Mitron Bank - Credit Card Allocation Analysis

📌 Objective: Identify customers eligible for credit cards based on spending behavior.

📊 Dataset: Includes customer details (dim_customer) & spending patterns (fact_spend).

1 Importing Required Libraries

We first import necessary Python libraries such as Pandas, NumPy, Matplotlib, and Seaborn for data analysis and visualization.

2 Data Loading & Overview

We load the datasets and explore the first few rows to understand the structure.

3 Data Cleaning & Preprocessing

Handling missing values, removing duplicates, and formatting columns for consistency.

4 Exploratory Data Analysis (EDA)

Visualizing customer spending trends, income distribution, and identifying key patterns.

5 Graphs & Insights

Key visualizations used in the analysis.

6 Conclusion & Credit Card Allocation Strategy

Based on the analysis, we identify key customer segments suitable for credit card allocation. Factors considered: **High Utilization %, Consistent Spending, and Income Stability.**

Importing required library

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib.patches import FancyBboxPatch
import matplotlib.ticker as mticker
```

Importing Data

In [2]:
```
df_dim = pd.read_csv('dim_customers.csv')
df_fact = pd.read_csv('fact_spends.csv')

dim_customer = pd.DataFrame(df_dim)
fact_spends = pd.DataFrame(df_fact)
```

Data Cleaning & Preprocessing

Checking NULL & DUPLICATE values

Data - dim_customer

In [3]:
```
dim_customer.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4000 entries, 0 to 3999
Data columns (total 7 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   customer_id     4000 non-null   object
 1   age_group       4000 non-null   object
 2   city            4000 non-null   object
 3   occupation      4000 non-null   object
 4   gender          4000 non-null   object
 5   marital status  4000 non-null   object
 6   avg_income      4000 non-null   int64 
dtypes: int64(1), object(6)
memory usage: 218.9+ KB
```

In [4]:
```
dim_customer.head(25)
```

Out[4]:

	customer_id	age_group	city	occupation	gender	marital status	avg_income
0	ATQCUS1825	45+	Bengaluru	Salaried IT Employees	Male	Married	73523
1	ATQCUS0809	25-34	Hyderabad	Salaried Other Employees	Male	Married	39922
2	ATQCUS0663	25-34	Chennai	Salaried Other Employees	Male	Married	37702
3	ATQCUS0452	25-34	Delhi NCR	Government Employees	Male	Married	54090
4	ATQCUS3350	21-24	Bengaluru	Freelancers	Male	Single	28376
5	ATQCUS3256	21-24	Delhi NCR	Salaried IT Employees	Male	Single	46586
6	ATQCUS3608	25-34	Chennai	Freelancers	Male	Single	34966
7	ATQCUS0611	25-34	Chennai	Salaried IT Employees	Male	Married	59078
8	ATQCUS3856	21-24	Bengaluru	Salaried Other Employees	Female	Single	30424
9	ATQCUS2640	35-45	Delhi NCR	Salaried IT Employees	Female	Married	67450
10	ATQCUS3008	45+	Mumbai	Government Employees	Female	Married	62542
11	ATQCUS2365	25-34	Chennai	Salaried IT Employees	Female	Married	62738
12	ATQCUS1141	35-45	Delhi NCR	Business Owners	Male	Married	72301
13	ATQCUS2853	35-45	Chennai	Business Owners	Female	Married	74783
14	ATQCUS3020	45+	Delhi NCR	Salaried IT Employees	Female	Married	73754
15	ATQCUS2442	25-34	Chennai	Government Employees	Female	Married	54575
16	ATQCUS3943	25-34	Delhi NCR	Salaried IT Employees	Female	Single	60269
17	ATQCUS1679	45+	Mumbai	Salaried Other Employees	Male	Married	41845
18	ATQCUS0744	25-34	Chennai	Freelancers	Male	Married	33831
19	ATQCUS3335	21-24	Bengaluru	Business Owners	Male	Single	54149

	customer_id	age_group	city	occupation	gender	marital status	avg_income
20	ATQCUS0846	25-34	Hyderabad	Business Owners	Male	Married	68634
21	ATQCUS2462	25-34	Hyderabad	Salaried IT Employees	Female	Married	65608
22	ATQCUS0916	35-45	Mumbai	Salaried IT Employees	Male	Married	63264
23	ATQCUS3766	21-24	Mumbai	Salaried IT Employees	Female	Single	47229
24	ATQCUS1285	35-45	Bengaluru	Business Owners	Male	Married	74263

In [5]:
```
dim_customer.describe()
```

Out[5]:

	avg_income
count	4000.000000
mean	51657.032250
std	14690.140645
min	24816.000000
25%	38701.000000
50%	50422.000000
75%	64773.250000
max	86600.000000

In [6]:
```
dim_customer.isnull().sum()
```

Out[6]:
```
customer_id       0
age_group         0
city              0
occupation        0
gender            0
marital status    0
avg_income        0
dtype: int64
```

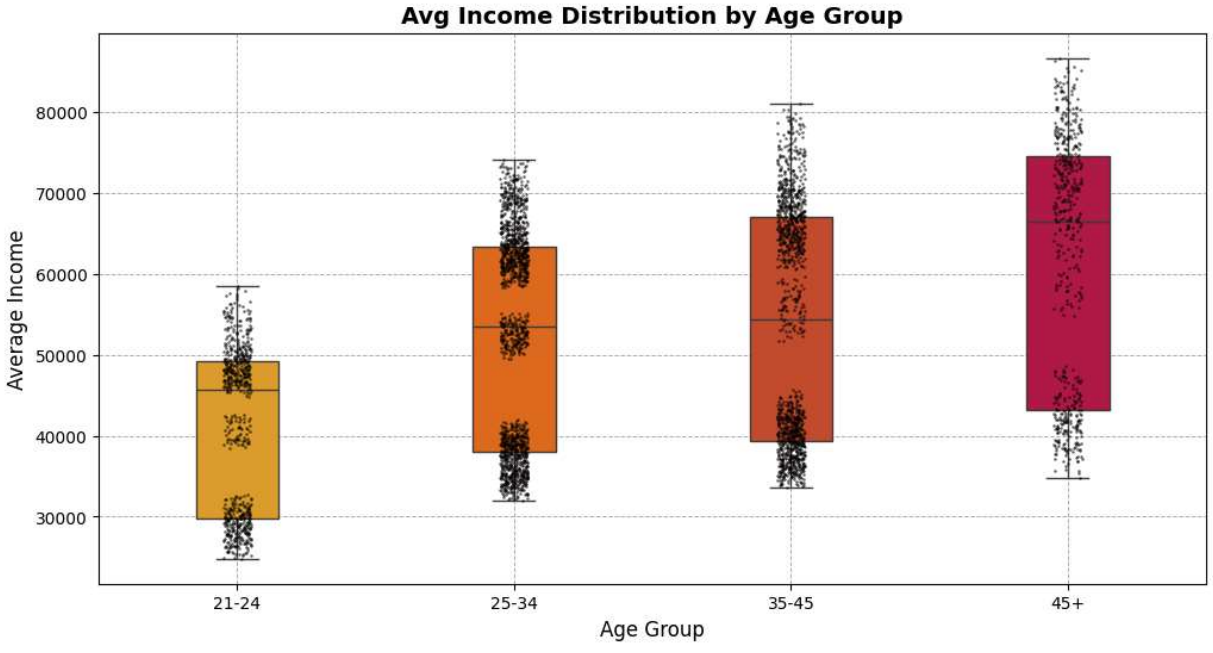
In [7]:
```
dim_customer[dim_customer.duplicated()]
```

Out[7]:

	customer_id	age_group	city	occupation	gender	marital status	avg_income

In [ ]:
```
plt.figure(figsize=(12, 6))

rocket_palette = ['#FDA50F','#FC6600','#DC3C18','#CA033E']

order = ['21-24','25-34','35-45','45+']

# Apply rocket colors to the boxplot
box = sns.boxplot(x=dim_customer['age_group'], y=dim_customer['avg_income'], width=

# Stripplot remains black for contrast
sns.stripplot(x=dim_customer['age_group'], y=dim_customer['avg_income'],
              alpha=0.6, color='#070504', size=1.8, jitter=0.05)

# Add grid
plt.grid(True, which='major', linestyle='--', linewidth=0.7, alpha=1)

# Title and labels
plt.title("Avg Income Distribution by Age Group", fontsize=14, fontweight='bold')
plt.xlabel("Age Group", fontsize=12)
plt.ylabel("Average Income", fontsize=12)

# Show plot
plt.show()
```

```
C:\Users\Professional\AppData\Local\Temp\ipykernel_5520\2552585308.py:8: FutureWarni
ng:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.1
4.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  box = sns.boxplot(x=dim_customer['age_group'], y=dim_customer['avg_income'], width
=0.3, palette=rocket_palette,order = order)
C:\Users\Professional\AppData\Local\Temp\ipykernel_5520\2552585308.py:8: UserWarnin
g: The palette list has more values (6) than needed (4), which may not be intended.
  box = sns.boxplot(x=dim_customer['age_group'], y=dim_customer['avg_income'], width
=0.3, palette=rocket_palette,order = order)
```



📊 Boxplot (Avg Income vs. Age Group)

Insight: Income increases with age, suggesting that older customers generally earn more.

The income range varies across age groups, with the widest variation in the 35-45 and 45+ groups.

Some outliers indicate customers with significantly higher or lower incomes than their peers.

Data - fact_spends

In [9]:
```
fact_spends.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 864000 entries, 0 to 863999
Data columns (total 5 columns):
 #   Column        Non-Null Count   Dtype 
---  ------        --------------   ----- 
 0   customer_id   864000 non-null  object
 1   month         864000 non-null  object
 2   category      864000 non-null  object
 3   payment_type  864000 non-null  object
 4   spend         864000 non-null  int64 
dtypes: int64(1), object(4)
memory usage: 33.0+ MB
```

In [10]:
```
fact_spends.head(10)
```

Out[10]:

	customer_id	month	category	payment_type	spend
0	ATQCUS1371	July	Health & Wellness	Credit Card	1114
1	ATQCUS0368	October	Groceries	Credit Card	1466
2	ATQCUS0595	May	Health & Wellness	Credit Card	387
3	ATQCUS0667	October	Electronics	Credit Card	1137
4	ATQCUS3477	September	Bills	UPI	2102
5	ATQCUS1972	October	Health & Wellness	UPI	243
6	ATQCUS2843	June	Entertainment	Credit Card	268
7	ATQCUS2634	August	Apparel	Debit Card	737
8	ATQCUS2165	July	Food	UPI	506
9	ATQCUS0908	September	Others	UPI	380

In [11]:
```
fact_spends.describe()
```

Out[11]:

	spend
count	864000.000000
mean	614.464994
std	661.571676
min	6.000000
25%	191.000000
50%	395.000000
75%	793.000000
max	10313.000000

In [12]:
```
fact_spends[fact_spends.duplicated()]
```

Out[12]:

customer_id	month	category	payment_type	spend

In [13]:
```
customer_spends = pd.merge(fact_spends , dim_customer , on = 'customer_id')
```

In [14]:
```
customer_spends.head(10)
```

Out[14]:

	customer_id	month	category	payment_type	spend	age_group	city	occupat
0	ATQCUS1371	July	Health & Wellness	Credit Card	1114	35-45	Chennai	Salarie Employ
1	ATQCUS1371	May	Bills	UPI	787	35-45	Chennai	Salarie Employ
2	ATQCUS1371	September	Bills	Net Banking	1022	35-45	Chennai	Salarie Employ
3	ATQCUS1371	September	Bills	UPI	1942	35-45	Chennai	Salarie Employ
4	ATQCUS1371	August	Electronics	Credit Card	1865	35-45	Chennai	Salarie Employ
5	ATQCUS1371	October	Travel	Net Banking	149	35-45	Chennai	Salarie Employ
6	ATQCUS1371	October	Others	Credit Card	280	35-45	Chennai	Salarie Employ
7	ATQCUS1371	August	Travel	Net Banking	186	35-45	Chennai	Salarie Employ
8	ATQCUS1371	August	Groceries	Net Banking	440	35-45	Chennai	Salarie Employ
9	ATQCUS1371	July	Travel	Net Banking	163	35-45	Chennai	Salarie Employ

---

In this step below we are adding a new column called 'utilisation_%' to customer_spends dataset.

The 'utilisation_%' column represent the percentage of a customer's spending relative to their availabel income. This metric helps in understanding spending behavior and financial utilisation patterns among different customer segments.

In [15]:
```
customer_spends['utilisation_%'] = (customer_spends.apply(lambda x: x['spend']/x['a
customer_spends
```

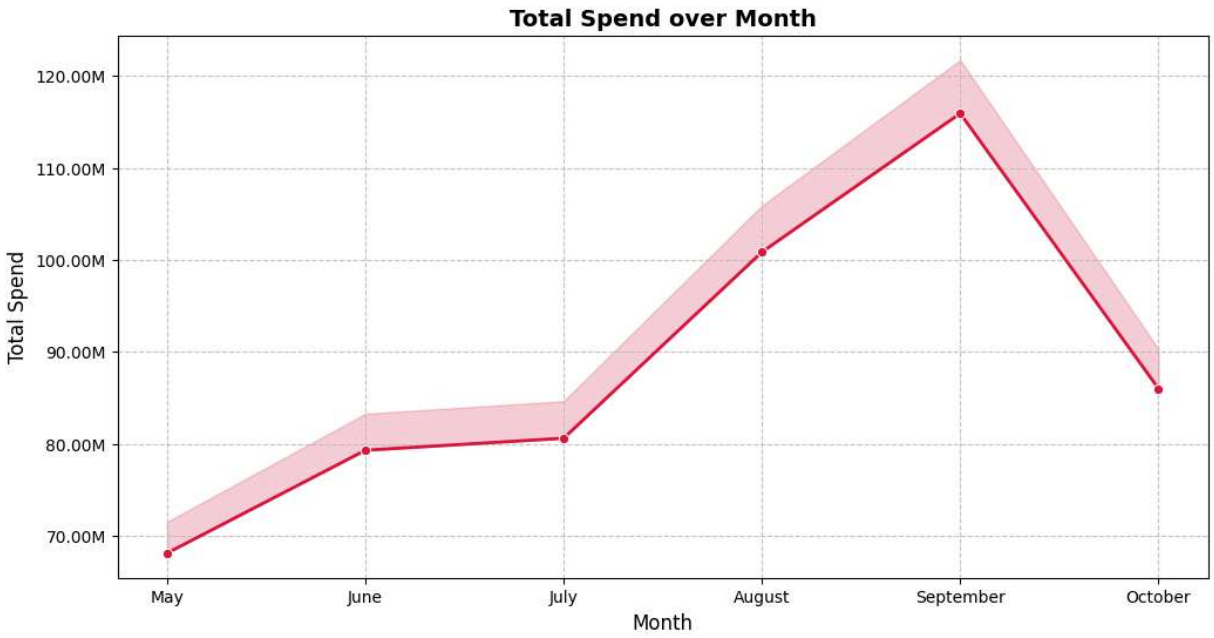
Out[15]:

	customer_id	month	category	payment_type	spend	age_group	city	o
0	ATQCUS1371	July	Health & Wellness	Credit Card	1114	35-45	Chennai	
1	ATQCUS1371	May	Bills	UPI	787	35-45	Chennai	
2	ATQCUS1371	September	Bills	Net Banking	1022	35-45	Chennai	
3	ATQCUS1371	September	Bills	UPI	1942	35-45	Chennai	
4	ATQCUS1371	August	Electronics	Credit Card	1865	35-45	Chennai	
...	...	...	...	...	...	...	...	
863995	ATQCUS0890	October	Groceries	Net Banking	764	35-45	Mumbai	
863996	ATQCUS0890	September	Health & Wellness	Debit Card	1188	35-45	Mumbai	
863997	ATQCUS0890	September	Electronics	UPI	2786	35-45	Mumbai	
863998	ATQCUS0890	August	Groceries	Debit Card	2593	35-45	Mumbai	
863999	ATQCUS0890	August	Groceries	UPI	3074	35-45	Mumbai	

864000 rows × 12 columns

In [16]:
```
# Define the correct order for months
month_order = ["May", "June", "July", "August", "September", "October"]

# Convert 'month' column to categorical type with the correct order
customer_spends['month'] = pd.Categorical(customer_spends['month'], categories=mont

# Aggregate total spend per month
monthly_spend = customer_spends.groupby("month")["spend"].sum().reset_index()

x_values = np.arange(len(monthly_spend["month"]))
y_values = monthly_spend["spend"]

# Plot the line chart
plt.figure(figsize=(12, 6))
sns.lineplot(x="month", y="spend", data=monthly_spend, marker="o", color="crimson",

plt.fill_between(x_values, y_values * 1.05, y_values * 1, color="crimson", alpha=0.
```

```
# Formatting Y-axis to show values in Millions (M)
ax = plt.gca()
ax.yaxis.set_major_formatter(mticker.FuncFormatter(lambda x, _: f"{x/1_000_000:.2f}

# Title and labels
plt.title("Total Spend over Month", fontsize=14, fontweight='bold')
plt.xlabel("Month", fontsize=12)
plt.ylabel("Total Spend", fontsize=12)

# Add grid for better readability
plt.grid(True, linestyle="--", alpha=0.7)

# Show plot
plt.show()
```



📊 Line Chart (Month vs Total Spend)

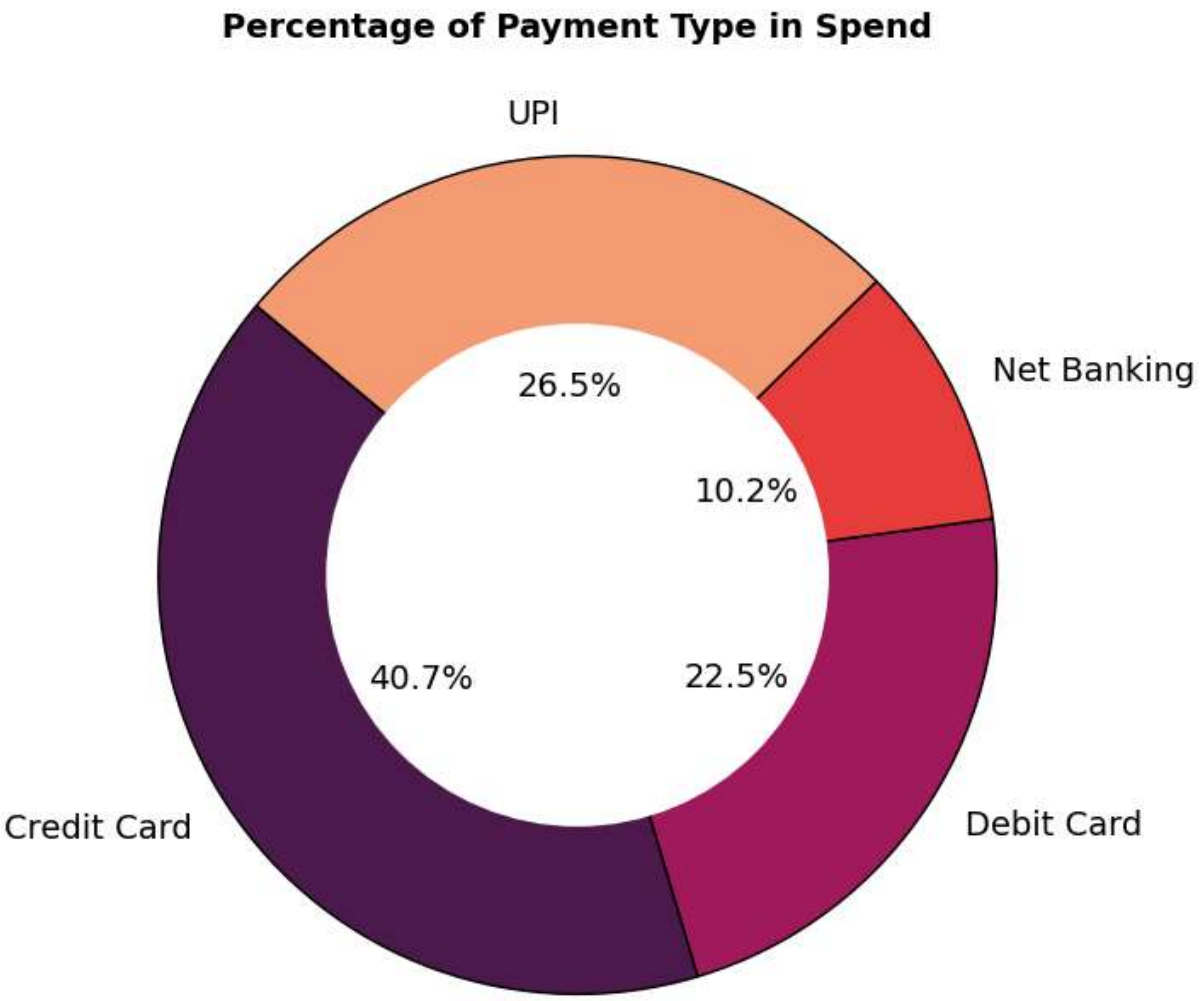
Insight: The trend shows a gradual increase in spending from May to September, peaking in September, followed by a decline in October.

This could indicate seasonal spending patterns or specific events driving higher expenditures in certain months.

In [17]:

```
# Aggregate Spend by Payment Type
payment_spend = customer_spends.groupby("payment_type")["spend"].sum()

# Get Labels & Values
labels = payment_spend.index
sizes = payment_spend.values

# Get 'rocket' color palette from Seaborn
```

```
num_colors = len(labels)  # Ensure we get the exact number of colors needed
colors = sns.color_palette("rocket", num_colors)

# Create Donut Chart
plt.figure(figsize=(8, 8))

plt.pie(sizes, labels=labels, autopct='%1.1f%%', colors=colors, startangle=140,
        wedgeprops={'edgecolor': 'black'}, textprops={'fontsize': 14}, pctdistance=

# Add White Circle in Center for Donut Effect
center_circle = plt.Circle((0, 0), 0.60, fc='white')
plt.gca().add_artist(center_circle)

# Title and labels
plt.title("Percentage of Payment Type in Spend", fontsize=14, fontweight='bold')
plt.show()
```



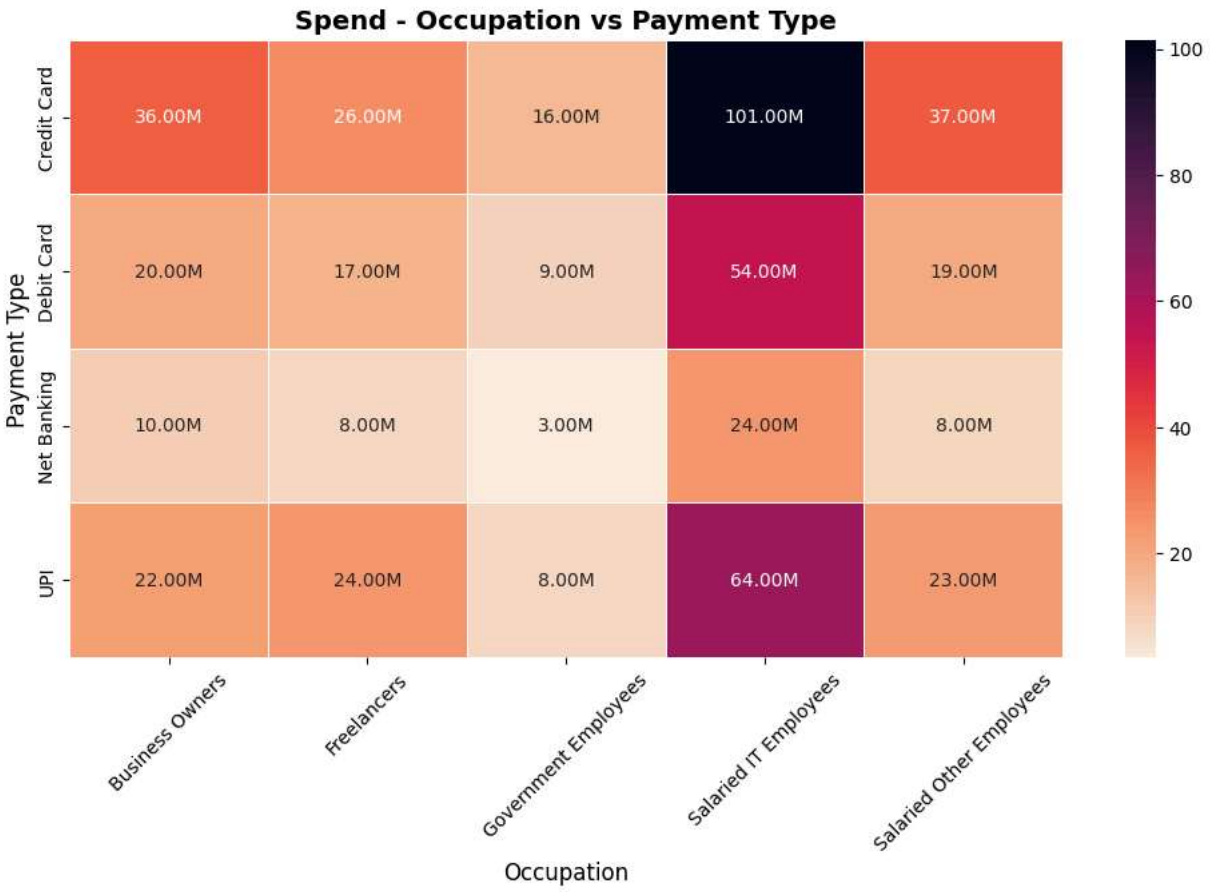
📊 Donut Chart (Total Spend % through Payment Type)

Insight: Credit Cards dominate transactions with 40.7% of total spend, followed by UPI (26.5%) and Debit Cards (22.5%).

Net Banking has the least share (10.2%), indicating a preference for card-based and digital transactions over traditional online banking.

This insight can help Mitron Bank in designing better credit card offers and UPI-based incentives to encourage more spending.

In [18]:
```
order = ['May','June','July','August','September','October']

# Convert 'month' column to categorical with a specific order
customer_spends["month"] = pd.Categorical(customer_spends["month"],ordered=True)

# Step 1: Aggregate total spend per category per month
category_month_spend = customer_spends.groupby(["payment_type", "occupation"])["spe

category_month_spend["spend"] = category_month_spend["spend"] / 1_000_000

# Step 2: Pivot data for heatmap format
pivot_table = category_month_spend.pivot(index="payment_type", columns="occupation"

# Step 3: Plot heatmap
plt.figure(figsize=(12, 6))
ax = sns.heatmap(pivot_table, cmap="rocket_r", annot=True, fmt=".0f", linewidths=0.

for text in ax.texts:
    text.set_text(f"{float(text.get_text()):.2f}M")

# Title and labels
plt.title("Spend - Occupation vs Payment Type", fontsize=14, fontweight='bold')
plt.xlabel("Occupation",fontsize=12)
plt.ylabel("Payment Type",fontsize=12)
plt.xticks(rotation=45)

plt.show()
```



📊 Heatmap (Total Spend across different payment Categories for various Occupation)

Insight: IT Employees spend the most using Credit Cards (101M) and UPI (64M), highlighting their digital payment preference.

Business Owners also have significant spending on Credit Cards (36M) and UPI (22M).

Net Banking is the least preferred payment mode, showing relatively lower values across all customer categories.

This data helps in targeting high-spending customer segments with tailored financial products and marketing strategies.

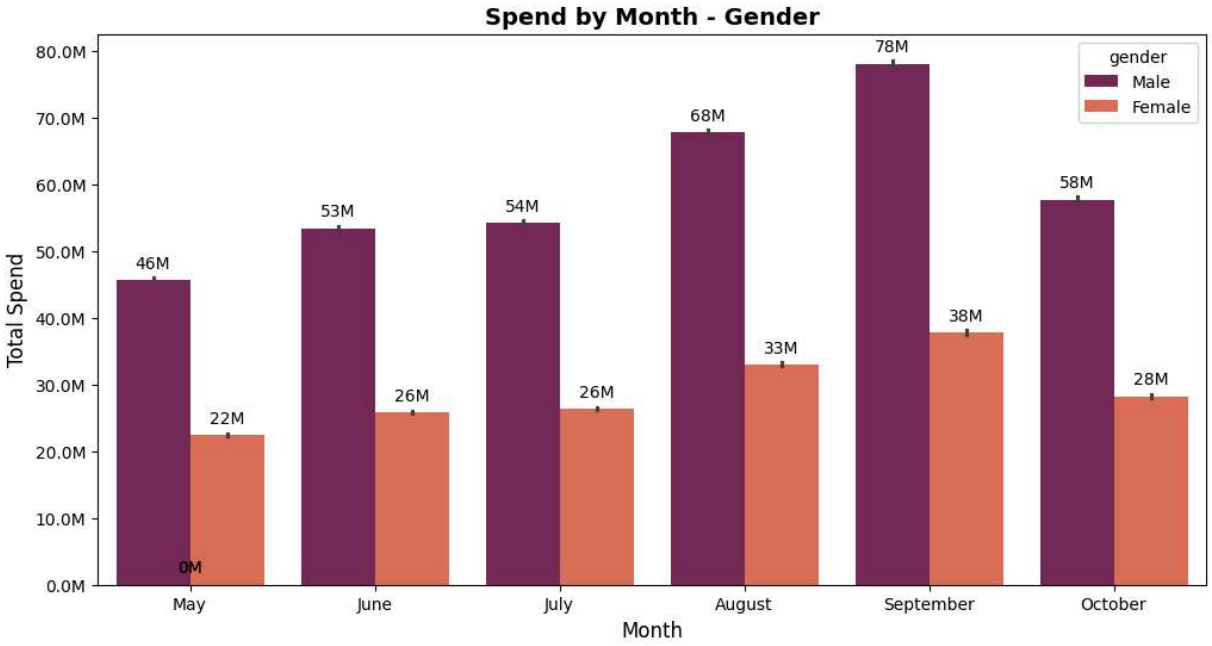
Note: The following annotation function is used to label bar heights in millions for better readability in the chart. Instead of displaying raw numbers, it formats values in millions (M) by dividing them by 1,000,000 and rounding them to the nearest whole number.

In [19]:
```
def annotation(ax, scale=1000000, fmt='{:.0f}M'):
    for p in ax.patches:
        ax.annotate(f'{p.get_height()/1000000:.0f}M',
        (p.get_x() + p.get_width() / 2., p.get_height()),
```

```
        ha='center', va='bottom',
        xytext=(0, 5), textcoords='offset points')
```

In [20]:
```
def annotation_h(ax, fmt='{:.0f}M'):
    for p in ax.patches:
        ax.annotate(f'{p.get_width():.0f}M',
                    (p.get_width(), p.get_y() + p.get_height() / 2.),
                    ha='left', va='center',
                    xytext=(5, 0), textcoords='offset points')
```

In [21]:
```
plt.figure(figsize = (12,6))

order = ['May','June','July','August','September','October']

ax = sns.barplot(data = customer_spends , x = 'month' , y = 'spend' ,
                 estimator = sum , hue = 'gender', palette = 'rocket', order = orde

ax.yaxis.set_major_formatter(mticker.FuncFormatter(lambda x, _: f'{x/1_000_000:.1f}

annotation(ax)

# Title and labels
plt.title("Spend by Month - Gender", fontsize=14, fontweight='bold')
plt.xlabel("Month", fontsize=12)
plt.ylabel("Total Spend", fontsize=12)
plt.show()
```



📊 Grouped Bar Chart (Total Spend by Gender across different Months)

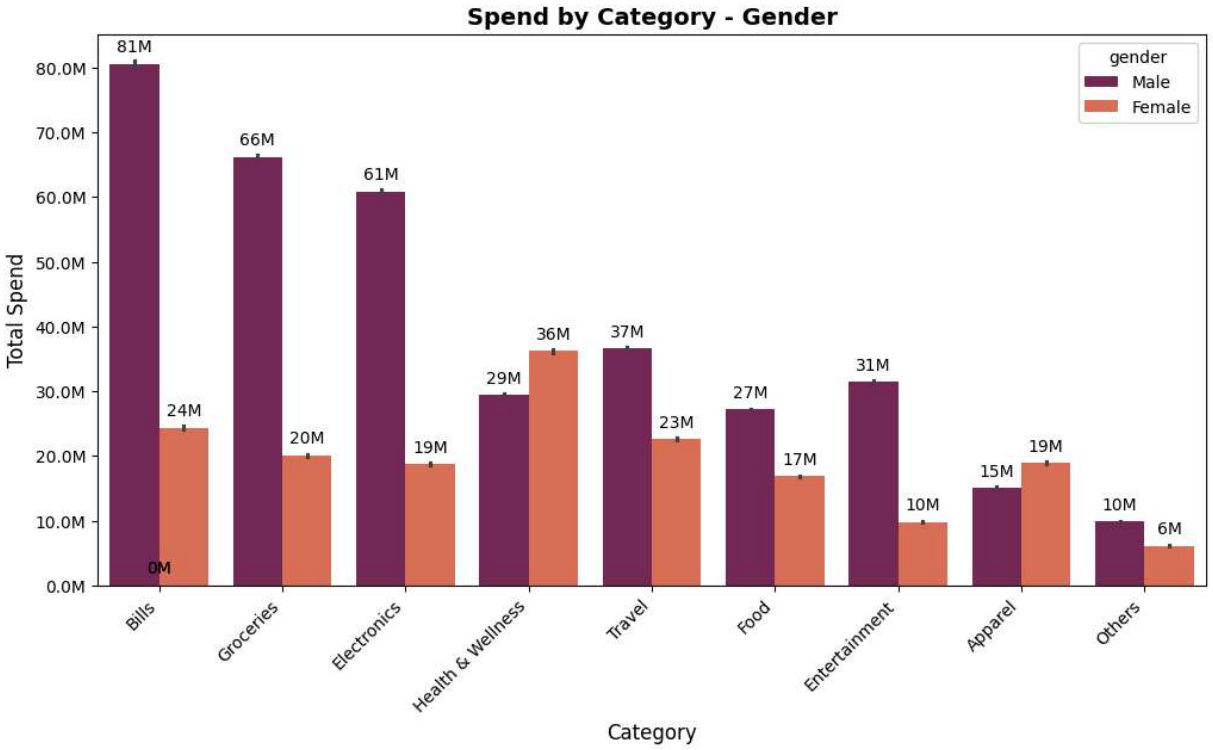
Insight: Male customers consistently spend more than female customers in all months.

September recorded the highest spending for both genders, with Males at 78M and Females at 38M.

Spending by females shows a gradual increase from May to September but drops in October.

Male spending also peaks in September but remains significantly higher than female spending throughout the months.

In [22]:
```
plt.figure(figsize = (12,6))

spend_order = customer_spends.groupby('category')['spend'].sum().sort_values(ascend


ax = sns.barplot(data = customer_spends , x = 'category', y = 'spend',
                 estimator = sum , hue = 'gender' , palette = 'rocket' , order = sp

ax.yaxis.set_major_formatter(mticker.FuncFormatter(lambda x, _: f'{x/1_000_000:.1f}

annotation(ax)

# Title and labels
plt.title("Spend by Category - Gender", fontsize=14, fontweight='bold')
plt.xlabel("Category", fontsize=12)
plt.ylabel("Total Spend", fontsize=12)
plt.xticks(rotation=45, ha='right')
plt.show()
```



📊 Grouped Bar Chart (Total Spend by Gender across various Category)

Insight: Males spend significantly more on Bills (81M), Groceries (66M), and Electronics (61M) compared to females.

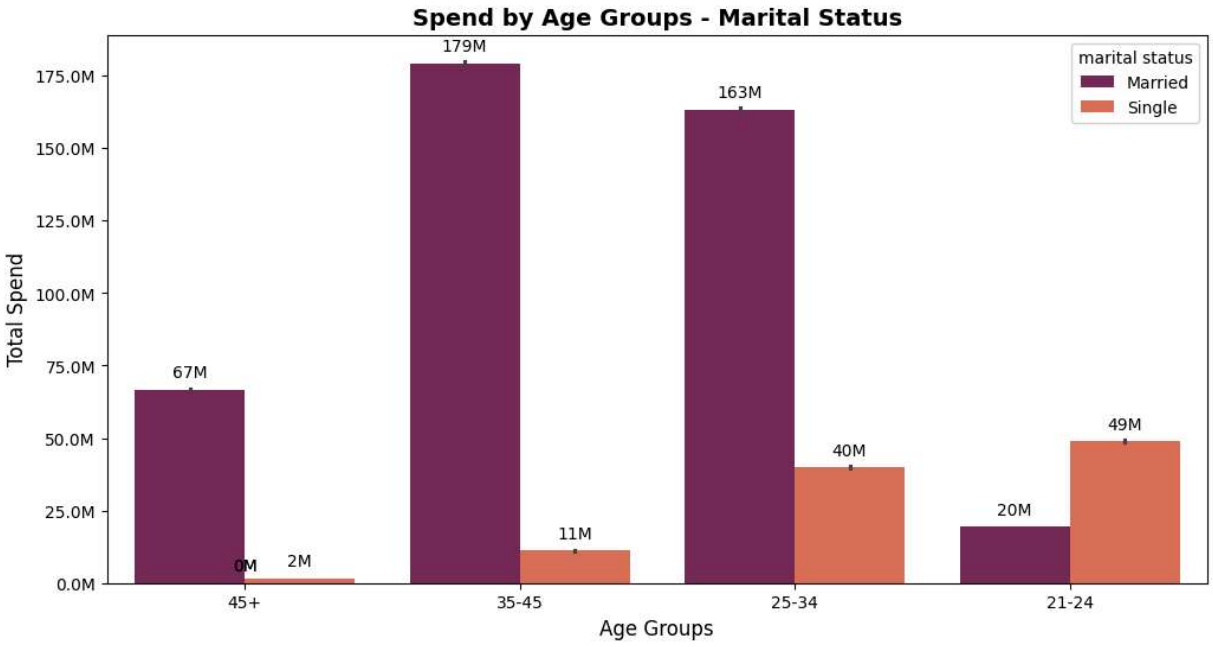
Females have higher spending in Health & Wellness (36M) compared to males (29M), possibly indicating a greater focus on personal care.

Spending in Travel is almost equal, with Males at 37M and Females at 36M.

Males spend significantly more on Entertainment (31M vs. 10M for females), suggesting a stronger preference for leisure activities.

In Apparel, females spend more (19M vs. 15M for males), which aligns with traditional shopping trends.

In [23]:
```
order = ['45+','35-45','25-34','21-24']
plt.figure(figsize = (12,6))

ax = sns.barplot(data = customer_spends , x = 'age_group' , y = 'spend' ,
                 estimator = sum , hue = 'marital status', palette = 'rocket',order

ax.yaxis.set_major_formatter(mticker.FuncFormatter(lambda x, _: f'{x/1_000_000:.1f}

annotation(ax)

# Title and labels
plt.title("Spend by Age Groups - Marital Status", fontsize=14, fontweight='bold')
plt.xlabel("Age Groups", fontsize=12)
plt.ylabel("Total Spend", fontsize=12)
plt.show()
```



📊 Grouped Bar Chart (Total Spend by Marital Status across various Age Group)

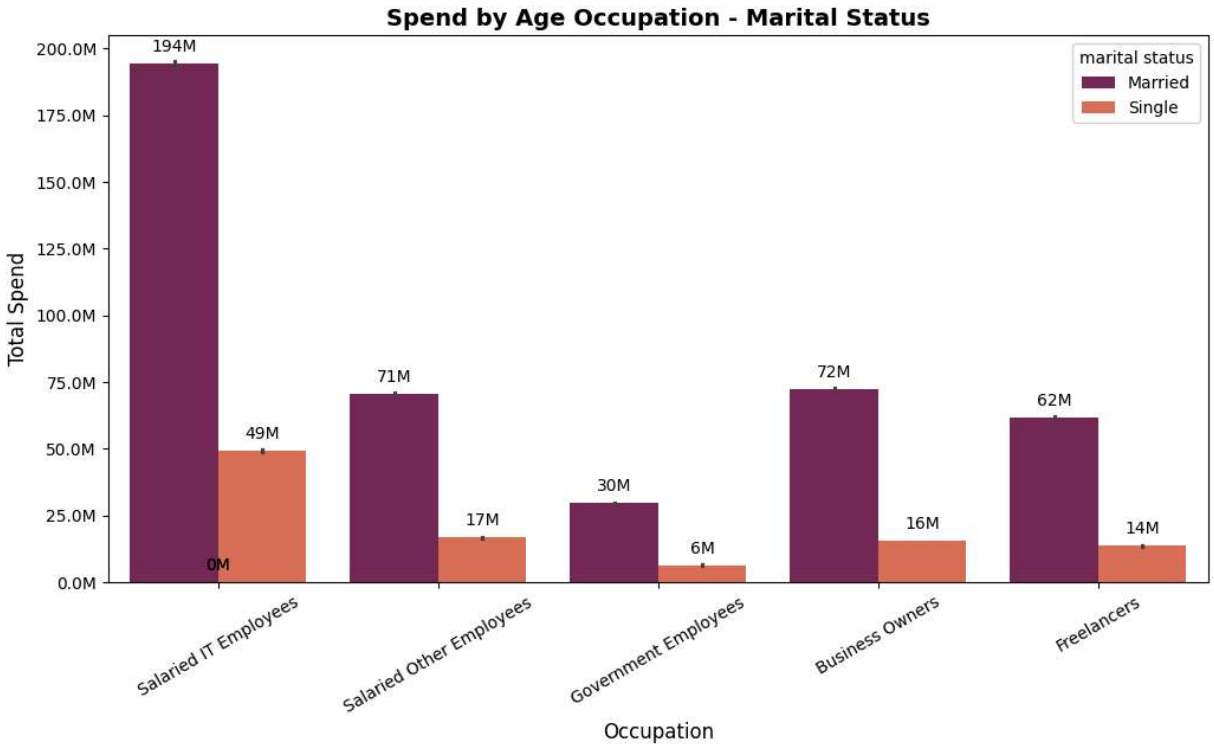
Insight: Married individuals in the 35-45 age group spend the most (179M), followed by the 25-34 group (163M).

Single individuals in the 21-24 age group spend significantly more (49M) compared to their married counterparts (20M).

Spending among singles decreases with age, while married individuals exhibit higher spending trends in older age groups (especially 35-45 and 45+).

The 45+ single group spends almost nothing (2M), while their married counterparts still maintain a spending level of 67M.

In [24]:
```
plt.figure(figsize = (12,6))

ax = sns.barplot(data = customer_spends , x = 'occupation' , y = 'spend' ,
                 estimator = sum , hue = 'marital status', palette = 'rocket')

ax.yaxis.set_major_formatter(mticker.FuncFormatter(lambda x, _: f'{x/1_000_000:.1f}

annotation(ax)

# Title and labels
plt.title("Spend by Age Occupation - Marital Status", fontsize=14, fontweight='bold
plt.xlabel("Occupation", fontsize=12)
plt.ylabel("Total Spend", fontsize=12)
plt.xticks(rotation=30)
plt.show()
```



📊 Grouped Bar Chart (Total Spend by Marital Status across various Occupation)

Insight: Salaried IT Employees (Married) have the highest spending (almost 190M), while their single counterparts spend 49M.

Married Salaried Other Employees (71M) spend significantly more than Single Salaried Other Employees (17M).

Government Employees show the least spending across both marital statuses, with Married spending 30M and Single spending only 6M.

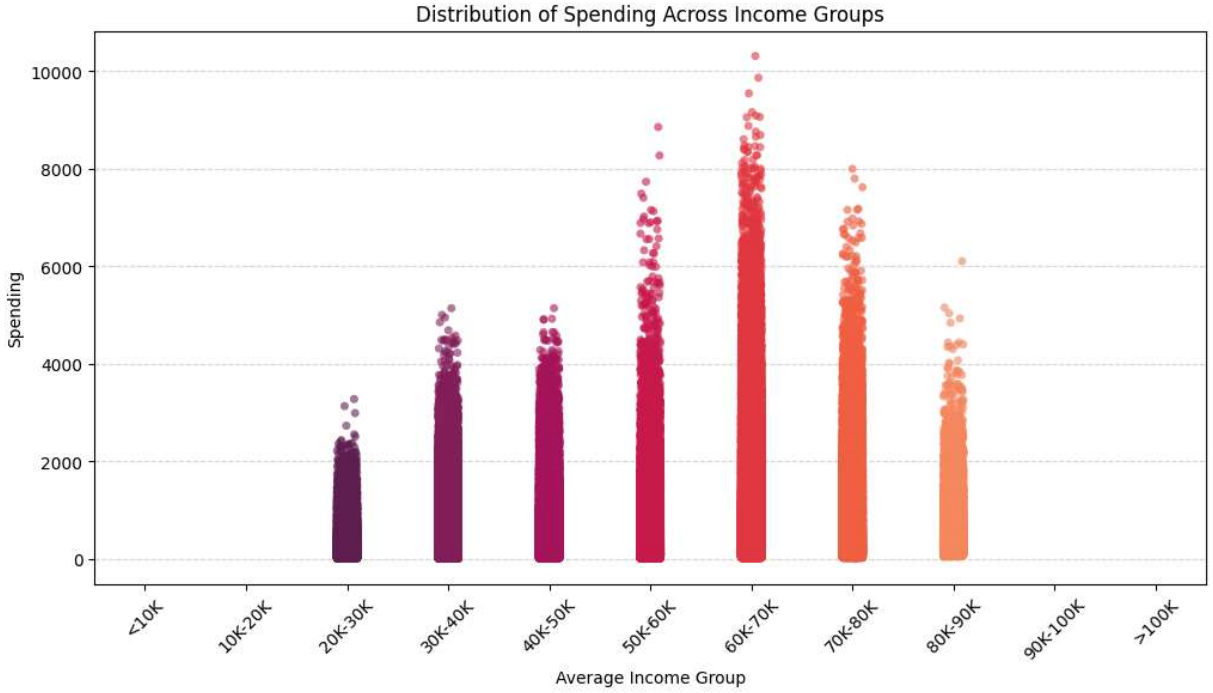
Business Owners (72M) and Freelancers (62M) who are married tend to spend more compared to their single counterparts (16M and 14M, respectively).

Note:
It has been difficult to plot large amounts of average salary data for 4000 customers, so I have divided all the average salaries into groups for better visualization.

In [25]:
```
bins = [0, 10000, 20000, 30000, 40000, 50000, 60000, 70000, 80000, 90000, 100000, f
labels = ["<10K", "10K-20K", "20K-30K", "30K-40K", "40K-50K", "50K-60K", "60K-70K",
          "70K-80K", "80K-90K", "90K-100K", ">100K"]

# Create income groups
customer_spends["avg_income_group"] = pd.cut(customer_spends["avg_income"], bins=bi
```

In [26]:
```
# Plot the box plot
plt.figure(figsize=(12, 6))
sns.stripplot(x="avg_income_group", y="spend", data=customer_spends, jitter=True, a

# Labels & Title
plt.xlabel("Average Income Group")
plt.ylabel("Spending")
plt.title("Distribution of Spending Across Income Groups")
plt.xticks(rotation=45)  # Rotate x-axis labels for better readability
plt.grid(axis="y", linestyle="--", alpha=0.5)

plt.show()
```

```
C:\Users\Professional\AppData\Local\Temp\ipykernel_5520\3665154274.py:3: FutureWarni
ng:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.1
4.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.stripplot(x="avg_income_group", y="spend", data=customer_spends, jitter=True,
alpha=0.6,palette = 'rocket')
```



📊 Spending Across Income Groups

Insight: Higher-income groups (example- 60K-70K, 70K-80K) exhibit a wider range of spending, suggesting more variance.

Lower-income groups have more concentrated spending patterns.

Spending increases as income level rises, but there is also a spread within each category.

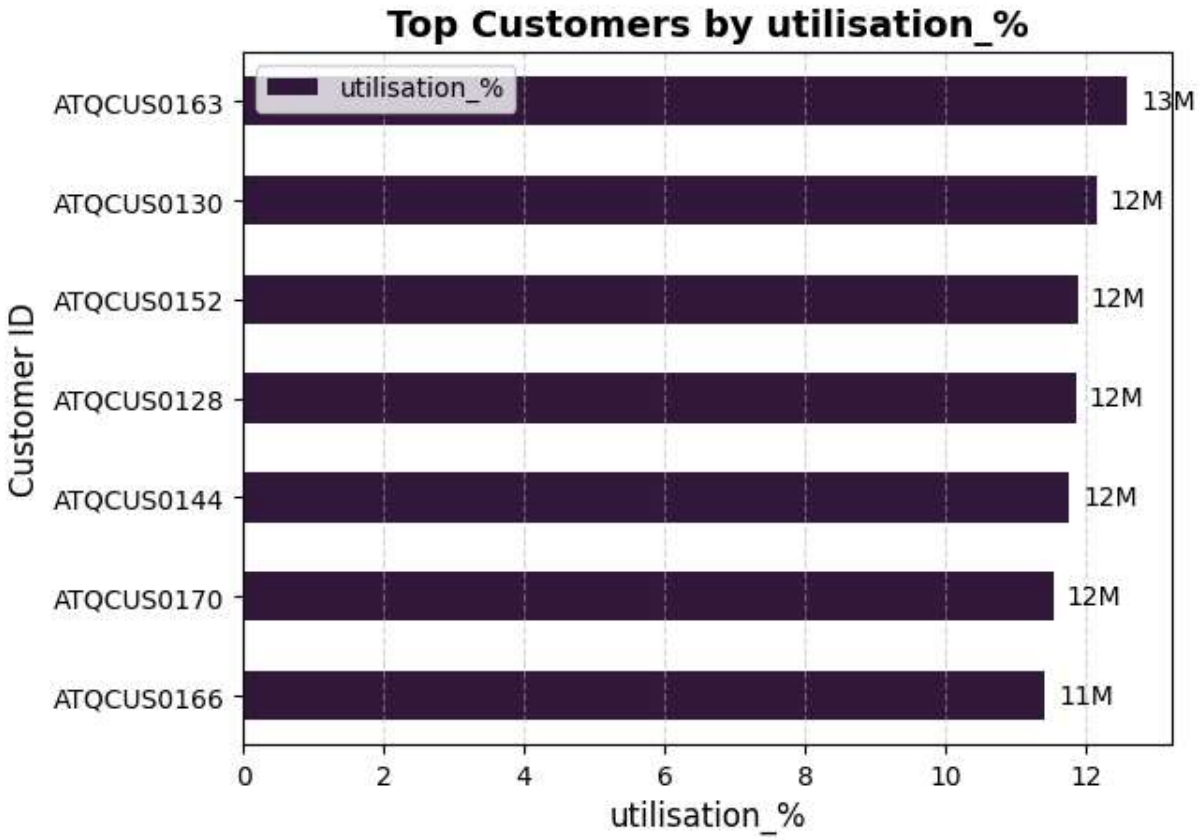
In [ ]:
```
# Filter rows where utilisation_% is >= 10
filtered_df = customer_spends[customer_spends["utilisation_%"] >= 10]

# Count occurrences of each customer_id
count_df = filtered_df.groupby("customer_id")["utilisation_%"].count().reset_index(

# Rename the count column
count_df.rename(columns={"utilisation_%": "count"}, inplace=True)

count_df = count_df[count_df["count"] >= 3]

# Merge the filtered data with the count dataframe
result_df = filtered_df.merge(count_df, on="customer_id")

top_customer = pd.DataFrame(result_df.groupby('customer_id')['utilisation_%'].mean(
top_customer
```

Out[ ]:

customer_id	utilisation_%
ATQCUS0128	11.863308
ATQCUS0130	12.143805
ATQCUS0144	11.767553
ATQCUS0152	11.885906
ATQCUS0163	12.594351
ATQCUS0166	11.411514
ATQCUS0170	11.539916

In [28]:
```
plt.figure(figsize=(25, 6))

rocket_colors = sns.color_palette("rocket", 6)

ax = top_customer.sort_values('utilisation_%').plot(kind="barh", color = rocket_col

annotation_h(ax)

# Title and labels
plt.title("Top Customers by utilisation_%", fontsize=14, fontweight='bold')
plt.xlabel("utilisation_%", fontsize=12)
plt.ylabel("Customer ID", fontsize=12)
plt.grid(axis="x", linestyle="--", alpha=0.5)
plt.show()
```

```
<Figure size 2500x600 with 0 Axes>
```



In [ ]:
```
order = ['May','June','July','August','September','October']

# Convert 'month' column to categorical with a specific order
customer_spends["month"] = pd.Categorical(customer_spends["month"], categories=orde

# Aggregate total spend per category per month
category_month_spend = customer_spends.groupby(["month", "category"])["spend"].sum(

category_month_spend["spend"] = category_month_spend["spend"] / 1_000_000

# Pivot data for heatmap format
pivot_table = category_month_spend.pivot(index="category", columns="month", values=

# Plot heatmap
plt.figure(figsize=(12, 6))
ax = sns.heatmap(pivot_table, cmap="rocket_r", annot=True, fmt=".0f", linewidths=0.

for text in ax.texts:
    text.set_text(f"{float(text.get_text()):.2f}M")

# Title and labels
plt.title("Monthly Spend by Category", fontsize=14, fontweight='bold')
plt.xlabel("Month",fontsize=12)
plt.ylabel("Category",fontsize=12)
plt.xticks(rotation=45)

plt.show()
```

📊 Monthly Spend by Category

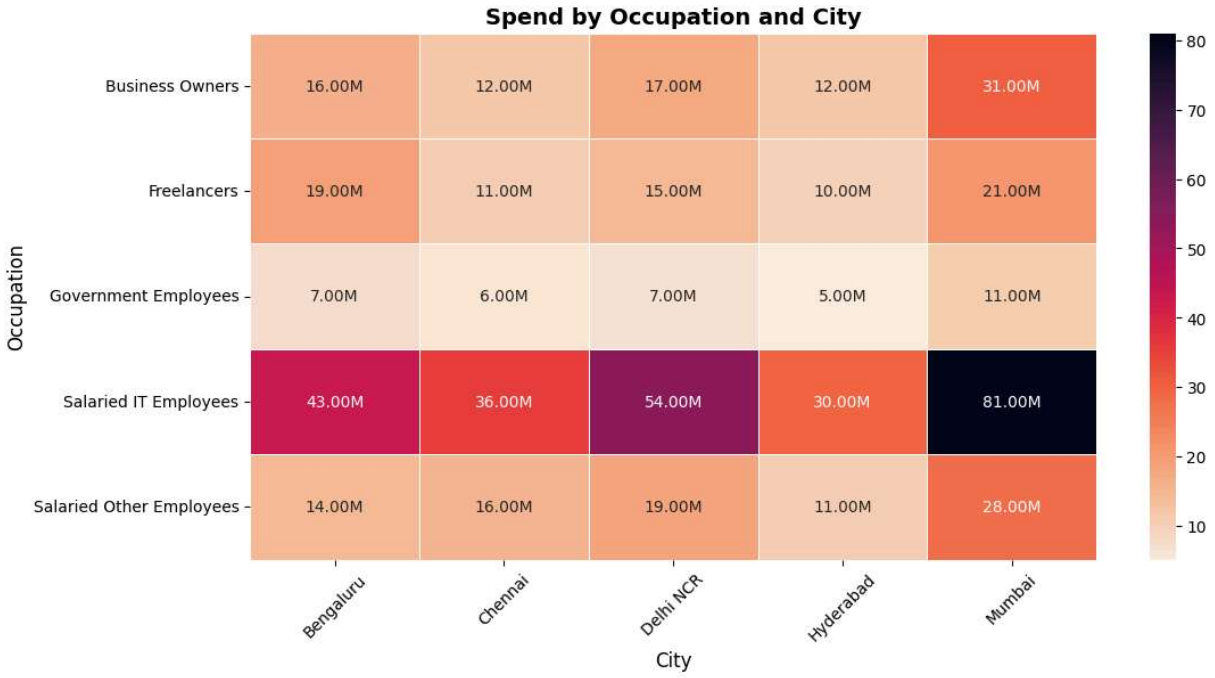
Insight: Bills and Groceries have consistently higher spending, especially in September and October.

Electronics spending peaks in August and September.

"Others" category has the lowest spending across all months.

In [ ]:
```
# Aggregate total spend per category per month
category_month_spend = customer_spends.groupby(["occupation", "city"])["spend"].sum

category_month_spend["spend"] = category_month_spend["spend"] / 1_000_000

# Pivot data for heatmap format
pivot_table = category_month_spend.pivot(index="occupation", columns="city", values

# Plot heatmap
plt.figure(figsize=(12, 6))
ax = sns.heatmap(pivot_table, cmap="rocket_r", annot=True, fmt=".0f", linewidths=0.

for text in ax.texts:
    text.set_text(f"{float(text.get_text()):.2f}M")

# Title and labels
plt.title("Spend by Occupation and City", fontsize=14, fontweight='bold')
plt.xlabel("City",fontsize=12)
plt.ylabel("Occupation",fontsize=12)
plt.xticks(rotation=45)
```

```
plt.show()
```



📊 Spend by Occupation and City

Insight: Salaried IT Employees have the highest spending, particularly in Mumbai (81M) and Delhi NCR (54M).

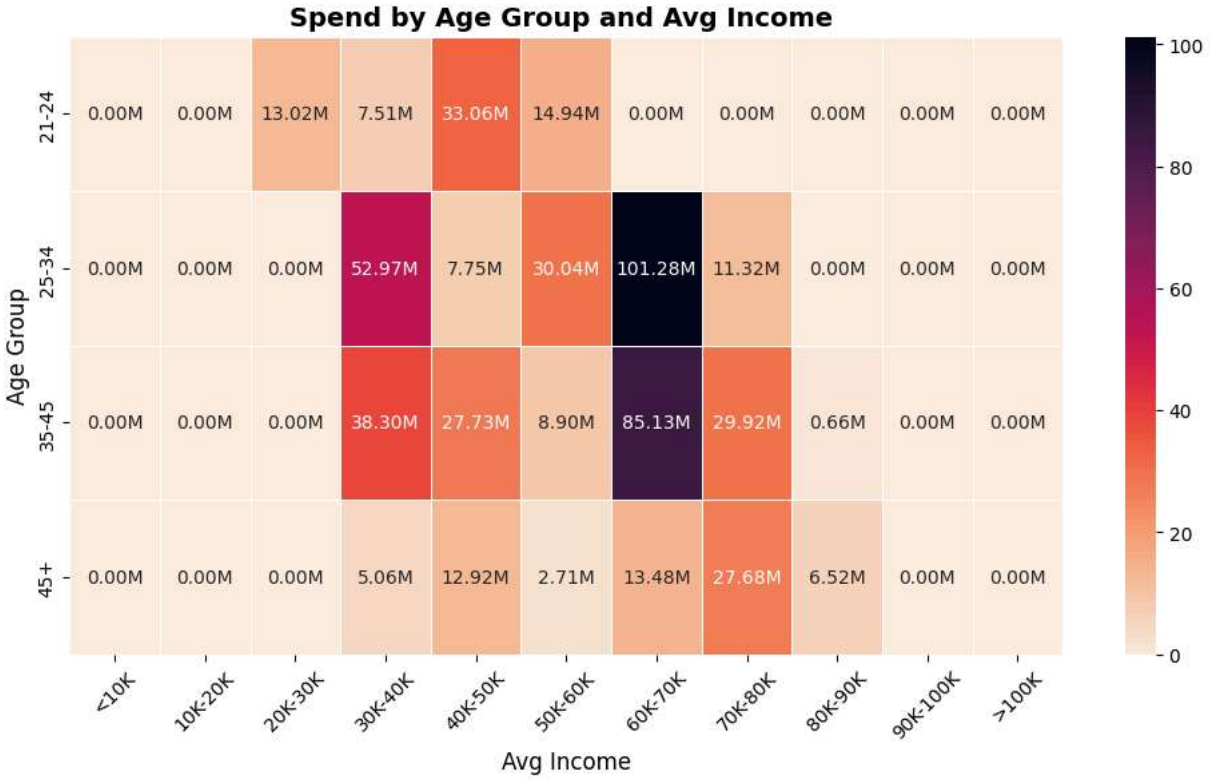
Freelancers and Business Owners exhibit moderate spending, with a peak in Mumbai (21M & 31M, respectively).

Government Employees have the lowest spending across all cities.

Bengaluru and Chennai show balanced spending trends across different occupations.

In [ ]:
```
# Aggregate total spend per category per month
category_month_spend = customer_spends.groupby(["age_group", "avg_income_group"])["

category_month_spend["spend"] = category_month_spend["spend"] / 1_000_000

# Pivot data for heatmap format
pivot_table = category_month_spend.pivot(index="age_group", columns="avg_income_gro

# Plot heatmap
plt.figure(figsize=(12, 6))
ax = sns.heatmap(pivot_table, cmap="rocket_r", annot=True, fmt=".2f", linewidths=0.

for text in ax.texts:
    text.set_text(f"{float(text.get_text()):.2f}M")

# Title and labels
```

```
plt.title("Spend by Age Group and Avg Income ", fontsize=14, fontweight='bold')
plt.xlabel("Avg Income",fontsize=12)
plt.ylabel("Age Group",fontsize=12)
plt.xticks(rotation=45)

plt.show()
```



📊 Spend by Age Group and Avg Income

Insight: The 25-34 and 35-45 age groups exhibit the highest spending across income categories, especially for individuals earning 60K-70K.

The highest spending is observed in the 25-34 age group with an income of 60K-70K (101.28M).

The 35-45 age group also shows significant spending in the 60K-70K income range (85.13M).

The 21-24 and 45+ age groups have lower spending levels, except for some moderate spending in the 30K-50K and 70K-80K income brackets.

Individuals earning below 20K or above 100K have very low or zero spending recorded, possibly due to fewer data points or different spending habits.

Note:

We identify customers who consistently spend 5% or more of their income across different categories.

To ensure consistency, we filter those who have maintained this spending pattern at least 3 times over 6 months.

After filtering, we extract their customer ID, spending amount, and category to analyze their spending behavior.

Finally, we aggregate the total spending per customer and category for further insights.

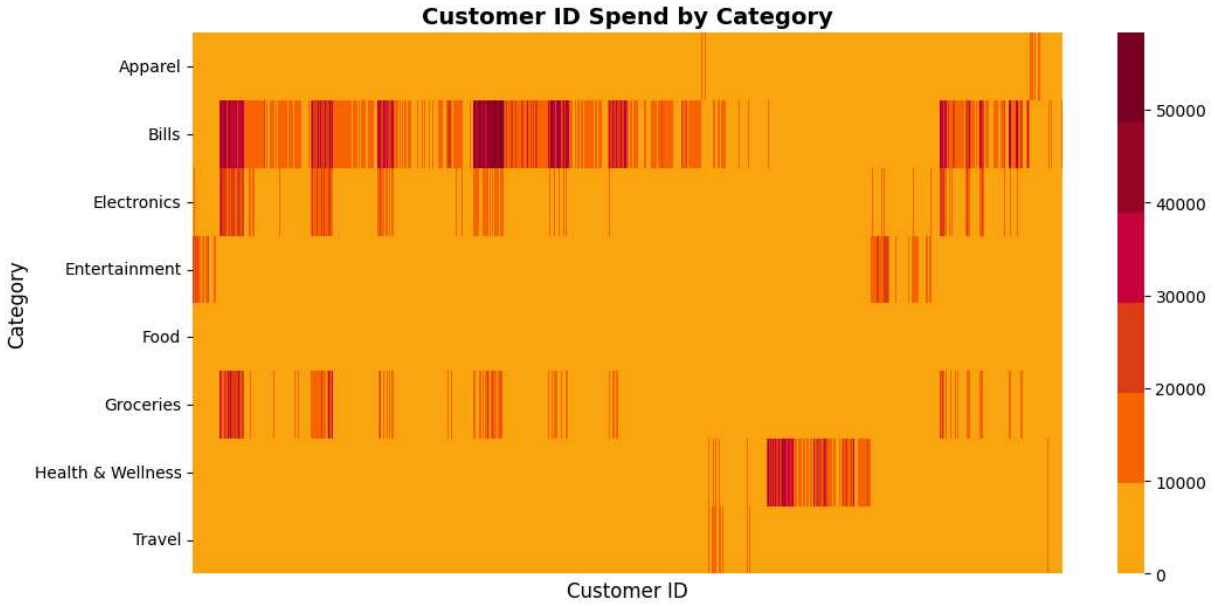
In [32]:
```
# Filter rows where utilisation_% is >= 5
filtered_df = customer_spends[customer_spends["utilisation_%"] >= 5]

# Count occurrences of each customer_id
count_df = filtered_df.groupby("customer_id")["utilisation_%"].count().reset_index(

# Rename the count column
count_df.rename(columns={"utilisation_%": "count"}, inplace=True)

count_df = count_df[count_df["count"] >= 3]

# Merge the filtered data with the count dataframe
top_utilisation_customer = filtered_df.merge(count_df, on="customer_id")
top_utilisation_customer = top_utilisation_customer[['customer_id','spend','categor
top_customer = top_utilisation_customer.groupby(['customer_id','category'])['spend'
top_customer
```

Out[32]:

	customer_id	category	spend
0	ATQCUS0001	Electronics	2867
1	ATQCUS0001	Entertainment	23957
2	ATQCUS0002	Electronics	5512
3	ATQCUS0002	Entertainment	10281
4	ATQCUS0002	Food	2774
...	...	...	...
4044	ATQCUS3963	Travel	3216
4045	ATQCUS3964	Bills	10217
4046	ATQCUS3964	Travel	6335
4047	ATQCUS3966	Bills	7133
4048	ATQCUS3966	Health & Wellness	3226

4049 rows × 3 columns

In [33]:
```
pivot_table = top_customer.pivot(index="category", columns="customer_id", values="s

plt.figure(figsize = (12,6))

sns.heatmap(data = pivot_table, cmap=rocket_palette, xticklabels=False, cbar=True,

# Title and labels
plt.title("Customer ID Spend by Category", fontsize=14, fontweight='bold')
plt.xlabel("Customer ID",fontsize=12)
plt.ylabel("Category",fontsize=12)

plt.show()
```



📊 Top Customer Spending on different Category

Insight: Popular Spending Categories: Categories like Bills, Electronics, and Groceries have higher concentrations of spending, suggesting these are essential or frequently used categories.

Customer Spending Behavior: Some customers show high spending across multiple categories, while others focus their spending on specific categories.

Variation in Spending Patterns: There are customers with consistent spending across months (seen as more continuous bars), while others have sporadic spending behavior in select months.

Potential Business Strategies: Understanding these high-utilization customers can help in targeted promotions, loyalty programs, and personalized offers to encourage more spending in specific categories.

Conclusion and Recommendations for Credit Card Allocation

Conclusion

High Spending Segments

- Salaried IT Employees and Business Owners spend the most, particularly in Mumbai and Delhi NCR.
- Married individuals aged 35-45 and 25-34 exhibit the highest spending across multiple categories.
- Males tend to spend more than females, especially on Bills, Groceries, and Electronics.
- Health & Wellness and Apparel are the top categories where female customers spend the most.

Payment Preferences

- Credit Cards are the most used payment method (40.7%), followed by UPI (26.5%) and Debit Cards (22.5%).
- IT Employees and Business Owners prefer digital payments like Credit Cards and UPI, while Net Banking is the least preferred mode.

Seasonal and Category-wise Spending Trends

- Spending peaks in September and declines in October.
- Bills, Groceries, and Electronics are the top spending categories, indicating essential and lifestyle-related expenditures.
- Entertainment, Travel, and Apparel spending is higher among younger and higher-income groups.

Income Influence on Spending

- Higher-income groups (60K-80K) exhibit a wide range of spending, making them prime candidates for premium credit card offers.
- Lower-income groups (<20K) have limited spending, meaning they may not require high credit limits.

Recommendations for Credit Card Allocation

1. Target High-Income and High-Spending Customers

Prioritize Salaried IT Employees, Business Owners, and Freelancers earning 60K-80K+ per month.

Offer premium credit cards with higher limits and reward programs to high-income individuals aged 25-45.

2. Customized Credit Card Offers

IT Professionals & Business Owners: Offer travel rewards, cashback on digital payments, and higher credit limits.

Married Individuals (35-45 Age Group): Provide family-focused benefits, such as grocery cashback and bill payment discounts.

Freelancers & Entrepreneurs: Introduce flexible repayment plans and business-oriented benefits.

. Focus on Digital Payment Users

Customers already using Credit Cards and UPI frequently should be offered credit limit enhancements and personalized discounts.

Introduce special offers for UPI-linked credit card transactions to encourage more spending.

4. Seasonal & Category-Based Promotions

Provide higher credit limits and cashback offers during peak spending months (August-September).

Offer targeted discounts in Bills, Electronics, and Grocery categories to align with major spending trends.

5. Low-Risk Credit Card Allocation

Avoid offering high-limit credit cards to low-income individuals (<20K) or those who exhibit inconsistent spending patterns.

In [ ]: