For this project, the data scientist ran a k-mean clustering analysis for iShopChangi shoppers who shopped between June 2020 and May 2021.
Took over the segmented/clustered data and find out the characteristics of each segment:

1. Spent on product categories
2. Proportion of Changi Rewards Tier
3. Net spend per order
4. Frequency of purchase
5. Recency of purchase
6. Size of shopping cart
7. Price sensitivity
8. Age of shoppers
9. Proportion of discounted transactions
10. Proportion of subscription opt-in
11. Gender of segment population
12. Gross spend of subscription opt-in vs. opt-out

In [ ]:
```
import pandas as pd
import numpy as np
pd.set_option('display.max_colwidth', None)
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', 1000)
pd.options.mode.chained_assignment = None

import matplotlib.pyplot as plt
import seaborn as sns
from datetime import datetime
import math
from scipy import stats

import re
```

1 Data preparation

1.1 Preparation [...]

1.2 Master Data

In [ ]:
```
def round_down(x):
    return math.floor(x)
```

In [ ]:
```
df_age = pd.read_csv('supplement query 1.csv', usecols=['CARDNO', 'DATEOFBIRTH'],
                     infer_datetime_format=True,
                     dtype={'CARDNO': 'str'})
df_age['DATEOFBIRTH'] = pd.to_datetime(df_age['DATEOFBIRTH'], format='%Y-%m-%d %H:%M:%S')
df_age['CARDNO'] = df_age['CARDNO'].apply(lambda x: '0000' + str(x) if len(str(x)) == 15 else x)

df_age.dropna(inplace=True)
```

In [ ]:
```
REF_YEAR = datetime.strptime('2020-12-31 23:59:59', '%Y-%m-%d %H:%M:%S')

df_age['age_Temp'] = REF_YEAR - df_age['DATEOFBIRTH']
df_age['age'] = (df_age['age_Temp'].dt.days / 365).apply(round_down)

df_age.drop(['age_Temp'], axis=1, inplace=True)
df_age.drop_duplicates(inplace=True)
df_age.reset_index(inplace=True, drop=True)
```

In [ ]:
```

```

2 Exploration

- Segment 0: Big-basket but low spending buyers
- Segment 1: Alcohol lovers (with promos)
- Segment 2: Beauty products lovers
- Segment 3: Alcohol lovers
- Segment 4: Big spenders
- Segment 5: Electronics lover
- Segment 6: Beauty products lovers (with promos)
- Segment 7: Frequent buyers, love promo too
- Segment 8: Random

2.1 Product Categories

- Segment 1, 2, 3, 5, 6 have very high concentration of product category. To search for cross-selling oppurunity within product category.
- Segment 0, 4, 7, 8 to search for cross-selling oppurtunity from check-out market

In [ ]:
```
df_seg_cat = df_master.groupby(['seg', 'PROGROUPED']).agg({'CARDNO': 'nunique'}).unstack().droplevel(
df_seg_cat['sum'] = df_seg_cat.sum(axis = 1)

df_seg_cat['CCD Share'] = df_seg_cat['CCD'] / df_seg_cat['sum']
df_seg_cat['CHILDREN Share'] = df_seg_cat['CHILDREN'] / df_seg_cat['sum']
df_seg_cat['ECP Share'] = df_seg_cat['ECP'] / df_seg_cat['sum']
df_seg_cat['FASHION Share'] = df_seg_cat['FASHION'] / df_seg_cat['sum']
df_seg_cat['LT Share'] = df_seg_cat['LT'] / df_seg_cat['sum']
df_seg_cat['LUXURY Share'] = df_seg_cat['LUXURY'] / df_seg_cat['sum']
df_seg_cat['OTH Share'] = df_seg_cat['OTH'] / df_seg_cat['sum']
df_seg_cat['PC Share'] = df_seg_cat['PC'] / df_seg_cat['sum']
df_seg_cat['PHARM Share'] = df_seg_cat['PHARM'] / df_seg_cat['sum']

df_seg_cat_share = df_seg_cat[['CCD Share', 'CHILDREN Share', 'ECP Share',
                  'FASHION Share', 'LT Share', 'LUXURY Share',
                  'OTH Share', 'PC Share', 'PHARM Share',]]
```

In [ ]:
```
f, ax1 = plt.subplots(figsize=(10, 10))
sns.heatmap(df_seg_cat_share, annot = True, fmt=".0%", cmap="Blues", ax=ax1)

bottom, top= ax1.get_ylim()
ax1.set_ylim(bottom + 0.5, top- 0.5)
```

2.2 CR Tiers

Seg 4 mainly made up of Gold and Platinum members

In [ ]:
```
df_member_tier = df_master[['CARDNO', 'CURRENTTIERCD', 'seg']].drop_duplicates()

df_cr_tier = df_member_tier.groupby(['seg', 'CURRENTTIERCD']).agg({
    'CARDNO': np.count_nonzero
}).unstack().droplevel(0, axis = 1).fillna(0)

df_cr_tier = df_cr_tier[['Platinum', 'Gold', 'Member', 'Staff_Platinum', 'Staff_Gold']]
df_cr_tier
```

2.3 Net Spend (per order)

Seg 4 has the highest spending, by a margin

In [ ]:
```
df_net_spend1 = df_master.groupby('ORDERID').agg({'NETSPEND': np.max, 'seg':np.min})
df_net_spend2 = df_net_spend1.groupby('seg').agg({'NETSPEND': [np.count_nonzero, np.sum, np.mean]}).d

df_net_spend2.sort_values('mean', ascending=False, inplace=True)
df_net_spend2
```

2.4 Frequency

Seg 7 is the only segment where all customers shopped more than once from iSC. Single spending shoppers formed 60%-80% of the remaning segments

In [ ]:
```
df_frequency1 = df_master.groupby(['CARDNO']).agg({'ORDERID': 'nunique', 'seg': np.max})
df_frequency1.columns = ['order_freq', 'seg']

def less_than_10(x):
    if x <= 9:
        return '0' + str(x)
    if x == 10:
        return str(x)
    if x > 10:
        return '>10'

df_frequency1['order_freq'] = df_frequency1['order_freq'].apply(less_than_10)

df_frequency2 = df_frequency1.groupby(['seg', 'order_freq']).agg({'order_freq': np.count_nonzero}
                                                                 ).unstack().fillna(0).droplevel(0, ax
```

In [ ]:
```
df_frequency2['sum'] = df_frequency2.sum(axis=1)

df_frequency2['01_share'] = df_frequency2['01'] / df_frequency2['sum']
df_frequency2['02_share'] = df_frequency2['02'] / df_frequency2['sum']
df_frequency2['03_share'] = df_frequency2['03'] / df_frequency2['sum']
df_frequency2['04_share'] = df_frequency2['04'] / df_frequency2['sum']
df_frequency2['05_share'] = df_frequency2['05'] / df_frequency2['sum']
df_frequency2['06_share'] = df_frequency2['06'] / df_frequency2['sum']
df_frequency2['07_share'] = df_frequency2['07'] / df_frequency2['sum']
df_frequency2['08_share'] = df_frequency2['08'] / df_frequency2['sum']
df_frequency2['09_share'] = df_frequency2['09'] / df_frequency2['sum']
df_frequency2['10_share'] = df_frequency2['10'] / df_frequency2['sum']
df_frequency2['>10_share'] = df_frequency2['>10'] / df_frequency2['sum']


df_frequency = df_frequency2[['01_share', '02_share', '03_share','04_share','05_share', '06_share',
                              '07_share', '08_share', '09_share','10_share','>10_share']].T
df_frequency.columns = ['seg ' + str(x) for x in range(0,9)]
```

In [ ]:
```
fig,ax = plt.subplots(5, 2, figsize=(15, 20))

for i in range(len(ax)):
    for j in [0,1]:
        ax[i,j].set_xticklabels(ax[0, 0].get_xticklabels(), rotation=90)
        ax[i,j].set_ylim([0, 1])

fig.subplots_adjust(hspace=0.5)

sns.barplot(data=df_frequency, x=df_frequency.index, y='seg 0', ax=ax[0,0])
sns.barplot(data=df_frequency, x=df_frequency.index, y='seg 1', ax=ax[0,1])
sns.barplot(data=df_frequency, x=df_frequency.index, y='seg 2', ax=ax[1,0])
sns.barplot(data=df_frequency, x=df_frequency.index, y='seg 3', ax=ax[1,1])
sns.barplot(data=df_frequency, x=df_frequency.index, y='seg 4', ax=ax[2,0])
sns.barplot(data=df_frequency, x=df_frequency.index, y='seg 5', ax=ax[2,1])
sns.barplot(data=df_frequency, x=df_frequency.index, y='seg 6', ax=ax[3,0])
sns.barplot(data=df_frequency, x=df_frequency.index, y='seg 7', ax=ax[3,1])
sns.barplot(data=df_frequency, x=df_frequency.index, y='seg 8', ax=ax[4,0])
```

2.5 Recency

No clear trend on number of days apart between two transactions

In [ ]:
```
PATTERN = r'.*STAFF.*'
promo_temp = df_master['PROMOTIONCD'].dropna()
staff_promo_cd = promo_temp[promo_temp.str.contains(PATTERN)].tolist()
```

In [ ]:
```
df_recency1 = df_master[~df_master['PROMOTIONCD'].isin(staff_promo_cd)][
    ['CARDNO', 'ORDERID','TRANSACTIONDTTM', 'seg']].drop_duplicates(
    ['CARDNO', 'ORDERID']).sort_values(['CARDNO', 'ORDERID']).reset_index(drop=True)

df_recency1['CARDNO_SHIFT'] = df_recency1['CARDNO'].shift(-1)
df_recency1['TRANSACTIONDTTM_SHIFT'] = df_recency1['TRANSACTIONDTTM'].shift(-1)
df_recency1['CARDNO_SAME'] = np.where((df_recency1['CARDNO'].str[:] == df_recency1['CARDNO_SHIFT'].st

df_recency1['time passed'] = np.where(df_recency1['CARDNO_SAME'] == 1,
                                      (df_recency1['TRANSACTIONDTTM_SHIFT'] - df_recency1['TRANSACTIOND
                                      0)
df_recency2 = df_recency1[df_recency1['time passed'] != 0]
#df_recency2.head(10)
```

In [ ]:
```
fig, ax = plt.subplots(5, 2, figsize = (15,20))
RECENCY_BINS = [x for x in range(0,200,30)]
for i in range(len(ax)):
    for j in [0,1]:
        ax[i,j].set_ylim([0, 1])
        ax[i,j].set_xticks(RECENCY_BINS)

sns.histplot(data = df_recency2[df_recency2['seg'] == 0], x='time passed',
             ax=ax[0,0], bins=RECENCY_BINS, hue='seg', stat='probability')
sns.histplot(data = df_recency2[df_recency2['seg'] == 1], x='time passed',
             ax=ax[0,1], bins=RECENCY_BINS, hue = 'seg', stat='probability')
sns.histplot(data = df_recency2[df_recency2['seg'] == 2], x='time passed',
             ax=ax[1,0], bins=RECENCY_BINS, hue = 'seg', stat='probability')
sns.histplot(data = df_recency2[df_recency2['seg'] == 3], x='time passed',
             ax=ax[1,1], bins=RECENCY_BINS, hue = 'seg', stat='probability')
sns.histplot(data = df_recency2[df_recency2['seg'] == 4], x='time passed',
             ax=ax[2,0], bins=RECENCY_BINS, hue = 'seg', stat='probability')
sns.histplot(data = df_recency2[df_recency2['seg'] == 5], x='time passed',
             ax=ax[2,1], bins=RECENCY_BINS, hue = 'seg', stat='probability')
sns.histplot(data = df_recency2[df_recency2['seg'] == 6], x='time passed',
             ax=ax[3,0], bins=RECENCY_BINS, hue = 'seg', stat='probability')
sns.histplot(data = df_recency2[df_recency2['seg'] == 7], x='time passed',
             ax=ax[3,1], bins=RECENCY_BINS, hue='seg', stat='probability')
sns.histplot(data = df_recency2[df_recency2['seg'] == 8], x='time passed',
             ax=ax[4,0], bins=RECENCY_BINS, hue='seg', stat='probability')
sns.histplot(data=df_recency2, x='time passed',
             ax=ax[4,1], bins=RECENCY_BINS, stat='probability')
```

In [ ]:
```
```

2.6 Size of cart

Seg 0 has the biggest cart per transaction, by a huge margin

In [ ]:
```
df_basket_size1 = df_master.groupby(['ORDERID']).agg({'ORDERDETAILSID': np.count_nonzero}).reset_inde
```

In [ ]:
```
df_basket_size1.columns = ['ORDERID', 'count']
```

In [ ]:
```
df_master_reduced = df_master[['CARDNO', 'ORDERID', 'seg']].drop_duplicates(['CARDNO', 'ORDERID'])
df_basket_size2 = df_master_reduced.merge(df_basket_size1, how='left', on='ORDERID')
```

In [ ]:
```
df_basket_size2.head()
```

In [ ]:
```
ax = plt.subplots(figsize = [10,10])

sns.boxplot(x="seg", y="count", data=df_basket_size2, showfliers=False)
```

In [ ]:
```
df_basket_size2.groupby(['seg']).agg({'count': [stats.mode, np.mean, np.median]}).droplevel(0, axis=1
```

2.7 Price sensitivity

No clear trend on price sensitivity

In [ ]:
```
df_pricing = df_master[
    ['CARDNO',
     'QUANTITY',
     'UNITPRICE',
     'seg',
     'PRODUCTTITLE',
     'BRAND',
     'PROGROUPED']
]

NO_OF_TOP_PROD = 5
top_lt_products = df_master[df_master['PROGROUPED'] == 'LT'].groupby('PRODUCTTITLE').agg(
    {'SUB_GROSSAMT': np.sum}).reset_index().sort_values(
    'SUB_GROSSAMT', ascending = False)['PRODUCTTITLE'].head(NO_OF_TOP_PROD).tolist()

df_pricing_top_prod = df_pricing.query('PRODUCTTITLE in @top_lt_products')

#df_pricing_top_prod.head()
```

In [ ]:
```
print(top_lt_products)
```

In [ ]:
```
df_yamazaki = df_pricing_top_prod[df_pricing_top_prod['PRODUCTTITLE'] == 'Yamazaki 12 Years Japanese

df_mean_price = df_yamazaki.groupby(['seg', 'UNITPRICE']).agg(
    {'UNITPRICE': np.count_nonzero}
).unstack().droplevel(0, axis=1).fillna(0)
```

In [ ]:
```
f, ax1 = plt.subplots(figsize=(10, 10))
sns.heatmap(df_mean_price,
            annot=True,
            fmt= ".0f",
            cmap="Blues",
            ax=ax1)

bottom, top = ax1.get_ylim()
ax1.set_ylim(
    bottom + 0.5,
    top - 0.5)
```

2.8 Age

No significant in age distribution across all segments.

In [ ]:
```
df_age_merged = df_master[['CARDNO', 'seg']].merge(
    df_age[['CARDNO', 'age']], how='left', on='CARDNO').drop_duplicates('CARDNO')
```

In [ ]:
```
f,ax = plt.subplots(5,2,figsize=(20, 20))
AGE_BINS = [x for x in range(0,100,5)]

sns.histplot(data=df_age_merged[df_age_merged['seg'] == 0], x='age', ax=ax[0,0], bins=AGE_BINS, hue='
sns.histplot(data=df_age_merged[df_age_merged['seg'] == 1], x='age', ax=ax[0,1], bins=AGE_BINS, hue='
sns.histplot(data=df_age_merged[df_age_merged['seg'] == 2], x='age', ax=ax[1,0], bins=AGE_BINS, hue='
sns.histplot(data=df_age_merged[df_age_merged['seg'] == 3], x='age', ax=ax[1,1], bins=AGE_BINS, hue='
sns.histplot(data=df_age_merged[df_age_merged['seg'] == 4], x='age', ax=ax[2,0], bins=AGE_BINS, hue='
sns.histplot(data=df_age_merged[df_age_merged['seg'] == 5], x='age', ax=ax[2,1], bins=AGE_BINS, hue='
sns.histplot(data=df_age_merged[df_age_merged['seg'] == 6], x='age', ax=ax[3,0], bins=AGE_BINS, hue='
sns.histplot(data=df_age_merged[df_age_merged['seg'] == 7], x='age', ax=ax[3,1], bins=AGE_BINS, hue='
sns.histplot(data=df_age_merged[df_age_merged['seg'] == 8], x='age', ax=ax[4,0], bins=AGE_BINS, hue='

sns.histplot(data=df_age_merged, x='age', ax=ax[4,1], bins=AGE_BINS)
```

2.9 Discount

Seg 6 and 7 have the highest proportion of discount over gross spend applied

In [ ]:
```
# discount by iSC
df_master_reduced = df_master.drop_duplicates(['ORDERID'])
df_iSC_disc = df_master_reduced.groupby('ORDERID').agg({'DISCOUNTAMT': np.sum}).reset_index()
```

In [ ]:
```
# discount given by tenants
df_tenant_disc = df_master.groupby(['ORDERID']).agg({'TENANTDISCAMT': np.sum}).reset_index()
df_disc = df_tenant_disc.merge(right=df_iSC_disc, how ='left',on='ORDERID')
```

In [ ]:
```
df_disc_merged = df_disc.merge(right=df_master[['ORDERID', 'GROSSAMT', 'seg']], how='left', on='ORDER

df_disc_merged['total_disc'] = df_disc_merged['TENANTDISCAMT'] + df_disc_merged['DISCOUNTAMT']
df_disc_merged['disc_shares'] = df_disc_merged['total_disc'] / df_disc_merged['GROSSAMT']
```

In [ ]:
```
df_disc_by_seg = df_disc_merged.groupby('seg').agg({
    'TENANTDISCAMT' : np.sum,
    'DISCOUNTAMT': np.sum,
    'total_disc': np.sum,
    'GROSSAMT': np.sum
})

df_disc_by_seg['disc_share'] = df_disc_by_seg['total_disc'] / df_disc_by_seg['GROSSAMT']

df_disc_by_seg.sort_values('disc_share', ascending = False)
```

In [ ]:
```

```

2.10 Subscriptions

Seg 4 has the highest proportion of opt-in - More engaged higher spend?

In [ ]:
```
df_subs = df_master.groupby(['seg', 'OPTIN']).agg({'OPTIN': np.count_nonzero}).unstack().droplevel(0,
```

In [ ]:
```
df_subs['y_shares'] = df_subs['Y'] / df_subs.sum(axis=1)
df_subs['n_shares'] = 1- df_subs['y_shares']
```

In [ ]:
```
df_subs.sort_values(['y_shares'], ascending=False, inplace=True)
df_subs
```

In [ ]:
```
sum_df = df_subs.sum(axis=0).to_list()
mean_subs = sum_df[1] / (sum_df[0] + sum_df[1])
```

In [ ]:
```
fig, ax = plt.subplots(figsize=(10,5))

df_subs[['y_shares','n_shares']].plot(kind='bar',stacked=True,ax=ax)
ax.axhline(y=mean_subs,c='r')

# net spend between optin and non-optin
```

2.11 Gender

seg 3 has significant more male shoppers than female
seg 6 has significant more female shoppers - Logical as this seg is made up of beauty products buyers

In [ ]:
```
df_gender = df_master.groupby(['seg', 'GENDERCD']).agg({
    'GENDERCD': np.count_nonzero
}).unstack().droplevel(0, axis=1).drop(['U'], axis=1)
```

In [ ]:
```
df_gender['m_shares'] = df_gender['M'] / df_gender.sum(axis=1)
df_gender['f_shares'] = 1 - df_gender['m_shares']
```

In [ ]:
```
df_gender.sort_values(['m_shares'], ascending=False, inplace=True)
df_gender
```

In [ ]:
```
sum_df_gender = df_gender.sum(axis=0).to_list()
mean_gender = sum_df_gender[1] / (sum_df_gender[0] + sum_df_gender[1])
```

In [ ]:
```
fig,ax = plt.subplots(figsize=(10,5))

df_gender[['m_shares', 'f_shares']].plot(kind='bar', stacked=True, ax=ax)
ax.axhline(y=mean_gender, c='r')
```

2.12 Gross Spend by Opt In

Counter-intuiative, shoppers who opt out of newsletter subsciptions has higher per capita spend than those who opted in

In [ ]:
```
df_opt_in = df_master[df_master['OPTIN'] == 'Y']
df_opt_out = df_master[df_master['OPTIN'] == 'N']
```

In [ ]:
```
# Computation for customers who opted in
df_opt_in_spend1 = df_opt_in.groupby(['ORDERID']).agg({'SUB_GROSSAMT': np.sum, 'seg': np.min})
df_opt_in_spend2 = df_opt_in_spend1.groupby('seg').agg({'SUB_GROSSAMT': np.sum}).reset_index()
df_opt_in_spend2.columns = ['seg', 'in']

df_opt_in_size = df_opt_in.groupby(['seg']).agg({'ORDERID': 'nunique'}).reset_index()
df_opt_in_size.columns = ['seg', 'in_count']
```

In [ ]:
```
# Computation for customers who opted out
df_opt_out_spend1 = df_opt_out.groupby(['ORDERID']).agg({'SUB_GROSSAMT': np.sum, 'seg': np.min})
df_opt_out_spend2 = df_opt_out_spend1.groupby('seg').agg({'SUB_GROSSAMT': np.sum}).reset_index()
df_opt_out_spend2.columns = ['seg', 'out']

df_opt_out_size = df_opt_out.groupby(['seg']).agg({'ORDERID': 'nunique'}).reset_index()
df_opt_out_size.columns = ['seg', 'out_count']
```

In [ ]:
```
df_opt_spend = df_opt_in_spend2.merge(right=df_opt_out_spend2, how='inner', on='seg')

df_opt_size = df_opt_in_size.merge(right=df_opt_out_size, on='seg',how='inner')
```

In [ ]:
```
df_opt = df_opt_spend.merge(right=df_opt_size, on='seg', how='inner')

df_opt['in_per_capita'] = df_opt['in'] / df_opt['in_count']
df_opt['out_per_capita'] = df_opt['out'] / df_opt['out_count']
df_opt['sales difference'] = df_opt['in_per_capita'] - df_opt['out_per_capita']
```

In [ ]:
```
df_opt.sort_values('in_per_capita', ascending=False)
```

In [ ]:
```

```

In [ ]:
```

```

In [ ]:
```

```