Modélisation SARIMA et Évaluation des Prévisions de Production Laitière :

Par : `Mourad Bahaddou`

1. Importer les packages

In [4]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [5]:
```
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
```

2. Charger les données

In [6]:
```
df = pd.read_csv("./monthly-milk-production (1).csv")
df
```

Out[6]:

	Month	Milk Production
0	1962-01-01	589.0
1	1962-02-01	561.0
2	1962-03-01	640.0
3	1962-04-01	656.0
4	1962-05-01	727.0
...	...	...
163	1975-08-01	858.0
164	1975-09-01	817.0
165	1975-10-01	827.0
166	1975-11-01	797.0
167	1975-12-01	843.0

168 rows × 2 columns

In [7]:
```
## dimention de la data
df.shape
```

Out[7]:
```
(168, 2)
```

In [8]:
```
## chercher les valeurs nulles
df.isnull().sum()
```

Out[8]:
```
Month             0
Milk Production    0
dtype: int64
```

In [9]:
```
df.describe
```

Out[9]:
```
<bound method NDFrame.describe of           Month  Milk Production
0    1962-01-01             589.0
1    1962-02-01             561.0
2    1962-03-01             640.0
3    1962-04-01             656.0
4    1962-05-01             727.0
..          ...               ...
163  1975-08-01             858.0
164  1975-09-01             817.0
165  1975-10-01             827.0
166  1975-11-01             797.0
167  1975-12-01             843.0

[168 rows x 2 columns]>
```

3. Division les données

Diviser les données en un échantillon d'apprentissage et un échantillon de validation est une étape importante qui permet :

la construction du modèle sur train .

faire des prévisions sur les dates de test.

comparer les prévisions au df_test['Milk Production'] pour mesurer la précision (ex. MSE, RMSE,...).

In [10]:
```
df_train=df[df['Month'] < '1974-01-01']
df_test = df[df['Month']>= '1974-01-01']
```

Echantillon d'apprentissage:

In [11]:
```
df_test
```

Out[11]:

	Month	Milk Production
144	1974-01-01	828.0
145	1974-02-01	778.0
146	1974-03-01	889.0
147	1974-04-01	902.0
148	1974-05-01	969.0
149	1974-06-01	947.0
150	1974-07-01	908.0
151	1974-08-01	867.0
152	1974-09-01	815.0
153	1974-10-01	812.0
154	1974-11-01	773.0
155	1974-12-01	813.0
156	1975-01-01	834.0
157	1975-02-01	782.0
158	1975-03-01	892.0
159	1975-04-01	903.0
160	1975-05-01	966.0
161	1975-06-01	937.0
162	1975-07-01	896.0
163	1975-08-01	858.0
164	1975-09-01	817.0
165	1975-10-01	827.0
166	1975-11-01	797.0
167	1975-12-01	843.0

Echantillon de validation :

In [12]:
```
df_train
```

Out[12]:

	Month	Milk Production
0	1962-01-01	589.0
1	1962-02-01	561.0
2	1962-03-01	640.0
3	1962-04-01	656.0
4	1962-05-01	727.0
...	...	...
139	1973-08-01	837.0
140	1973-09-01	784.0
141	1973-10-01	791.0
142	1973-11-01	760.0
143	1973-12-01	802.0

144 rows × 2 columns

4. Création et visualisation de la série temporelle

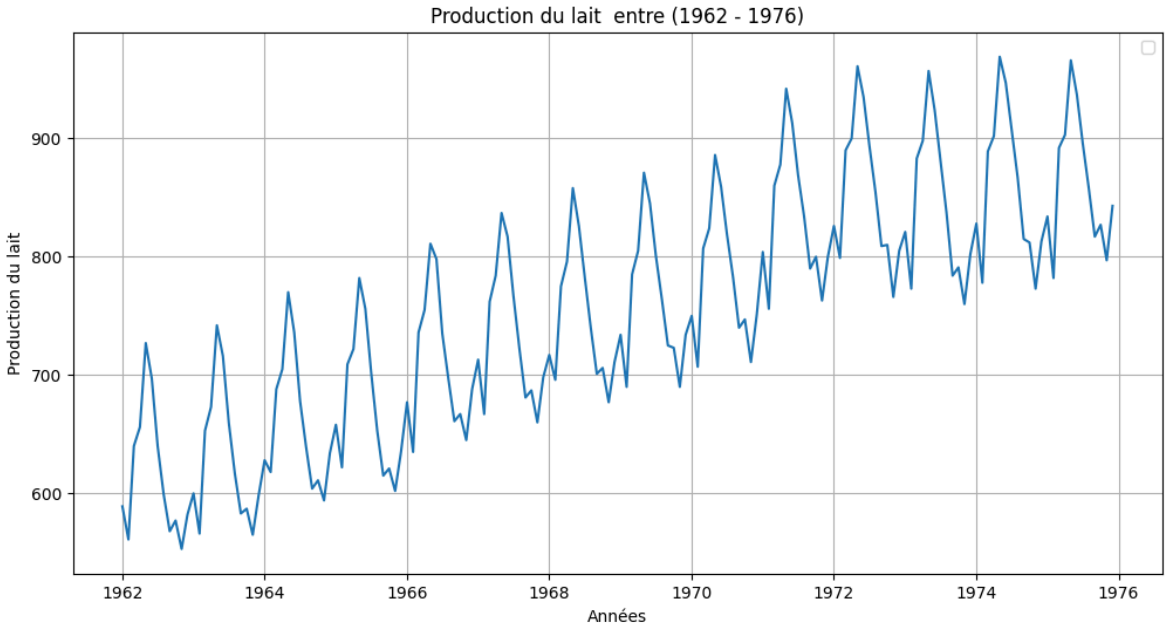
On commence par Créer un objet de type série temporelle

In [13]:
```
df['Month'] = pd.to_datetime(df['Month'])
serie_temporelle=df['Milk Production']
```

on visualise la série temporelle temporelle :

In [14]:
```
plt.figure(figsize=(12, 6))
plt.plot(df['Month'],serie_temporelle)
plt.title("Production du lait  entre (1962 - 1976)")
plt.xlabel("Années")
plt.ylabel("Production du lait")
plt.legend()
plt.grid(True)
plt.show()
```

```
C:\Users\lenovo\AppData\Local\Temp\ipykernel_12668\3294993755.py:6: UserWarning:
No artists with labels found to put in legend.  Note that artists whose label sta
rt with an underscore are ignored when legend() is called with no argument.
  plt.legend()
```



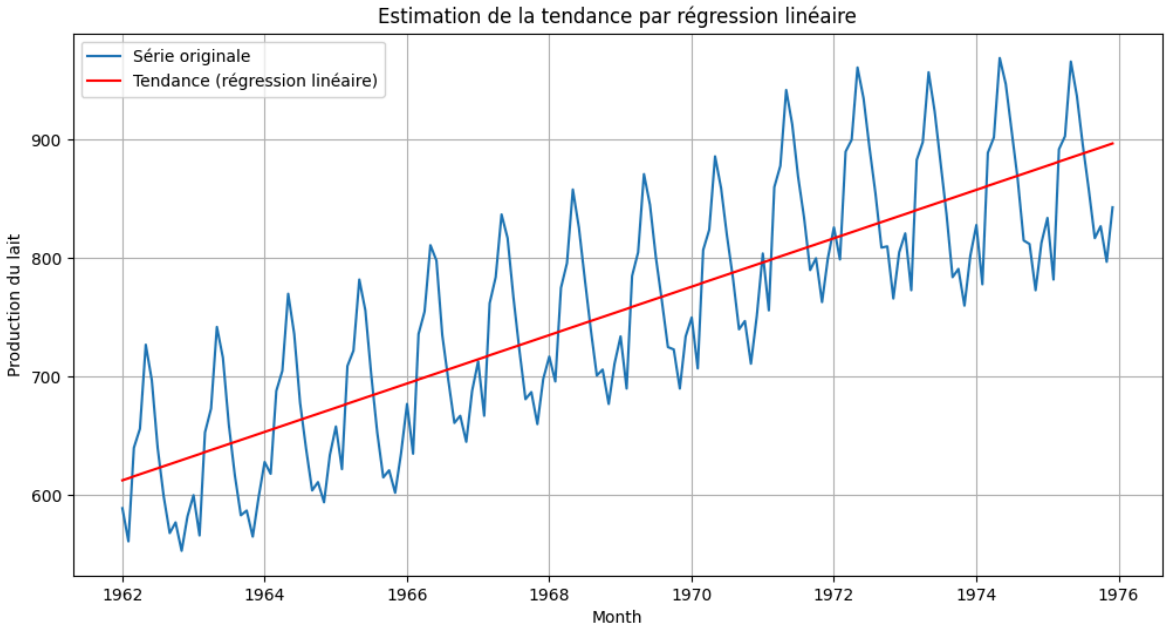
Calcule de la tendance en utilisant la regression linéaire

In [15]:
```
t=np.array(list(ti for ti in range(1,len(df)+1))) ##liste des mois
Xt=np.array(list(xi for xi in df['Milk Production']))## liste de production du l
a=((np.cov(t,Xt))[0,1])/np.var(t)
b=np.mean(Xt)-a*np.mean(t)
print(f"Équation de la tendance : Tt = {a:.2f} * t + {b:.2f}")
```

```
Équation de la tendance : Tt = 1.70 * t + 610.83
```

Visualisation de la tendance

In [16]:
```
Tt=a*t+b
plt.figure(figsize=(12, 6))
plt.plot(df['Month'],df['Milk Production'],label="Série originale")
plt.plot(df['Month'],Tt, color='red', label="Tendance (régression linéaire)")
plt.title("Estimation de la tendance par régression linéaire")
plt.xlabel("Month")
plt.ylabel("Production du lait")
plt.legend()
plt.grid(True)
plt.show()
```



Calculation des coefficients saisonniers

In [17]:
```
L=Xt/Tt
print(L)
```

```
[0.96158766 0.91333656 1.0390721  1.06211269 1.17383085 1.12230661
 1.0277077  0.95924734 0.90712992 0.91900432 0.87839661 0.92196718
 0.94792471 0.89180989 1.02613717 1.05474335 1.15978697 1.11617683
 1.02615423 0.95676579 0.90166212 0.90546398 0.86924525 0.9176115
 0.96113425 0.94337114 1.04750277 1.07061028 1.16630321 1.11193629
 1.02168261 0.96044886 0.90552459 0.91368663 0.88600888 0.94327702
 0.97651079 0.92075793 1.04690677 1.06342877 1.14892089 1.10794972
 1.0262495  0.95224633 0.89461093 0.90110688 0.87138378 0.9168908
 0.97513803 0.91240425 1.05494578 1.07954466 1.15680039 1.13549946
 1.0433269  0.98700056 0.93377054 0.93998546 0.90680544 0.96494914
 0.99763018 0.93104873 1.06113474 1.0891885  1.16007561 1.12968969
 1.05806209 0.99365144 0.93503408 0.94107211 0.90198289 0.95170058
 0.97534207 0.94458765 1.04937875 1.07533425 1.15643132 1.11075184
 1.05052276 0.99056824 0.93622866 0.94076705 0.90008138 0.94314975
 0.97146525 0.91117682 1.03430301 1.05828041 1.1424888  1.1059146
 1.0459975  0.99546703 0.94256011 0.93788373 0.89310305 0.94796528
 0.96650392 0.90909621 1.03541422 1.05492124 1.13182906 1.09495592
 1.04170745 0.99376584 0.93716591 0.94399533 0.89657234 0.94498338
 1.00951034 0.94721592 1.07522684 1.09539961 1.17275514 1.13424677
 1.07730537 1.03173771 0.97525118 0.98552452 0.93797648 0.98140726
 1.01119076 0.97610261 1.08501622 1.09493449 1.16672977 1.13282187
 1.08091739 1.03163937 0.97413458 0.97334303 0.91859053 0.9633923
 0.98054234 0.92134097 1.05031866 1.06600191 1.13374824 1.09244975
 1.03951791 0.98562078 0.92136248 0.92773246 0.88959719 0.93689181
 0.96534466 0.90525379 1.03236416 1.04539349 1.12083277 1.09323241
 1.04615376 0.99695968 0.93533372 0.93007327 0.88367878 0.92760039
 0.94971547 0.88877725 1.01183891 1.02234208 1.09156396 1.05676115
 1.00858393 0.96396154 0.91614549 0.92559171 0.8903185  0.93991668]
```

In [18]:
```
St = []
for i in range(len(L) // 12):
    H = []
    for j in range(12):
        idx = 12 * j + i
        if idx < len(L):
```

```
            H.append(L[idx] / 12)
    St.append(sum(H))
print(St)
```

```
[np.float64(0.9778733584675059), np.float64(0.9268540569505213), np.float64(1.050
446419062088), np.float64(1.0720416791048333), np.float64(1.1558083551073621), n
p.float64(1.1162249467603842), np.float64(1.04492928388969), np.float64(0.9865132
742200691), np.float64(0.930369590737721), np.float64(0.9357971256421435), np.flo
at64(0.8958119851825939), np.float64(0.9445155005055221), np.float64(0.9781864412
843021), np.float64(0.9261804931197379)]
```

5.Pridictions:Production du lait entre 1976-1977

In [19]:
```
t=np.array(list(ti for ti in range(len(df)+1,len(df)+15)))
Xt_predicted=[]
T=[]
S=[]
for j in t:
    T.append(a * j + b)
for i in range(12):
    S.append(St[i])
for i in range(12):
    Xt_pred = round(T[i]*S[i])
    Xt_predicted.append(Xt_pred)
print(Xt_predicted)
```

```
[879, 834, 947, 969, 1046, 1013, 950, 898, 849, 855, 820, 866]
```

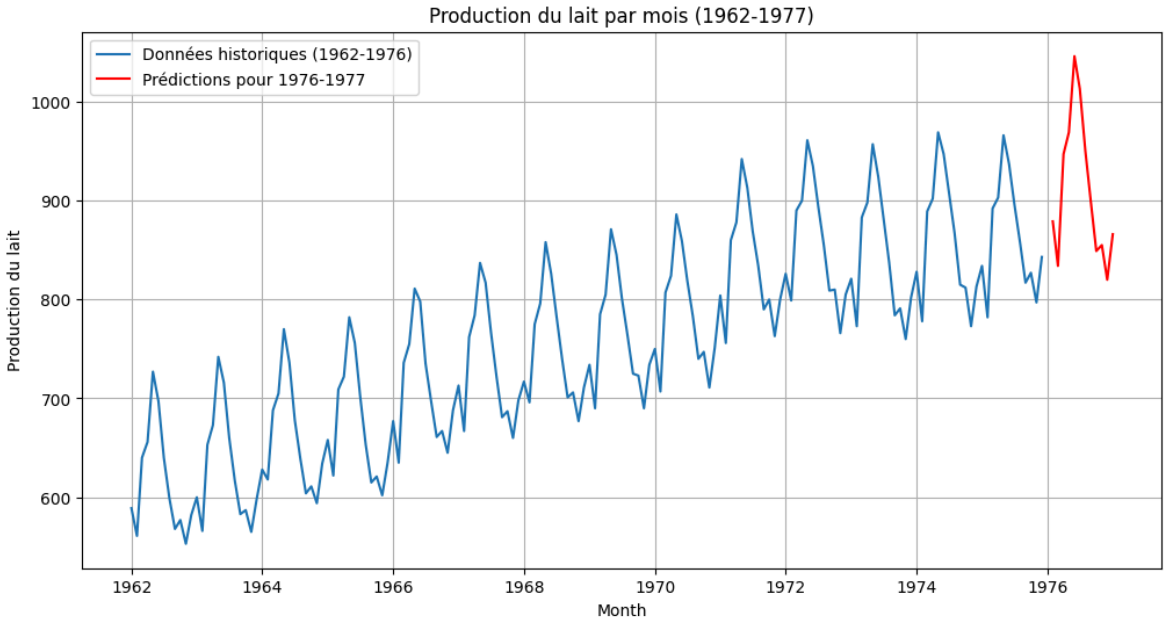
In [20]:
```
plt.figure(figsize=(12, 6))
plt.plot(df['Month'], df['Milk Production'], label='Données historiques (1962-19


mois_1961 = pd.date_range(start='1976-01-01', end='1976-12-31', freq='M')


plt.plot(mois_1961, Xt_predicted, color='red', label='Prédictions pour 1976-1977

plt.title("Production du lait par mois (1962-1977)")
plt.xlabel("Month")
plt.ylabel("Production du lait")
plt.legend()
plt.grid(True)
plt.show()
```

```
C:\Users\lenovo\AppData\Local\Temp\ipykernel_12668\1329085697.py:5: FutureWarnin
g: 'M' is deprecated and will be removed in a future version, please use 'ME' ins
tead.
  mois_1961 = pd.date_range(start='1976-01-01', end='1976-12-31', freq='M')
```



Moyenne des coefficients saisonniers

In [21]:
```
s=0
for x in St:
    s+=x
St_c=s/12
print(St_c)
```

```
1.1617960425028728
```

Correction des coefficients saisonniers

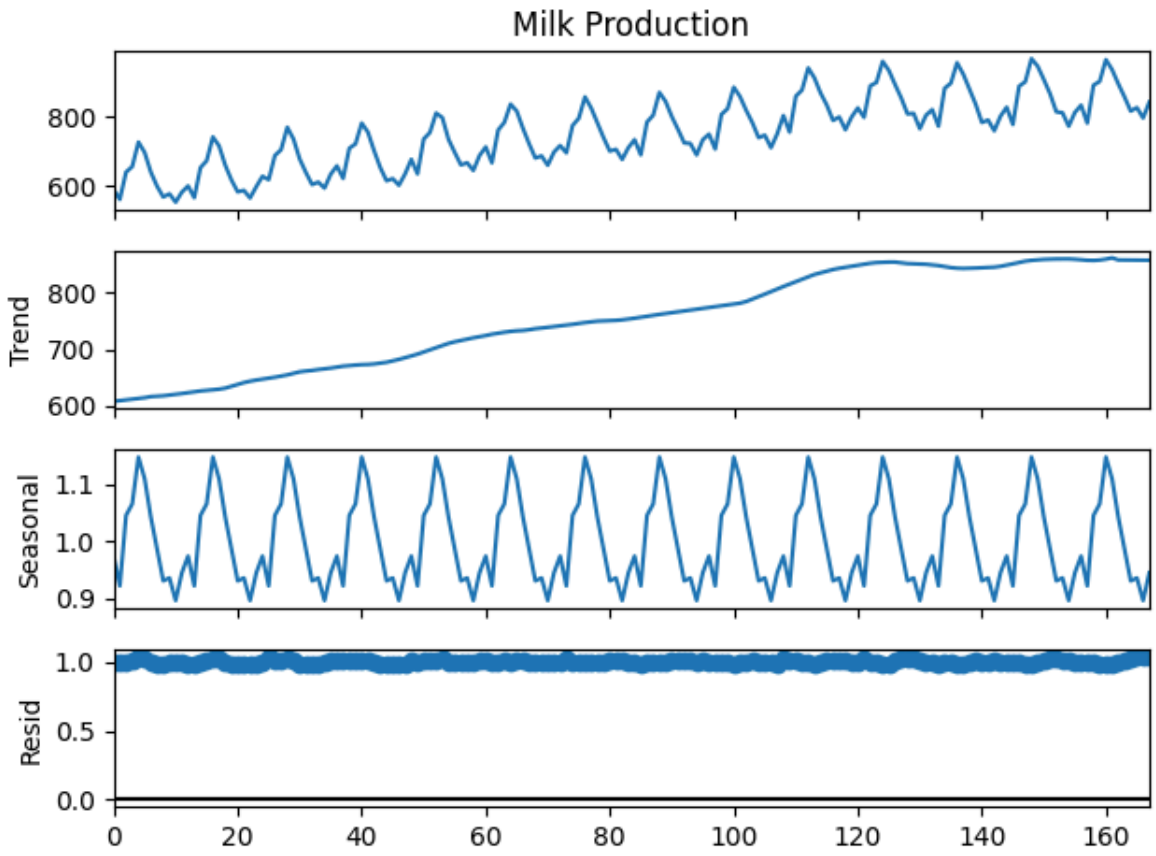
In [22]:
```
St_corrected=[]
for x in St:
    St_corrected.append(x/St_c)
print(St_corrected)
```

```
[np.float64(0.8416910737282769), np.float64(0.7977769100967044), np.float64(0.904
157339698883), np.float64(0.9227451634241407), np.float64(0.9948461802446742), n
p.float64(0.9607753047218905), np.float64(0.8994085413121091), np.float64(0.84912
77626448187), np.float64(0.8008028575595879), np.float64(0.8054745337453062), np.
float64(0.7710578728196853), np.float64(0.8129787552647707), np.float64(0.8419605
554663294), np.float64(0.7971971492728231)]
```

6. Analyser qualitative de la série.

In [23]:
```
# Décomposition multiplicative
result_mul = seasonal_decompose(df['Milk Production'], model='multiplicative', p

# Affichage des composantes
result_mul.plot()
plt.show()
```



analyse :

1)Série originale ("Milk Production") On observe une tendance globale à la hausse : la production de lait augmente entre le début et la fin de la période.

Il y a des fluctuations périodiques régulières : des pics et des creux qui reviennent chaque année.

2)Tendance (Trend) La courbe est croissante sur la première partie de la période (1962–1974).

Puis elle semble se stabiliser vers la fin (1975–1976).

Cela signifie que la production de lait augmente au fil du temps, mais que la croissance ralentit.

3)Saisonnalité (Seasonal) Le graphique présente une saisonnalité claire et stable : un motif répété tous les 12 mois.

Chaque année, il y a des pics de production à certaines périodes (probablement en été) et des creux à d'autres (probablement en hiver).

4)Résidus (Residus)

Les résidus sont faibles et proches de 1, ce qui signifie que la majorité de la variance est bien expliquée par la tendance + saisonnalité.

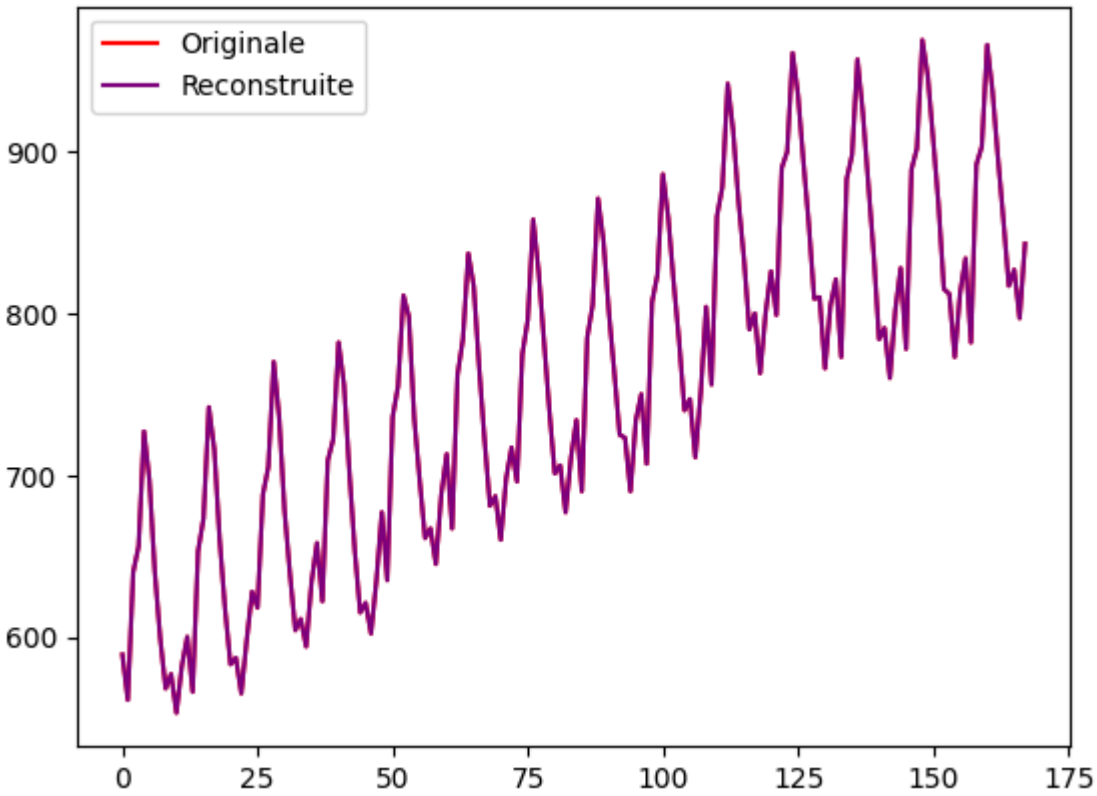
Pas de structure visible : donc le modèle de décomposition semble bien fonctionner.

7. Le modèle adéquat :

l'objectif est de tester pour savoir quel est le modèle adéquat pour la décomposition de notre série

In [24]:
```
trend = result_mul.trend
seasonal = result_mul.seasonal
residual = result_mul.resid
```

In [35]:
```
reconstructed = result_mul.trend * result_mul.seasonal * result_mul.resid
df['Milk Production'].plot(color='red',label='Originale')
reconstructed.plot(color='purple',label='Reconstruite')
plt.legend()
plt.show()
```



Déduction :

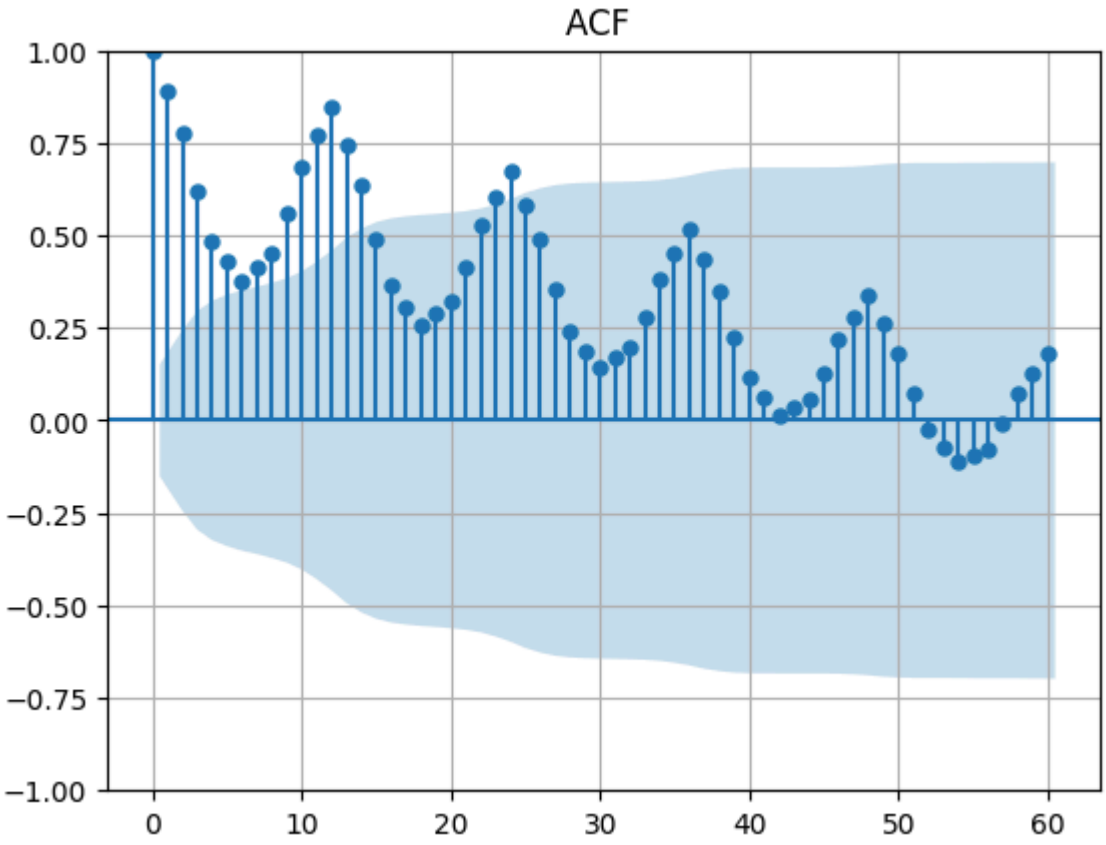
Le modèle adéquet est donc le modèle multuplicatif

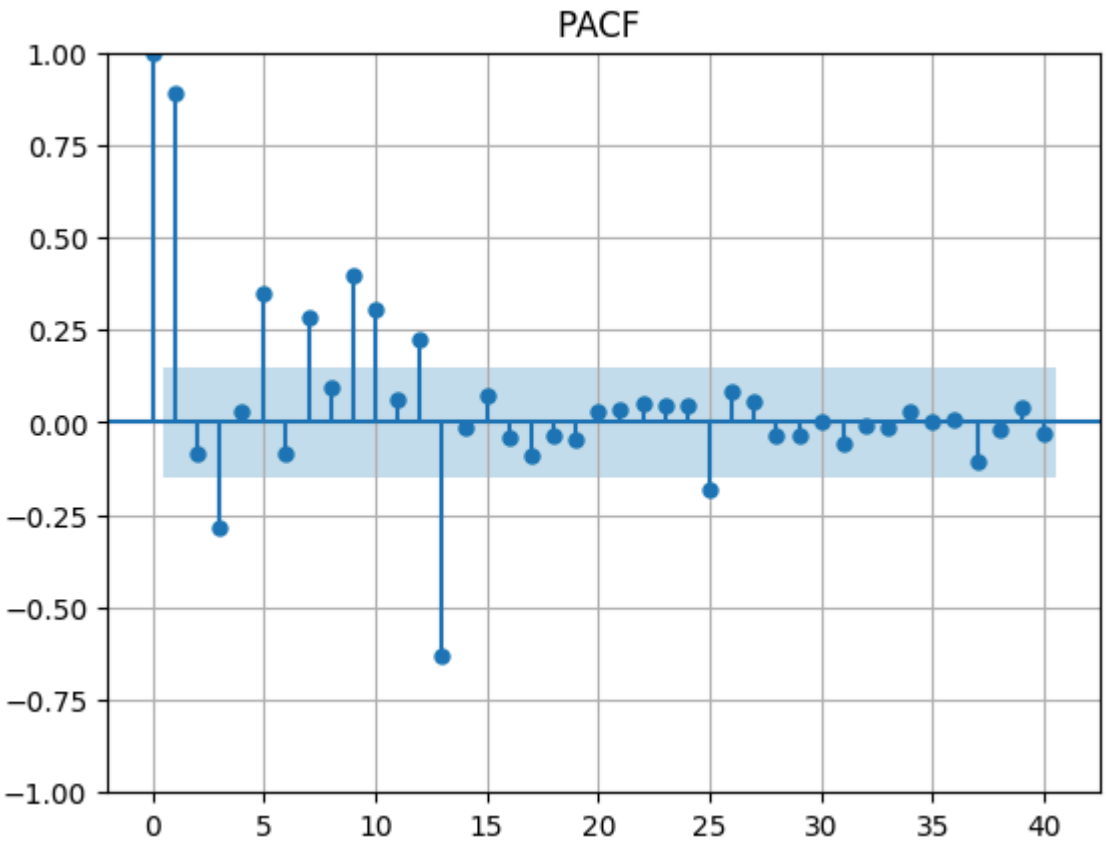
8. Les corrélogrammes simples et partiels ACF/PACF

In [26]:
```
plot_acf(df['Milk Production'], lags=60)
plt.title("ACF ")
plt.grid(True)
plt.show()

# PACF
plot_pacf(df['Milk Production'], lags=40, method='ywm')
```

```
plt.title("PACF ")
plt.grid(True)
plt.show()
```





Intérprétation des résultats :

pour L'ACF :

Décroissance lente don :la série est non-stationnaire (nécessite une différenciation, d=1).

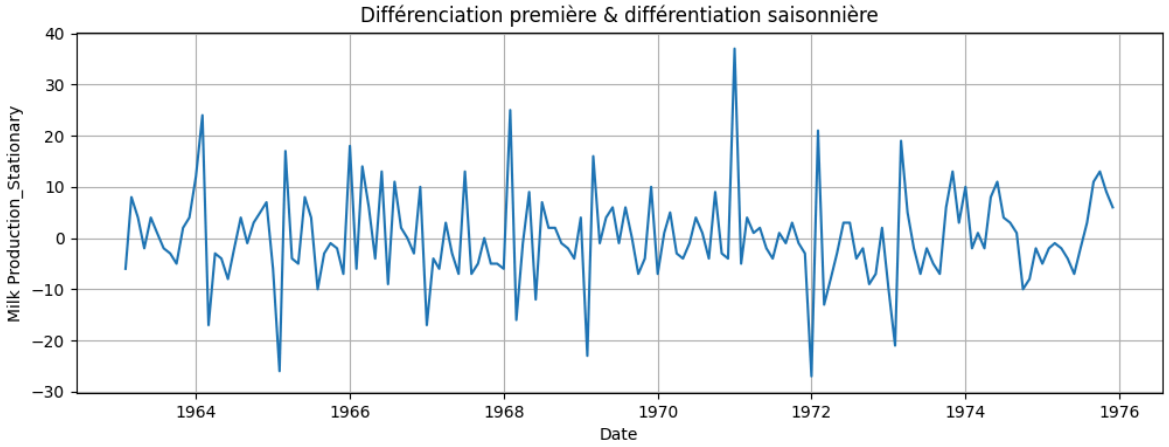
Plusieurs pics significatifs donc :possible une composante MA(q) ou effet de tendance/saisonnalité.

pour la PACF :

Coupe après quelques lags donc modèle AR(p) possible (p>1).

9. Stationarisation de la série

In [27]:
```
df["Milk Production_diff"]=df["Milk Production"].diff(1).dropna()
Xt_Seasonal_diff = df["Milk Production_diff"].diff(12).dropna()
plt.figure(figsize=(12,4))
adjusted_dates = df['Month'].iloc[13:]
plt.plot(adjusted_dates,Xt_Seasonal_diff)
plt.title("Différenciation première & différentiation saisonnière")
plt.grid(True)
plt.xlabel("Date")
plt.ylabel("Milk Production_Stationary")
plt.show()
```



Test de Dickey_Fuller

Le test de Dicky_Fuller se base sur le test de la stationnarité de la série le test d'hypothèse est le suivant: H_0(Hypothèse nulle):la série n'est pas stationnaire H_1 (hypothèse alternative) : la série est stationnaire. Statistique du test: Le test produit :

Une statistique ADF : un nombre réel (négatif en général).

Des valeurs critiques à différents niveaux de confiance (1%, 5%, 10%).

Un p-value : utilisée pour décider si H_0 doit être rejetée.

deux méthodes pour prendre la décision se présentent:

Méthode 1 : Comparaison avec la valeur critique Si ADF_stat < valeur_critique ⇒ on rejette H_0 ⇒ la série est stationnaire

Sinon ⇒ on ne rejette pas H_0 ⇒ la série n'est pas stationnaire

Méthode 2 : p-value Si p-value < alpha (ex. 0.05 ou 0.01) ⇒ on rejette H_0 ⇒ la série est stationnaire

Sinon ⇒ on ne rejette pas H_0

In [28]:
```
from statsmodels.tsa.stattools import adfuller
adf=adfuller(Xt_Seasonal_diff,maxlag=1)
print("ADF Statistic est :",adf[0])
print("la p_value est :",adf[1])
print("les valeurs critiques sont :",adf[4])
```

```
ADF Statistic est : -15.305129986852691
la p_value est : 4.260242160097419e-28
les valeurs critiques sont : {'1%': np.float64(-3.473542528196209), '5%': np.floa
t64(-2.880497674144038), '10%': np.float64(-2.576878053634677)}
```

Déduction:

on remarque que p_value= 4.260242160097419e-28<<0.05 :Alors on rejette H_0(que la série bien stationnaire)

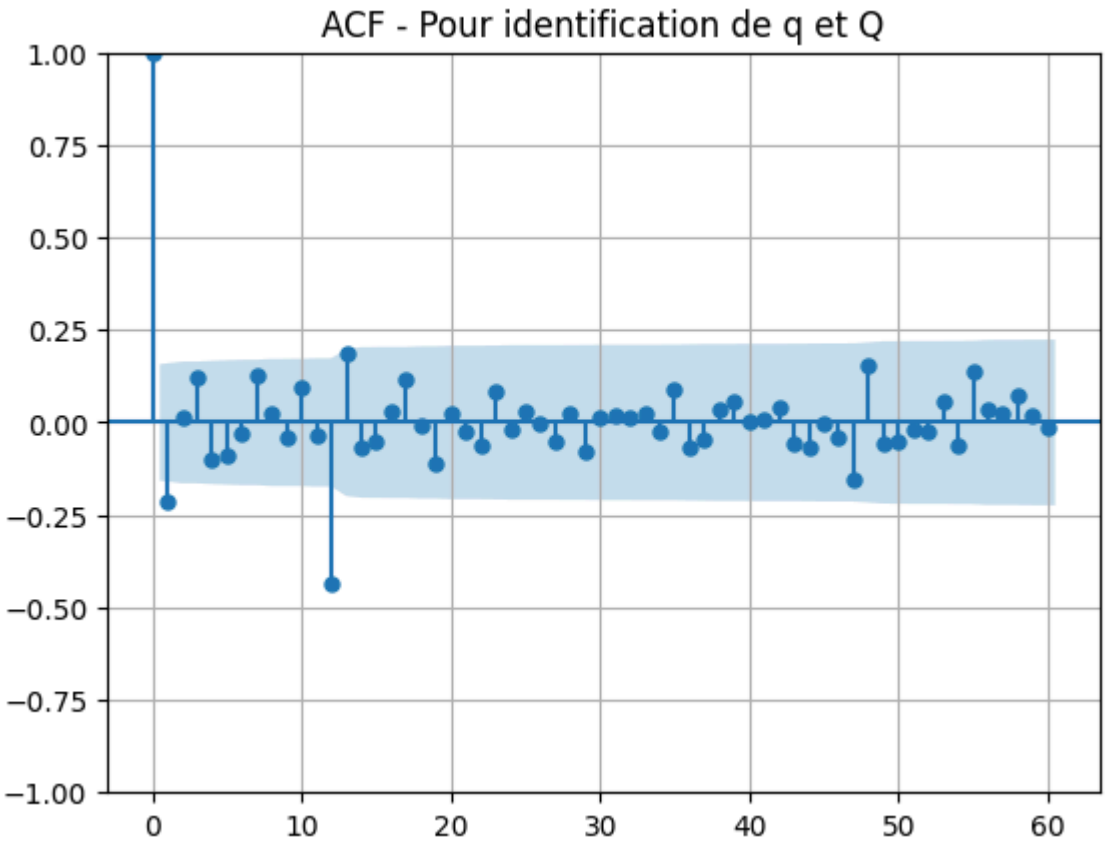
ou encore ADF<VCi ∀i : Alors on rejette H_0 c_à_d que la série est bien stationnaire

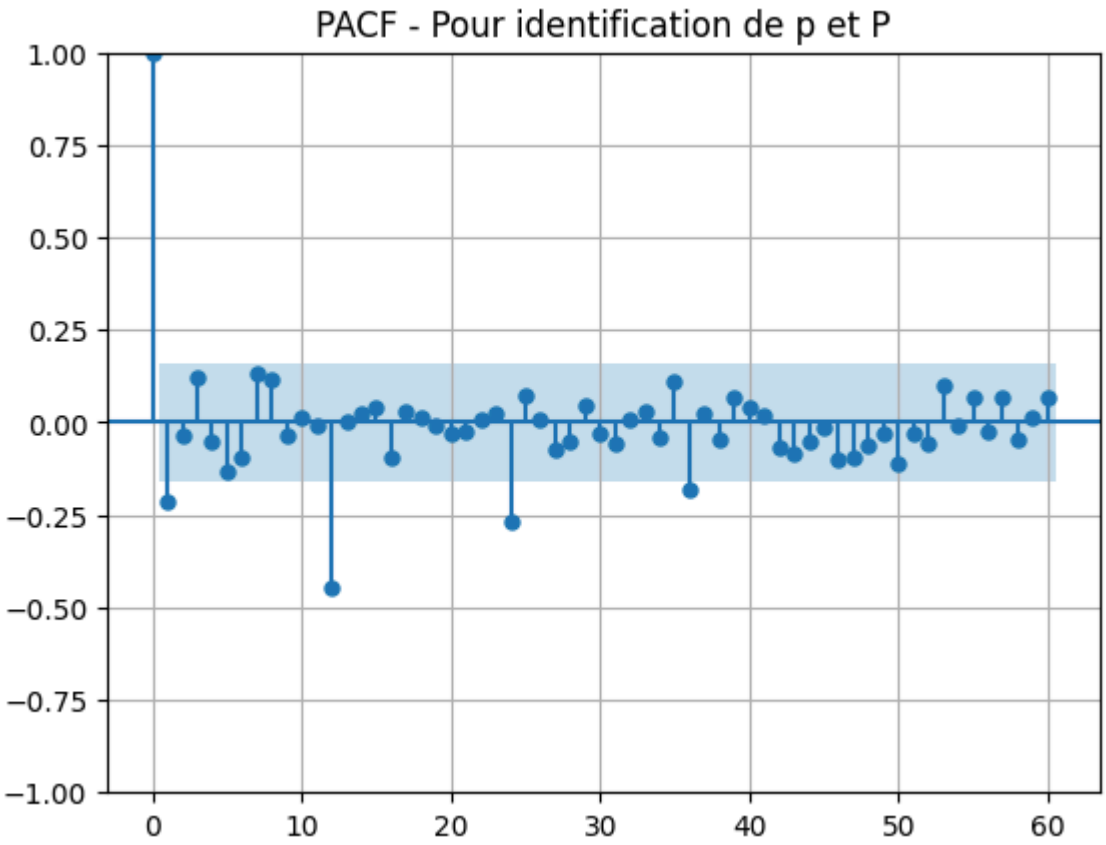
avec VCi ∈ { -3.473542528196209, -2.880497674144038, -2.576878053634677}

10. Construction du Modèle SARIMA(p,d,q)(P,D,Q)s

In [29]:
```
#ACF
plot_acf(Xt_Seasonal_diff, lags=60)
plt.title("ACF - Pour identification de q et Q")
plt.grid(True)
plt.show()

# PACF
plot_pacf(Xt_Seasonal_diff, lags=60, method='ywm')
plt.title("PACF - Pour identification de p et P")
plt.grid(True)
plt.show()
```





Détermination des paramètres du modèle:

à partir de la visualisation de l'ACF et la PACF on peut éstimer ler paramètres du modèle

p,P à partir de PACF et q,Q à partir de la PACF on déduit alors que :

(p= 0 ou p=1 ou p=2) ,(P=1 ou P=0) ,(q=0 ou q=1 ou q=2) ,(Q=0 ou Q=1 ou Q=2) ,d=1 ,D=1

après avoir tester ses valeurs pour plusieurs modèles :

le modèle adéquat est SARIMA(1,1,0)(0,1,1)12

In [30]:
```
import statsmodels.api as sm

model = sm.tsa.SARIMAX(df['Milk Production']
                       ,order=(1, 1, 0),
                      seasonal_order=(0, 1, 1, 12),
                      enforce_stationarity=False,
                      enforce_invertibility=False)


results = model.fit(disp=False)


print(results.summary())
```

```
                                     SARIMAX Results
==========================================================================================
Dep. Variable:                      Milk Production   No. Observations:
168
Model:             SARIMAX(1, 1, 0)x(0, 1, [1], 12)   Log Likelihood
-483.131
Date:                              Wed, 28 May 2025   AIC
972.262
Time:                                      23:11:04   BIC
981.130
Sample:                                           0   HQIC
975.866
                                              - 168
Covariance Type:                                opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.2618      0.082     -3.202      0.001      -0.422      -0.102
ma.S.L12      -0.6177      0.072     -8.579      0.000      -0.759      -0.477
sigma2        52.1076      5.038     10.342      0.000      42.233      61.983
==========================================================================================
==
Ljung-Box (L1) (Q):                   0.01   Jarque-Bera (JB):                29.
09
Prob(Q):                              0.94   Prob(JB):                         0.
00
Heteroskedasticity (H):               0.97   Skew:                             0.
66
Prob(H) (two-sided):                  0.93   Kurtosis:                         4.
78
==========================================================================================
==

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-st
ep).
```

à partir des résultats du modèle SARIMA(1,1,0)(0,1,1) on a :

1)test de Ljung_Box :montre la normalité des résidus car Prob(Q)=0.94 > 0.05 ce qui est un point

fort cette normalité nous permet une meilleure estimation des intervalles de confiances et des tests d'hypothèses

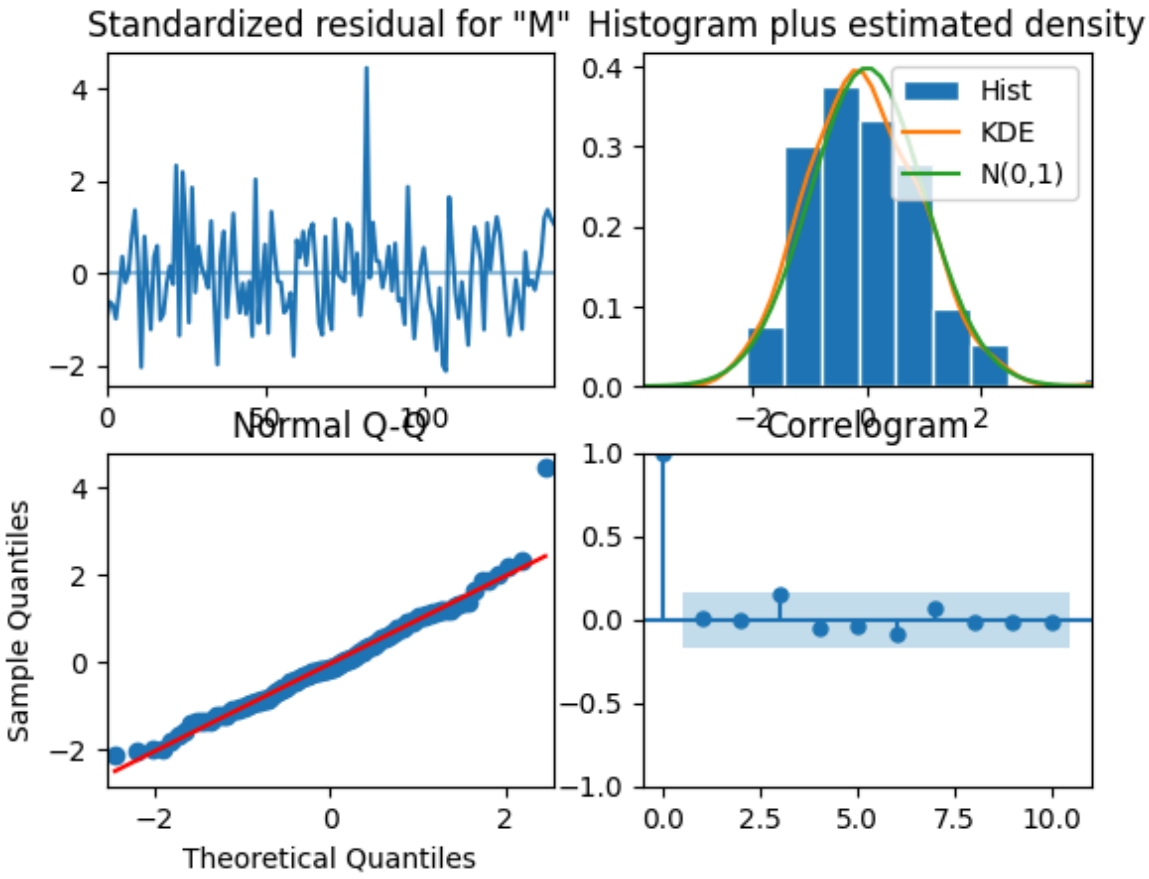
2)|z| > 1.96 :donc tous les coefficients sont significatifs

3)Heteroscédasticité : Prob(H) =0.93 > 0.O5 donc On ne rejette pas l'hypothèse nulle d'homoscédasticité

c_à_d que la variance est stable .

4)Ainsi ce modèle c'est le meilleur modèle avec AIC et BIC faibles

In [31]:
```
output = results.plot_diagnostics()
```



11. Les prévisions :

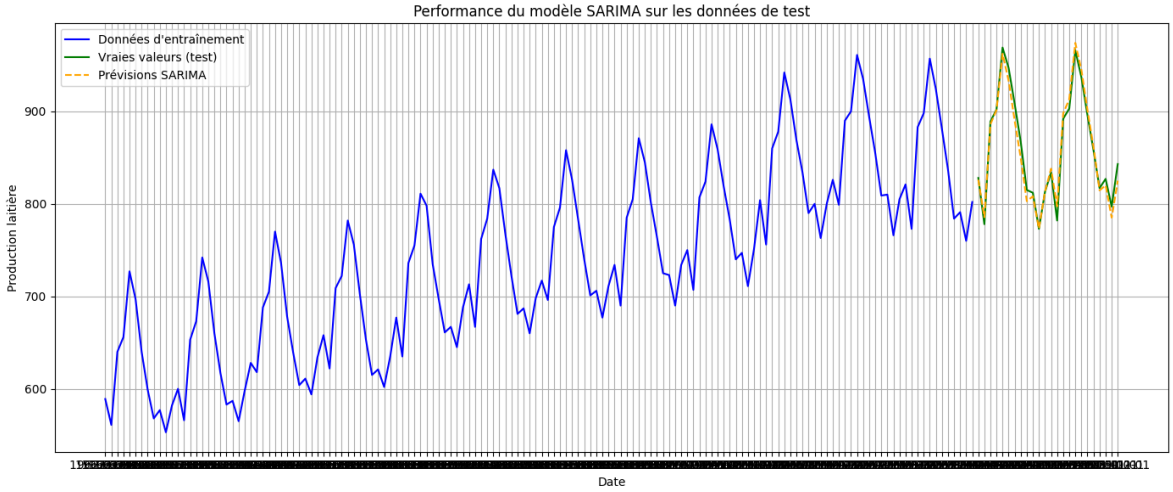
In [32]:
```
model = sm.tsa.SARIMAX(df_train['Milk Production'],
                       order=(1, 1, 0),
                       seasonal_order=(0, 1, 1, 12),
                       enforce_stationarity=False,
                       enforce_invertibility=False)

results = model.fit(disp=False)
```

```
n_forecast = len(df_test)
forecast = results.get_forecast(steps=n_forecast)
pred_mean = forecast.predicted_mean
pred_ci = forecast.conf_int()

plt.figure(figsize=(14, 6))

plt.plot(df_train['Month'], df_train['Milk Production'], label='Données d\'entra
plt.plot(df_test['Month'], df_test['Milk Production'], label='Vraies valeurs (te
plt.plot(df_test['Month'], pred_mean, label='Prévisions SARIMA', color='orange',
plt.title("Performance du modèle SARIMA sur les données de test")
plt.xlabel("Date")
plt.ylabel("Production laitière")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```



Evaluer les prévisions :Quelques métriques de performance

In [36]:
```
from sklearn.metrics import mean_absolute_error, mean_squared_error

mae = mean_absolute_error(df_test['Milk Production'], pred_mean)
rmse = mean_squared_error(df_test['Milk Production'], pred_mean)

print(f"MAE: {mae:.2f}")
print(f"RMSE: {rmse:.2f}")
```

```
MAE: 7.71
RMSE: 90.47
```