Department of Computer Science and Engineering (Data Science)

Subject: Machine Learning – I (DJ19DSC402)

AY: 2022-23

Name: Yash Thakar **SAP ID.: 60009210205**

Colab Link: https://colab.research.google.com/drive/1XPme7jyd-Iy_GtXfdAjB1hYkZQKiwYPQ?usp=sharing

Importing Libraries and Data:

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

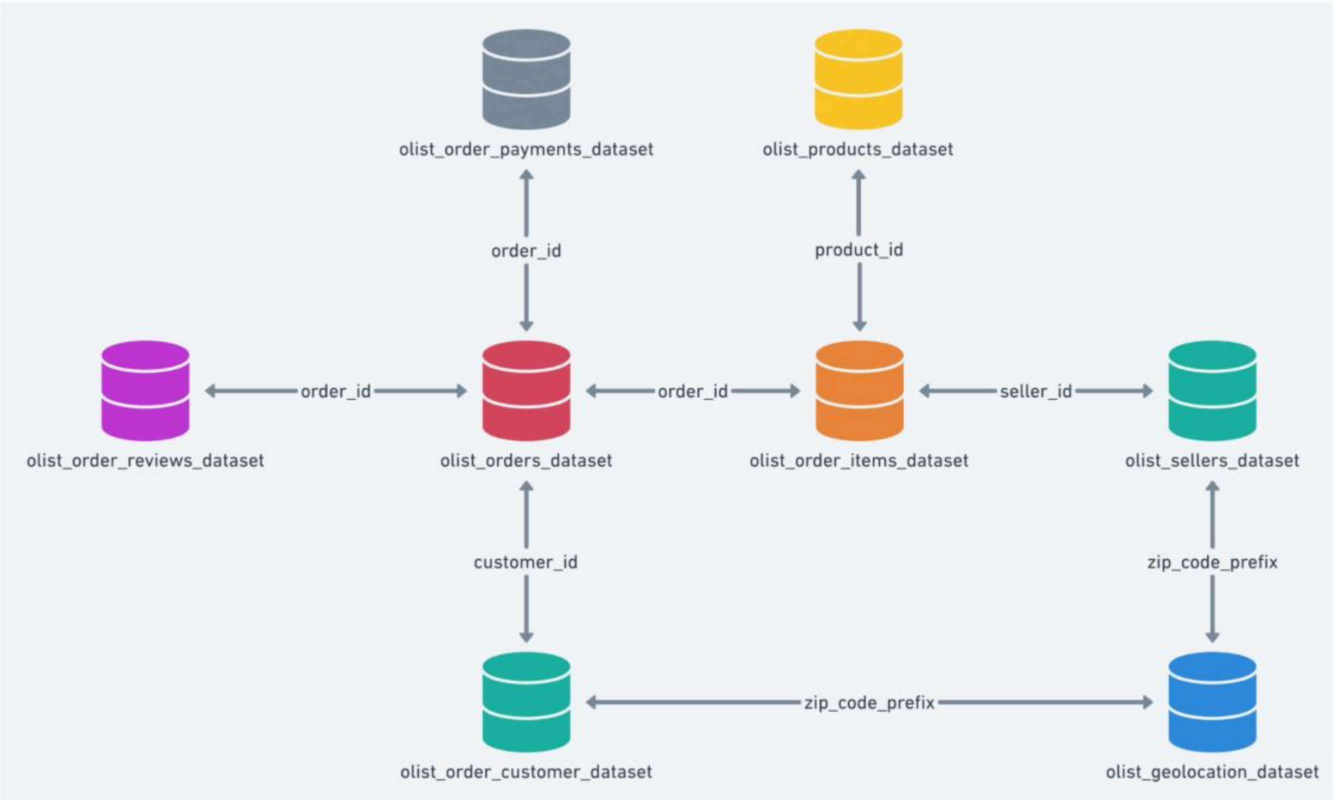
```
In [2]: path = './data/'
        olist_customer = pd.read_csv(path + 'olist_customers_dataset.csv')
        olist_geolocation = pd.read_csv(path + 'olist_geolocation_dataset.csv')
        olist_order_items = pd.read_csv(path + 'olist_order_items_dataset.csv')
        olist_order_payments = pd.read_csv(path + 'olist_order_payments_dataset.csv')
        olist_order_reviews = pd.read_csv(path + 'olist_order_reviews_dataset.csv')
        olist_orders = pd.read_csv(path + 'olist_orders_dataset.csv')
        olist_products = pd.read_csv(path + 'olist_products_dataset.csv')
        olist_sellers = pd.read_csv(path + 'olist_sellers_dataset.csv')
        olist_translation = pd.read_csv(path + 'product_category_name_translation.csv')
```

Viewing attributes of all the datasets:

```
In [3]: dfs = [olist_customer, olist_geolocation, olist_order_items, olist_order_payments, olist_order_reviews, olist_orders, olist_prod
        dfs_names = ['olist_customer','olist_geolocation', 'olist_order_items', 'olist_order_payments', 'olist_order_reviews', 'olist_or
        j = 0
        for i in dfs:
            print (dfs_names[j])
            j = j+1
            print(i.shape)
            print(i.columns)
            print()
```

```
olist_customer
(99441, 5)
Index(['customer_id', 'customer_unique_id', 'customer_zip_code_prefix',
       'customer_city', 'customer_state'],
      dtype='object')

olist_geolocation
(1000163, 5)
Index(['geolocation_zip_code_prefix', 'geolocation_lat', 'geolocation_lng',
       'geolocation_city', 'geolocation_state'],
      dtype='object')

olist_order_items
(112650, 7)
Index(['order_id', 'order_item_id', 'product_id', 'seller_id',
       'shipping_limit_date', 'price', 'freight_value'],
      dtype='object')

olist_order_payments
(103886, 5)
Index(['order_id', 'payment_sequential', 'payment_type',
       'payment_installments', 'payment_value'],
      dtype='object')

olist_order_reviews
(99224, 7)
Index(['review_id', 'order_id', 'review_score', 'review_comment_title',
       'review_comment_message', 'review_creation_date',
       'review_answer_timestamp'],
      dtype='object')

olist_orders
(99441, 8)
Index(['order_id', 'customer_id', 'order_status', 'order_purchase_timestamp',
       'order_approved_at', 'order_delivered_carrier_date',
       'order_delivered_customer_date', 'order_estimated_delivery_date'],
      dtype='object')

olist_products
(32951, 9)
Index(['product_id', 'product_category_name', 'product_name_lenght',
       'product_description_lenght', 'product_photos_qty', 'product_weight_g',
       'product_length_cm', 'product_height_cm', 'product_width_cm'],
      dtype='object')

olist_sellers
(3095, 4)
Index(['seller_id', 'seller_zip_code_prefix', 'seller_city', 'seller_state'], dtype='object')

olist_translation
(71, 2)
Index(['product_category_name', 'product_category_name_english'], dtype='object')
```

Understanding the DataScehma:



Number of unique customer ordering on olist:

```
In [4]: olist_customer["customer_unique_id"].nunique()
Out[4]: 96096
```

Merging datasets based on the above DataSchema:

```
In [4]: olist_customer["customer_unique_id"].nunique()
Out[4]: 96096

In [5]: df= pd.merge(olist_customer, olist_orders, on="customer_id", how='inner')
        df= df.merge(olist_order_reviews, on="order_id", how='inner')
        df= df.merge(olist_order_items, on="order_id", how='inner')
        df= df.merge(olist_products, on="product_id", how='inner')
        df= df.merge(olist_order_payments, on="order_id", how='inner')
        df= df.merge(olist_sellers, on='seller_id', how='inner')
        df= df.merge(olist_translation, on='product_category_name', how='inner')
        df.shape
Out[5]: (115609, 40)

In [6]: df.head()
Out[6]:
```

	customer_id	customer_unique_id	customer_zip_code_prefix	customer_city	customer_state	
0	06b8999e2fba1a1fbc88172c00ba8bc7	861eff4711a542e4b93843c6dd7febb0	14409	franca	SP	00e7ee1b050b8499577073ael
1	8912fc0c3bbf1e2fbf35819e21706718	9eae34bbd3a474ec5d07949ca7de67c0	68030	santarem	PA	c1d2b34febe9cd269e378117
2	8912fc0c3bbf1e2fbf35819e21706718	9eae34bbd3a474ec5d07949ca7de67c0	68030	santarem	PA	c1d2b34febe9cd269e378117
3	f0ac8e5a239118859b1734e1087cbb1f	3c799d181c34d51f6d44bbbc563024db	92480	nova santa rita	RS	b1a5d5365d330d10485e0203
4	6bc8d08963a135220ed6c6d098831f84	23397e992b09769faf5e66f9e171a241	25931	mage	RJ	2e604b3614664aa66867856dl

5 rows × 40 columns

Re-assigning appropriate datatypes and Renaming misspelled feature names:

In [7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 115609 entries, 0 to 115608
Data columns (total 40 columns):
 #   Column                         Non-Null Count   Dtype  
---  ------                         --------------   -----  
 0   customer_id                    115609 non-null  object 
 1   customer_unique_id             115609 non-null  object 
 2   customer_zip_code_prefix       115609 non-null  int64  
 3   customer_city                  115609 non-null  object 
 4   customer_state                 115609 non-null  object 
 5   order_id                       115609 non-null  object 
 6   order_status                   115609 non-null  object 
 7   order_purchase_timestamp       115609 non-null  object 
 8   order_approved_at              115595 non-null  object 
 9   order_delivered_carrier_date   114414 non-null  object 
 10  order_delivered_customer_date  113209 non-null  object 
 11  order_estimated_delivery_date  115609 non-null  object 
 12  review_id                      115609 non-null  object 
 13  review_score                   115609 non-null  int64  
 14  review_comment_title           13801 non-null   object 
 15  review_comment_message         48906 non-null   object 
 16  review_creation_date           115609 non-null  object 
 17  review_answer_timestamp        115609 non-null  object 
 18  order_item_id                  115609 non-null  int64  
 19  product_id                     115609 non-null  object 
 20  seller_id                      115609 non-null  object 
 21  shipping_limit_date            115609 non-null  object 
 22  price                          115609 non-null  float64
 23  freight_value                  115609 non-null  float64
 24  product_category_name          115609 non-null  object 
 25  product_name_lenght            115609 non-null  float64
 26  product_description_lenght     115609 non-null  float64
 27  product_photos_qty             115609 non-null  float64
 28  product_weight_g               115608 non-null  float64
 29  product_length_cm              115608 non-null  float64
 30  product_height_cm              115608 non-null  float64
 31  product_width_cm               115608 non-null  float64
 32  payment_sequential             115609 non-null  int64  
 33  payment_type                   115609 non-null  object 
 34  payment_installments           115609 non-null  int64  
 35  payment_value                  115609 non-null  float64
 36  seller_zip_code_prefix         115609 non-null  int64  
 37  seller_city                    115609 non-null  object 
 38  seller_state                   115609 non-null  object 
 39  product_category_name_english  115609 non-null  object 
dtypes: float64(10), int64(6), object(24)
memory usage: 36.2+ MB
```

In [8]:
```
for feature in ['order_purchase_timestamp', 'order_approved_at', 'order_delivered_carrier_date',
                'order_delivered_customer_date', 'order_estimated_delivery_date', 'shipping_limit_date',
                'review_creation_date', 'review_answer_timestamp']:
    df[feature] = pd.to_datetime(df[feature], errors = 'raise', utc = False)
```

In [9]:
```
df.rename(columns = {'product_name_lenght':'product_name_length',
                     'product_description_lenght':'product_description_length'}, inplace = True)
```

Missing Value Analysis and Removal:

We see that 100% of the data is present for most of the important features and the missing data is mostly due to customers choosing not to leave a review for the products they purchased and some failed deliveries. Thus, Dropping the features with high percentage of missing values

In [11]:
```
df.shape
```

Out[11]:
```
(115609, 40)
```

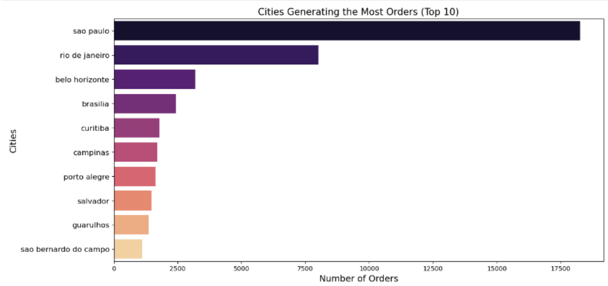
In [12]:
```
Total = df.isnull().sum().sort_values(ascending = False)
Percent = (df.isnull().sum()*100/df.isnull().count()).sort_values(ascending = False)

missing_data = pd.concat([Total, Percent], axis = 1)
print(missing_data)
```

```
                                   0          1
review_comment_title          101808  88.062348
review_comment_message         66703  57.697065
order_delivered_customer_date   2400   2.075963
order_delivered_carrier_date    1195   1.033657
order_approved_at                 14   0.012110
product_height_cm                  1   0.000865
product_weight_g                   1   0.000865
product_length_cm                  1   0.000865
product_width_cm                   1   0.000865
customer_id                        0   0.000000
product_name_length                0   0.000000
product_description_length         0   0.000000
product_photos_qty                 0   0.000000
payment_sequential                 0   0.000000
freight_value                      0   0.000000
payment_type                       0   0.000000
payment_installments               0   0.000000
payment_value                      0   0.000000
seller_zip_code_prefix             0   0.000000
seller_city                        0   0.000000
seller_state                       0   0.000000
product_category_name              0   0.000000
seller_id                          0   0.000000
price                              0   0.000000
order_estimated_delivery_date      0   0.000000
customer_zip_code_prefix           0   0.000000
customer_city                      0   0.000000
customer_state                     0   0.000000
order_id                           0   0.000000
order_status                       0   0.000000
order_purchase_timestamp           0   0.000000
review_id                          0   0.000000
shipping_limit_date                0   0.000000
review_score                       0   0.000000
review_creation_date               0   0.000000
review_answer_timestamp            0   0.000000
order_item_id                      0   0.000000
product_id                         0   0.000000
customer_unique_id                 0   0.000000
product_category_name_english      0   0.000000
```

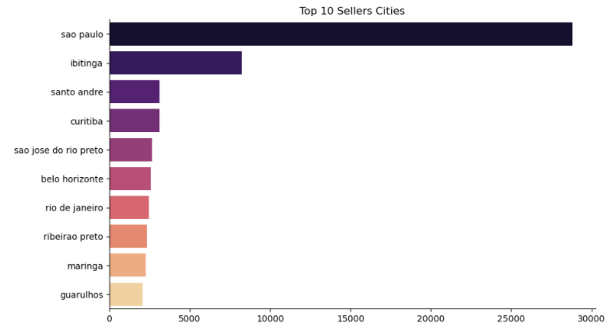
In [13]:
```
df.drop(['review_comment_title', 'review_comment_message'], axis = 1, inplace = True)
```

Knowing and understanding your major markets can be a huge deal for e-commerce businesses and thus we plot the top 10 cities that are generating the highest orders

```
In [14]: top_orders_cities = df.groupby("customer_city")["order_id"].count().reset_index().sort_values("order_id", ascending = False)
plt.figure(figsize = (14, 7))
sns.barplot(x = "order_id", y="customer_city", data = top_orders_cities[:10], palette = 'magma')
plt.xlabel("Number of Orders", fontsize = 14)
plt.ylabel("Cities", fontsize = 14)
plt.yticks(fontsize = 12)
plt.title("Cities Generating the Most Orders (Top 10)", fontsize = 15)
plt.show()
```



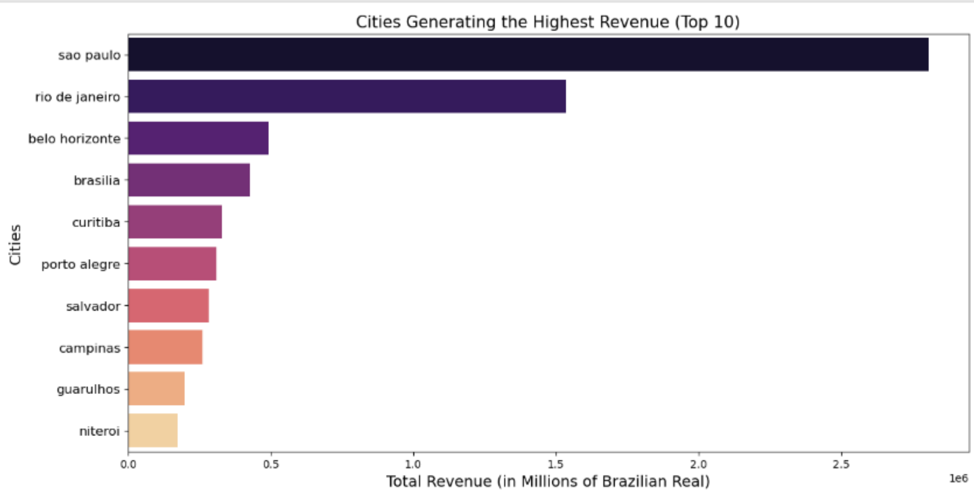

```
In [80]: plt.figure(figsize=[10, 6])
sns.barplot(x = df.seller_city.value_counts().values[:10], y= df.seller_city.value_counts().index[:10], palette= 'magma')
plt.title('Top 10 Sellers Cities')
sns.despine()
```



We can clearly see that most of the orders are coming from Brazil's biggest metropolitan cities - Sao Paulo and Rio de Janerio, while most sellers are located in sao paulo and ibitinga, this can be a major insight to improve logistical operations.

Similarly lets analyse the cities that generate most revenue

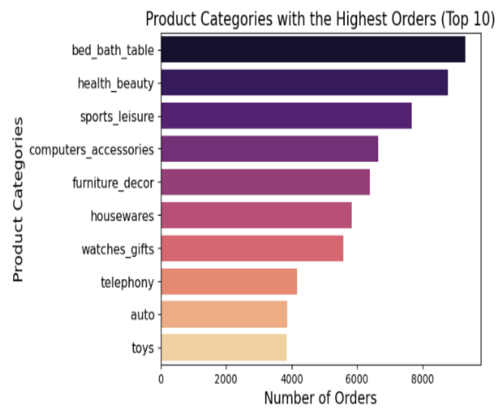
```
In [15]: top_revenue_cities = df.groupby("customer_city")["payment_value"].sum().reset_index().sort_values("payment_value", ascending = F
plt.figure(figsize = (14, 7))
sns.barplot(x = "payment_value", y = "customer_city", data = top_revenue_cities[:10], palette = 'magma')
plt.xlabel("Total Revenue (in Millions of Brazilian Real)", fontsize = 14)
plt.ylabel("Cities", fontsize = 14)
plt.yticks(fontsize = 12)
plt.title("Cities Generating the Highest Revenue (Top 10)", fontsize = 15)
plt.show()
```



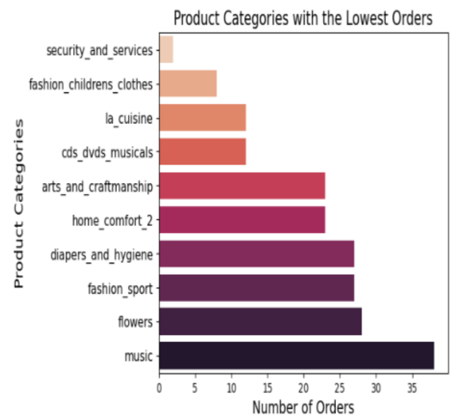
As expected, the cities that generated the most orders, also generated the most revenue**.**

Understanding Product categories can be of great importance so as to understand which products should the business be focusing on:

```
In [18]: prodCat_TopOrders = df.groupby(df['product_category_name_english'])['order_id'].nunique().reset_index().sort_values('order_id', 
         sns.barplot(x = 'order_id', y = 'product_category_name_english', data = prodCat_TopOrders[:10], palette = 'magma')
         plt.xlabel("Number of Orders", fontsize = 14)
         plt.ylabel("Product Categories", fontsize = 14)
         plt.tick_params(axis = 'y', labelsize = 12)
         plt.title("Product Categories with the Highest Orders (Top 10)", fontsize = 15)

Out[18]: Text(0.5, 1.0, 'Product Categories with the Highest Orders (Top 10)')
```




```
In [52]: prodCat_TopOrders = df.groupby(df['product_category_name_english'])['order_id'].nunique().reset_index().sort_values('order_id', 
         sns.barplot(x = 'order_id', y = 'product_category_name_english', data = prodCat_TopOrders[:10], palette = 'rocket_r')
         plt.xlabel("Number of Orders", fontsize = 14)
         plt.ylabel("Product Categories", fontsize = 14)
         plt.tick_params(axis = 'y', labelsize = 12)
         plt.title("Product Categories with the Lowest Orders", fontsize = 15)

Out[52]: Text(0.5, 1.0, 'Product Categories with the Lowest Orders')
```

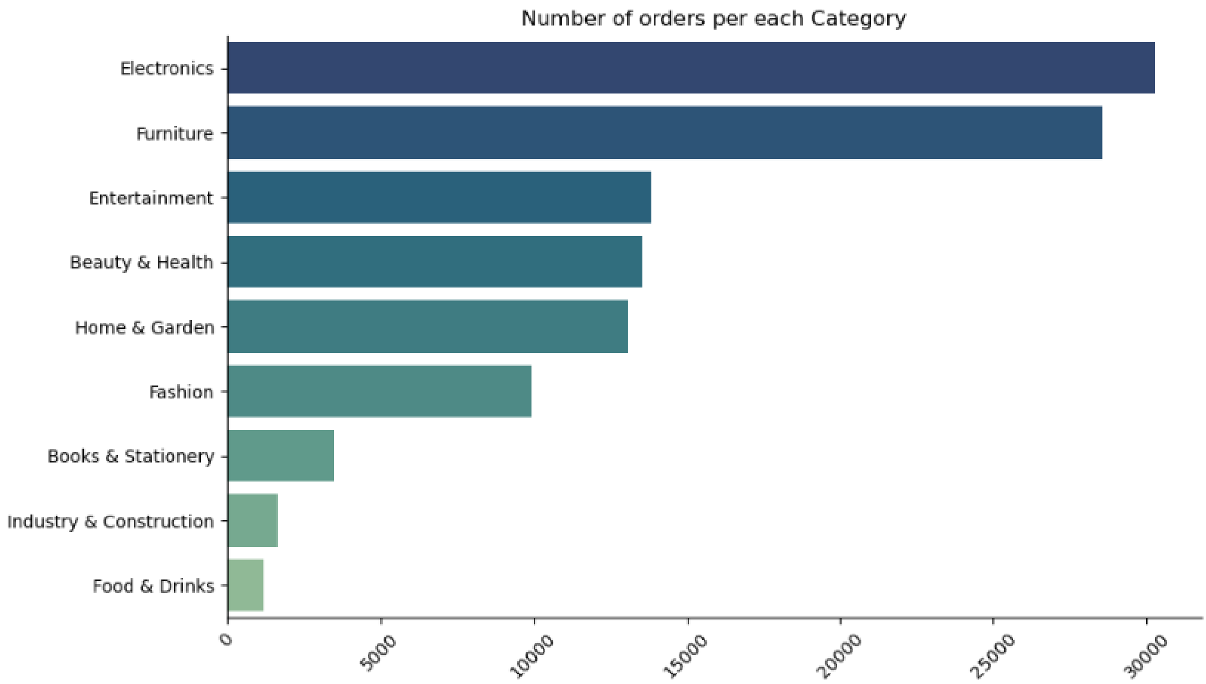


The best Product category are bed_bath_table and the worst is security_and_services.
But since here many categories can be grouped into one major category to simplify our understanding we do so manually.

```
In [20]: def classify_cat(x):

             if x in ['office_furniture', 'furniture_decor', 'furniture_living_room', 'kitchen_dining_laundry_garden_furniture', 'bed_batl
                 return 'Furniture'

             elif x in ['auto', 'computers_accessories', 'musical_instruments', 'consoles_games', 'watches_gifts', 'air_conditioning', 'te
                 return 'Electronics'

             elif x in ['fashio_female_clothing', 'fashion_male_clothing', 'fashion_bags_accessories', 'fashion_shoes', 'fashion_sport',
                 return 'Fashion'

             elif x in ['housewares', 'home_confort', 'home_appliances', 'home_appliances_2', 'flowers', 'costruction_tools_garden', 'gar
                 return 'Home & Garden'

             elif x in ['sports_leisure', 'toys', 'cds_dvds_musicals', 'music', 'dvds_blu_ray', 'cine_photo', 'party_supplies', 'christma
                 return 'Entertainment'

             elif x in ['health_beauty', 'perfumery', 'diapers_and_hygiene']:
                 return 'Beauty & Health'

             elif x in ['food_drink', 'drinks', 'food']:
                 return 'Food & Drinks'

             elif x in ['books_general_interest', 'books_technical', 'books_imported', 'stationery']:
                 return 'Books & Stationery'

             elif x in ['construction_tools_construction', 'construction_tools_safety', 'industry_commerce_and_business', 'agro_industry_a
                 return 'Industry & Construction'

         df['product_category'] = df.product_category_name_english.apply(classify_cat)
```

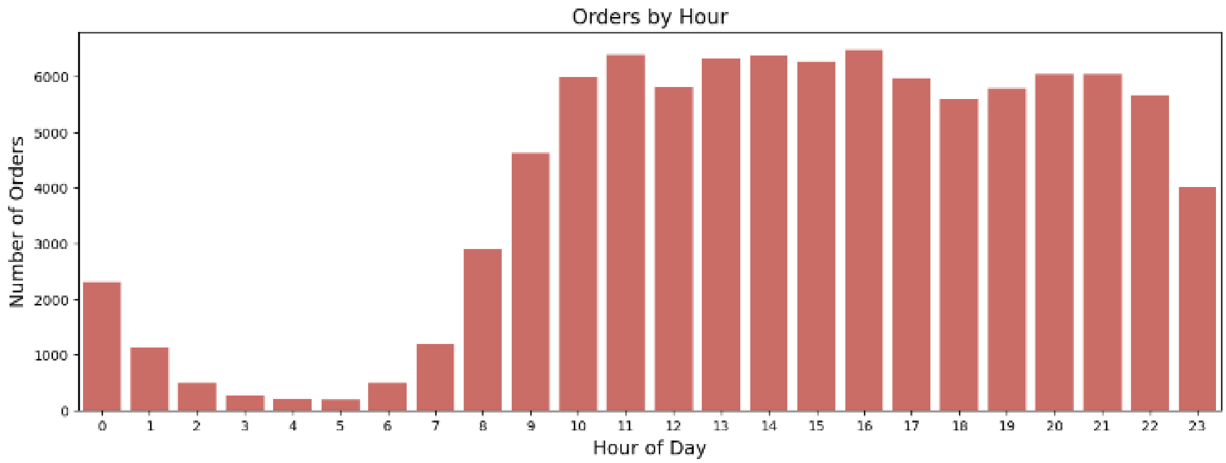
Now we can observe in the below bar chart that the Electronics is the best performing category followed by Furniture while Foods and Drinks have the least amount of orders on the E-commerce platform

In [21]:
```
plt.figure(figsize=[10, 6])
sns.barplot(x = df.product_category.value_counts().values, y = df.product_category.value_counts().index, palette= 'crest_r')
plt.title('Number of orders per each Category')
plt.xticks(rotation = 45)
sns.despine()
```

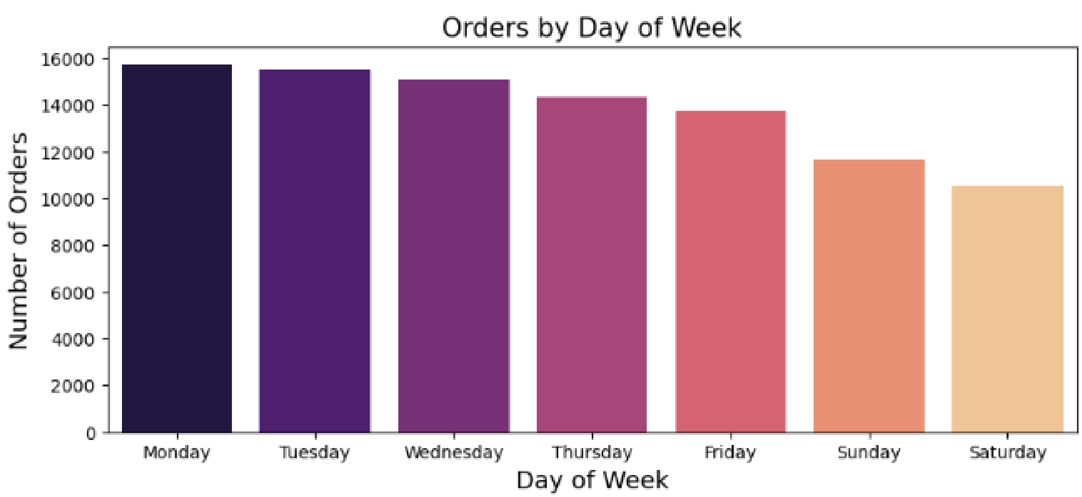


Analysing times at which the customer is likely to order from the platform

```
In [16]: clrp = sns.color_palette("hls", 1)

         orders_byHour = df.groupby(df.order_purchase_timestamp.dt.hour)['order_id'].nunique().reset_index()
         plt.figure(figsize = (15, 5))
         sns.barplot(x = 'order_purchase_timestamp', y = 'order_id', data = orders_byHour, palette = clrp)
         plt.xlabel("Hour of Day", fontsize = 14)
         plt.ylabel("Number of Orders", fontsize = 14)
         plt.title("Orders by Hour", fontsize = 15)
         plt.show()
```




```
In [17]: orders_byDays = df.groupby(df.order_purchase_timestamp.dt.day_name())['order_id'].nunique().reset_index().sort_values('order_id'
         plt.figure(figsize = (10, 4))
         sns.barplot(x = 'order_purchase_timestamp', y = 'order_id', data = orders_byDays, palette = 'magma')
         plt.xlabel("Day of Week", fontsize = 14)
         plt.ylabel("Number of Orders", fontsize = 14)
         plt.title("Orders by Day of Week", fontsize = 15)
         plt.show()
```



We observe that the amount of orders pouring in on Monday to Wednesday is significantly larger than the remaining half of the week.
This can be a useful insight when arranging the logistics for delivery.
And the hourly engagement of orders can be useful if the E-commerce platform focuses on increasing its Food&Drink Product Category by providing one-day or faster deliverys

While we are on the matter of delivery lets take into account reviews and Delivery times, as these are some quantitative factors that affect customer satisfaction:

In [24]:
```
# Distribution of review scores
plt.figure(figsize=(8, 6))
sns.countplot(x='review_score', data=df)
plt.title('Distribution of Review Scores', fontsize=16)
plt.xlabel('Review Score', fontsize=14)
plt.ylabel('Count', fontsize=14)
plt.tick_params(axis='both', which='major', labelsize=12)
sns.despine()
plt.show()
```

We can observe that customers are mostly satisfied as there are large number of 4 and 5 Star reviews

To measure Delivery times,

```
In [28]: deliveryTime = (df["order_delivered_customer_date"] - df["order_purchase_timestamp"])
         deliveryTime_Seconds = deliveryTime.apply(lambda x: x.total_seconds())
         df['deliveryTime_Days'] = round(deliveryTime_Seconds/86400, 2)
```

```
In [29]: df['deliveryTime_Days'].describe()
```

```
Out[29]: count    113209.000000
         mean         12.442129
         std           9.356006
         min           0.530000
         25%           6.740000
         50%          10.190000
         75%          15.500000
         max         208.350000
         Name: deliveryTime_Days, dtype: float64
```

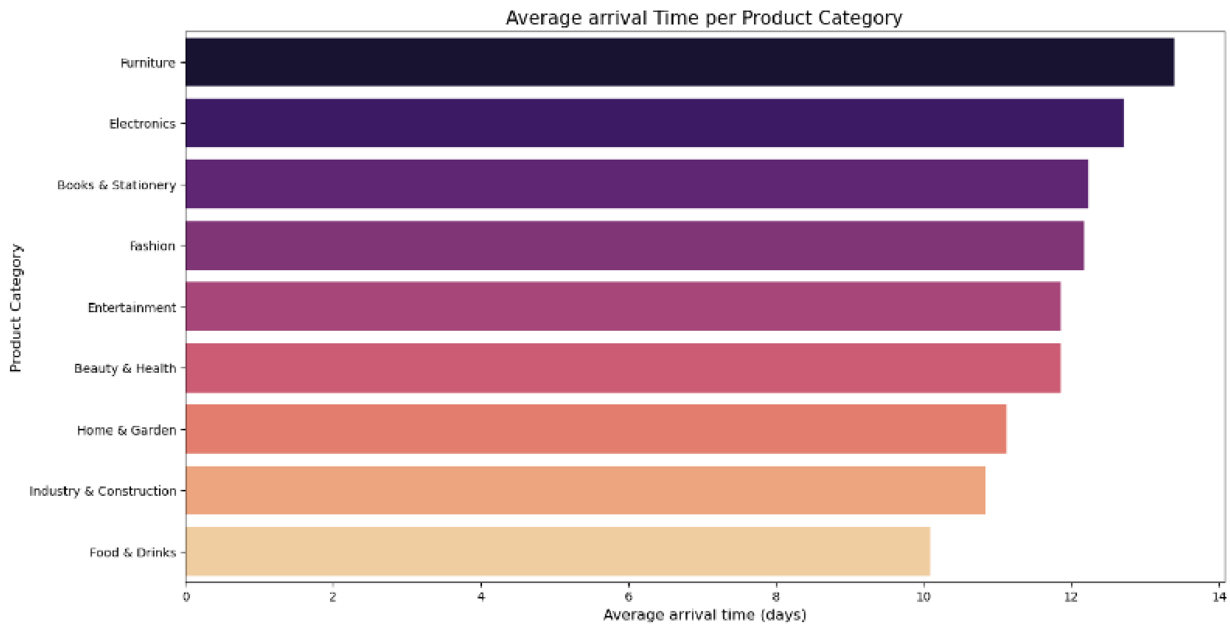
```
In [30]: sns.boxplot(df.deliveryTime_Days, orient = 'h', showfliers = True)
         plt.xlabel("Number of Days", fontsize = 14)
         plt.yticks([])
         plt.title('Distribution of Delivery Time Across All Orders', fontsize = 15)
         plt.show()
```

We can some to the conclusion from the Box and Whisker Plot, that on an average a delivery takes 12 days to reach the customer, which is significantly slower by modern standards.

We can also observe a huge amount of Outliers in the Box Plot pointing towards inconsistent Delivery Times

Plotting Average arrival / Delivery times for each product Category:

```
In [86]: df['arrival_days'] = (df['order_delivered_customer_date'].dt.date - df['order_purchase_timestamp'].dt.date).dt.days

         # Group product category by average arrival time
         ship_per_cat = df.groupby('product_category')[['arrival_days']].mean().sort_values(by='arrival_days', ascending=False)
         ship_per_cat.reset_index(inplace=True)

         plt.figure(figsize=[15, 8])
         sns.barplot(x = ship_per_cat.arrival_days, y=  ship_per_cat.product_category, palette= 'magma')

         plt.title('Average arrival Time per Product Category', fontsize= 15)
         plt.xlabel('Average arrival time (days)',fontsize= 12)
         plt.ylabel('Product Category', fontsize= 12)

Out[86]: Text(0, 0.5, 'Product Category')
```

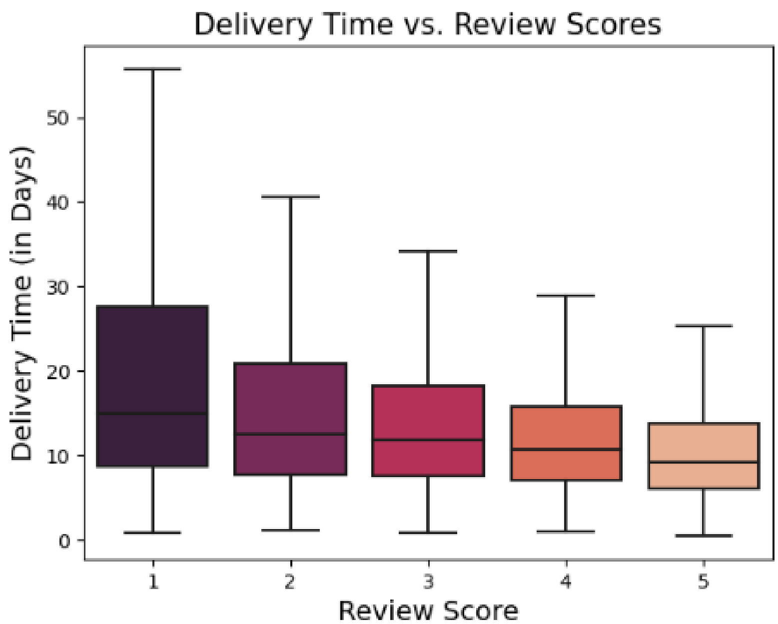


We can observe that the Best Performing Categories of Furniture and Electronics have Delivery times equal to or greater than the average, this could be a huge drawback going further as many of the customers will be experiencing slow delivery times.

Also to increase orders and thus revenue from the Food&Drinks Category we need to significantly shorten delivery durations to deliver foods that can be perceived fresh by the customer,
Also faster delivery times in this category would mean customers will more likely to be order Food & Drinks to satisfy their impulsive cravings, thus boosting revenue.

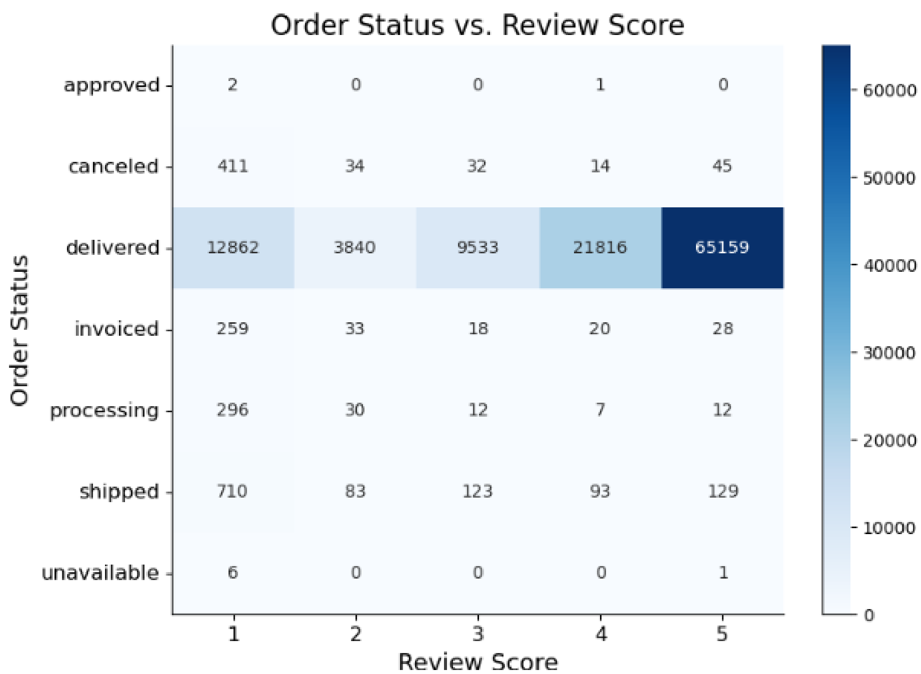
Now lets take a look at how Delivery time affect Review Scores:

```
In [33]: sns.boxplot(x = "review_score", y = "deliveryTime_Days", data = df, showfliers = False, palette = 'rocket')
         plt.xlabel("Review Score", fontsize = 14)
         plt.ylabel("Delivery Time (in Days)", fontsize = 14)
         plt.title("Delivery Time vs. Review Scores", fontsize = 15)
         plt.show()
```



As observed in the above Box Plot, there's a slight correlation between delivery times and review scores. The longer it takes for an order to be delivered, the more likely it is to receive a low review score.

```
In [78]: # Contingency table of order status vs. review score
cont_table = pd.crosstab(df['order_status'], df['review_score'])
plt.figure(figsize=(8, 6))
sns.heatmap(cont_table, cmap='Blues', annot=True, fmt='d')
plt.title('Order Status vs. Review Score', fontsize=16)
plt.xlabel('Review Score', fontsize=14)
plt.ylabel('Order Status', fontsize=14)
plt.tick_params(axis='both', which='major', labelsize=12)
sns.despine()
plt.show()
```



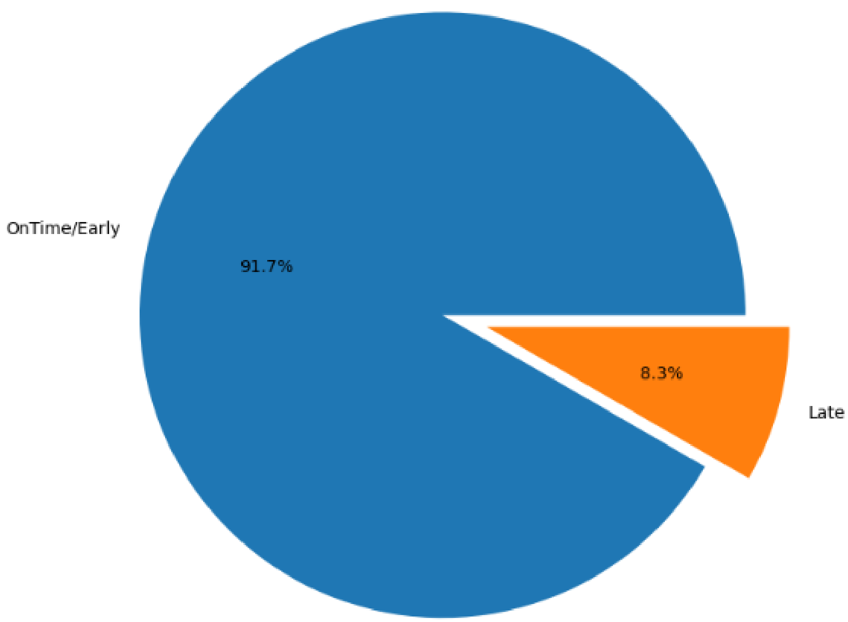
We can observe that the proportion of negative reviews for a cancelled order or a order that shows shipped/invoiced/processing as its order status is significantly high and is a sign of unsatisfied customers.

On classifying the delivers into On-time and Late we can observe that 91.7% of deliveries only make it on time, which is a number that needs improvement to improve customer retention:

```
In [34]: df['seller_to_carrier_status'] = (df['shipping_limit_date'].dt.date - df['order_delivered_carrier_date'].dt.date).dt.days

         # Now calssify the duration into 'OnTime/Early' & 'Late'
         df['seller_to_carrier_status'] = df['seller_to_carrier_status'].apply(lambda x : 'OnTime/Early' if x >=0 else 'Late')
```

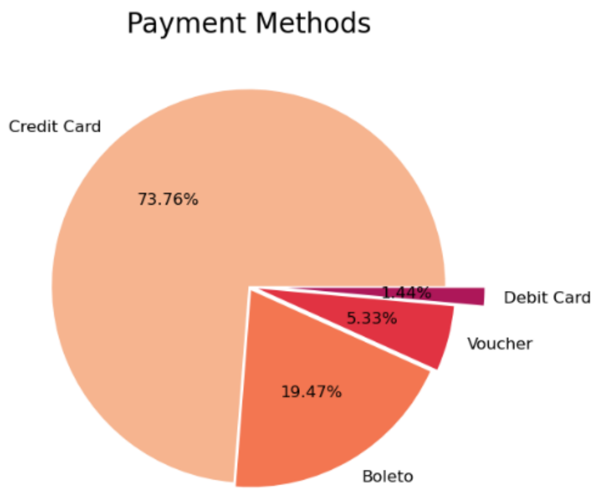
```
In [35]: # First get difference between estimated delivery date and actual delivery date in days
         df['arrival_status'] = (df['order_estimated_delivery_date'].dt.date - df['order_delivered_customer_date'].dt.date).dt.days

         # Now Classify the duration in 'OnTime/Early' & 'Late'
         df['arrival_status'] = df['arrival_status'].apply(lambda x : 'OnTime/Early' if x >=0 else 'Late')
```

```
In [36]: plt.figure(figsize=[30,8])
         Values = df.arrival_status.value_counts().values
         Labels = df.arrival_status.value_counts().index
         plt.pie(Values, explode=(0.05, 0.1), labels= ['OnTime/Early', 'Late'], autopct='%1.1f%%')
         plt.show()
```

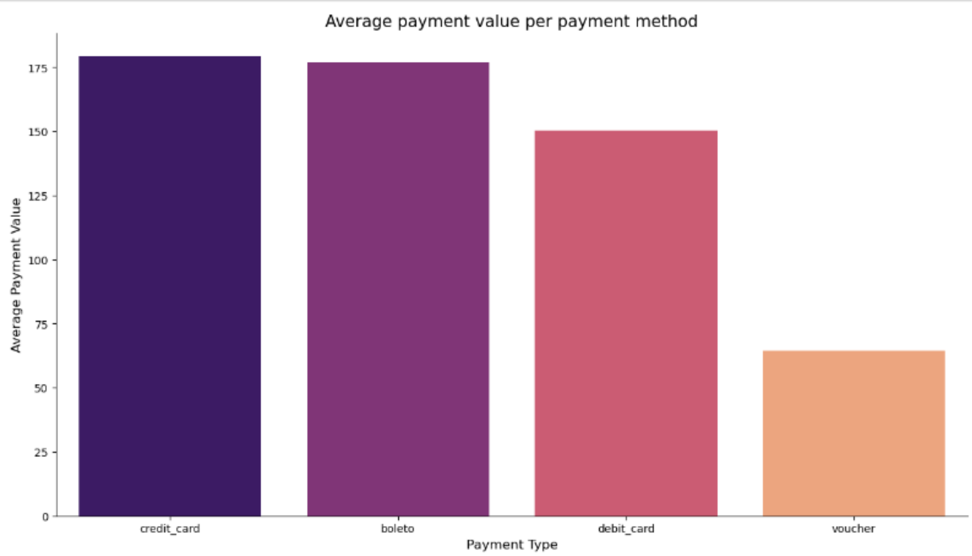


The e-commerce platform has multiple payment methods available, the way in which customers pay can indicate customer preferences and point out customer traits:

In [75]:
```
plt.figure(figsize = (10, 6.5))
payment_type_counts = df.payment_type.value_counts()
plt.pie(x = payment_type_counts.to_list(), labels = ['Credit Card', 'Boleto', 'Voucher', 'Debit Card'],
        autopct = '%1.2f%%', explode = (0, 0.025, 0.05, 0.2), colors = sns.color_palette('rocket_r'),
        textprops = {'fontsize': 12})
plt.title("Payment Methods", fontsize = 20)
plt.show()
```

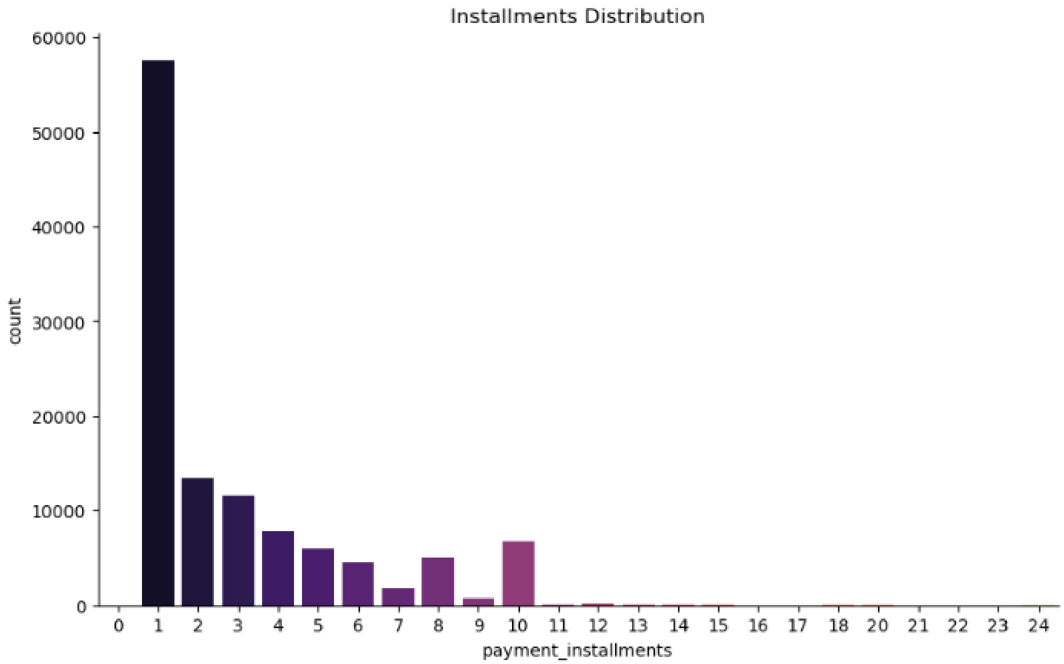


In [87]:
```
# Group each payment type by average payment value
payment_methods = df.groupby('payment_type')[['payment_value']].mean().sort_values(by='payment_value', ascending=False)
payment_methods.reset_index(inplace=True)

# plot Average payments per payment method
plt.figure(figsize=[15, 8])
sns.barplot(x = payment_methods.payment_type, y= payment_methods.payment_value, palette= 'magma')
plt.title('Average payment value per payment method', fontsize= 15)
plt.xlabel('Payment Type', fontsize= 12)
plt.ylabel('Average Payment Value', fontsize= 12)
sns.despine()
```



We can observe that the even though a significantly more amount of customers pay via credit_card over Boleto, the average payment value per payment method is approximately the same for both but credit_card fees that are payable to the service providers are likely to eat away a chunk of the profits.

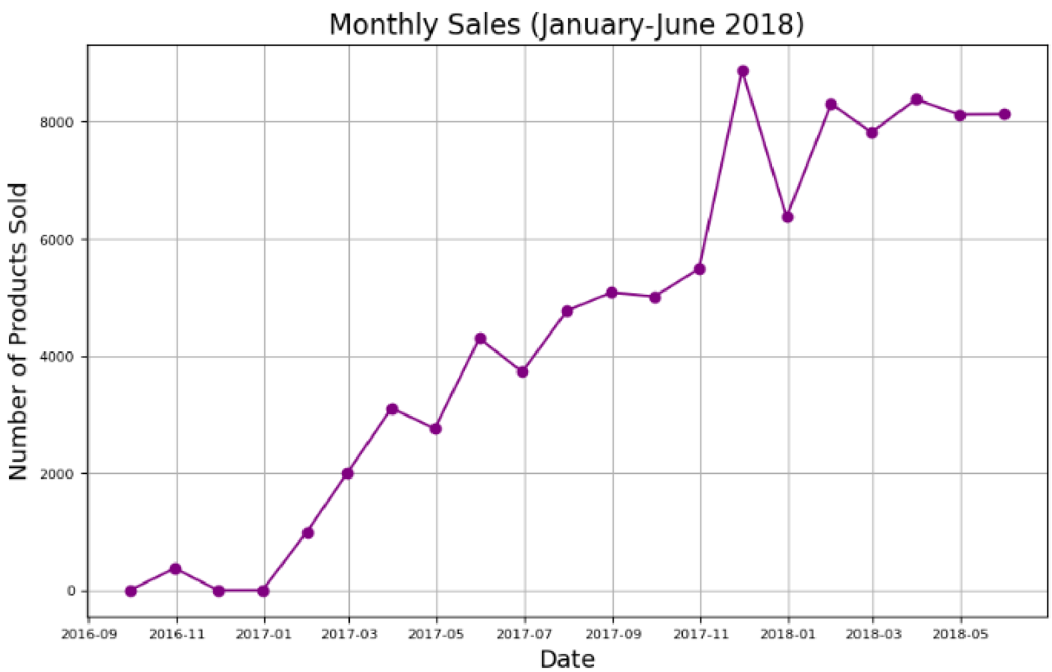
In [79]:
```
plt.figure(figsize=[10, 6])
sns.countplot(x = df.payment_installments, palette= 'magma')
plt.title('Installments Distribution')
sns.despine()
```



We can observe here that a significant number of customer pay in installments ranging from 1 – 12

Analysing Revenue and Sales:

```
In [40]: monthly_sales = df.resample('M', on='order_purchase_timestamp')['order_item_id'].count()

         monthly_sales = monthly_sales['2016-01':'2018-05']


         plt.figure(figsize=(10,6))
         plt.plot(monthly_sales.index, monthly_sales.values, '-o', color='purple')

         plt.title('Sales', fontsize=16)
         plt.xlabel('Date', fontsize=14)
         plt.ylabel('Number of Products Sold', fontsize=14)

         plt.grid(True)
```

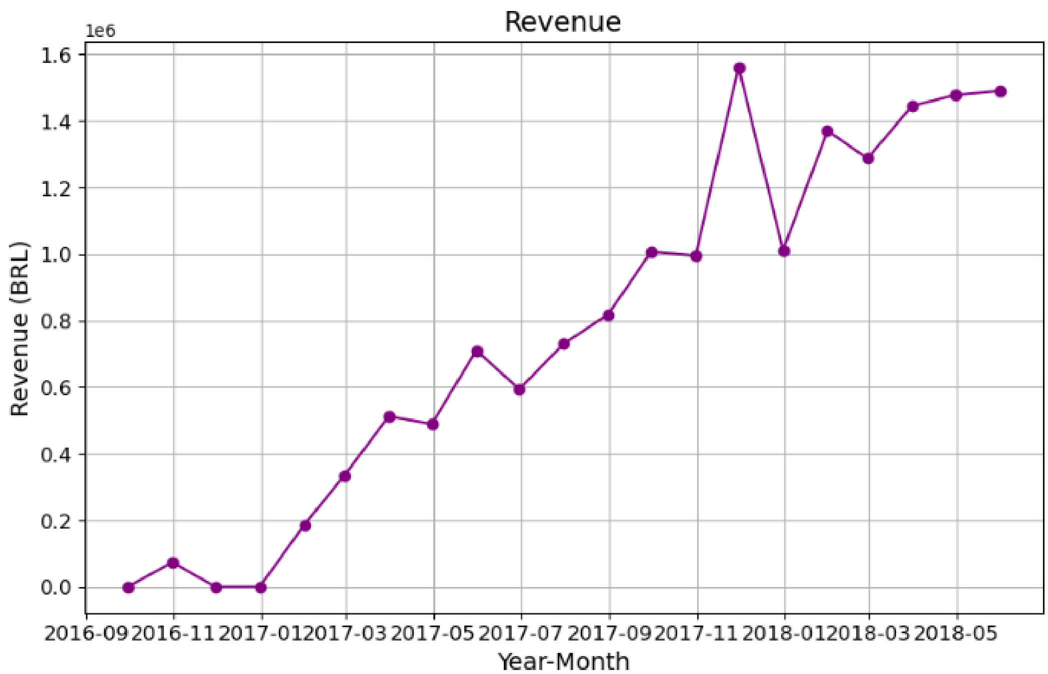


There was a huge growth in the sales from 2016 to 2017,hitting its peak in the end of 2017 after which the sales fell down and became inconsistent.

With a major hit in sales observed in the start of a 2018 right after the company made record sales.

A similar trend can be observed in the Revenue:

```
In [50]: monthly_revenue = df['payment_value'].resample('M').sum()

         monthly_revenue = monthly_revenue['2016-01':'2018-05']


         plt.figure(figsize=(10,6))
         plt.plot(monthly_revenue.index, monthly_revenue.values, '-o', color='purple')

         plt.title('Revenue', fontsize=16)
         plt.xlabel('Year-Month', fontsize=14)
         plt.ylabel('Revenue (BRL)', fontsize=14)

         plt.grid(True)
         plt.show()
```



Although Revenue is constantly growing after the dip in the beginning of 2018/04 to 2018/06 in spite of a sales being a little low throughout this period, this might indicate organizational efficiency and improved margins in the business.