Project Phase 2 Machine Learning

Name: Abdelrahman Mohamed

ID: 202101081

- ***Project Steps***

1- Import libraries
2- Load the Data and Knew Some info about it
3- Implement linear regression algorithm
4- Apply agglomerative clustering and draw the dendrogram
5- Use the resulting clusters as initial centroids for the K-Means clustering algorithm.
6- Show Agglomerative Clustering and K-Means Clustering together
7- Evaluate Clustering Models
8- Implement a For Loop with a Stop Condition

- First dataset for the linear regression algorithm

real estate dataset, containing information such as transaction date, house age, distance to the nearest MRT station, number of convenience stores, latitude, longitude, and house price of unit area.

- Second dataset for the **clustering algorithms**

Absenteeism Dataset. The columns include information such as ID, reason for absence, month of absence, day of the week, seasons, transportation expense, distance from residence to work, service time, age, work load average per day, hit target, disciplinary failure, education, son, social drinker, social smoker, pet, weight, height, body mass index, and absenteeism time in hours.

- FULL CODE
- **Import libraries**

import libraries

In [1]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import numpy as np
import matplotlib.pyplot as plt
import numpy as np
from scipy.cluster.hierarchy import dendrogram, linkage
from sklearn.cluster import KMeans
from sklearn.cluster import AgglomerativeClustering
import os
from scipy.spatial.distance import cdist
```

Read the data and Knew some information

In [2]:
```
# Replace 'yourDataset.csv' with your actual dataset filename
df = pd.read_csv(r"Real estate valuation data set.csv", delimiter = ";")
df.head()
```

Out[2]:

	No	X1 transaction date	X2 house age	X3 distance to the nearest MRT station	X4 number of convenience stores	X5 latitude	X6 longitude	Y house price of unit area
0	1	2012.917	32.0	84.87882	10	24.98298	121.54024	37.9
1	2	2012.917	19.5	306.59470	9	24.98034	121.53951	42.2
2	3	2013.583	13.3	561.98450	5	24.98746	121.54391	47.3
3	4	2013.500	13.3	561.98450	5	24.98746	121.54391	54.8
4	5	2012.833	5.0	390.56840	5	24.97937	121.54245	43.1

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 414 entries, 0 to 413
Data columns (total 8 columns):
 #   Column                                  Non-Null Count  Dtype  
---  ------                                  --------------  -----  
 0   No                                      414 non-null    int64  
 1   X1 transaction date                     414 non-null    float64
 2   X2 house age                            414 non-null    float64
 3   X3 distance to the nearest MRT station  414 non-null    float64
 4   X4 number of convenience stores         414 non-null    int64  
 5   X5 latitude                             414 non-null    float64
 6   X6 longitude                            414 non-null    float64
 7   Y house price of unit area              414 non-null    float64
dtypes: float64(6), int64(2)
memory usage: 26.0 KB
```

Implement the linear regression algorithm

In [5]:
```
# Replace 'yourDataset.csv' with your actual dataset filename
df = pd.read_csv(r"Real estate valuation data set.csv", delimiter = ";")
df.head()
```

Out[5]:

	No	X1 transaction date	X2 house age	X3 distance to the nearest MRT station	X4 number of convenience stores	X5 latitude	X6 longitude	Y house price of unit area
0	1	2012.917	32.0	84.87882	10	24.98298	121.54024	37.9
1	2	2012.917	19.5	306.59470	9	24.98034	121.53951	42.2
2	3	2013.583	13.3	561.98450	5	24.98746	121.54391	47.3
3	4	2013.500	13.3	561.98450	5	24.98746	121.54391	54.8
4	5	2012.833	5.0	390.56840	5	24.97937	121.54245	43.1

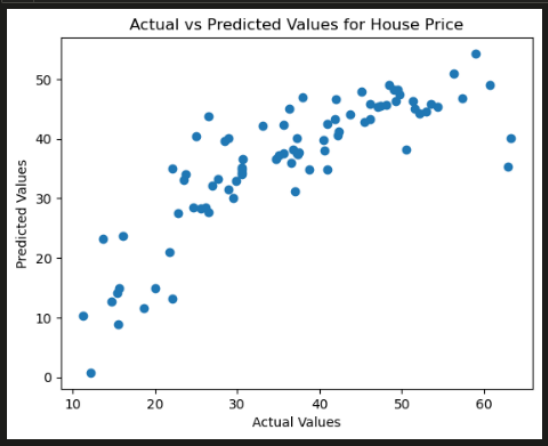
In [6]:
```
# Define features and target
features = ['X1 transaction date','X2 house age', 'X3 distance to the nearest MRT station', 'X4 number of convenience stores
target = 'Y house price of unit area'

X = df[features]
y = df[target]

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [7]:
```
# Initialize and train the linear regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)

print(f'Mean Squared Error: {mse}')
print(f'Root Mean Squared Error: {rmse}')
```

```
Mean Squared Error: 53.50561912450112
Root Mean Squared Error: 7.314753524521596
```

Root Mean Squared Error: 7.314753524521596

In [8]:
```
# Plot the predicted vs actual values
plt.scatter(y_test, y_pred)
plt.xlabel("Actual Values")
plt.ylabel("Predicted Values")
plt.title("Actual vs Predicted Values for House Price")
plt.show()
```



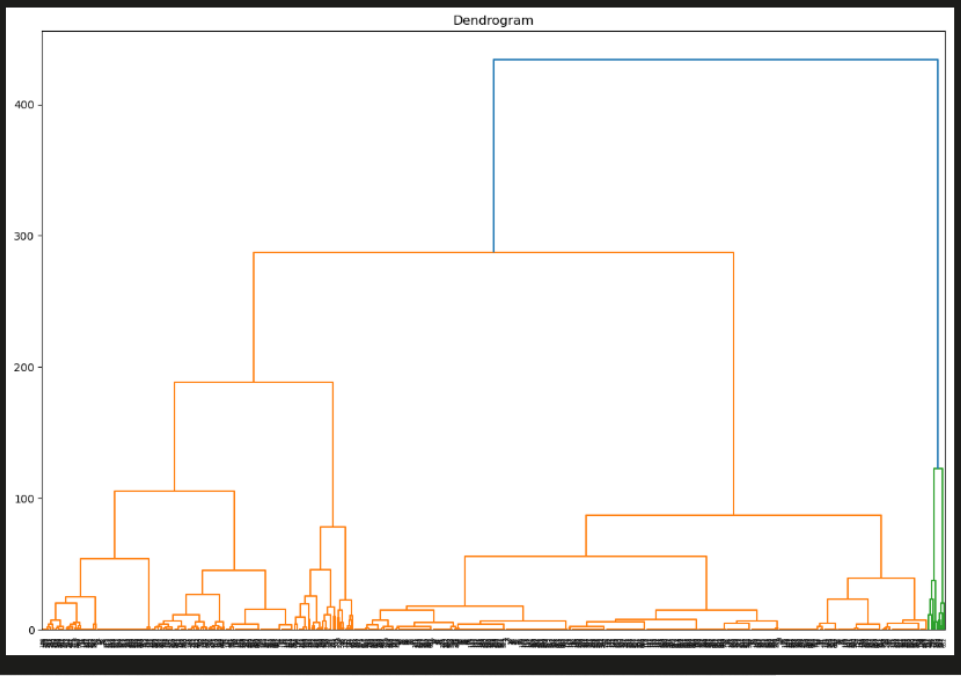
- Apply agglomerative clustering and draw the dendrogram

Apply agglomerative clustering and draw the dendrogram

In [10]:
```
#  This can provide insights into whether certain reasons for absence are associated with specific patterns of absenteeism d

data = pd.read_csv(r"Absenteeism_at_work.csv", sep=";")
# data and feature selection
X = data.iloc[:, [1, -1]].values   # Using only two features for simplicity

agg_cluster = AgglomerativeClustering(distance_threshold=0, n_clusters=None, linkage='ward')
agg_labels = agg_cluster.fit_predict(X)

print(agg_labels)

plt.figure(figsize=(15, 10))
linked = linkage(X, 'ward')
dendrogram(linked, orientation='top', distance_sort='descending', show_leaf_counts=True)
plt.title('Dendrogram')
plt.show()
```

```
[739 738 737 736 554 735 733 732 591 524 727 477 725 724 699 722 734 720
 378 721 717 395 731 715 714 419 710 705 657 718 708 706 723 352 707 383
 700 349 698 704 702 697 695 693 696 691 410 694 728 503 347 687 692 685
 686 683 681 690 682 701 680 689 677 675 679 678 674 671 670 684 668 666
 665 669 663 584 661 660 672 658 659 328 655 348 646 499 688 439 656 650
 664 654 647 709 645 552 643 649 651 640 667 371 639 638 635 652 644 633
 637 634 629 350 631 641 495 625 703 510 627 553 475 623 620 619 618 617
 616 615 614 624 611 613 622 612 471 626 520 521 726 719 621 441 551 630
 517 518 550 609 519 516 509 610 511 507 628 549 505 504 403 515 482 496
 493 379 599 607 173 494 384 472 322 506 662 479 512 514 466 467 490 341
 415 548 539 339 608 470 606 533 462 547 235 478 461 642 469 605 445 359
 251 480 455 489 453 673 474 546 604 508 454 444 545 411 238 452 443 442
 220 716 476 247 450 440 481 676 361 465 436 435 451 457 459 429 437 427
 544 543 254 513 425 334 636 502 422 492 491 310 428 423 420 729 438 219
 417 414 430 501 240 412 460 433 413 434 409 464 259 407 404 416 313 473
 408 458 397 207 583 632 396 394 431 237 201 205 390 418 603 393 197 319
 392 560 424 388 209 387 380 486 329 381 602 311 189 426 386 376 468 372
 601 389 369 600 432 373 367 366 255 463 391 399 185 400 374 344 485 343
 382 365 195 312 405 317 385 447 331 299 368 308 730 363 316 309 542 598
 398 597 370 483 596 595 307 713 155 594 330 302 315 541 154 402 653 536
 320 449 204 301 327 561 191 593 592 540 272 245 198 300 306 303 225 171
 274 270 377 537 563 531 217 532 527 406 525 523 500 375 269 305 538 535
 253 252 157 125 362 648 529 231 164 180 360 179 456 250 149 268 216 271
 273 213 211 126 109 304 227 218 214 208 194 488 215 342 585 103 421 212
 345 188 249 210 163 151 295 484 291 581 203 267 298 206 358 364 559 401
 199 134 133 102 196 565 239 534 448  81  97 193 326 557 314 229 104 108
 192 101 266 567 590 325 124 497 228 265 589  96 332  50  98  95 113 226
 190 187 712 181 297 588 200 132  48 587 586 186 321 530 236 323 296 152
 153 117 498 575 150 112 106 264  74 148  75 147 234 579  73 224  92 351
 293 170 223 340 346 233 318 169 357 202 335 184 183 292 107  47 241 294
 246 528 263  36 145 232  99 146 356 711 248 354 580 111 131 221  72 577
 446 576 573 261  55 222 123 487 287 571 572  51 522 116 100 159  65 158
```

```
246 528 263  36 145 232  99 146 356 711 248 354 580 111 131 221  72 577
446 576 573 261  55 222 123 487 287 571 572  51 522 116 100 159  65 158
338 242 582  61 289 260 162 182 290 324 569 119 578  91 288 143 526 144
156 574 118  90  89  49 161 568  44  66 286 285  80 142 105 337 570  52
284 564  78 283 141  77 566 279 336 333 115 282 281 178  38 355 165 129
160 140 353 562 230 257 175 555 280  57  76 139  70 174 556  86 172 114
 58  85  84 256 177 122  79 168 258 128  39  69  23  28  34 275  53 244
558 127  37 278  94  63 243 277 121  24 262  25 110  54 130  56  93 120
138 136  60  71 176 276  87  45 135 167  59  83  46 166  67  64 137  68
 32  88  82  26  33  41  12  40  29  35  62  16  43  42  18  30  31  15
 19  14  27  20  17  11   8   7  22  21   5   3  13   6   2   9  10   4
  1   0]
```



- Use the resulting clusters as initial centroids for the K-Means clustering algorithm

Use the resulting clusters as initial centroids for the K-Means clustering algorithm.

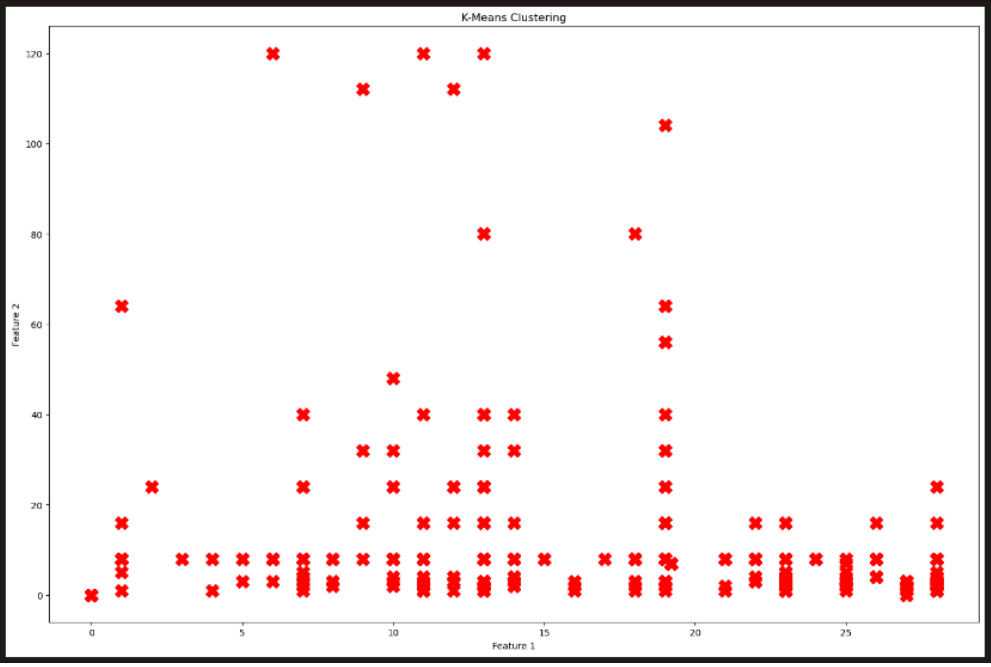
In [11]:

```
# the goal is to identify groups of instances that share similarities in terms of both the reason for absence and the durati
#  This can provide insights into whether certain reasons for absence are associated with specific patterns of absenteeism d

X = data.iloc[:, [1, -1]].values   # Using only two features for simplicity

# Agglomerative Clustering
agg_cluster = AgglomerativeClustering(distance_threshold=0, n_clusters=None, linkage='ward')
agg_labels = agg_cluster.fit_predict(X)

# Use the output of agglomerative clustering to determine the number of clusters
num_clusters = len(np.unique(agg_labels))

# K-Means with initial centroids from Agglomerative Clustering
kmeans = KMeans(n_clusters=num_clusters, random_state=42)
kmeans_labels = kmeans.fit_predict(X)
kmeans_centers = kmeans.cluster_centers_

# The labels are integers ranging from 0 to (n_clusters - 1)
print(kmeans_labels)

plt.figure(figsize=(15, 10))
plt.scatter(X[:, 0], X[:, 1], c=kmeans_labels, cmap='viridis', edgecolors='k', s=20, alpha=0.5, linewidths=0.5)
plt.scatter(kmeans_centers[:, 0], kmeans_centers[:, 1], c='red', marker='X', s=200, label='Cluster Centers')
plt.title('K-Means Clustering')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')

plt.tight_layout()
plt.show()
```

```
C:\Users\Lenovo\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init` will
change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Lenovo\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1440: UserWarning: KMeans is known to have a memory leak
on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OM
P_NUM_THREADS=3.
  warnings.warn(

[ 49   0   2   5   2   2   8  12  13   8  16  16  23  17  12  28   2  33
  23   2  36  17  39  19  40  48  46  19  40   2  40  40  12  40   2  46
  67  68  15  12   2  12  12  75   2  78  78  12   8  91   0   0   2   2
   0   0 100  78   0  46  78  12  78  78   0  46  78  78  78   2   2 115
  47  78 131  28 133  54  46 128  46  46 122  33 115 151   8 153  46 157
  78  33 161  19 145 128 167  78  46 172   2  54   2  54 182   8 128 128
   2   8  50  75   2  46 196 199  40  50   2 133 199  50  50  50  50  50
  50  50  50  40  40  50   2  50 229  46 197  46  16  23 133  47 133   8
  23  16 133   2  46  47 217  40  46  46  75 133  75 262 263   8 122 122
 210  39 270 271 272 122 157 275  19  46  28 281   2  23 252  23 257 161
 290 133 210 145 145  46 145 281  46 133  19 281  46  78  28 145  15 182
```

```
252  78  78 468 647  54  40 314   2  78   2 640 312 638 327 327 161 327
 36 100  50  40  50 525 122 525  50 122  78  16  50 128  50  50 133  50
502  50  73 525 525 217 516   2 525 486  50  50 122  50 610 217  50 608
 50   2 327  50  50 327  50   2 217 217 601  50  50 525 327 525 217  75
122  50 525  50 593 516 525  73  50  47 327  50  78 525   2 525 525  47
281 525 525 516 573   8 122  36  75  75 122  50 167 566  78   2 562   8
 50  75 559  50  19  75 382  50   8 262 133 552  54 252 133  28 172   8
 50   2 541 145   0  50   0 537 536  36   8  36 533  36   2 133  50  73
  2  72  36  17  73 525  54  36   8 402  75  50   0   0 262 516 133 161
217   8 508 507 197  49 502 157  19 500 499   2   8   8 351  28  68   0
  0   0]
```

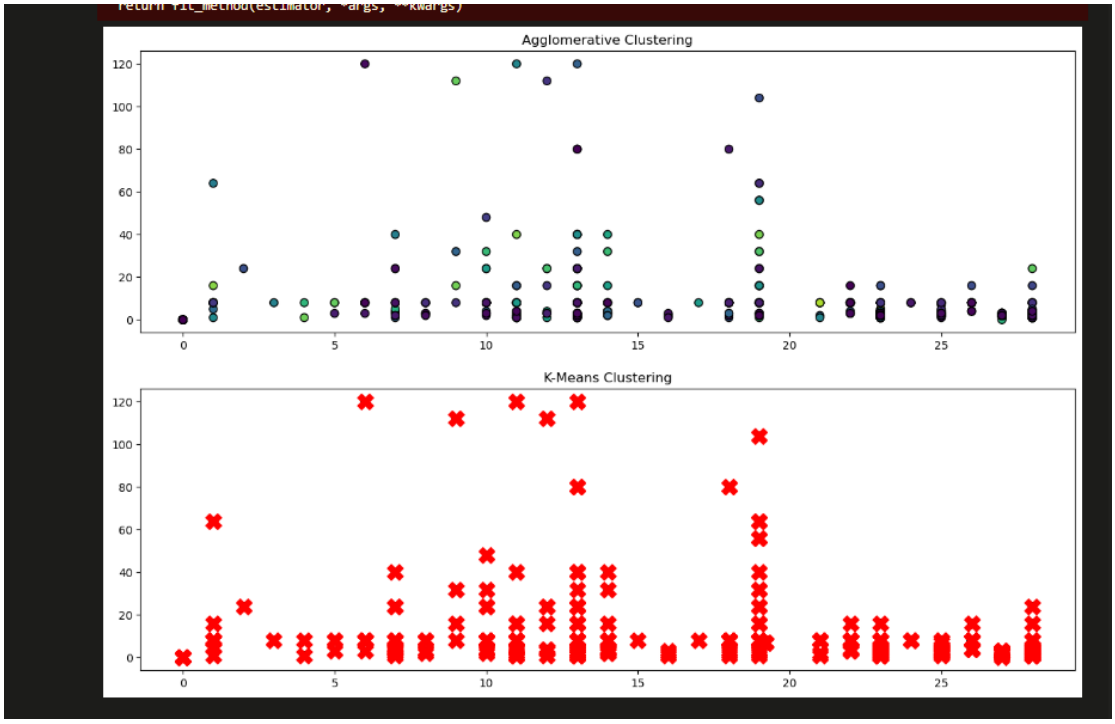
```
C:\Users\Lenovo\anaconda3\lib\site-packages\sklearn\base.py:1152: ConvergenceWarning: Number of distinct clusters (126) found s
maller than n_clusters (740). Possibly due to duplicate points in X.
  return fit_method(estimator, *args, **kwargs)
```



Agglomerative Clustering and K-Means Clustering together

In [12]:

```
X = data.iloc[:, [1, -1]].values   # Using only two features for simplicity

# Agglomerative Clustering
agg_cluster = AgglomerativeClustering(distance_threshold=0, n_clusters=None, linkage='ward')
agg_labels = agg_cluster.fit_predict(X)

# K-Means with initial centroids from Agglomerative Clustering
kmeans = KMeans(n_clusters=len(np.unique(agg_labels)), init='k-means++', random_state=42)
kmeans.fit(X)

# Plot the results
plt.figure(figsize=(15, 10))

# Plot Agglomerative Clustering results
plt.subplot(2, 1, 1)
plt.scatter(X[:, 0], X[:, 1], c=agg_labels, cmap='viridis', edgecolors='k', s=50)
plt.title('Agglomerative Clustering')

# Plot K-Means results
plt.subplot(2, 1, 2)
plt.scatter(X[:, 0], X[:, 1], c=kmeans.labels_, cmap='viridis', edgecolors='k', s=50)
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], c='red', marker='X', s=200, label='K-Means Centroi
plt.title('K-Means Clustering')

plt.show()
```

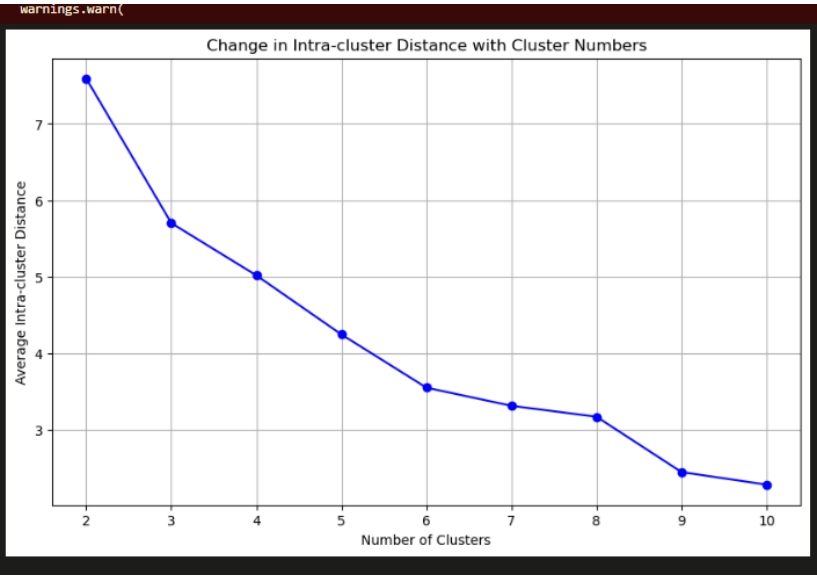


Evaluate Clustering Models

In [13]:

```
import os
os.environ['JOBLIB_START_METHOD'] = 'forkserver'
os.environ['OMP_NUM_THREADS'] = '3'

X = data.iloc[:, [1, -1]].values   # Using only two features for simplicity

# Calculate and visualize intra-cluster distance for different cluster numbers
cluster_numbers = range(2, 11)
intra_cluster_distances = []

for num_clusters in cluster_numbers:
    kmeans = KMeans(n_clusters=num_clusters, n_init=1, random_state=42)
    kmeans.fit(X)
    intra_cluster_distances.append(np.mean(np.min(cdist(X, kmeans.cluster_centers_, 'euclidean'), axis=1)))

# Visualize the change in intra-cluster distance
plt.figure(figsize=(10, 6))
plt.plot(cluster_numbers, intra_cluster_distances, marker='o', linestyle='-', color='b')
plt.title('Change in Intra-cluster Distance with Cluster Numbers')
plt.xlabel('Number of Clusters')
plt.ylabel('Average Intra-cluster Distance')
plt.grid(True)
plt.show()
```

```
warnings.warn(
```



Implement a For Loop with a Stop Condition

In [14]:

```
import os
os.environ['JOBLIB_START_METHOD'] = 'forkserver'
os.environ['OMP_NUM_THREADS'] = '3'

X = data.iloc[:, [1, -1]].values   # Using only two features for simplicity

# Initialize variables for the loop
max_iterations = 20
threshold = 0.001
prev_intra_cluster_distance = float('inf')

# Calculate and visualize intra-cluster distance for different cluster numbers
cluster_numbers = range(2, max_iterations + 2)
intra_cluster_distances = []

for num_clusters in cluster_numbers:
    kmeans = KMeans(n_clusters=num_clusters, n_init=1, random_state=42)
    kmeans.fit(X)
    intra_cluster_distance = np.mean(np.min(cdist(X, kmeans.cluster_centers_, 'euclidean'), axis=1))
    intra_cluster_distances.append(intra_cluster_distance)

    # Check if the change in intra-cluster distance is below the threshold
    if abs(prev_intra_cluster_distance - intra_cluster_distance) < threshold:
        print(f"Stopping at {num_clusters} clusters due to stabilization.")
        break

    prev_intra_cluster_distance = intra_cluster_distance

# Visualize the change in intra-cluster distance
plt.figure(figsize=(10, 6))
plt.plot(cluster_numbers, intra_cluster_distances, marker='o', linestyle='-', color='b')
plt.title('Change in Intra-cluster Distance with Cluster Numbers')
plt.xlabel('Number of Clusters')
plt.ylabel('Average Intra-cluster Distance')
plt.grid(True)
plt.show()
```

```
warnings.warn(
```