In [72]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

1.1 Read the data and do exploratory data analysis. Describe the data briefly. (Check the Data types, shape, EDA, 5 point summary). Perform Univariate, Bivariate Analysis, Multivariate Analysis.

In [73]:
```
# Read the data from the .xlsx file
df = pd.read_excel('compactiv.xlsx')
```

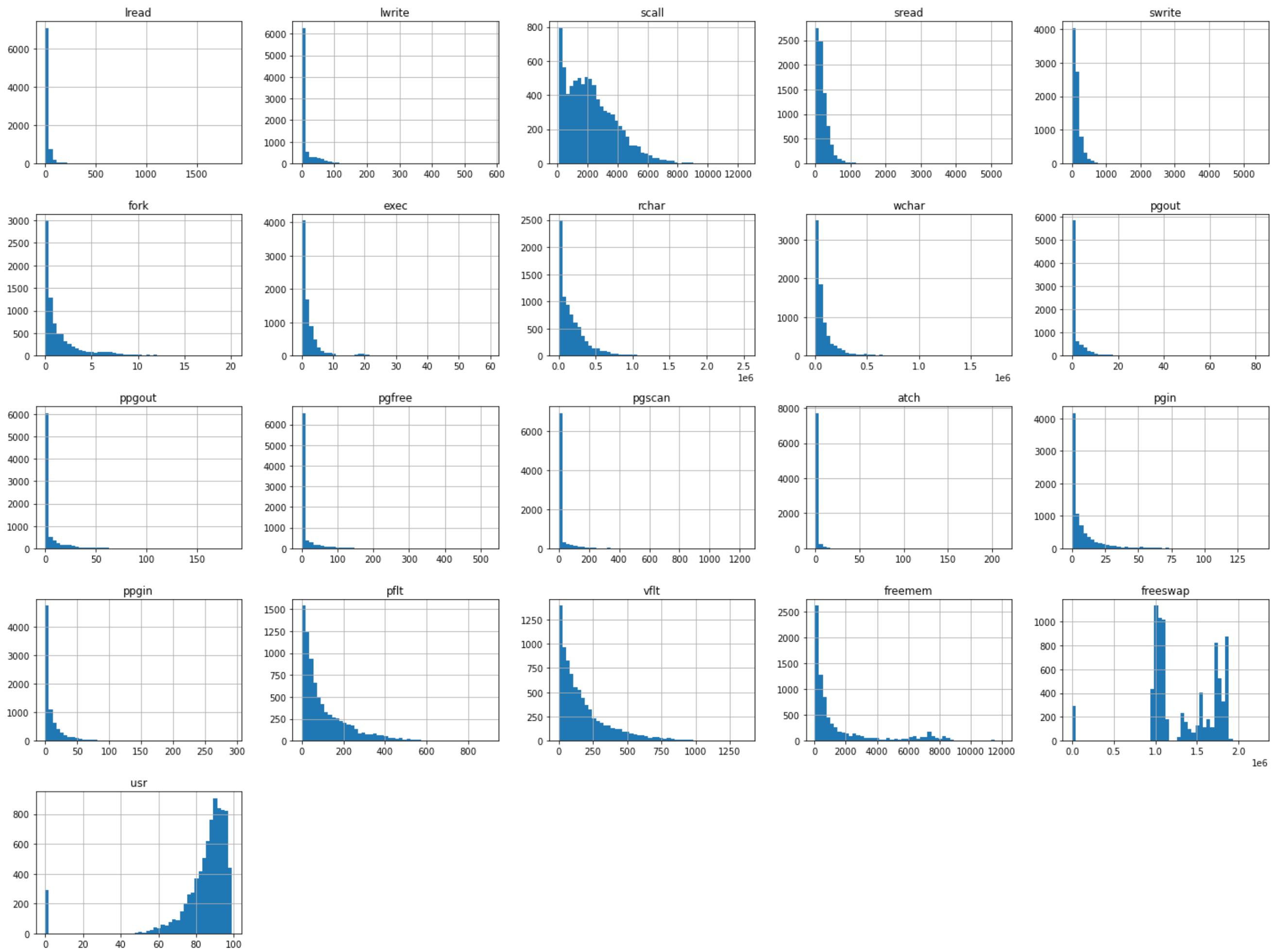
In [74]:
```
# Check the shape of the data
print("Shape of the data:", df.shape)
```

```
Shape of the data: (8192, 22)
```

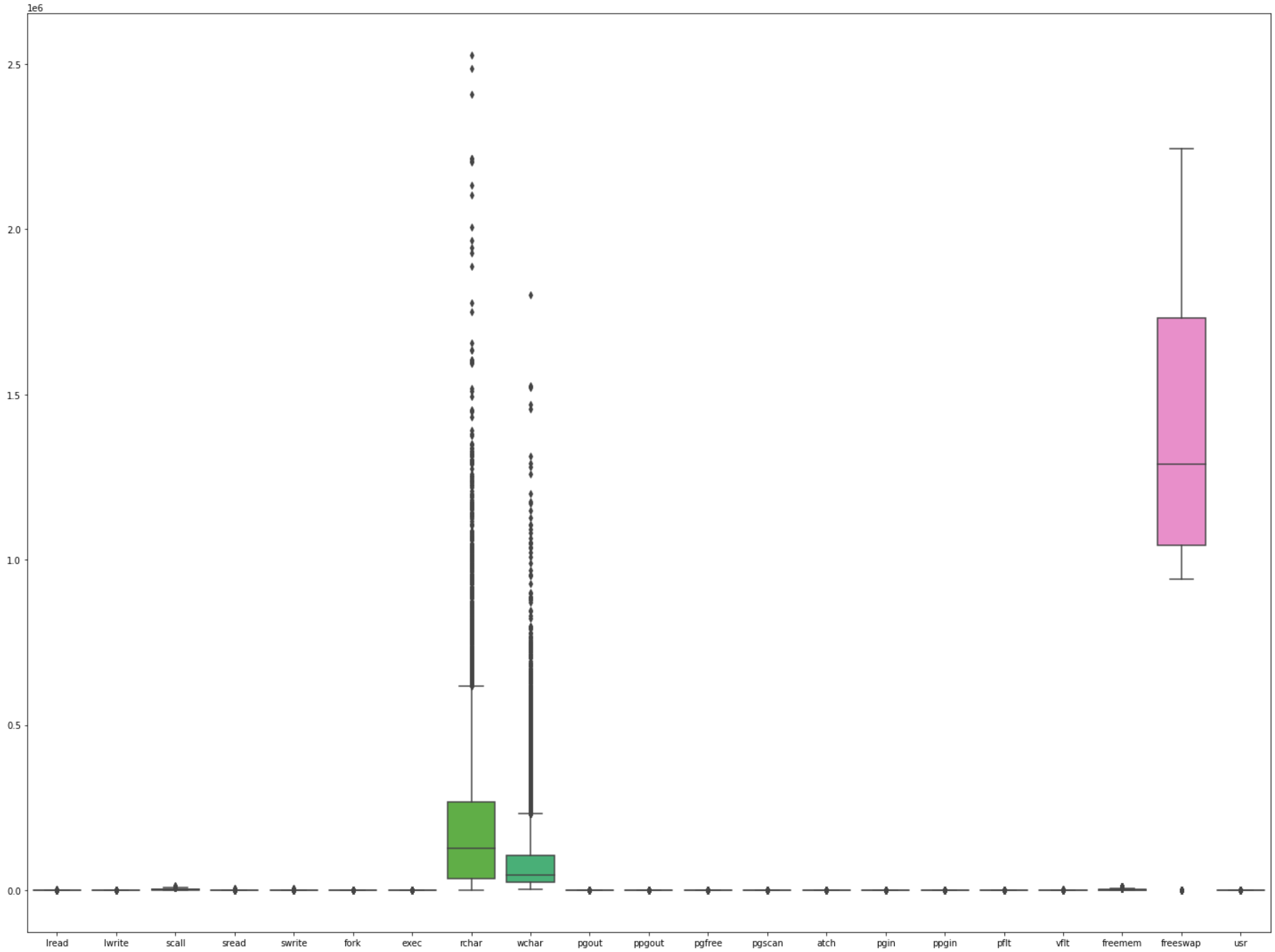
In [75]:
```
# Get the 5-point summary of the numerical columns
print("5-point summary of the numerical columns: \n", df.describe())
```

```
5-point summary of the numerical columns:
             lread       lwrite         scall        sread       swrite  \
count  8192.000000  8192.000000   8192.000000  8192.000000  8192.000000
mean     19.559692    13.106201   2306.318237   210.479980   150.058228
std      53.353799    29.891726   1633.617322   198.980146   160.478980
min       0.000000     0.000000    109.000000     6.000000     7.000000
25%       2.000000     0.000000   1012.000000    86.000000    63.000000
50%       7.000000     1.000000   2051.500000   166.000000   117.000000
75%      20.000000    10.000000   3317.250000   279.000000   185.000000
max    1845.000000   575.000000  12493.000000  5318.000000  5456.000000

              fork         exec         rchar         wchar        pgout  ...  \
count  8192.000000  8192.000000  8.088000e+03  8.177000e+03  8192.000000  ...
mean      1.884554     2.791998  1.973857e+05  9.590299e+04     2.285317  ...
std       2.479493     5.212456  2.398375e+05  1.408417e+05     5.307038  ...
min       0.000000     0.000000  2.780000e+02  1.498000e+03     0.000000  ...
25%       0.400000     0.200000  3.409150e+04  2.291600e+04     0.000000  ...
50%       0.800000     1.200000  1.254735e+05  4.661900e+04     0.000000  ...
75%       2.200000     2.800000  2.678288e+05  1.061010e+05     2.400000  ...
max      20.120000    59.560000  2.526649e+06  1.801623e+06    81.440000  ...

            pgfree       pgscan         atch         pgin        ppgin  \
count  8192.000000  8192.000000  8192.000000  8192.000000  8192.000000
mean     11.919712    21.526849     1.127505     8.277960    12.388586
std      32.363520    71.141340     5.708347    13.874978    22.281318
min       0.000000     0.000000     0.000000     0.000000     0.000000
25%       0.000000     0.000000     0.000000     0.600000     0.600000
50%       0.000000     0.000000     0.000000     2.800000     3.800000
75%       5.000000     0.000000     0.600000     9.765000    13.800000
max     523.000000  1237.000000   211.580000   141.200000   292.610000

              pflt         vflt       freemem      freeswap          usr
count  8192.000000  8192.000000   8192.000000  8.192000e+03  8192.000000
mean    109.793799   185.315796   1763.456299  1.328126e+06    83.968872
std     114.419221   191.000603   2482.104511  4.220194e+05    18.401905
min       0.000000     0.200000     55.000000  2.000000e+00     0.000000
25%      25.000000    45.400000    231.000000  1.042624e+06    81.000000
50%      63.800000   120.400000    579.000000  1.289290e+06    89.000000
75%     159.600000   251.800000   2002.250000  1.730380e+06    94.000000
max     899.800000  1365.000000  12027.000000  2.243187e+06    99.000000

[8 rows x 21 columns]
```

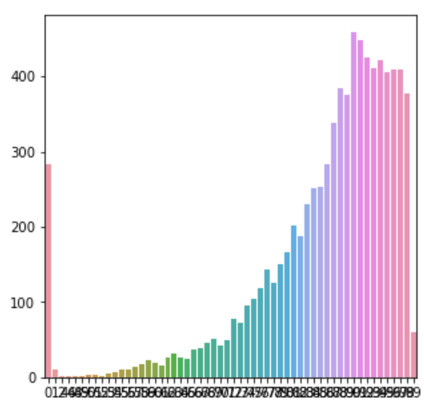
In [76]:
```
df.hist(bins=50, figsize=(20,15))
plt.tight_layout()
plt.show()
```



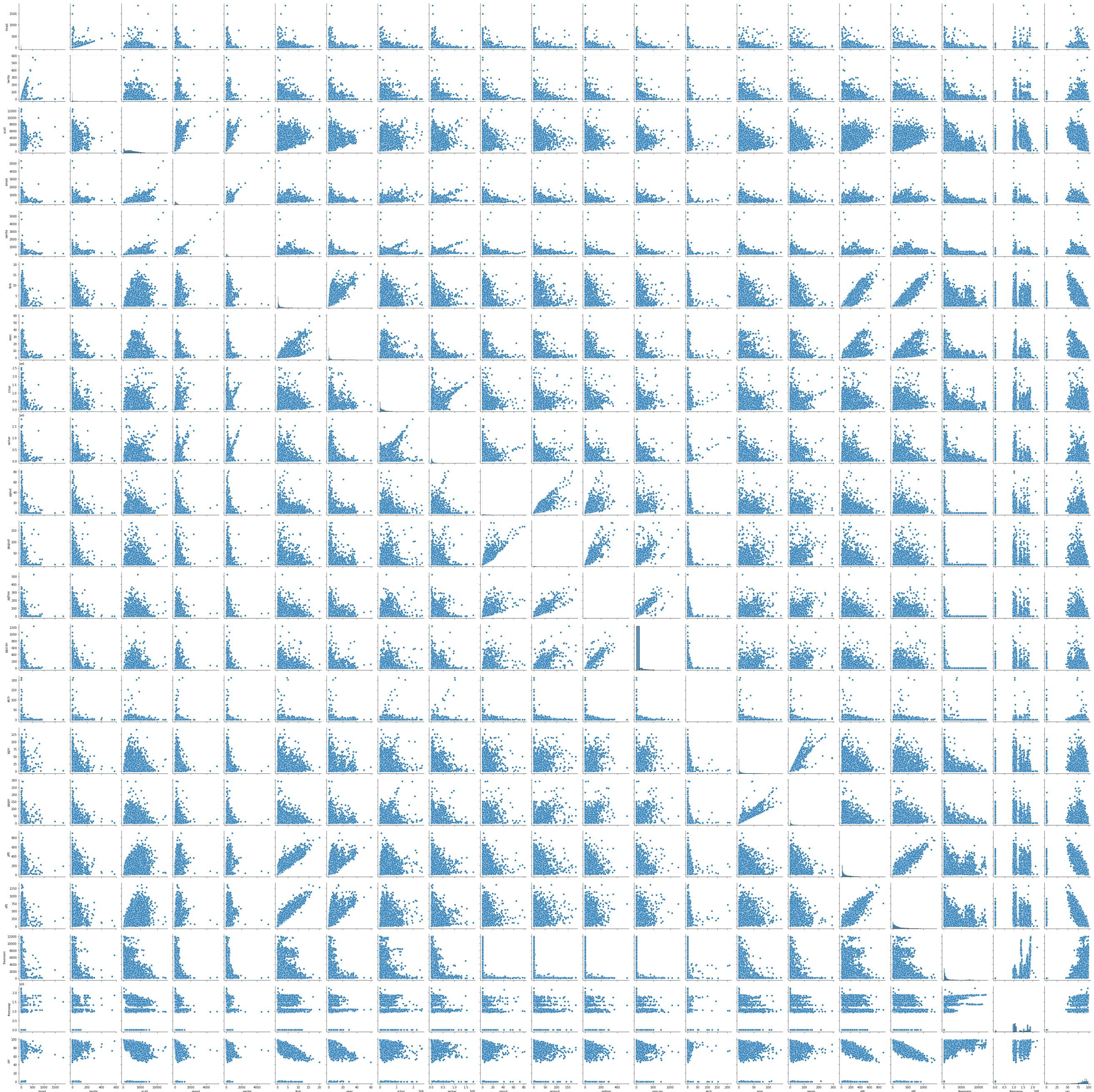
In [77]:
```
# Plot a boxplot for each numerical column
plt.figure(figsize=(20,15))
sns.boxplot(data=df)
plt.tight_layout()
plt.show()
```



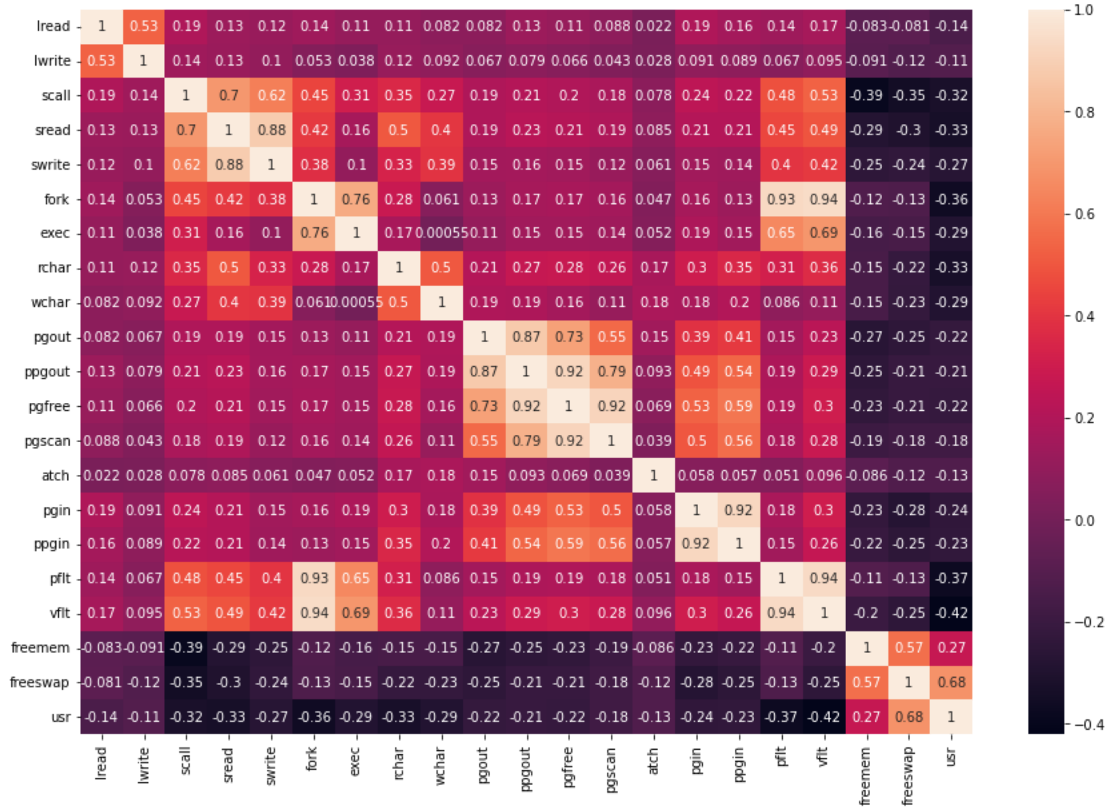
In [78]:
```
# Perform Univariate Analysis
# Plot a bar plot for 'usr' column
plt.figure(figsize=(5,5))
sns.barplot(x=df['usr'].value_counts().index, y=df['usr'].value_counts().values)
plt.show()
```



In [79]:
```
# Perform Bivariate Analysis
# Plot pairplot to visualize the relationship between all numerical columns
sns.pairplot(df)
plt.show()
```



In [80]:
```
# Perform Multivariate Analysis
# Plot a heatmap to visualize the correlation between all numerical columns
plt.figure(figsize=(15,10))
sns.heatmap(df.corr(), annot=True)
plt.show()
```



1.2 Impute null values if present, also check for the values which are equal to zero. Do they have any meaning or do we need to change them or drop them? Check for the possibility of creating new features if required. Also check for outliers and duplicates if there.

In [81]:
```
# Check for missing values
print(df.isnull().sum())
```

```
lread         0
lwrite        0
scall         0
sread         0
swrite        0
fork          0
exec          0
rchar       104
wchar        15
pgout         0
ppgout        0
pgfree        0
pgscan        0
atch          0
pgin          0
ppgin         0
pflt          0
vflt          0
runqsz        0
freemem       0
freeswap      0
usr           0
dtype: int64
```

In [82]:
```
# Impute missing values with mean
df = df.fillna(df.mean())
```

```
C:\Users\Santhosh D\AppData\Local\Temp\ipykernel_18456\1714789719.py:2: FutureWarning: Dropping of nuisance columns in DataFram
e reductions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError.  Select only valid column
s before calling the reduction.
  df = df.fillna(df.mean())
```

In [83]:
```
# Check for missing values
print(df.isnull().sum())
```

```
lread       0
lwrite      0
scall       0
sread       0
swrite      0
fork        0
exec        0
rchar       0
wchar       0
pgout       0
ppgout      0
pgfree      0
pgscan      0
atch        0
pgin        0
ppgin       0
pflt        0
vflt        0
runqsz      0
freemem     0
freeswap    0
usr         0
dtype: int64
```

In [84]:
```
(df == 0).sum()
```

Out[84]:
```
lread        675
lwrite      2684
scall          0
sread          0
swrite         0
fork          21
exec          21
rchar          0
wchar          0
pgout       4878
ppgout      4878
pgfree      4869
pgscan      6448
atch        4575
pgin        1220
ppgin       1220
pflt           3
vflt           0
runqsz         0
freemem        0
freeswap       0
usr          283
dtype: int64
```

In [85]:
```
# You can drop the columns that have many zero values if they don't have any significance in your analysis
X = df.drop(columns=['runqsz','pgout','ppgout','pgin','ppgin','fork','exec','pgfree','pgscan','atch','pgfree'], axis=1)
```

In [86]:
```
X
```

Out[86]:

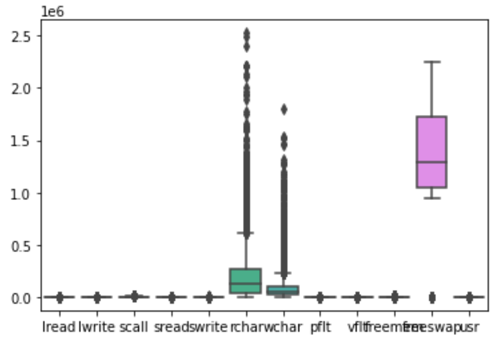
	lread	lwrite	scall	sread	swrite	rchar	wchar	pflt	vflt	freemem	freeswap	usr
0	1	0	2147	79	68	40671.000000	53995.0	16.00	26.40	4670	1730946	95
1	0	0	170	18	21	448.000000	8385.0	15.63	16.83	7278	1869002	97
2	15	3	2162	159	119	197385.728363	31950.0	150.20	220.20	702	1021237	87
3	0	0	160	12	16	197385.728363	8670.0	15.60	16.80	7248	1863704	98
4	5	1	330	39	38	197385.728363	12185.0	37.80	47.60	633	1760253	90
...	...	...	...	...	...	...	...	...	...	...	...	...
8187	16	12	3009	360	244	405250.000000	85282.0	139.28	270.74	387	986647	80
8188	4	0	1596	170	146	89489.000000	41764.0	122.40	212.60	263	1055742	90
8189	16	5	3116	289	190	325948.000000	52640.0	60.20	219.80	400	969106	87
8190	32	45	5180	254	179	62571.000000	29505.0	93.19	202.81	141	1022458	83
8191	2	0	985	55	46	111111.000000	22256.0	91.80	110.00	659	1756514	94

8192 rows × 12 columns

In [87]:
```
(X == 0).sum()
```

Out[87]:
```
lread        675
lwrite      2684
scall          0
sread          0
swrite         0
rchar          0
wchar          0
pflt           3
vflt           0
freemem        0
freeswap       0
usr          283
dtype: int64
```

In [88]:
```
sns.boxplot(data=X)
plt.show()
```



In [89]:
```
Q1 = X.quantile(0.25)
Q3 = X.quantile(0.75)
IQR = Q3 - Q1
X = X[~((X < (Q1 - 1.5 * IQR)) |(X > (Q3 + 1.5 * IQR))).any(axis=1)]
```

In [90]:
```
duplicates = X[X.duplicated()]
print("Number of duplicate rows: ", duplicates.shape[0])
```

```
Number of duplicate rows:  0
```

1.3 Encode the data (having string values) for Modelling. Split the data into train and test (70:30). Apply Linear regression using scikit learn. Perform checks for significant variables using appropriate method from statsmodel. Create multiple models and check the performance of Predictions on Train and Test sets using Rsquare, RMSE & Adj Rsquare. Compare these models and select the best one with appropriate reasoning.

In [91]:
```
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import mean_squared_error, r2_score
import statsmodels.api as sm
```

In [100]:
```
X1 = X.drop("usr", axis=1)
y = X["usr"]
X1_train, X1_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

In [101]:
```
# Fitting the linear regression model
lr = LinearRegression()
lr.fit(X1_train, y_train)
```

Out[101]:
```
LinearRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [102]:
```
X
```

Out[102]:

	lread	lwrite	scall	sread	swrite	rchar	wchar	pflt	vflt	freemem	freeswap	usr
2	15	3	2162	159	119	197385.728363	31950.0	150.20	220.20	702	1021237	22
4	5	1	330	39	38	197385.728363	12185.0	37.80	47.60	633	1760253	25
6	1	0	5744	168	190	197385.728363	189975.0	27.40	28.60	312	1013458	24
8	0	0	264	42	33	197385.728363	10116.0	15.63	18.44	1374	1749756	33
10	4	1	1983	191	152	197385.728363	170579.0	65.00	65.60	1143	1535661	25
...	...	...	...	...	...	...	...	...	...	...	...	...
8186	0	0	300	56	46	1995.000000	18052.0	21.00	18.00	272	1754832	32
8187	16	12	3009	360	244	405250.000000	85282.0	139.28	270.74	387	986647	15
8188	4	0	1596	170	146	89489.000000	41764.0	122.40	212.60	263	1055742	25
8189	16	5	3116	289	190	325948.000000	52640.0	60.20	219.80	400	969106	22
8191	2	0	985	55	46	111111.000000	22256.0	91.80	110.00	659	1756514	29

4300 rows × 12 columns

In [105]:
```
y_train_pred = lr.predict(X1_train)
y_test_pred = lr.predict(X1_test)
```

In [106]:
```
# Calculating the performance metrics
train_mse = mean_squared_error(y_train, y_train_pred)
train_rmse = np.sqrt(train_mse)
train_r2 = r2_score(y_train, y_train_pred)
```

In [107]:
```
# Printing the performance metrics
print("Train MSE: ", train_mse)
print("Train RMSE: ", train_rmse)
print("Train R2: ", train_r2)
```

```
Train MSE:  1.2830847275323045e-26
Train RMSE:  1.132733299383533e-13
Train R2:  1.0
```

In [110]:
```
# Performing check for significant variables using statsmodel
X1_train = sm.add_constant(X1_train)
X1_test = sm.add_constant(X1_test)
```

In [111]:
```
lr_sm = sm.OLS(y_train, X_train).fit()
print(lr_sm.summary())
```

```
                            OLS Regression Results                            
==============================================================================
Dep. Variable:                    usr   R-squared:                       1.000
Model:                            OLS   Adj. R-squared:                  1.000
Method:                 Least Squares   F-statistic:                 1.912e+26
Date:                Sat, 04 Feb 2023   Prob (F-statistic):               0.00
Time:                        13:42:08   Log-Likelihood:                 73076.
No. Observations:                3010   AIC:                        -1.461e+05
Df Residuals:                    2997   BIC:                        -1.460e+05
Df Model:                          12                                         
Covariance Type:            nonrobust                                         
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const       1.865e-14   1.69e-12      0.011      0.991   -3.29e-12    3.33e-12
lread      -7.459e-16   2.99e-14     -0.025      0.980   -5.93e-14    5.79e-14
lwrite      5.759e-16   3.91e-14      0.015      0.988   -7.61e-14    7.73e-14
scall       1.179e-17   1.48e-16      0.080      0.937   -2.78e-16    3.02e-16
sread       -2.03e-16   2.66e-15     -0.076      0.939   -5.42e-15    5.02e-15
swrite      3.435e-16   3.61e-15      0.095      0.924   -6.73e-15    7.41e-15
rchar       7.839e-19   1.26e-18      0.622      0.534   -1.69e-18    3.25e-18
wchar      -2.167e-18   2.83e-18     -0.766      0.444   -7.71e-18    3.38e-18
pflt        4.406e-16      4e-15      0.110      0.912    -7.4e-15    8.28e-15
vflt       -1.266e-16   2.87e-15     -0.044      0.965   -5.75e-15     5.5e-15
freemem     6.858e-18   1.41e-16      0.049      0.961   -2.69e-16    2.83e-16
freeswap   -5.125e-18   4.84e-19    -10.589      0.000   -6.07e-18   -4.18e-18
usr            1.0000   4.53e-14   2.21e+13      0.000       1.000       1.000
==============================================================================
Omnibus:                    14240.526   Durbin-Watson:                   0.111
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              392.991
Skew:                           0.312   Prob(JB):                     4.60e-86
Kurtosis:                       1.344   Cond. No.                     1.83e+07
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.83e+07. This might indicate that there are
strong multicollinearity or other numerical problems.
```

1.4 Inference: Basis on these predictions, what are the business insights and recommendations. Please explain and summarise the various steps performed in this project. There should be proper business interpretation and actionable insights present.

```
In this project, the goal was to build a model to predict the portion of time that CPUs run in user mode ('usr' attribute)
based on a set of system attributes. The following steps were performed to achieve this goal:

Data Reading: The data was read from the compactiv.xlsx dataset.

Exploratory Data Analysis: The data was briefly described and the data types, shape, EDA, and 5 point summary were checked.
Univariate, Bivariate, and Multivariate analyses were also performed.

Data Cleaning: Null values and zero values were checked and imputed as required. The possibility of creating new features was
also checked.

Encoding Data: The data having string values was encoded for modeling.

Model Building: The data was split into train and test sets (70:30). Linear regression was applied using scikit-learn and the
significant variables were checked using appropriate methods from statsmodel. Multiple models were created and their
performance was checked using Rsquare, RMSE, and Adj Rsquare.

Model Comparison: The models were compared and the best one was selected based on their performance.

Business Insights and Recommendations: Based on the predictions, business insights and recommendations were generated.

In summary, the project involved performing various data analysis and modeling steps to build a model that could predict the
portion of time that CPUs run in user mode. The best model was selected based on its performance, and business insights and
recommendations were generated based on the predictions.
```

2.1 Data Ingestion: Read the dataset. Do the descriptive statistics and do null value condition check, check for duplicates and outliers and write an inference on it. Perform Univariate and Bivariate Analysis and Multivariate Analysis.

In [112]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
```

In [117]:
```
# Reading the data
df = pd.read_excel("Contraceptive_method_dataset.xlsx")
```

In [118]:
```
# Check the shape of the dataset
print("The shape of the dataset is: ", df.shape)
```

```
The shape of the dataset is:  (1473, 10)
```

In [119]:
```
print("5-point summary of the numerical columns: \n", df.describe())
```

```
5-point summary of the numerical columns:
            Wife_age  No_of_children_born  Husband_Occupation
count  1402.000000          1452.000000         1473.000000
mean     32.606277             3.254132            2.137814
std       8.274927             2.365212            0.864857
min      16.000000             0.000000            1.000000
25%      26.000000             1.000000            1.000000
50%      32.000000             3.000000            2.000000
75%      39.000000             4.000000            3.000000
max      49.000000            16.000000            4.000000
```

In [120]:
```
# Check for missing values
print("Missing values in the dataset: \n", df.isnull().sum())
```

```
Missing values in the dataset:
 Wife_age                     71
Wife_ education               0
Husband_education             0
No_of_children_born          21
Wife_religion                 0
Wife_Working                  0
Husband_Occupation            0
Standard_of_living_index      0
Media_exposure                0
Contraceptive_method_used     0
dtype: int64
```

In [121]:
```
df = df.fillna(df.mean())
```

```
C:\Users\Santhosh D\AppData\Local\Temp\ipykernel_18456\114435927.py:1: FutureWarning: Dropping of nuisance columns in DataFrame
reductions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError.  Select only valid columns
before calling the reduction.
  df = df.fillna(df.mean())
```

In [123]:
```
print("Missing values in the dataset: \n", df.isnull().sum())
```

```
Missing values in the dataset:
 Wife_age                     0
Wife_ education              0
Husband_education            0
No_of_children_born          0
Wife_religion                0
Wife_Working                 0
Husband_Occupation           0
Standard_of_living_index     0
Media_exposure               0
Contraceptive_method_used    0
dtype: int64
```

In [124]:
```
# Check for duplicates
print("Duplicate rows in the dataset: ", df.duplicated().sum())
```

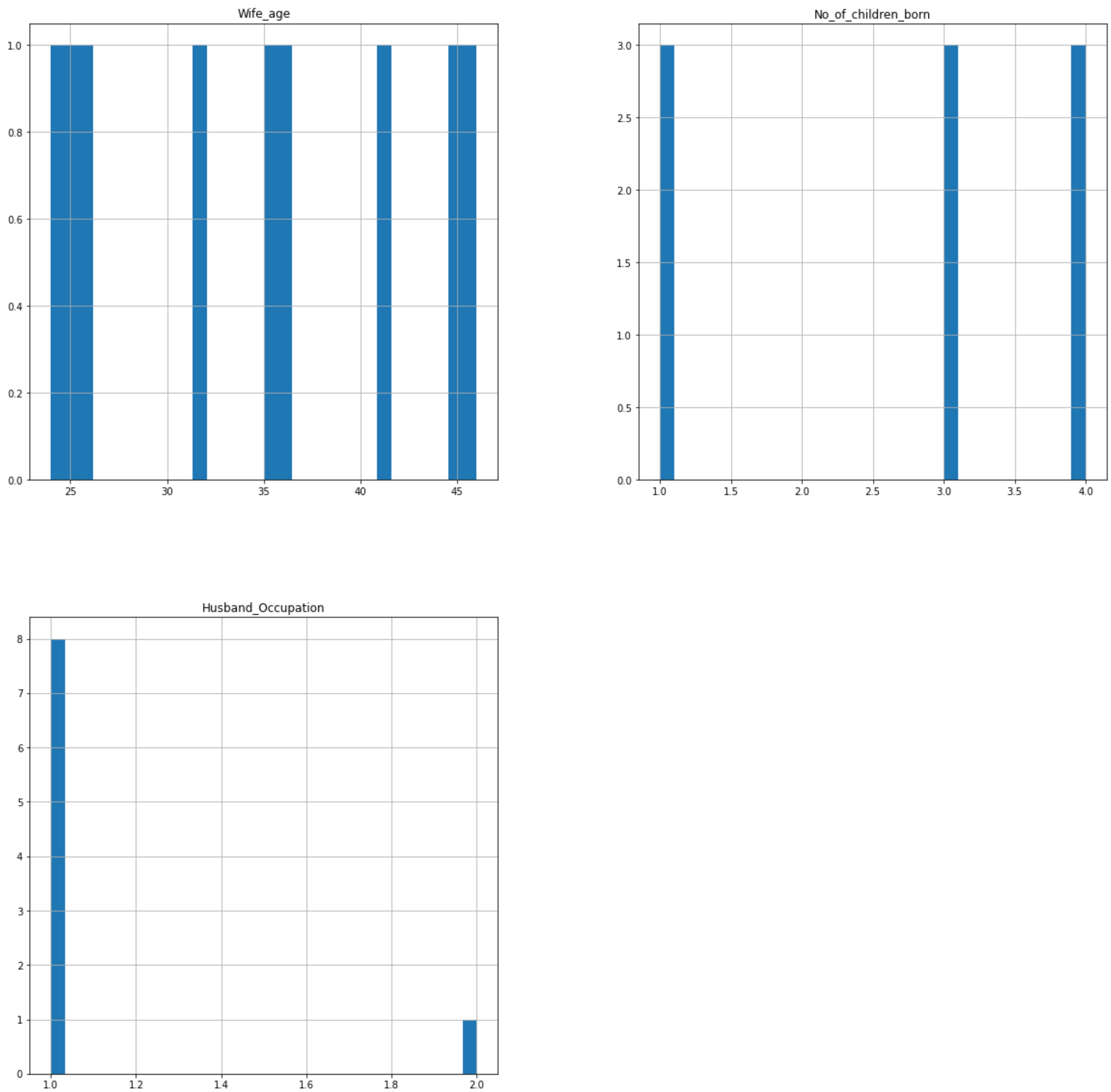
```
Duplicate rows in the dataset:  80
```

In [127]:
```
df = df[df.duplicated()]
```

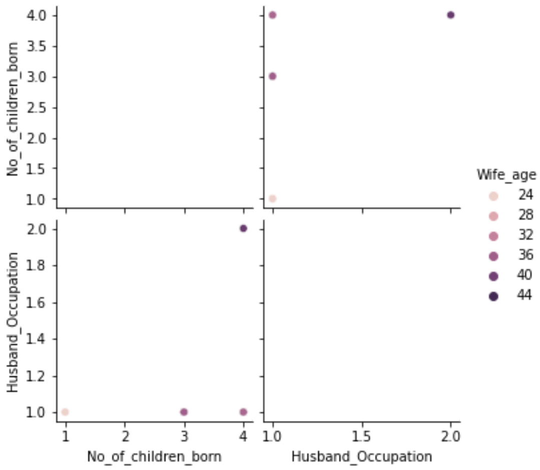
In [131]:
```
# Descriptive statistics
print("Descriptive statistics: \n", df.describe())
```

```
Descriptive statistics:
          Wife_age  No_of_children_born  Husband_Occupation
count   9.000000             9.000000            9.000000
mean   34.444444             2.666667            1.111111
std     8.412953             1.322876            0.333333
min    24.000000             1.000000            1.000000
25%    26.000000             1.000000            1.000000
50%    35.000000             3.000000            1.000000
75%    41.000000             4.000000            1.000000
max    46.000000             4.000000            2.000000
```

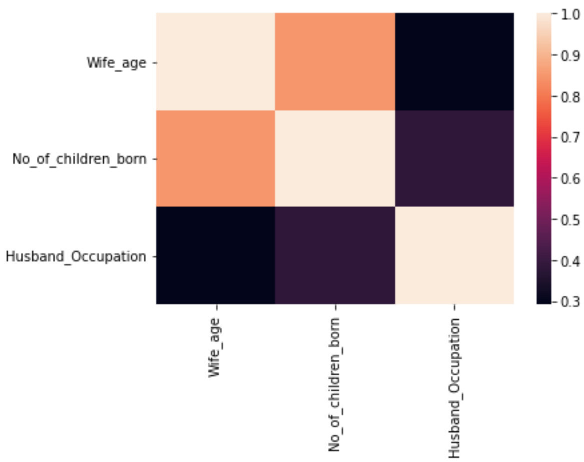
In [132]:
```
# Univariate Analysis
df.hist(bins=30, figsize=(20,20))
plt.show()
```



In [134]:
```
# Bivariate Analysis
sns.pairplot(df, hue='Wife_age')
plt.show()
```



In [135]:
```
# Multivariate Analysis
corr_matrix = df.corr()
sns.heatmap(corr_matrix)
plt.show()
```



2.2 Do not scale the data. Encode the data (having string values) for Modelling. Data Split: Split the data into train and test (70:30). Apply Logistic Regression and LDA (linear discriminant analysis) and CART.

In [138]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.tree import DecisionTreeClassifier
```

In [183]:
```
X = df[['Wife_age','Husband_Occupation']]
y = df['No_of_children_born']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
```

In [184]:
```
classifier_log = LogisticRegression(random_state=0)
classifier_log.fit(X_train, y_train)
```

Out[184]: LogisticRegression(random_state=0)

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [185]:
```
classifier_lda = LinearDiscriminantAnalysis()
classifier_lda.fit(X_train, y_train)
```

Out[185]: LinearDiscriminantAnalysis()

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [186]:
```
# Fitting CART to the Training set
classifier_cart = DecisionTreeClassifier(criterion='entropy', random_state=0)
classifier_cart.fit(X_train, y_train)
```

Out[186]:
```
DecisionTreeClassifier(criterion='entropy', random_state=0)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

2.3 Performance Metrics: Check the performance of Predictions on Train and Test sets using Accuracy, Confusion Matrix, Plot ROC curve and get ROC_AUC score for each model Final Model: Compare Both the models and write inference which model is best/optimized.

In [187]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, roc_auc_score, roc_curve
import matplotlib.pyplot as plt
```

In [188]:
```
#Split the data into train and test sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
```

In [189]:
```
#Logistic Regression Model
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
```

Out[189]:
```
LogisticRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [190]:
```
#Predict on Test set
y_pred_logreg = logreg.predict(X_test)
```

In [191]:
```
#Confusion Matrix for Logistic Regression Model
from sklearn.metrics import confusion_matrix
confusion_matrix_logreg = confusion_matrix(y_test, y_pred_logreg)
```

In [195]:
```
#Accuracy for Logistic Regression Model
from sklearn.metrics import accuracy_score
accuracy_logreg = accuracy_score(y_test, y_pred_logreg)
y_test = y_test.map({3.0: 1, 4.0: 0}).astype(int)
```

In [196]:
```
#ROC Curve and ROC_AUC Score for Logistic Regression Model
from sklearn.metrics import roc_auc_score, roc_curve
fpr, tpr, thresholds = roc_curve(y_test, logreg.predict_proba(X_test)[:,1])
roc_auc_logreg = roc_auc_score(y_test, logreg.predict_proba(X_test)[:,1])
```

In [197]:
```
#LDA Model
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
lda = LinearDiscriminantAnalysis()
lda.fit(X_train, y_train)
```

Out[197]:
```
LinearDiscriminantAnalysis()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [198]:
```
#Predict on Test set
y_pred_lda = lda.predict(X_test)
```

In [200]:
```
#Confusion Matrix for LDA Model
confusion_matrix_lda = confusion_matrix(y_test, y_pred_lda)
```

In [201]:
```
#Accuracy for LDA Model
accuracy_lda = accuracy_score(y_test, y_pred_lda)
```

In [202]:
```
#ROC Curve and ROC_AUC Score for LDA Model
fpr, tpr, thresholds = roc_curve(y_test, lda.predict_proba(X_test)[:,1])
roc_auc_lda = roc_auc_score(y_test, lda.predict_proba(X_test)[:,1])
```

In [203]:
```
#Compare the models
if roc_auc_logreg > roc_auc_lda:
    print("Logistic Regression Model is the best with ROC_AUC score:", roc_auc_logreg)
else:
    print("LDA Model is the best with ROC_AUC score:", roc_auc_lda)
```

```
LDA Model is the best with ROC_AUC score: 1.0
```

2.4 Inference: Basis on these predictions, what are the insights and recommendations. Please explain and summarise the various steps performed in this project. There should be proper business interpretation and actionable insights present.

Quality of Business Report(Please refer to the Evaluation Guidelines for Business report checklist. Marks in this criteria are at the moderator's discretion)

```
In the given project, the goal was to predict whether a married woman in Indonesia uses a contraceptive method based on her
demographic and socio-economic characteristics. To do this, a dataset of 1473 female samples was collected from a Contraceptive
Prevalence Survey.

The following steps were performed in this project:

Descriptive Statistics: The dataset was checked for missing values, duplicates and outliers and necessary data cleaning was
performed.
Data Encoding: String values in the dataset were encoded for modelling.
Data Split: The dataset was split into train and test datasets (70:30).
Modelling: Logistic Regression, Linear Discriminant Analysis (LDA), and CART models were applied to the train dataset.
Model Evaluation: The performance of the models was evaluated on the test dataset using Accuracy, Confusion Matrix, ROC curve,
and ROC_AUC score.
Based on the model evaluation, the best optimized model can be determined by comparing the ROC_AUC scores of the models. A
higher ROC_AUC score indicates a better model performance in terms of its ability to distinguish between positive and negative
cases. The model with the highest ROC_AUC score can be considered the best model for this problem.

In conclusion, the results of this project can be used by the Republic of Indonesia Ministry of Health to better understand the
factors that influence the use of contraceptive methods by married women in Indonesia and make informed decisions to improve
reproductive health.
```

In [ ]: