Hola 😀

Soy **Hesus Garcia** como "Jesús" pero con H. Sé que puede ser confuso al principio, pero una vez que lo recuerdes, ¡nunca lo olvidarás! 😊 . Como revisor de código de Triple-Ten, estoy emocionado de examinar tus proyectos y ayudarte a mejorar tus habilidades en programación. si has cometido algún error, no te preocupes, pues ¡estoy aquí para ayudarte a corregirlo y hacer que tu código brille! 🌟. Si encuentro algún detalle en tu código, te lo señalaré para que lo corrijas, ya que mi objetivo es ayudarte a prepararte para un ambiente de trabajo real, donde el líder de tu equipo actuaría de la misma manera. Si no puedes solucionar el problema, te proporcionaré más información en la próxima oportunidad. Cuando encuentres un comentario, **por favor, no los muevas, no los modifiques ni los borres**.

Revisaré cuidadosamente todas las implementaciones que has realizado para cumplir con los requisitos y te proporcionaré mis comentarios de la siguiente manera:

Comentario del revisor Si todo está perfecto.

Comentario del revisor Si tu código está bien pero se puede mejorar o hay algún detalle que le hace falta.

Comentario del revisor Si de pronto hace falta algo o existe algún problema con tu código o conclusiones.

Puedes responderme de esta forma:

Respuesta del estudiante

</br>

¡Empecemos! 🚀

Descripcion

Trabajas en la compañía de extracción de petróleo OilyGiant. Tu tarea es encontrar los mejores lugares donde abrir 200 pozos nuevos de petróleo. Para completar esta tarea, tendrás que realizar los siguientes pasos:

- Leer los archivos con los parámetros recogidos de pozos petrolíferos en la región seleccionada: calidad de crudo y volumen de reservas.
- Crear un modelo para predecir el volumen de reservas en pozos nuevos.
- Elegir los pozos petrolíferos que tienen los valores estimados más altos.
- Elegir la región con el beneficio total más alto para los pozos petrolíferos seleccionados.

Tienes datos sobre muestras de crudo de tres regiones. Ya se conocen los parámetros de cada pozo petrolero de la región. Crea un modelo que ayude a elegir la región con el mayor margen de beneficio. Analiza los beneficios y riesgos potenciales utilizando la técnica bootstrapping.

Inicializacion

In [1]:

```
# Cargar todas las librerías

import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from scipy.stats import zscore
from scipy.stats import norm
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.pipeline import make_pipeline
from sklearn.utils import resample
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import StandardScaler
```

Respuesta del estudiante Aca cargamos las librerias necesarias para el proyecto.

Cargar los Datos

In [2]:

```
# Carga el archivo de datos en un DataFrame

try:
    data_0 = pd.read_csv('geo_data_0.csv')
    data_1 = pd.read_csv('geo_data_1.csv')
    data_2 = pd.read_csv('geo_data_2.csv')
except:
    data_0 = pd.read_csv('/datasets/geo_data_0.csv')
    data_1 = pd.read_csv('/datasets/geo_data_1.csv')
    data_2 = pd.read_csv('/datasets/geo_data_2.csv')
```

Respuesta del estudiante Ahora cargamos los datos por cada dataframe que nos compartieron.

Exploracion de los Datos

In [3]:

```
# imprime la información general/resumida sobre el DataFrame

print(data_0.shape)
print(data_1.shape)
print(data_2.shape)
```

```
(100000, 5)
(100000, 5)
```

```
(100000, 5)
```

In [4]:

```
# Imprimimos una muestra de los datos del DataFrame

print(data_0.head(10))
print(data_1.head(10))
print(data_2.head(10))
```

```
      id        f0        f1        f2     product
0  txEyH  0.705745 -0.497823  1.221170  105.280062
1  2acmU  1.334711 -0.340164  4.365080   73.037750
2  409Wp  1.022732  0.151990  1.419926   85.265647
3  iJLyR -0.032172  0.139033  2.978566  168.620776
4  Xdl7t  1.988431  0.155413  4.751769  154.036647
5  wX4Hy  0.969570  0.489775 -0.735383   64.741541
6  tL6pL  0.645075  0.530656  1.780266   49.055285
7  BYPU6 -0.400648  0.808337 -5.624670   72.943292
8  j9Oui  0.643105 -0.551583  2.372141  113.356160
9  OLuZU  2.173381  0.563698  9.441852  127.910945
      id         f0         f1        f2     product
0  kBEdx -15.001348  -8.276000 -0.005876    3.179103
1  62mP7  14.272088  -3.475083  0.999183   26.953261
2  vyE1P   6.263187  -5.948386  5.001160  134.766305
3  KcrkZ -13.081196 -11.506057  4.999415  137.945408
4  AHL4O  12.702195  -8.147433  5.004363  134.766305
5  HHckp  -3.327590  -2.205276  3.003647   84.038886
6  h5Ujo -11.142655 -10.133399  4.002382  110.992147
7  muH9x   4.234715  -0.001354  2.004588   53.906522
8  YiRkx  13.355129  -0.332068  4.998647  134.766305
9  jG6Gi   1.069227 -11.025667  4.997844  137.945408
      id        f0        f1        f2     product
0  fwXo0 -1.146987  0.963328 -0.828965   27.758673
1  WJtFt  0.262778  0.269839 -2.530187   56.069697
2  ovLUW  0.194587  0.289035 -5.586433   62.871910
3  q6cA6  2.236060 -0.553760  0.930038  114.572842
4  WPMUX -0.515993  1.716266  5.899011  149.600746
5  LzZXx -0.758092  0.710691  2.585887   90.222465
6  WBHRv -0.574891  0.317727  1.773745   45.641478
7  XO8fn -1.906649 -2.458350 -0.177097   72.480640
8  ybmQ5  1.776292 -0.279356  3.004156  106.616832
9  OilcN -1.214452 -0.439314  5.922514   52.954532
```

In [5]:

```
# Obtener información sobre los datos

print(data_0.info())
print(data_1.info())
print(data_2.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 5 columns):
 #   Column   Non-Null Count   Dtype
---  ------   --------------   -----
 0   id       100000 non-null  object
 1   f0       100000 non-null  float64
 2   f1       100000 non-null  float64
 3   f2       100000 non-null  float64
 4   product  100000 non-null  float64
dtypes: float64(4), object(1)
memory usage: 3.8+ MB
None
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 5 columns):
 #   Column   Non-Null Count   Dtype
---  ------   --------------   -----
 0   id       100000 non-null  object
 1   f0       100000 non-null  float64
 2   f1       100000 non-null  float64
 3   f2       100000 non-null  float64
 4   product  100000 non-null  float64
dtypes: float64(4), object(1)
memory usage: 3.8+ MB
None
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 5 columns):
 #   Column   Non-Null Count   Dtype
---  ------   --------------   -----
 0   id       100000 non-null  object
 1   f0       100000 non-null  float64
 2   f1       100000 non-null  float64
 3   f2       100000 non-null  float64
 4   product  100000 non-null  float64
dtypes: float64(4), object(1)
memory usage: 3.8+ MB
None
```

Respuesta del estudiante En las 3 celdas anteriores visualizamos la forma, contenido inicial y tipo de data contenida en las `DF` por Region.

In [36]:

```
# Elimina la columna 'id'
#data_0 = data_0.drop('id', axis=1)
#data_1 = data_1.drop('id', axis=1)
#data_2 = data_2.drop('id', axis=1)

# Verificar si hay valores nulos en cada columna

null_values = data_0.isnull().sum(), data_1.isnull().sum(), data_2.isnull().sum()

print("Valores nulos por columna:")
null_values
```

```
Valores nulos por columna:
```

Out[36]:

```
(f0         0
 f1         0
 f2         0
 product    0
 dtype: int64,
 f0         0
 f1         0
 f2         0
 product    0
 dtype: int64,
 f0         0
 f1         0
 f2         0
 product    0
 dtype: int64)
```

Respuesta del estudiante Nos aseguramos de que no haya nulos presentes en los `DF` ademas

de quitar la columna `ID` ya que no es relevante para la investigacion.

In [7]:

```
# Asegúrate de tener solo las columnas relevantes para la detección de anomalías
features = ['f0', 'f1', 'f2', 'product']
data = data_0[features]

# Calcular la puntuación Z para cada característica
z_scores = np.abs(zscore(data))

# Define un umbral para identificar valores atípicos (puedes ajustar este umbral)
threshold = 3

# Encuentra las posiciones de los valores atípicos
outliers = np.where(z_scores > threshold)

# Imprime las posiciones de los valores atípicos
print("Posiciones de valores atípicos:")
for feature, positions in zip(features, outliers):
    print(f"Característica: {feature}, Posiciones: {positions}")
```

```
Posiciones de valores atípicos:
Característica: f0, Posiciones: [  131   232   947  1054  2227  3426  4384  4475  4515
6002  6404  6496
  6760  7122  8953 11404 11773 11882 13949 14128 14813 15348 15353 15492
 16373 16748 17717 17908 18296 19439 20275 21210 21253 21745 22500 22669
 23465 23472 24012 24284 24395 24458 24685 25156 25827 26773 26835 27238
 27658 29047 29529 29668 30946 32357 32407 32540 32867 34331 34703 34958
 35127 35273 35659 36033 36172 36573 37820 37854 38886 39384 39904 40928
 41364 41435 41519 42379 42500 42538 43388 43808 44651 44754 46299 46784
 47362 49785 50265 50377 50502 51140 51464 51494 51905 52356 52455 52477
 52820 54228 54328 55233 55480 55529 56960 57338 57591 57748 58060 58269
 59136 59536 59770 59780 60351 61554 61800 62404 65487 65507 65743 66543
 68207 68527 70439 70797 71028 71080 71202 72059 72133 73887 74439 74589
 74946 75469 75714 76089 76748 76826 78808 78948 79037 80159 80239 80361
 80729 81257 81450 82305 83095 84701 85054 85572 86449 86554 88683 88795
 89040 89189 89232 89454 90290 90310 90859 91398 92002 92052 92571 93073
 93554 93716 94655 95537 95898 96481 96732 96963 98773 99628]
Característica: f1, Posiciones: [2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2]
```

In [8]:

```
# Asegúrate de tener solo las columnas relevantes para la detección de anomalías
features = ['f0', 'f1', 'f2', 'product']
data = data_1[features]

# Calcular la puntuación Z para cada característica
z_scores = np.abs(zscore(data))

# Define un umbral para identificar valores atípicos (puedes ajustar este umbral)
threshold = 3

# Encuentra las posiciones de los valores atípicos
outliers = np.where(z_scores > threshold)

# Imprime las posiciones de los valores atípicos
print("Posiciones de valores atípicos:")
```

```
for feature, positions in zip(features, outliers):
    print(f"Característica: {feature}, Posiciones: {positions}")
```

```
Posiciones de valores atípicos:
Característica: f0, Posiciones: [  769  1201  1576  2254  2367  2379  2688  3262  3333
3692  4062  4097
  4785  5042  5085  5630  5631  5815  6370  7220  7227  7479  7919  8600
  8637  8645  8667  9100  9272  9765 10232 10521 10662 10671 12242 13256
 13376 13939 14647 15143 15166 15875 15949 16032 16042 16900 17083 17924
 18818 19524 19672 19742 19866 19982 20143 20341 21307 21445 21779 22071
 23165 23834 24452 25851 26237 26292 26531 26843 27255 27593 28336 28698
 28978 29148 29470 29682 30461 30533 31688 32386 32549 32634 33033 33351
 33562 34189 34283 34720 35253 35283 35481 35621 35716 35734 35744 36493
 36631 37079 37538 37708 37861 38245 38607 38750 38836 39193 39561 40652
 40678 41475 41691 41916 42299 42357 43106 43235 43770 44439 44991 45057
 45769 46583 47208 47992 48467 48593 48643 48944 49027 49047 49242 49484
 50059 50841 51260 51261 51356 52190 52618 52680 53084 54467 54788 54812
 56046 56948 57368 57986 58032 58184 58373 58726 58833 59374 60075 60372
 61300 61798 61808 61853 62081 62230 62231 62483 63769 64226 64259 65022
 65138 65398 65798 66430 66696 67188 67324 67767 68006 68143 68842 69795
 70925 71167 71857 72100 72533 72604 72677 72697 73383 73471 73750 73836
 73882 74227 75417 75657 75762 75846 76428 76810 76971 77363 78361 78465
 79228 79516 79991 80395 80428 80495 80938 81412 81680 81725 82454 83389
 83437 83601 84636 84943 85263 85292 86432 87001 87294 87591 88075 88119
 88601 88656 88665 89596 89632 89905 90128 90871 90916 90945 91077 91127
 92138 92260 92497 93001 93323 93394 93432 93634 94098 94169 94482 94884
 95062 95585 95807 95818 95952 95996 96081 97110 97143 97417 97830 97930
 98188 98801 98855 99250 99349]
Característica: f1, Posiciones: [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1
 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1
 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1]
```

In [9]:

```
# Asegúrate de tener solo las columnas relevantes para la detección de anomalías
features = ['f0', 'f1', 'f2', 'product']
data = data_2[features]

# Calcular la puntuación Z para cada característica
z_scores = np.abs(zscore(data))

# Define un umbral para identificar valores atípicos (puedes ajustar este umbral)
threshold = 3

# Encuentra las posiciones de los valores atípicos
outliers = np.where(z_scores > threshold)

# Imprime las posiciones de los valores atípicos
print("Posiciones de valores atípicos:")
for feature, positions in zip(features, outliers):
    print(f"Característica: {feature}, Posiciones: {positions}")
```

```
Posiciones de valores atípicos:
Característica: f0, Posiciones: [  110   130   304   314   319   334   426   443   586
620   644   709
   771  1056  1311  1349  1545  1546  1610  1904  2032  2145  2361  2646
  2709  2897  2990  3445  3540  3720  3924  4025  4055  4591  4674  4744
  4752  4761  5016  5172  5293  5296  5357  5375  5434  5443  5567  5944
  6109  6207  6219  6558  6647  6689  6713  6951  7110  7176  7661  7891
```

```
  8021  8028  8165  8292  8317  8417  8770  8839  8869  8901  8920  9268
  9756  9818 10138 10452 10758 11058 11062 11210 11316 11323 11377 11634
 11727 11963 12056 12160 12239 12278 12327 12334 12544 12575 12730 12907
 13067 13469 13638 14121 14846 15110 15488 15594 15772 16022 16061 16437
 16553 16613 16710 16723 16773 17020 17106 17415 17638 17665 17808 18215
 18391 18485 18534 18747 18867 18951 18963 19033 19096 19314 19407 19467
 19674 19698 19757 20819 21349 21383 21540 21547 21803 22012 22102 22134
 22349 22481 22533 22578 22597 22647 22670 22699 22764 22860 23017 23027
 23271 23282 23330 23440 23500 23552 23636 23646 23862 24018 24067 24099
 24358 24583 24658 24703 24794 24920 25319 25463 25481 25492 25521 25715
 25718 25790 25821 26089 26398 26489 26548 26761 27080 27192 27315 27451
 27471 27480 27508 27800 28016 28136 28213 28236 28457 28528 28596 28602
 28676 28738 29022 29246 29335 29338 29463 29557 29584 29655 29713 29734
 29787 29888 29926 30173 30187 30240 30292 30389 30559 30605 30799 30899
 30975 31113 31116 31183 31289 31425 32074 32130 32441 32473 32603 32739
 32791 32792 33006 33145 33197 33284 33489 33520 33820 34027 34173 34321
 34362 34408 34543 34650 35019 35073 35092 35112 35139 35301 35557 35567
 35814 36323 36376 36646 36740 36870 37077 37104 37255 37722 37791 37874
 38230 38322 38567 38586 38758 38875 38881 38916 38979 39015 39115 39122
 39274 39420 39484 39657 39828 39890 40212 40231 40399 40644 40747 41153
 41174 41182 41414 41423 41560 41820 42178 42218 42713 42787 42812 42816
 42930 43126 43172 43177 43543 43639 43761 43812 43845 43931 44054 44077
 44162 44341 44558 44679 44721 44873 45078 45225 45265 45539 45561 46170
 46528 46582 46642 46649 46694 46744 46748 46826 46994 47377 47433 47528
 47580 47670 47811 47950 48202 48249 48483 48599 48957 48997 49062 49066
 49083 49258 49461 49538 49568 49692 49742 50100 50499 50710 50772 50936
 50967 51088 51271 51385 51504 51576 51641 51723 51755 51771 51787 51886
 51900 51930 52025 52133 52161 52256 52298 52353 52545 52722 53151 53243
 53375 53429 53444 53613 53973 53994 54008 54085 54119 54137 54214 54310
 54494 54772 54814 54842 54955 54981 55016 55101 55228 55304 55309 55346
 55381 55415 55463 55828 56188 56265 56569 56649 56711 56741 56795 56803
 56881 57047 57092 57164 57334 57408 57416 57718 57790 57850 57917 57921
 57930 58024 58024 58254 58429 58531 58534 58558 58565 58684 58756 58774
 58980 59016 59089 59396 59583 59788 59837 59899 60079 60108 60192 60240
 60354 60524 60767 60775 60793 60823 60952 61401 61486 61596 61619 61702
 61743 61768 61915 61960 62009 62101 62215 62216 62496 62522 62626 63074
 63109 63123 63201 63325 63336 63600 63658 63671 63783 63875 63953 64075
 64376 64380 64432 64504 64669 65021 65792 66047 66136 66386 66604 66726
 66816 66984 67036 67100 67195 67203 67303 67524 67647 67688 67774 67862
 68126 68598 68745 68774 68817 68843 68925 69019 69192 69367 69502 69512
 69796 69873 69960 70337 70444 70700 70712 70862 71030 71032 71109 71370
 71378 71403 71479 71862 72086 72393 72501 72512 72514 72747 72869 72997
 73021 73068 73112 73221 73244 73426 73453 73636 73729 73749 73783 73874
 73985 74140 74281 74840 74951 75079 75457 75652 75837 76300 76363 76543
 77004 77271 77724 77766 78115 78293 78653 79163 79261 79352 79361 79518
 79605 79854 79931 80011 80568 80853 80891 81077 81130 81374 81503 81529
 81656 81719 81792 81886 81967 82389 82531 82630 82661 82682 82770 82841
 82947 83263 83266 83618 83671 84047 84177 84212 84291 84441 84497 84597
 84814 84817 84888 84900 84970 85200 85202 85241 85501 85545 85691 85877
 85912 86321 86489 86523 86685 86809 87447 87965 88051 88087 88091 88126
 88529 88697 88710 88748 88870 89019 89165 89195 89216 89364 89554 89590
 89604 89615 90046 90186 90357 90417 90648 91005 91189 91246 91268 91269
 91270 91307 91445 91759 92154 92444 92446 92560 92766 92824 92918 92938
 92957 93161 93180 93286 93527 93708 94078 94219 94239 94342 94733 94755
 95032 95066 95257 95638 95828 95970 96053 96415 96515 96773 96836 97217
 97249 97502 97513 97563 97585 97721 97738 97762 97820 97907 98189 98280
 98309 98511 98529 98619 98714 98733 98765 98768 98942 99174 99362 99393
 99533 99626 99714 99941]
Característica: f1, Posiciones: [1 0 1 1 2 0 2 2 1 1 1 2 0 2 0 2 2 2 1 2 1 0 1 2 2 2 1 0
2 1 2 0 2 1 0 2 1
 0 2 1 1 0 2 0 1 2 1 2 2 2 0 2 2 0 2 1 0 1 0 2 0 0 1 1 2 2 1 1 1 0 0 2 2 2
 0 2 1 2 2 2 2 1 2 1 1 2 0 0 2 1 2 0 2 2 2 1 2 0 2 1 0 1 1 0 0 0 1 2 0 0 0
 0 2 2 0 2 2 2 1 0 2 1 1 2 0 0 0 0 1 1 2 0 2 2 1 0 2 0 0 2 2 2 0 0 1 1 0 1
 0 0 2 1 0 1 1 0 1 2 1 0 2 1 1 0 0 2 1 0 0 0 1 2 2 0 2 2 1 0 0 2 0 1 0 1 2
 2 2 0 0 1 1 0 2 0 0 0 0 1 0 1 2 1 1 2 1 0 1 2 0 1 1 0 0 1 0 1 2 1 2 1 1 0
```

```
1 0 0 0 1 2 2 0 0 2 1 1 2 0 2 1 1 0 2 1 0 1 1 0 1 1 1 2 1 0 2 2 1 1 0 2 1
1 1 2 0 1 2 2 1 0 0 0 0 1 1 0 1 1 1 1 0 1 1 1 0 1 2 1 1 2 2 2 0 0 1 2 1 2
0 2 0 1 1 0 1 1 0 2 1 2 1 0 1 2 1 1 1 0 0 1 0 0 2 2 0 1 0 0 2 0 1 1 0 0 1
2 0 1 2 0 1 2 0 0 0 0 0 2 2 0 0 2 1 0 0 1 0 1 0 1 1 0 0 0 1 1 0 0 0 1 1 1
0 2 2 0 0 1 1 0 2 2 1 1 1 1 1 2 2 1 0 0 0 0 0 0 2 1 1 2 2 2 1 1 1 2 2 2 1
1 1 2 1 1 0 1 1 2 0 0 2 0 0 0 1 0 2 0 1 1 2 0 1 0 1 0 1 0 1 0 2 0 1 0 0 0
2 0 1 1 1 1 1 0 2 2 0 2 1 0 2 0 1 1 2 1 0 0 1 0 1 1 0 0 1 0 0 0 1 0 1 0 0
0 0 1 0 2 1 1 2 1 1 1 2 0 0 2 1 1 1 0 0 1 2 2 2 2 2 2 2 0 1 2 1 0 0 1 0 1
1 1 0 0 1 1 1 0 1 0 0 0 1 1 1 1 2 0 2 0 2 0 0 1 2 2 0 1 2 0 1 0 0 1 1 2 2
1 0 1 0 1 1 1 2 0 0 0 2 1 0 2 1 0 0 1 2 1 0 0 1 1 1 2 0 0 0 0 1 0 0 0 1 1
2 2 0 0 2 0 1 2 0 2 1 2 2 1 1 1 0 1 0 1 2 1 2 2 0 1 2 1 2 0 1 2 0 2 1 0 0
2 0 0 1 1 0 0 1 0 0 1 2 1 0 2 1 0 0 0 2 1 1 1 2 1 0 0 1 1 2 1 0 1 0 1 1 0
2 0 2 0 2 1 2 0 2 1 0 1 1 2 1 1 2 0 1 1 1 0 1 1 1 0 1 1 1 2 0 1 1 2 1 0 0
0 1 1 2 1 0 2 2 1 2 2 2 0 1 0 1 0 2 0 1 2 1 1 0 0 0 0 1 0 1 0 2 2 2 1 1 1
0 0 2 1 0 0 1 1]
```

Respuesta del estudiante Como hemos encontrado una cantidad significativa de datos atipicos y anomalias en los datos vamos a reemplazarlos por la mediana de estos para intentar ver si mejora la precision de los resultados.

In [10]:

```python
# Función para reemplazar valores atípicos por la mediana en un DataFrame
def replace_outliers_with_median(data, threshold):
    # Calcular la puntuación Z para cada característica
    z_scores = np.abs(zscore(data))

    # Encontrar las posiciones de los valores atípicos
    outliers = np.where(z_scores > threshold)

    # Reemplazar valores atípicos por la mediana
    data_no_outliers = data.copy()
    for feature, positions in zip(data.columns, outliers):
        data_no_outliers[feature].iloc[positions] = data[feature].median()

    return data_no_outliers

# Establecer umbrales específicos para cada región
thresholds = [0.3, 4, 0.8]

# Aplicar la función a cada DataFrame
data_0_no_outliers = replace_outliers_with_median(data_0, thresholds[0])
data_1_no_outliers = replace_outliers_with_median(data_1, thresholds[1])
data_2_no_outliers = replace_outliers_with_median(data_2, thresholds[2])

# Verificar la forma de los DataFrames después de reemplazar outliers por la mediana
print(f"Forma de data_0 después de reemplazar outliers por la mediana: {data_0_no_outli
print(f"Forma de data_1 después de reemplazar outliers por la mediana: {data_1_no_outli
print(f"Forma de data_2 después de reemplazar outliers por la mediana: {data_2_no_outli
```

```
Forma de data_0 después de reemplazar outliers por la mediana: (100000, 4)
Forma de data_1 después de reemplazar outliers por la mediana: (100000, 4)
Forma de data_2 después de reemplazar outliers por la mediana: (100000, 4)
```

In [11]:

```python
# Función para visualizar la relación entre dos variables y sus distribuciones individu
def visualize_relationship(feature, target, data, region):
    sns.jointplot(x=feature, y=target, data=data, kind='hex')
    plt.suptitle(f'Región {region} - Relación entre {feature} y {target}', y=1.02)
    plt.show()
```

```python
# Visualizar la relación para la Región 0
for feature in ['f0', 'f1', 'f2']:
    visualize_relationship(feature, 'product', data_0, 0)

# Visualizar la relación para la Región 1
for feature in ['f0', 'f1', 'f2']:
    visualize_relationship(feature, 'product', data_1, 1)

# Visualizar la relación para la Región 2
for feature in ['f0', 'f1', 'f2']:
    visualize_relationship(feature, 'product', data_2, 2)

# Calcular la correlación de Pearson para cada característica con respecto al target
correlation_0 = data_0[['f0', 'f1', 'f2', 'product']].corr()['product']
correlation_1 = data_1[['f0', 'f1', 'f2', 'product']].corr()['product']
correlation_2 = data_2[['f0', 'f1', 'f2', 'product']].corr()['product']

print("Correlación de Pearson para la Región 0:")
print(correlation_0)

print("\nCorrelación de Pearson para la Región 1:")
print(correlation_1)

print("\nCorrelación de Pearson para la Región 2:")
print(correlation_2)
```