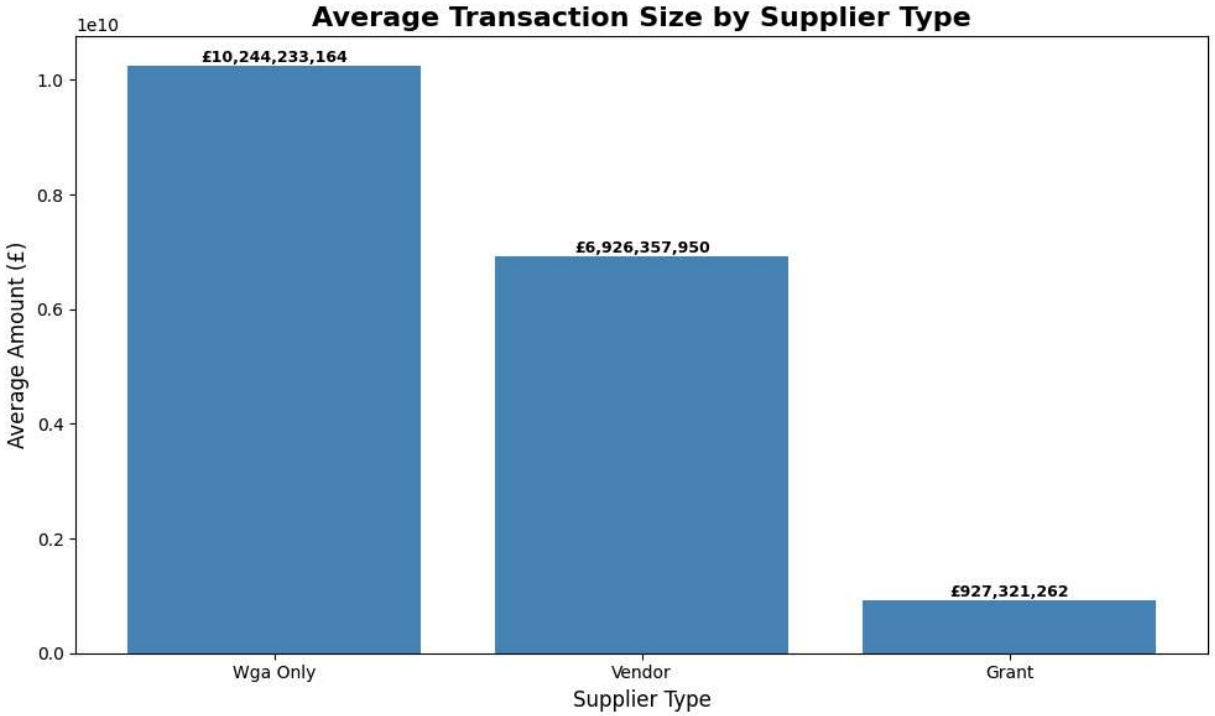


What's the average transaction size per supplier type?

In [7]:
```
supplier_type_spend = df.groupby('Supplier Type')['Amount'].mean().reset_index()
supplier_type_spend = supplier_type_spend.sort_values('Amount', ascending=False)

# Adding percentage column
supplier_type_spend['Percentage'] = (supplier_type_spend['Amount'] / supplier_type_

print("Average Spend by Supplier Type:")
print(supplier_type_spend)
```

```
Average Spend by Supplier Type:
  Supplier Type        Amount  Percentage
2      Wga Only  2.717303e+07       72.64
1        Vendor  7.536842e+06       20.15
0         Grant  2.695701e+06        7.21
```

In [5]:
```
# Plot bar chart
fig, ax = plt.subplots(figsize=(10, 6))

bars = ax.bar(supplier_type_spend['Supplier Type'], supplier_type_spend['Amount'],

# Add labels and title
ax.set_title('Average Transaction Size by Supplier Type', fontsize=16, fontweight='
ax.set_xlabel('Supplier Type', fontsize=12)
ax.set_ylabel('Average Amount (£)', fontsize=12)

# Add value labels on top of bars
for bar, amount in zip(bars, supplier_type_spend['Amount']):
    ax.text(bar.get_x() + bar.get_width()/2, bar.get_height(),
            f'£{amount:,.0f}', va='bottom', ha='center',
            fontsize=9, fontweight='bold')

plt.tight_layout()
plt.show()
```



In [ ]:
```
# Find the dominant supplier type
dominant = supplier_type_spend.iloc[0]

print(f"🏆 Dominant Supplier Type: {dominant['Supplier Type']}")
print(f" Total Spend: £{dominant['Amount']:,.2f}")
print(f" Percentage of Budget: {dominant['Percentage']}%")

# Simple conclusion
if dominant['Percentage'] > 50:
    print(f"\n✅ {dominant['Supplier Type']} DOMINATES spending with more than 50%
else:
    print(f"\n✅ Spending is relatively DISTRIBUTED across supplier types")
```

```
🏆 Dominant Supplier Type: Wga Only
💰 Total Spend: £27,173,032.27
📊 Percentage of Budget: 72.64%

✅ Wga Only DOMINATES spending with more than 50% of total budget!
```

In [ ]:

Top 10 suppliers by total spend ?

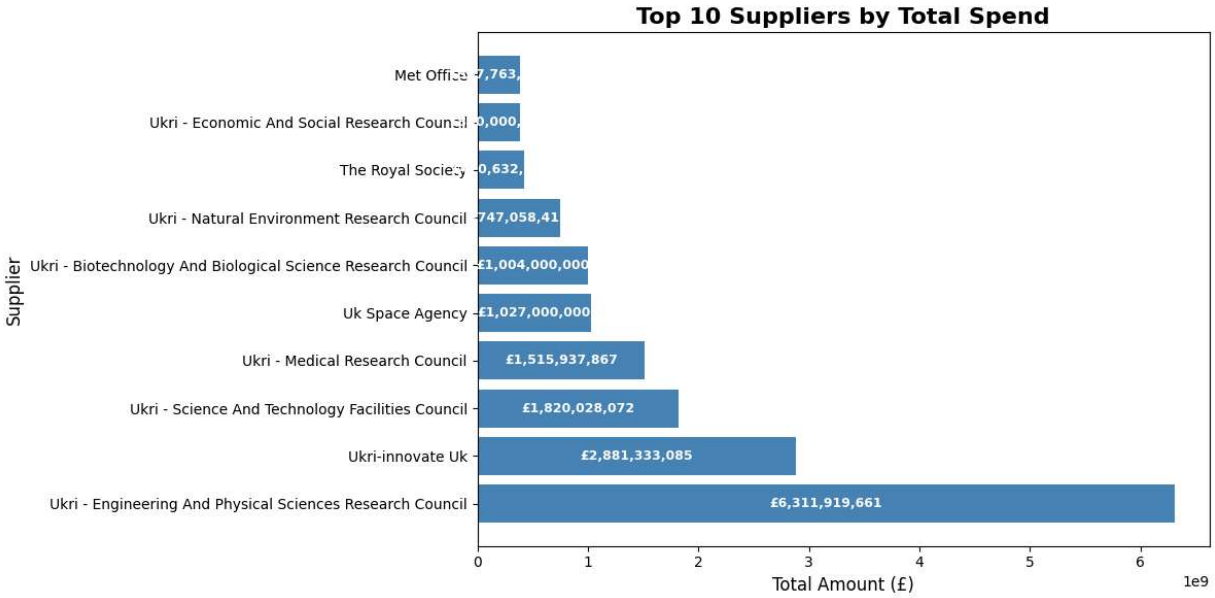
In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Loading cleaned data from processed data folder
df = pd.read_csv('../data/processed data/master_spend_cleand_data.csv')
```

In [3]:
```
top_10_supplires= df.groupby('Supplier')['Amount'].sum().reset_index()
top_10_supplires = top_10_supplires.sort_values('Amount', ascending=False).head(10)
top_10_supplires
```

Out[3]:

	Supplier	Amount
257	Ukri - Engineering And Physical Sciences Resea...	6.311920e+09
261	Ukri-innovate Uk	2.881333e+09
260	Ukri - Science And Technology Facilities Council	1.820028e+09
258	Ukri - Medical Research Council	1.515938e+09
253	Uk Space Agency	1.027000e+09
255	Ukri - Biotechnology And Biological Science Re...	1.004000e+09
259	Ukri - Natural Environment Research Council	7.470584e+08
236	The Royal Society	4.206324e+08
256	Ukri - Economic And Social Research Council	3.900000e+08
142	Met Office	3.877640e+08

In [4]:
```
# Plot horizontal bar chart
fig, ax = plt.subplots(figsize=(12, 6))

bars = ax.barh(top_10_supplires['Supplier'], top_10_supplires['Amount'], color='ste

# Add labels and title
ax.set_title('Top 10 Suppliers by Total Spend', fontsize=16, fontweight='bold')
ax.set_xlabel('Total Amount (£)', fontsize=12)
ax.set_ylabel('Supplier', fontsize=12)

# Add value labels inside bars
for bar, amount in zip(bars, top_10_supplires['Amount']):
    ax.text(bar.get_width() * 0.5, bar.get_y() + bar.get_height()/2,
            f'£{amount:,.0f}', va='center', ha='center',
            fontsize=9, color='white', fontweight='bold')

plt.tight_layout()
plt.show()
```



What percentage of total budget do they represent?

In [18]:
```
# Calculate total budget
total_budget = df['Amount'].sum()

# Add percentage column to top 10 suppliers
top_10_supplires['Percentage'] = (top_10_supplires['Amount'] / total_budget * 100).

print(f"Total Budget: £{total_budget:,.2f}")
print(top_10_supplires)
```

```
Total Budget: £18,097,912,376.53
                                                Supplier        Amount  \
257  Ukri - Engineering And Physical Sciences Resea...  6.311920e+09
261                                  Ukri-innovate Uk  2.881333e+09
260   Ukri - Science And Technology Facilities Council  1.820028e+09
258                    Ukri - Medical Research Council  1.515938e+09
253                                  Uk Space Agency  1.027000e+09
255  Ukri - Biotechnology And Biological Science Re...  1.004000e+09
259        Ukri - Natural Environment Research Council  7.470584e+08
236                                The Royal Society  4.206324e+08
256        Ukri - Economic And Social Research Council  3.900000e+08
142                                     Met Office  3.877640e+08

     Percentage
257       34.88
261       15.92
260       10.06
258        8.38
253        5.67
255        5.55
259        4.13
236        2.32
256        2.15
142        2.14
```

Is spending concentrated with few suppliers or distributed?

In [ ]:
```python
# Compare top 10 vs rest of suppliers
top10_total = top_10_supplires['Amount'].sum()
rest_total = total_budget - top10_total

print(f"Top 10 suppliers:  £{top10_total:,.2f} ({top10_total/total_budget*100:.1f}%
print(f"Rest of suppliers: £{rest_total:,.2f} ({rest_total/total_budget*100:.1f}%)"

# Conclusion
if top10_total/total_budget > 0.5:
    print("\n✅ Spending is CONCENTRATED with few suppliers")
else:
    print("\n✅ Spending is DISTRIBUTED across many suppliers")


# ChatGPT was used to generate the final output using if statements and to format l
```

```
Top 10 suppliers:  £16,505,673,492.11 (91.2%)
Rest of suppliers: £1,592,238,884.42 (8.8%)

✅ Spending is CONCENTRATED with few suppliers
```

In [ ]: