In [52]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches

%matplotlib inline
```

In [4]:
```
df = pd.read_csv('C:/Users/Ezra Muir/Documents/Training-Work/Python/Nov_Learn/SQL T
df
```

Out[4]:

	id	tran_dt	tran_amt	cust_id	acc_id	loan_id
0	2	2021-12-31	100001.0	1	1	NaN
1	3	2022-02-23	6000.0	1	1	NaN
2	4	2015-06-19	700.0	1	1	NaN
3	5	2022-08-31	90.0	2	2	2.0
4	6	2020-10-30	300.0	3	3	2.0
5	7	2015-06-25	200.0	3	3	2.0

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6 entries, 0 to 5
Data columns (total 6 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   id        6 non-null      int64
 1   tran_dt   6 non-null      object
 2   tran_amt  6 non-null      float64
 3   cust_id   6 non-null      int64
 4   acc_id    6 non-null      int64
 5   loan_id   3 non-null      float64
dtypes: float64(2), int64(3), object(1)
memory usage: 416.0+ bytes
```

In [6]:
```
# tran_dt to category type
df['tran_dt'] = df['tran_dt'].astype('category')
```

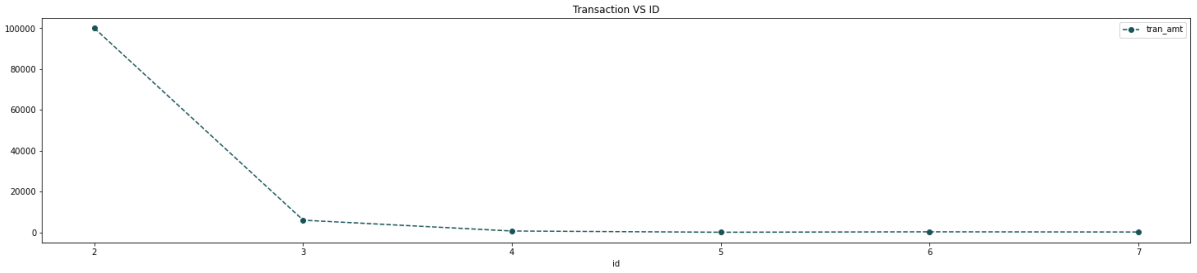
In [7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6 entries, 0 to 5
Data columns (total 6 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   id        6 non-null      int64
 1   tran_dt   6 non-null      category
 2   tran_amt  6 non-null      float64
 3   cust_id   6 non-null      int64
 4   acc_id    6 non-null      int64
 5   loan_id   3 non-null      float64
dtypes: category(1), float64(2), int64(3)
memory usage: 594.0 bytes
```

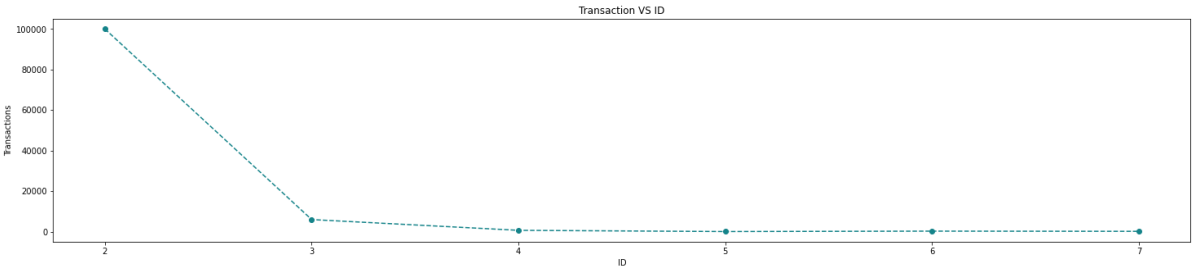
Line Plots

In [11]:
```
# Using plot
df.plot(x='id', y='tran_amt', figsize=(25, 5), title='Transaction VS ID', linestyle

plt.show()
```



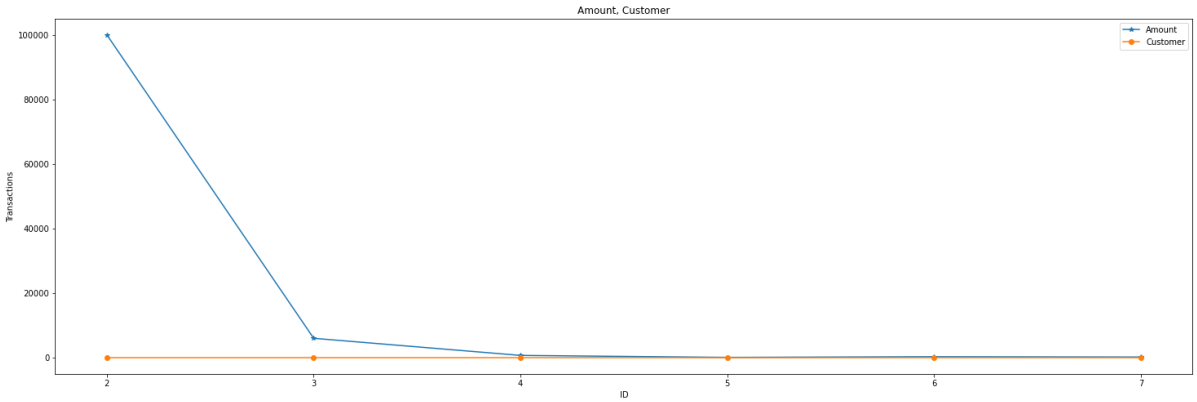
In [12]:
```
# Using plt and .
fig = plt.figure(figsize=(25, 5))
x = df['id']; y = df['tran_amt']
plt.plot(x,y, linestyle='--', marker='o', color='#15848a')
plt.xlabel('ID')
plt.ylabel('Transactions')
plt.title('Transaction VS ID')
plt.show()
```



Overlaying Line Plots

In [15]:
```
plt.figure(figsize=(25,8))
x = df['id']
```

```
y1 = df['tran_amt']
# plotting the line 1 points
plt.plot(x, y1, label="Amount", marker="*")
# line 2 points
y2 = df['cust_id']
# plotting the line 2 points
plt.plot(x, y2, label="Customer", marker="o")
plt.xlabel('ID')
# Set the y axis label of the current axis.
plt.ylabel('Transactions')
# Set a title of the current axes.
plt.title('Amount, Customer')
# show a legend on the plot
plt.legend()
# Display a figure.
plt.show()
```

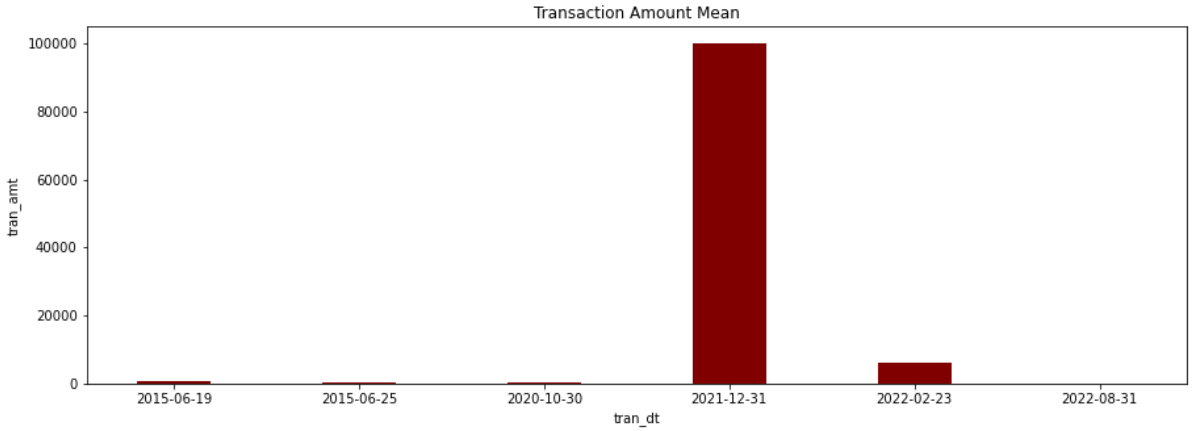


Bar Plots

In [17]:

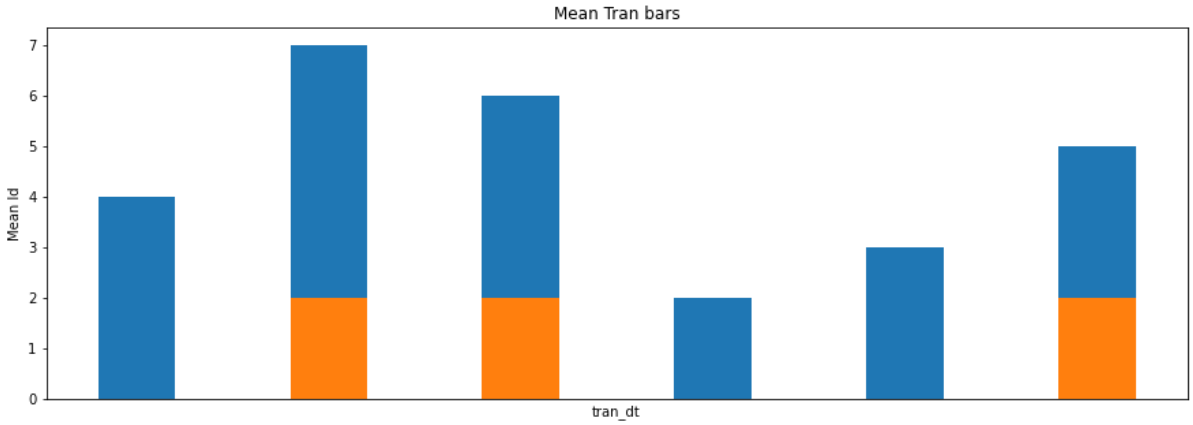
```
category = df['tran_dt'].unique()
meanTran = df[['tran_dt', 'tran_amt']].groupby('tran_dt').mean().reset_index()
fig = plt.figure(figsize=(15, 5))

# bar plot
plt.bar(x='tran_dt', height='tran_amt', data=meanTran, color='maroon', width=0.4)
plt.xlabel('tran_dt')
plt.ylabel('tran_amt')
plt.title('Transaction Amount Mean')
plt.show()
```



Stacked Bar plot over another

In [23]:
```
fig = plt.figure(figsize=(15, 5))
MeanTran = df[['tran_dt', 'id', 'loan_id']].groupby('tran_dt'). mean().reset_index(
plt.bar(x='tran_dt', height='id', data=MeanTran, width=0.4)
plt.bar(x='tran_dt', height='loan_id', data=MeanTran, width=0.4)
plt.xlabel("tran_dt")
plt.xlabel("tran_dt")
plt.ylabel("Mean Id")
plt.title("Mean Tran bars")
plt.xticks('tran_dt', rotation='65')
plt.show()
```



Stacked bar plot side by side

In [46]:
```
# fig = plt.figure(figsize=(20,5))
# N = 3
# ind = np.arange(N); width = 0.4
# MeanSepal = df[['id', 'cust_id', 'acc_id']].groupby('id'). mean().reset_index()
# category = tuple(MeanSepal.id)
# plt.bar(x = ind, height = 'cust_id', data = MeanSepal, width = width, label = 'Me
# plt.bar(x = ind+width , height = 'acc_id', data = MeanSepal, width = width, label
# plt.xlabel("id")
# plt.ylabel("MeanLength-Cm")
```

```
# plt.title("Sepal-Means bars")
# #plt.xticks(ind + width / 2, ('Iris-setosa', 'Iris-versicolor', 'Iris-virginica')
# plt.xticks(ind + width / 2, category)
# plt.legend(loc='best')
# plt.show()
```

Scatter Plots

In [47]:

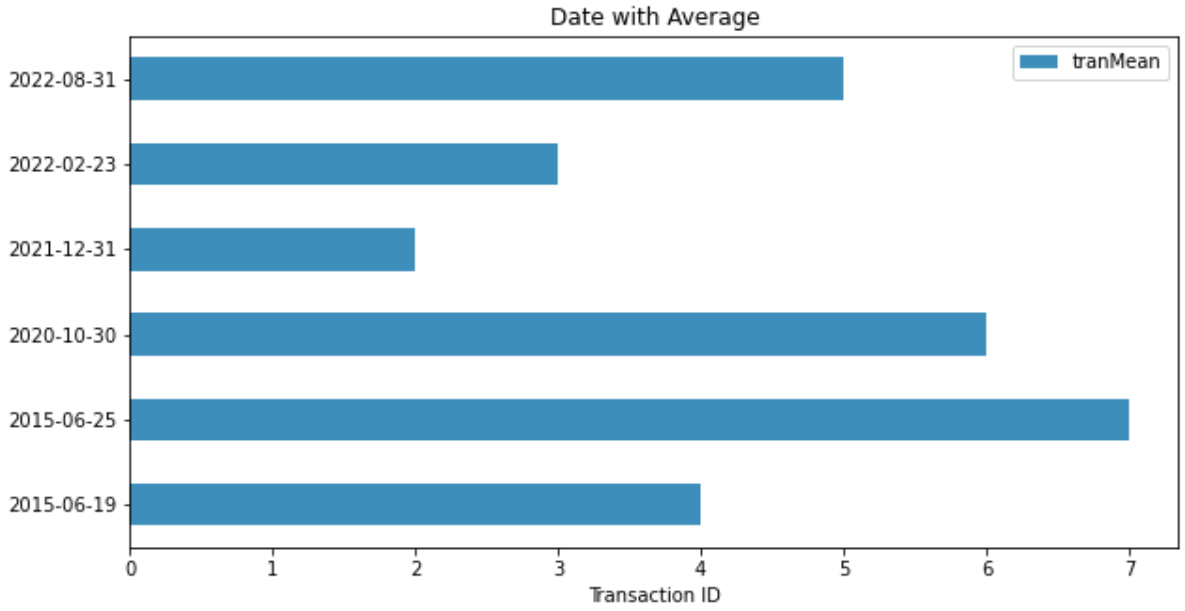
```
# x1 =df['id']
# x2 =df['cust_id']
# y1 =df['id']
# y2 = df['acc_id']
# #colurs for each category of the species
# colors = {'Id':'red', 'Customer ID':'blue', 'Account ID':'green'}
# # colour patches for the legend prepration
# red_patch = mpatches.Patch(color='red', label='Transaction ID')
# blue_patch = mpatches.Patch(color='blue', label='Customer ID')
# green_patch = mpatches.Patch(color='green', label='Account ID')
# f = plt.figure(figsize=(10,5))
# #Assign subplots
# ax1 = f.add_subplot(121, title = "Customer Scatter")
# ax2 = f.add_subplot(122, title = "Account Scatter")
# # Add a plot for each ax
# #Plot1
# ax1.scatter(x1, y1, c=df['id'].apply(lambda x: colors[x]))
# ax1.set_xlabel('ID Length'); ax1.set_ylabel('Customer Width')
# ax1.legend(handles=[red_patch, blue_patch,green_patch], loc = 'upper left')
# #Plot2
# ax2.scatter(x2, y2, c=df['id'].apply(lambda x: colors[x]))
# ax2.set_xlabel('ID Length');ax2.set_ylabel('Account Width')
# ax2.legend(handles=[red_patch, blue_patch,green_patch], loc = 'upper left')
# #wrap up to show
# plt.tight_layout()
# plt.show()

# x1 =df['id']
# x2 =df['cust_id']
# y1 =df['id']
# y2 = df['acc_id']
# #colurs for each category of the species
# colors = {'Id':'red', 'Customer ID':'blue', 'Account ID':'green'}
# # colour patches for the legend prepration
# red_patch = mpatches.Patch(color='red', label='Transaction ID')
# blue_patch = mpatches.Patch(color='blue', label='Customer ID')
# green_patch = mpatches.Patch(color='green', label='Account ID')
# f = plt.figure(figsize=(10,5))
# #Assign subplots
# ax1 = f.add_subplot(121, title = "Customer Scatter")
# ax2 = f.add_subplot(122, title = "Account Scatter")
# # Add a plot for each ax
# #Plot1
# ax1.scatter(x1, y1, c=df['id'].apply(lambda x: colors[x]))
# ax1.set_xlabel('ID Length'); ax1.set_ylabel('Customer Width')
# ax1.legend(handles=[red_patch, blue_patch,green_patch], loc = 'upper left')
```

```
# #Plot2
# ax2.scatter(x2, y2, c=df['id'].apply(lambda x: colors[x]))
# ax2.set_xlabel('ID Length');ax2.set_ylabel('Account Width')
# ax2.legend(handles=[red_patch, blue_patch,green_patch], loc = 'upper left')
# #wrap up to show
# plt.tight_layout()
# plt.show()
```

Horizontal Bar Plot

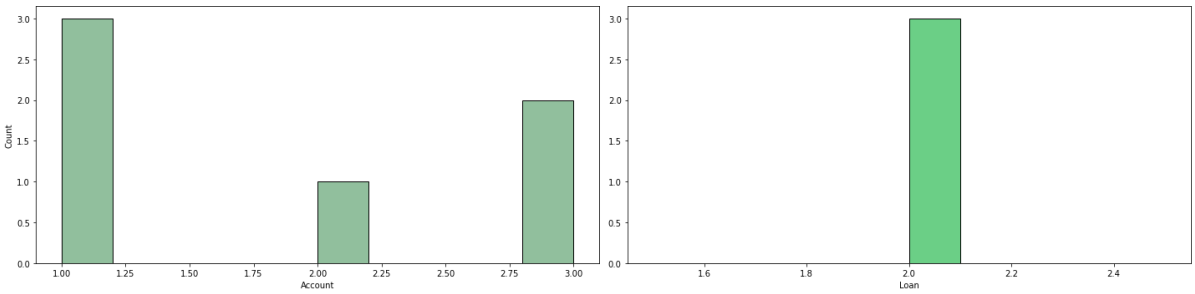
In [51]:
```
tranMean = df[['tran_dt', 'id']].groupby("tran_dt").mean().reset_index()
tranMean.columns = ['tran_dt', 'tranMean']
tranMean.plot(kind='barh', figsize=(10, 5), color='#3d8eba', title="Date with Avera
dates = list(tranMean['tran_dt'])
y_pos = np.arange(len(tranMean))
plt.yticks(y_pos, dates)
plt.xlabel('Transaction ID')
plt.show()
```



Side by Side Histograms

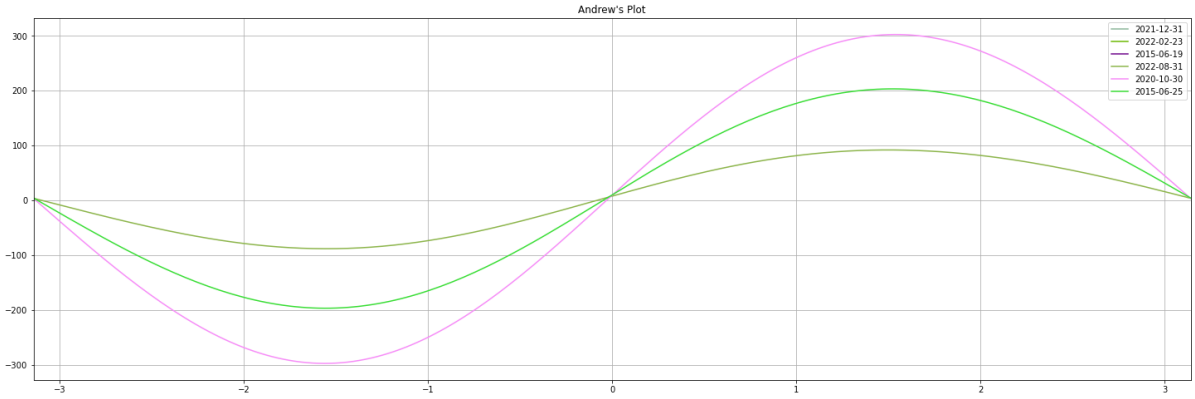
In [61]:
```
fig, (ax1, ax3) = plt.subplots(1,2, figsize=(20,5))

ax1.hist(df['acc_id'], edgecolor = 'black', align = 'mid', color = '#91bf9d')
ax1.set_xlabel('Account')
ax1.set_ylabel('Count')
ax3.hist(df['loan_id'], edgecolor = 'black', align = 'mid', color = '#6bcf86')
ax3.set_xlabel('Loan')
plt.tight_layout()
plt.show()
```



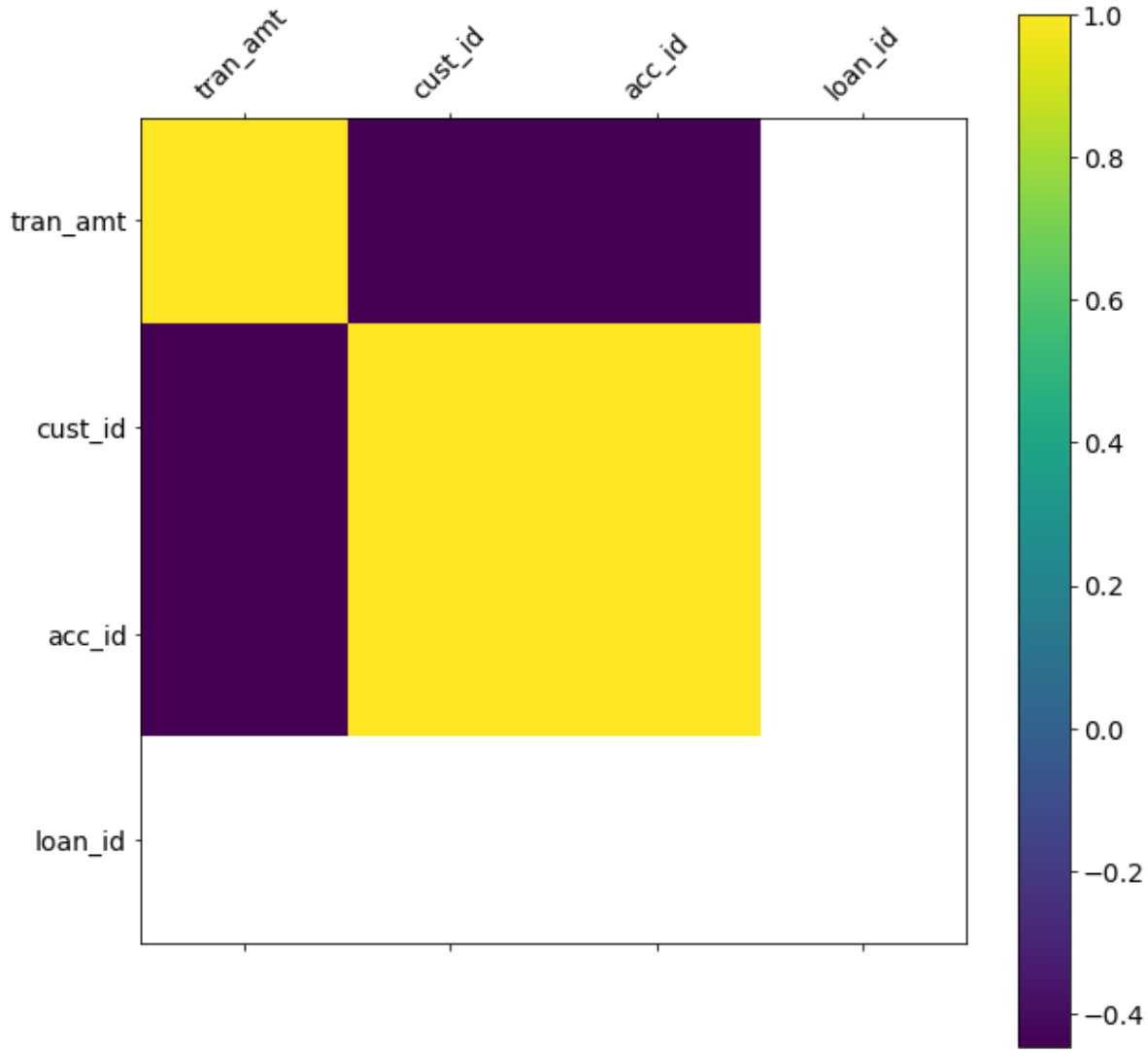
Andrew's Curve

In [62]:
```
# Andrews curves are used for visualizing high-dimensional data by mapping each obs
plt.figure(figsize = (25,8))
pd.plotting.andrews_curves(df, 'tran_dt')
plt.title("Andrew's Plot")
plt.show()
```



Correlation Matrix Plot

In [65]:
```
d = df.drop('id', axis = 1)
fig = plt.figure(figsize=(10,10))
plt.matshow(d.corr(),fignum=fig.number)
plt.xticks(range(d.select_dtypes(['number']).shape[1]), d.select_dtypes(['number'])
plt.yticks(range(d.select_dtypes(['number']).shape[1]), d.select_dtypes(['number'])
cb = plt.colorbar()
cb.ax.tick_params(labelsize=14)
plt.show()
```



Pie Charts

In [75]:
```
# # Pie chart, where the slices will be ordered and plotted counter-clockwise:
# labels = df['tran_dt'].unique()
# sizes1 = df.groupby('tran_dt').mean()['id']
# sizes2 = df.groupby('tran_dt').mean()['cust_id']
# sizes3 = df.groupby('tran_dt').mean()['acc_id']
# sizes4 = df.groupby('tran_dt').mean()['loan_id']
# explode = (0, 0.1, 0)  # only "explode" the 2nd slice ()

# fig, (ax1, ax2, ax3, ax4) = plt.subplots(1,4, figsize = (25,5))
# #plot1
# ax1.pie(sizes1, explode=explode, labels=labels, autopct='%1.1f%%',shadow=True, st
# ax1.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
# ax1.set_title("Mean-SepalLength")
# #Plot2
# ax2.pie(sizes2, explode=explode, labels=labels, autopct='%1.1f%%',shadow=True, st
# ax2.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
# ax2.set_title("Mean-SepalWidth")
# #plot3
```

```
# ax3.pie(sizes3, explode=explode, labels=labels, autopct='%1.1f%%', shadow=True, s
# ax3.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
# ax3.set_title("Mean-PetalLength")

# #plot4
# ax4.pie(sizes4, explode=explode, labels=labels, autopct='%1.1f%%', shadow=True, s
# ax4.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
# ax4.set_title("Mean-PetalWidth")

# plt.show()
```

In [ ]: