In [45]:

```
# import libraries and load the data
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
pd.set_option('max_columns',150)
store_dataAfterdrop = pd.read_excel('Submitive_Data.xlsx')
store_dataAfterdrop.head()
```

Out[45]:

	Unnamed: 0	emp_title	application_type	home_ownership	loan_status	term	installment	grade
0	0	1	1	1	1	1	1214.15	1
1	1	2	1	1	2	2	489.51	1
2	2	27	1	2	3	2	466.20	1
3	3	4	1	1	1	1	586.74	2
4	4	5	1	2	1	2	497.04	3

In [46]:

```
X= store_dataAfterdrop[store_dataAfterdrop.columns.difference(['loan_status'])]

#Designate the outcome or target variable as y
y = store_dataAfterdrop.loan_status
print(X.shape)
print(y.shape)
labels_true = y
print(X)
print(labels_true)
```

```
20
3               1       0.869          0     1             53705
34
4               1       0.242          0     2            449124
24
...           ...         ...        ...   ...               ...
...
66873           1       0.770          0     1            317714
28
66874           1       0.590          0     1            380950
20
66875           1       0.040          0     1            127710
14
66876           1       0.727          0     1            122026
37
66877           1       0.865          0     1            128773
13

       total_bal_ex_mort  total_bc_limit  total_il_high_credit_limit
`
```

In [47]:

```
store_dataAfterdrop.describe()
```

Out[47]:

	Unnamed: 0	emp_title	application_type	home_ownership	loan_status	ter
count	66878.000000	66878.000000	66878.000000	66878.000000	66878.000000	66878.0000(
mean	38427.006175	6308.206092	1.109348	1.692201	1.996337	1.28261
std	22275.277353	8044.688782	0.312078	0.661418	0.963236	0.45027
min	0.000000	1.000000	1.000000	1.000000	1.000000	1.0000(
25%	19115.250000	228.000000	1.000000	1.000000	1.000000	1.0000(
50%	38363.500000	1941.500000	1.000000	2.000000	2.000000	1.0000(
75%	57677.750000	10685.750000	1.000000	2.000000	3.000000	2.0000(
max	77158.000000	28186.000000	2.000000	5.000000	4.000000	2.0000(

In [48]:

```
store_dataAfterdrop.drop(['loan_status'], axis=1, inplace=True)
store_dataAfterdrop.describe()
```

Out[48]:

	Unnamed: 0	emp_title	application_type	home_ownership	term	installmer
count	66878.000000	66878.000000	66878.000000	66878.000000	66878.000000	66878.0000(
mean	38427.006175	6308.206092	1.109348	1.692201	1.282619	403.1794(
std	22275.277353	8044.688782	0.312078	0.661418	0.450276	246.05053
min	0.000000	1.000000	1.000000	1.000000	1.000000	7.6100(
25%	19115.250000	228.000000	1.000000	1.000000	1.000000	225.8300(
50%	38363.500000	1941.500000	1.000000	2.000000	1.000000	342.1700(
75%	57677.750000	10685.750000	1.000000	2.000000	2.000000	527.5100(
max	77158.000000	28186.000000	2.000000	5.000000	2.000000	1546.5200(

In [49]:

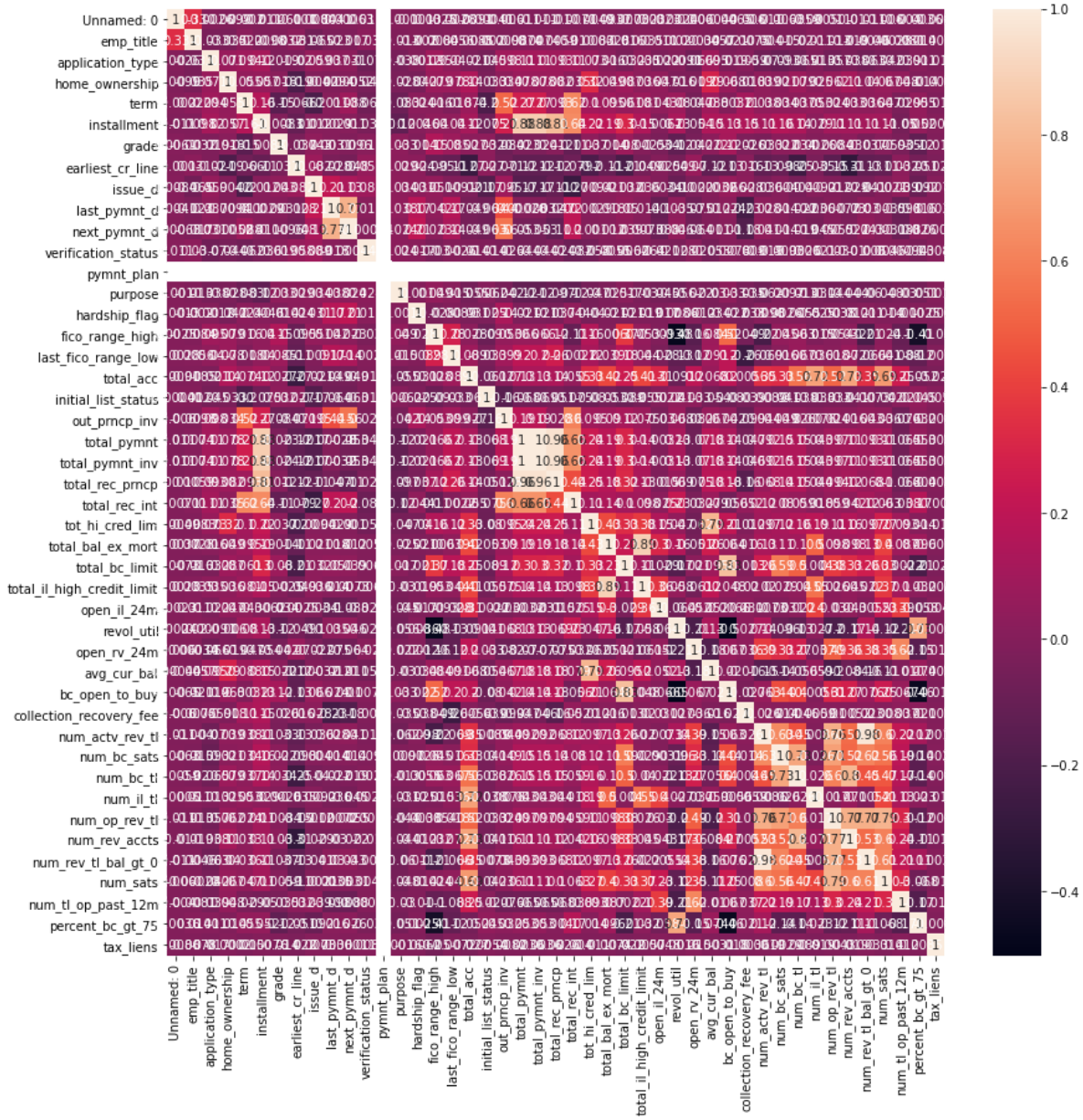
```
store_dataAfterdrop.isna().mean() * 100
```

Out[49]:

```
Unnamed: 0                    0.0
emp_title                     0.0
application_type              0.0
home_ownership                0.0
term                          0.0
installment                   0.0
grade                         0.0
earliest_cr_line              0.0
issue_d                       0.0
last_pymnt_d                  0.0
next_pymnt_d                  0.0
verification_status           0.0
pymnt_plan                    0.0
purpose                       0.0
hardship_flag                 0.0
fico_range_high               0.0
last_fico_range_low           0.0
total_acc                     0.0
initial_list_status           0.0
out_prncp_inv                 0.0
total_pymnt                   0.0
total_pymnt_inv               0.0
total_rec_prncp               0.0
total_rec_int                 0.0
tot_hi_cred_lim               0.0
total_bal_ex_mort             0.0
total_bc_limit                0.0
total_il_high_credit_limit    0.0
open_il_24m                   0.0
revol_util                    0.0
open_rv_24m                   0.0
avg_cur_bal                   0.0
bc_open_to_buy                0.0
collection_recovery_fee       0.0
num_actv_rev_tl               0.0
num_bc_sats                   0.0
num_bc_tl                     0.0
num_il_tl                     0.0
num_op_rev_tl                 0.0
num_rev_accts                 0.0
num_rev_tl_bal_gt_0           0.0
num_sats                      0.0
num_tl_op_past_12m            0.0
percent_bc_gt_75              0.0
tax_liens                     0.0
dtype: float64
```

In [50]:

```
plt.figure(figsize=(14,14))
sns.heatmap(store_dataAfterdrop.corr(), annot=True)
plt.show()
```



In [51]:

```
from sklearn.cluster import KMeans


k_means = KMeans(3)
k_means.fit(X)

labels_pred = k_means.predict(X)
print(labels_pred)
```

```
[2 2 1 ... 2 2 0]
```

In [52]:

```
from sklearn.metrics.cluster import adjusted_rand_score


score = adjusted_rand_score(labels_true, labels_pred)
score
```

Out[52]:

```
0.6547074534249879
```

In [53]:

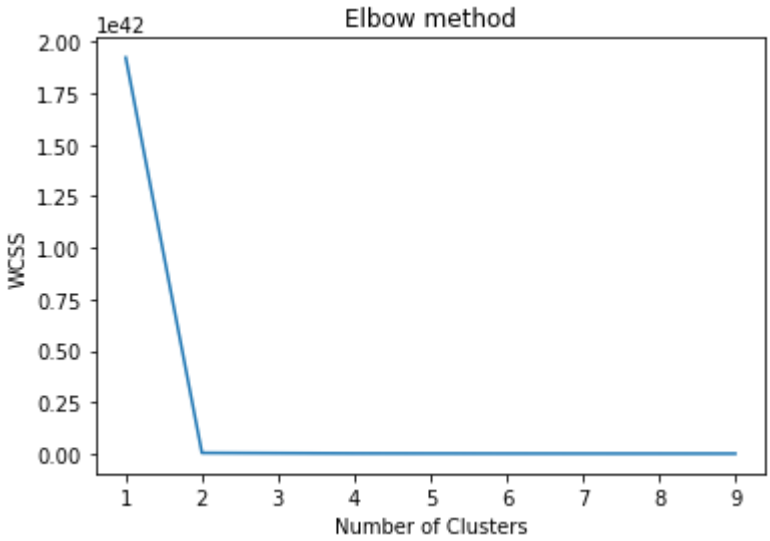
```
from sklearn.metrics.cluster import adjusted_mutual_info_score


score = adjusted_mutual_info_score(labels_true, labels_pred)
score
```

Out[53]:

```
0.6669185516003795
```

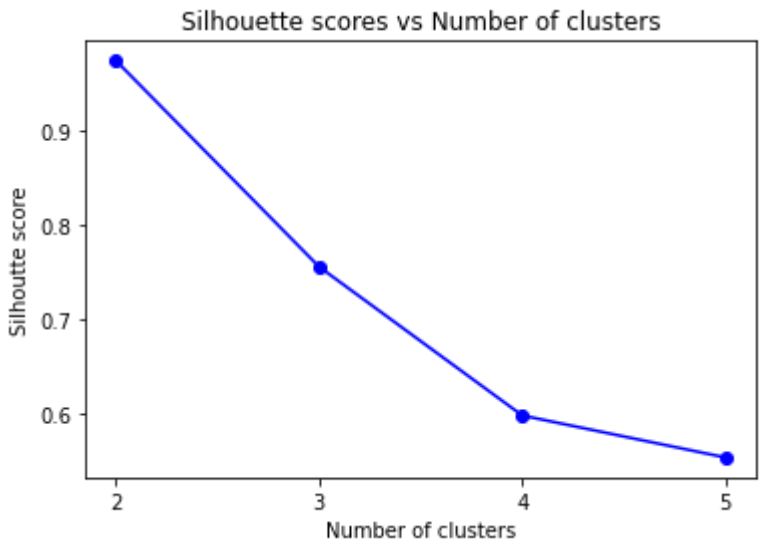
In [54]:

```
from sklearn.cluster import KMeans

kmeans_models = [KMeans(n_clusters=k).fit(store_dataAfterdrop) for k in range (1, 10
innertia = [model.inertia_ for model in kmeans_models]

plt.plot(range(1, 10), innertia)
plt.title('Elbow method')
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS')
plt.show()
```



In [55]:

```
from sklearn.metrics import silhouette_score
from sklearn.cluster import KMeans
silhoutte_scores = [silhouette_score(store_dataAfterdrop, model.labels_) for model i
plt.plot(range(2,6), silhoutte_scores, "bo-")
plt.xticks([2, 3, 4, 5])
plt.title('Silhouette scores vs Number of clusters')
plt.xlabel('Number of clusters')
plt.ylabel('Silhoutte score')
plt.show()
```



In [ ]:

```
```

In [56]:

```
from sklearn.metrics import silhouette_score

kmeans = KMeans(n_clusters=2, random_state=23)
kmeans.fit(store_dataAfterdrop)

print('Silhoutte score of our model is ' + str(silhouette_score(store_dataAfterdrop,
```

```
Silhoutte score of our model is 0.9753017123493052
```

In [57]:

```
#add the cluster ids to the table
store_dataAfterdrop['cluster_id'] = kmeans.labels_
```

In [58]:

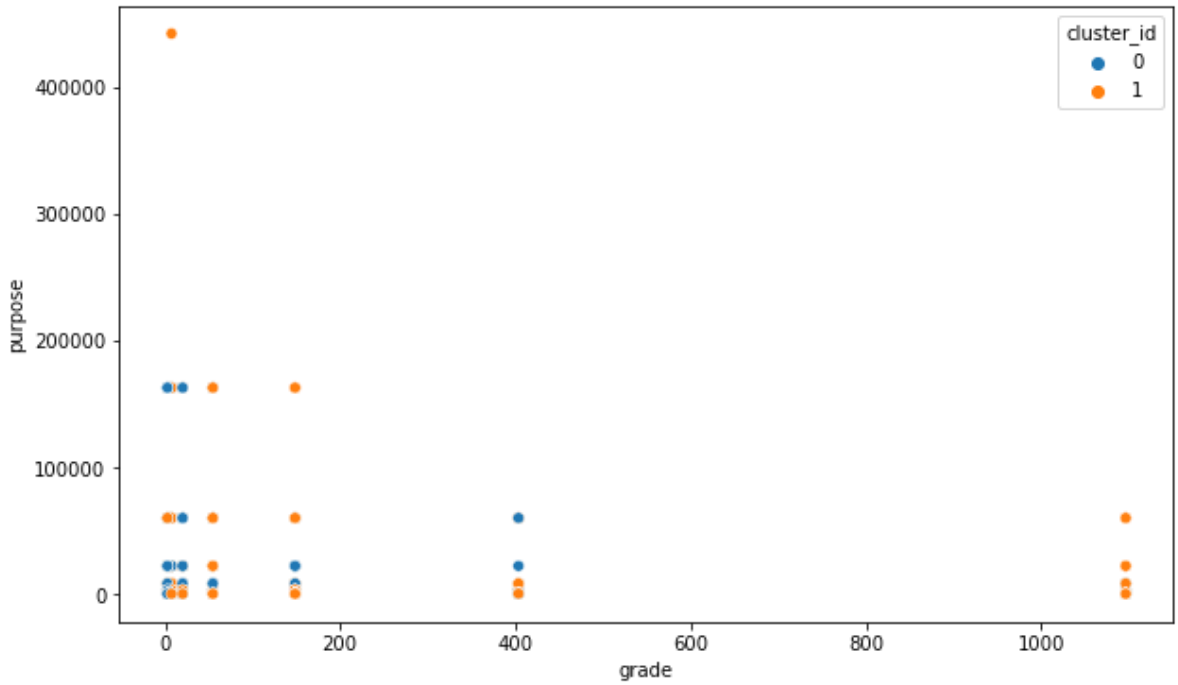
```
cols = ['home_ownership', 'application_type', 'term', 'installment', 'grade','purpos
```

In [ ]:

```
for col in cols:
    store_dataAfterdrop[col] = np.exp(store_dataAfterdrop[col])
```

In [61]:

```
plt.figure(figsize=(10,6))
sns.scatterplot(data=store_dataAfterdrop, x='grade', y='purpose', hue='cluster_id')
plt.title('')
plt.show()
```



In [67]:

```
from sklearn.pipeline import Pipeline
from sklearn.ensemble import RandomForestClassifier
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split

# Without stratification divide into first partition and test set.
X_train, X_test, y_train, y_test= train_test_split(X,y, test_size=0.2,random_state=1
estimators = [("PCA", PCA()), ("clf_RFC", RandomForestClassifier())]

pipe = Pipeline(estimators)
print(pipe)


#we can use a pipeline like a classifier or transformer.
pipe.fit(X_train, y_train)

y_hat = pipe.predict(X_test)

from sklearn.metrics import accuracy_score

accuracy = accuracy_score(y_test, y_hat)
print(accuracy)
```

```
Pipeline(steps=[('PCA', PCA()), ('clf_RFC', RandomForestClassifier
())])
0.9558911483253588
```

In [64]:

```
from sklearn.cluster import FeatureAgglomeration



agglo = FeatureAgglomeration(n_clusters = 2,linkage="average")
agglo.fit(X)

X_reduced = agglo.transform(X)

k_means = KMeans(4)
k_means.fit(X_reduced)

labels_pred = k_means.predict(X_reduced)
print(labels_pred)
```

```
[0 0 2 ... 0 0 3]
```

In [65]:

```
from sklearn.metrics.cluster import adjusted_rand_score


score_reduced = adjusted_rand_score(labels_true, labels_pred)
print("Full features score", score)
print("Reduced Features score: ", score_reduced)
```

```
Full features score 0.6669185516003795
Reduced Features score:  0.4740806518489913
```

In [ ]:

```

```