TOÁN ỨNG DỤNG VÀ THỐNG KÊ

Final Project - Câu 1

Thông tin sinh viên

- Họ và tên: Nguyễn Thái Bảo
- Mã số sinh viên: 23120023
- Lớp: 23CTT1

Sử dụng thư viện

- Đọc dữ liệu: pandas
- Trực quan hóa dữ liệu: matplotlib
- Tính toán trên số thực: math

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import math
```

Các hàm tiện ích

Xử lý ma trận

In [2]:
```
def create_matrix(rows, cols, default_value = 0):
    """Tạo ma trận với rows hàng và cols cột"""
    return [[default_value for _ in range(cols)] for _ in range(rows)]
```

```python
def matrix_multiply(A, B):
    """Nhân ma trận A và B"""

    # Trường hợp B là vector
    if not isinstance(B[0], list):
        # Chuyển B thành ma trận cột
        B = [[b] for b in B]

    # Lấy kích thước của ma trận
    rows_A = len(A)
    cols_A = len(A[0])
    rows_B = len(B)
    cols_B = len(B[0]) if isinstance(B[0], list) else 1

    # Kiểm tra điều kiện nhân ma trận
    if cols_A != rows_B:
        raise ValueError(f"Kích thước không phù hợp: ({rows_A}x{cols_A}) và ({rows_B}x{cols_B})")

    # Nhân ma trận
    C = create_matrix(rows_A, cols_B)
    for i in range(rows_A):
        for j in range(cols_B):
            for k in range(cols_A):
                C[i][j] += A[i][k] * B[k][j]

    # Nếu ma trận kết quả chỉ có 1 cột, trả về vector
    if cols_B == 1:
        return [row[0] for row in C]

    return C


def print_matrix(A):
    """Hàm in ma trận"""
    for row in A:
        print(f"{'  '.join(f'{x:.6f}' for x in row)}")


def inverse(P):
    """Tính ma trận nghịch đảo bằng phương pháp Gauss - Jordan"""
    n = len(P)
```

```
    # Tạo ma trận mở rộng [P | I]
    P_augmented = []
    for i in range(n):
        row = P[i][:]
        for j in range(n):
            if i == j:
                row.append(1.0)
            else:
                row.append(0.0)
        P_augmented.append(row)

    # Biến đổi Gauss - Jordan
    for i in range(n):
        # Tìm pivot
        max_val = abs(P_augmented[i][i])
        max_row = i
        for k in range(i + 1, n):
            if abs(P_augmented[k][i]) > max_val:
                max_val = abs(P_augmented[k][i])
                max_row = k

        # Hoán đổi hàng
        if max_row != i:
            P_augmented[i], P_augmented[max_row] = P_augmented[max_row], P_augmented[i]

        # Chuẩn hóa hàng pivot
        pivot = P_augmented[i][i]
        for j in range(i, 2*n):
            P_augmented[i][j] /= pivot

        # Khử tất cả các hàng khác
        for k in range(n):
            if k != i:
                factor = P_augmented[k][i]
                for j in range(i, 2*n):
                    P_augmented[k][j] -= factor * P_augmented[i][j]

    # Lấy ma trận nghịch đảo
    P_inverse = []
    for i in range(n):
        P_inverse.append(P_augmented[i][n:])
```

```
    return P_inverse


def transpose(A):
    """Tính ma trận chuyển vị"""

    # Kiểm tra nếu A là vector
    if not isinstance(A[0], list):
        return [[a] for a in A]  # Chuyển vector hàng thành vector cột

    # Tính ma trận chuyển vị
    rows = len(A)
    cols = len(A[0])
    A_T = [[0 for _ in range(rows)] for _ in range(cols)]

    for i in range(rows):
        for j in range(cols):
            A_T[j][i] = A[i][j]

    return A_T
```

Xử lý vector

In [3]:
```
def sum_of_squares(values):
    """Tính tổng bình phương của list"""
    return sum(x * x for x in values)
```

Tiếp cận bộ dữ liệu

Tải dữ liệu

In [4]:
```
df = pd.read_csv('customer_purchase_behaviors.csv')
```

Tổng quan bộ dữ liệu

In [5]:
```
df
```

Out[5]:

	user_id	age	annual_income	purchase_amount	loyalty_score	region	purchase_frequency
0	1	25	45000	200	4.5	North	12
1	2	34	55000	350	7.0	South	18
2	3	45	65000	500	8.0	West	22
3	4	22	30000	150	3.0	East	10
4	5	29	47000	220	4.8	North	13
...	...	...	...	...	...	...	...
233	234	40	60000	450	7.2	West	20
234	235	38	59000	430	6.9	North	20
235	236	54	74000	630	9.4	South	27
236	237	32	52000	360	5.8	West	18
237	238	31	51000	340	5.6	North	17

238 rows × 7 columns

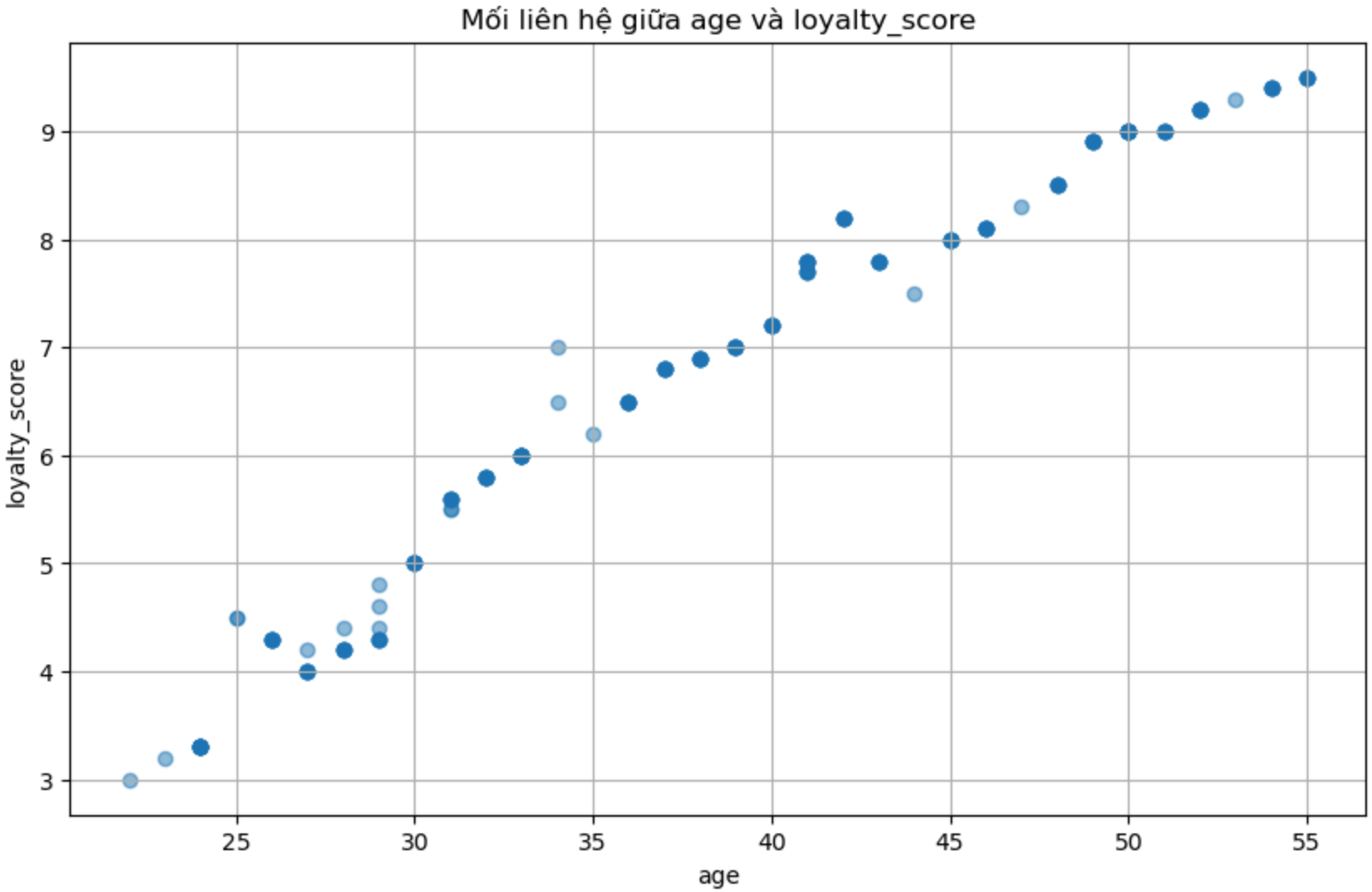
Giải quyết các yêu cầu

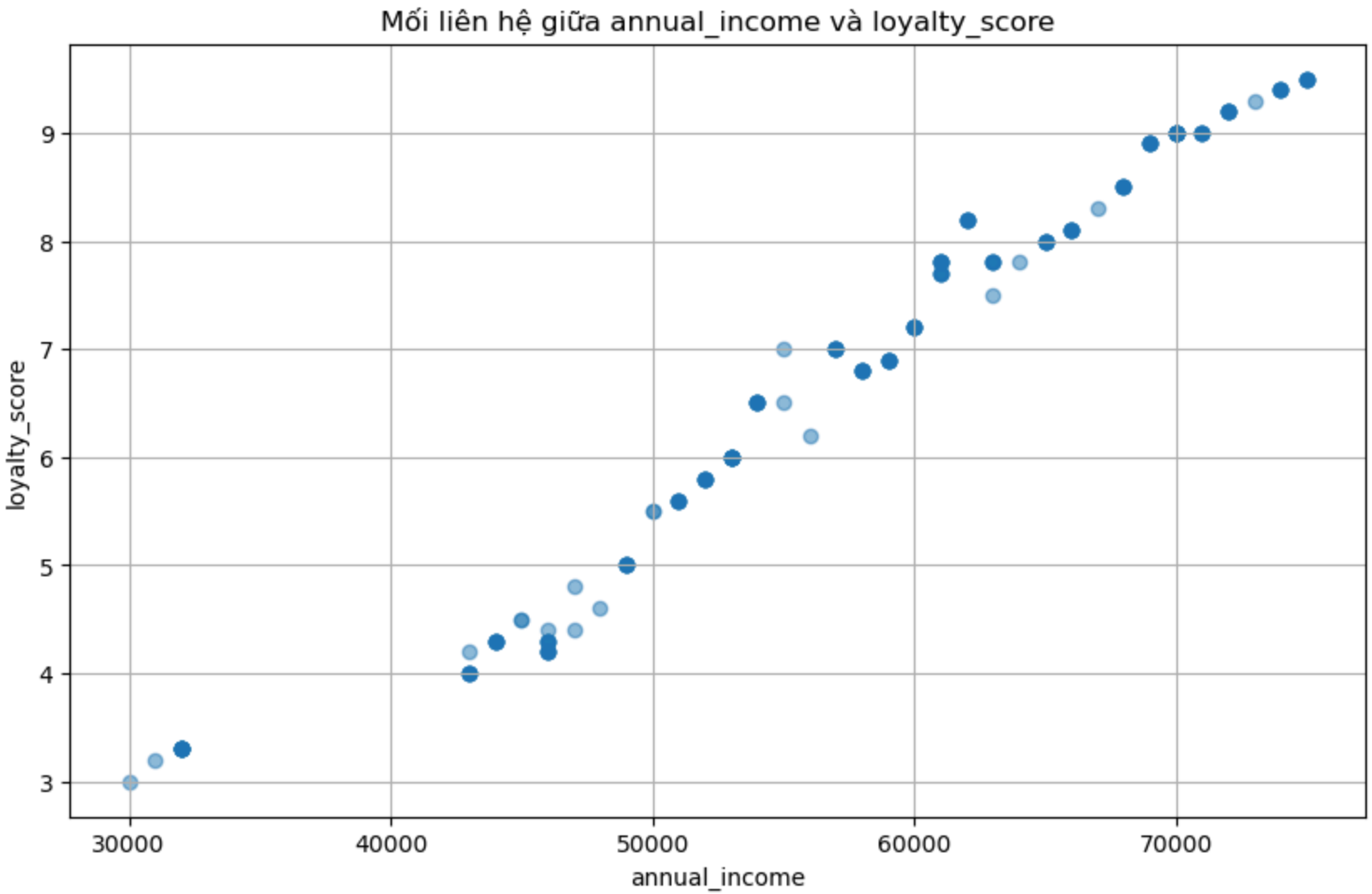
a. Mô tả đầu vào (input) và đầu ra (output) của mô hình cần được xây dựng

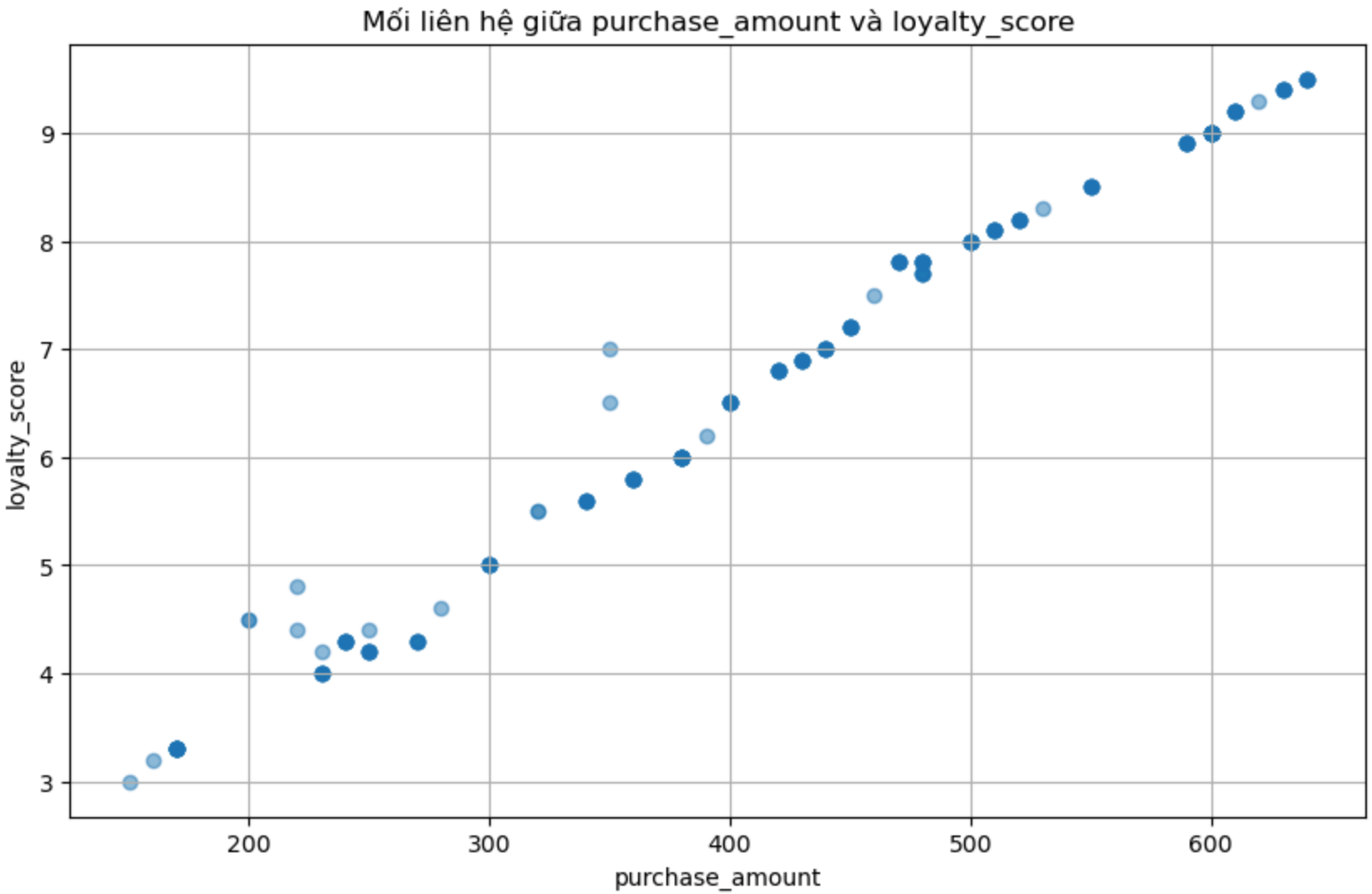
- **Input**: Vì chúng ta bỏ qua cột `user_id` và `region` trong file dữ liệu nên các cột dữ liệu đầu vào được sử dụng là `age`, `annual_income`, `purchase_amount` và `purchase_frequency`.
- **Output**: Đầu ra của mô hình là điểm thân thiết của khách hàng `loyalty_score` được dự đoán thông qua thói quen mua hàng, thể hiện ở các đặc trưng được nêu ở Input.

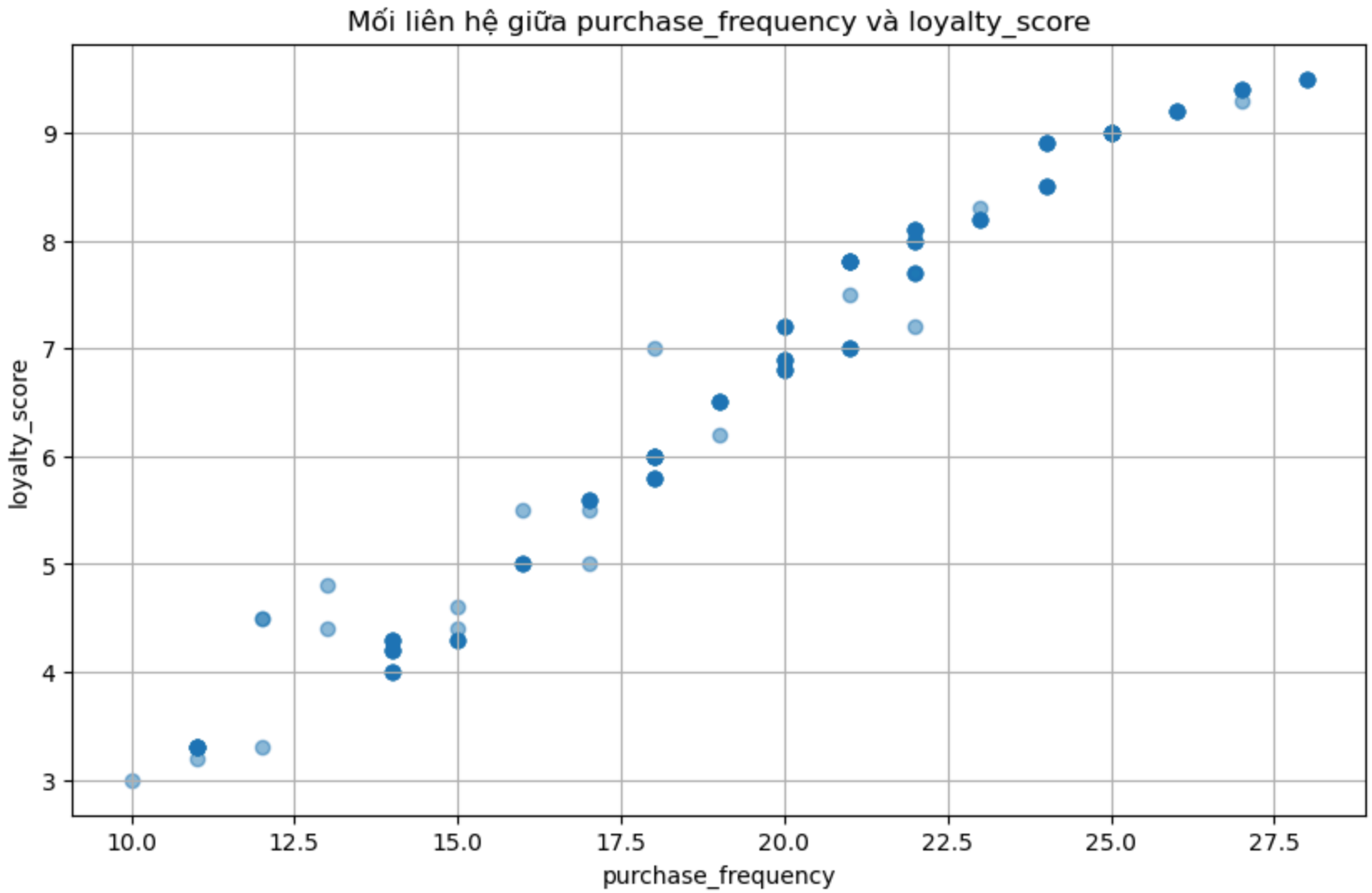
b. Sử dụng thư viện `matplotlib`, để xem mối liên hệ giữa đặc trưng thứ i và đầu ra của tập dữ liệu, vẽ biểu đồ thể hiện các điểm dữ liệu cho từng cặp (X_i, Y), trong đó X_i là đặc trưng thứ i của tập dữ liệu, và Y là đầu ra của tập dữ liệu.

In [6]:
```python
# Xác định cột mục tiêu là 'loyalty_score'
target = 'loyalty_score'

# Bỏ qua cột user_id và region
features = [col for col in df.columns if col not in ['user_id', 'region', target]]

# Vẽ biểu đồ phân tán cho mỗi cặp (feature, target)
for feature in features:
    plt.figure(figsize=(10, 6))
    plt.scatter(df[feature], df[target], alpha=0.5)
    plt.title(f'Mối liên hệ giữa {feature} và {target}')
    plt.xlabel(feature)
    plt.ylabel(target)
    plt.grid()
    plt.show()
```


Mối liên hệ giữa age và loyalty_score
loyalty_score
age
3
4
5
6
7
8
9
25
30
35
40
45
50
55


Mối liên hệ giữa annual_income và loyalty_score
loyalty_score
annual_income
3
4
5
6
7
8
9
30000
40000
50000
60000
70000


Mối liên hệ giữa purchase_amount và loyalty_score
loyalty_score
purchase_amount
3
4
5
6
7
8
9
200
300
400
500
600


Mối liên hệ giữa purchase_frequency và loyalty_score
loyalty_score
purchase_frequency
3
4
5
6
7
8
9
10.0
12.5
15.0
17.5
20.0
22.5
25.0
27.5

Nhận xét:

- Cả 4 đặc trưng đều thể hiện mối quan hệ tuyến tính với đầu ra của tập dữ liệu (điểm thân thiết của khách hàng).
- Đặc trưng `purchase_amount` là đặc trưng thể hiện quan hệ tuyến tính mạnh và rõ ràng nhất với các điểm dữ liệu tập trung gần nhau quanh một đường thẳng.
- Mối quan hệ tuyến tính của `age` và `purchase_frequency` có độ phân tán khá lớn, đặc biệt là đặc trưng `purchase_frequency`.

c) Xây dựng mô hình hồi quy tuyến tính dạng đơn giản nhất, $y = w_0 + w_1x_1 + \ldots + w_nx_n$ với n là số lượng đặc trưng trong tập dữ liệu, trong đó sử dụng toàn bộ tất cả các biến đầu vào được mô tả ở câu (a)

Các hàm cần dùng

In [7]:
```python
def ols(X, y):
    """
    Hàm triển khai OLS để tính hệ số hồi quy tuyến tính
    Công thức: beta = (X^T X)^(-1) X^T y
    """
     # Tính ma trận X^T
    X_T = transpose(X)

    # Tính (X^T X)
    X_T_X = matrix_multiply(X_T, X)
    X_T_X_inverse = inverse(X_T_X)

    # Tính ma trận (X^T y)
    X_T_y = matrix_multiply(X_T, y)

    # Tính hệ số beta
    beta = matrix_multiply(X_T_X_inverse, X_T_y)

    return beta


def predict(X, coefficients):
    """ Tính giá trị dự đoán y_hat = X * beta """

    # Thêm cột hằng số 1 để tính hệ số chặn
    X_with_intercept = [[1] + row for row in X]
```

```
    # Tính giá trị dự đoán y_hat = X * beta
    y_pred = []
    for row in X_with_intercept:
        pred = 0
        for j in range(len(coefficients)):
            pred += row[j] * coefficients[j]
        y_pred.append(pred)

    return y_pred


def evaluate_model(y_true, y_pred):
    """
    Tính chỉ số đánh giá mô hình: chọn chuẩn vector phần dư
    """

    # Tính các phần dư
    residuals = [y_true[i] - y_pred[i] for i in range(len(y_true))]

    # print("Phần dư:", residuals)
    # print("Giá trị dự đoán:", y_pred)
    # print("Giá trị thực tế:", y_true)

    # Tính chuẩn vector của phần dư
    residuals_norm = math.sqrt(sum_of_squares(residuals))

    return residuals_norm
```

In [8]:
```
# Lấy ra list các cột trong df
df_columns = df.columns.tolist()

# Cột mục tiêu là 'loyalty_score'
target = 'loyalty_score'

# Bỏ qua cột user_id và region, các cột còn là các biến dự đoán
features = [col for col in df.columns if col not in ['user_id', 'region', target]]

# Tách dữ liệu
X = df[features].values.tolist()
y = df[target].values.tolist()
```

```python
# Hiển thị thông tin cơ bản
print(f"\nSố lượng mẫu: {len(X) - 1}")
print(f"Số lượng đặc trưng: {len(X[0])}")

# Thêm cột hằng số 1 để tính hệ số chặn (intercept)
X_with_intercept = [[1] + row for row in X]

# Tính các hệ số của mô hình
coefficients = ols(X_with_intercept, y)

# Tính giá trị dự đoán
y_pred = predict(X, coefficients)

# Đánh giá mô hình
residuals_norm = evaluate_model(y, y_pred)

# In kết quả
print("\nHệ số mô hình:")
print(f"\tHệ số chặn (Intercept): {coefficients[0]:.5f}")
for i, name in enumerate(features):
    print(f"\t{name}: {coefficients[i+1]:.5f}")

# In chỉ số đánh giá mô hình
print("\nĐánh giá mô hình:")
print(f"\tChuẩn vector phần dư: {residuals_norm:.5f}")

# Phương trình hồi quy
print("\nPhương trình hồi quy:")
equation = f"loyalty_score = {coefficients[0]:.5f}"
for i, name in enumerate(features):
    coefficient = coefficients[i+1]
    sign = "+" if coefficient >= 0 else "-"
    equation += f" {sign} {abs(coefficient):.5f} * {name}"
print(equation)
```

```
Số lượng mẫu: 237
Số lượng đặc trưng: 4

Hệ số mô hình:
        Hệ số chặn (Intercept): 0.55541
        age: 0.00504
        annual_income: 0.00003
        purchase_amount: 0.01231
        purchase_frequency: -0.05978

Đánh giá mô hình:
        Chuẩn vector phần dư: 2.96641

Phương trình hồi quy:
loyalty_score = 0.55541 + 0.00504 * age + 0.00003 * annual_income + 0.01231 * purchase_amount - 0.05978 * purchase_fr
equency
```

Vậy ta có mô hình hồi quy tuyến tính dạng đơn giản nhất (sử dụng toàn bộ tất cả các biến đầu vào):

loyalty_score = 0.55541 + 0.00504 * age + 0.00003 * annual_income + 0.01231 * purchase_amount - 0.05978 * purchase_frequency

d. Xét mô hình hồi quy tuyến tính $y = w_0 + w_1 x_1$ chỉ sử dụng 1 đặc trưng duy nhất, hãy tìm đặc trưng mà mô hình hồi quy tuyến tính thể hiện tốt nhất

Ta so sánh mức độ thể hiện của các mô hình dựa vào chuẩn vector phần dư (theo nội dung học). Mô hình có chuẩn vector phần dư càng nhỏ thì thể hiện càng tốt và ngược lại.

In [9]:
```python
# Lưu đặc trưng tốt nhất (tên, chuẩn vector phần dư, hệ số)
best_feature = None

for feature in features:
    print(f"\nĐặc trưng: {feature}")

    # Lấy ra giá trị của đặc trưng hiện tại
    X_single_feature = [[row[features.index(feature)]] for row in X]

    # Thêm cột hằng số 1 để tính hệ số chặn w0
    X_single_feature_with_intercept = [[1] + row for row in X_single_feature]

    # Tính các hệ số
```

```
    coefficients_single = ols(X_single_feature_with_intercept, y)

    # Tính giá trị dự đoán
    y_pred_single = predict(X_single_feature, coefficients_single)

    # Đánh giá mô hình
    residuals_norm_single = evaluate_model(y, y_pred_single)

    # Lưu đặc trưng mà mô hình thể hiện tốt nhất
    if best_feature is None or residuals_norm_single < best_feature[1]:
        best_feature = (feature, residuals_norm_single, coefficients_single)

    # In kết quả
    print(f"\tHệ số chặn (Intercept): {coefficients_single[0]:.5f}")
    print(f"\tHệ số của x_1: {coefficients_single[1]:.5f}")
    print(f"\tChuẩn vector phần dư: {residuals_norm_single:.5f}")

# In đặc trưng tốt nhất
if best_feature:
    print(f"\nĐặc trưng tốt nhất: {best_feature[0]}, với chuẩn vector phần dư: {best_feature[1]:.5f}")

# Phương trình hồi quy
if best_feature:
    feature = best_feature[0]
    coefficient = best_feature[2]
    equation_best = f"loyalty_score = {coefficient[0]:.5f} {'+' if coefficient[1] >= 0 else '-'} {abs(coefficient[1])
    print(f"\nPhương trình hồi quy cho đặc trưng tốt nhất ({feature}):")
    print(equation_best)
```

```
Đặc trưng: age
        Hệ số chặn (Intercept): -0.91772
        Hệ số của x_1: 0.19939
        Chuẩn vector phần dư: 5.54697

Đặc trưng: annual_income
        Hệ số chặn (Intercept): -2.61616
        Hệ số của x_1: 0.00016
        Chuẩn vector phần dư: 5.15179

Đặc trưng: purchase_amount
        Hệ số chặn (Intercept): 1.05653
        Hệ số của x_1: 0.01348
        Chuẩn vector phần dư: 3.15841

Đặc trưng: purchase_frequency
        Hệ số chặn (Intercept): -1.33863
        Hệ số của x_1: 0.41078
        Chuẩn vector phần dư: 4.70057

Đặc trưng tốt nhất: purchase_amount, với chuẩn vector phần dư: 3.15841

Phương trình hồi quy cho đặc trưng tốt nhất (purchase_amount):
loyalty_score = 1.05653 + 0.01348 * purchase_amount
```

Vậy đặc trưng mà mô hình hồi quy tuyến tính thể hiện tốt nhất là `purchase_amount`.

e. Thiết kế một mô hình hồi quy tuyến tính khác với những mô hình trên mà cho kết quả tốt nhất.

Vì ta chỉ cần tính chất "tuyến tính" cho các tham số w_i, còn x_i có thể ở bất kì dạng nào nên ta tiến hành một số phép biến đổi lên x_i, chẳng hạn như: bình phương, lập phương, lấy căn bậc hai, căn bậc 3, lấy log.

Ta tiến hành thay đổi cho từng đặc trưng và chọn ra phép biến đổi tốt nhất cho đặc trưng đó, từ đó ta có được tập các phép biến đổi tốt nhất tương ứng cho từng đặc trưng để xây dựng mô hình hồi quy tuyến tính mới với kết quả tốt hơn các mô hình trước đó.

Kết quả các phép biến đổi tốt nhất là:

- age: Logarith
- annual_income: Bình phương
- purchase_amount: Gốc

- purchase_frequency: Gốc

Mô hình hồi quy:

loyalty_score = -7.7399237400 + 3.1074771436 * age (Logarith) - 0.0000000001 * annual_income (Bình phương) + 0.0097509272 * purchase_amount (Gốc) - 0.0261511036 * purchase_frequency (Gốc)

Chuẩn vector phần dư: 2.8604340657 (e) < 2.9664091744 (c) < 3.1584141192 (d)

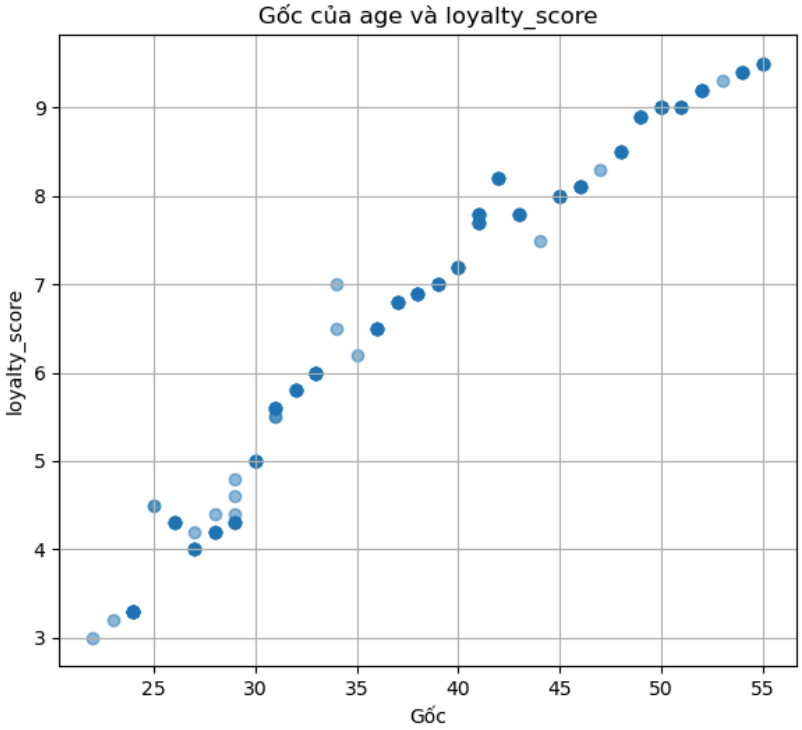
In [10]:
```
# Lần lượt lưu các phương pháp biến đổi tốt nhất cho từng đặc trưng
best_methods = {}

for feature in features:
    print(f"\nĐặc trưng: {feature}")

    # Lấy dữ liệu của đặc trưng hiện tại
    feature_data = df[feature].values.tolist()

    # Biến đổi dữ liệu
    transformations = {
        'Gốc': feature_data,
        'Bình phương': [x ** 2 for x in feature_data],
        'Căn bậc hai': [math.sqrt(x) if x >= 0 else 0 for x in feature_data],
        'Logarith': [math.log(x) if x > 0 else 0 for x in feature_data],
        'Lập phương': [x ** 3 for x in feature_data],
        'Căn bậc ba': [x ** (1/3) for x in feature_data]
    }

    # Vẽ biểu đồ cho từng biến đổi để so sánh trực quan
    plt.figure(figsize=(15, 20))
    for i, (transform_name, transformed_data) in enumerate(transformations.items()):
        plt.subplot(3, 2, i + 1)
        plt.scatter(transformed_data, df[target], alpha=0.5)
        plt.title(f'{transform_name} của {feature} và {target}')
        plt.xlabel(transform_name)
        plt.ylabel(target)
        plt.grid()

    # Tính chuẩn vector phần dư cho từng biến đổi
    for transform_name, transformed_data in transformations.items():
```

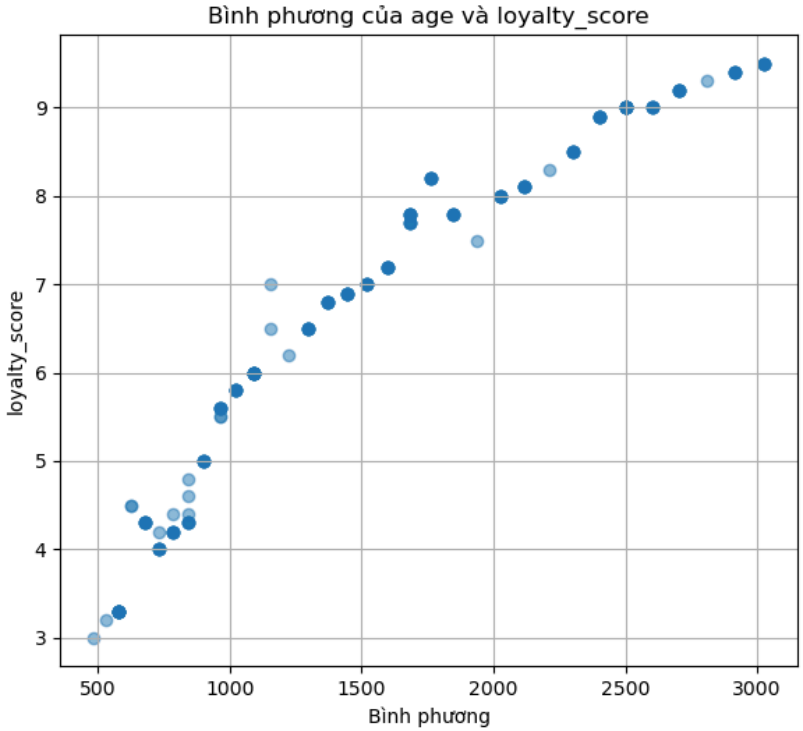
```
        # Tách dữ liệu thành
        X_transformed = [[x] for x in transformed_data]

        # Thêm cột hằng số 1 để tính hệ số chặn w0
        X_transformed_with_intercept = [[1] + row for row in X_transformed]

        # Tính các hệ số
        coefficients_transformed = ols(X_transformed_with_intercept, y)

        # Tính giá trị dự đoán
        y_pred_transformed = predict(X_transformed, coefficients_transformed)

        # Đánh giá mô hình
        residuals_norm_transformed = evaluate_model(y, y_pred_transformed)

        # In kết quả
        # print(f"\t{transform_name}:")
        # print(f"\t\tHệ số chặn (Intercept): {coefficients_transformed[0]:.5f}")
        # print(f"\t\tHệ số của {feature}: {coefficients_transformed[1]:.5f}")
        # print(f"\t\tChuẩn vector phần dư: {residuals_norm_transformed:.5f}")

        # So sánh giữa các biến đổi và in ra kết quả tốt nhất
        if transform_name == 'Gốc':
            best_transform = transform_name
            best_residual = residuals_norm_transformed
        else:
            if residuals_norm_transformed < best_residual:
                best_transform = transform_name
                best_residual = residuals_norm_transformed

    print(f"Biến đổi tốt nhất: {best_transform}, với chuẩn vector của phần dư: {best_residual:.5f}")
    best_methods[feature] = best_transform


# In ra các phương pháp biến đổi tốt nhất cho từng đặc trưng
print("\nCác phương pháp biến đổi tốt nhất cho từng đặc trưng:")
for feature, method in best_methods.items():
    print(f"\t{feature}: {method}")

# Thực hiện hồi quy tuyến tính
X_best = []
```

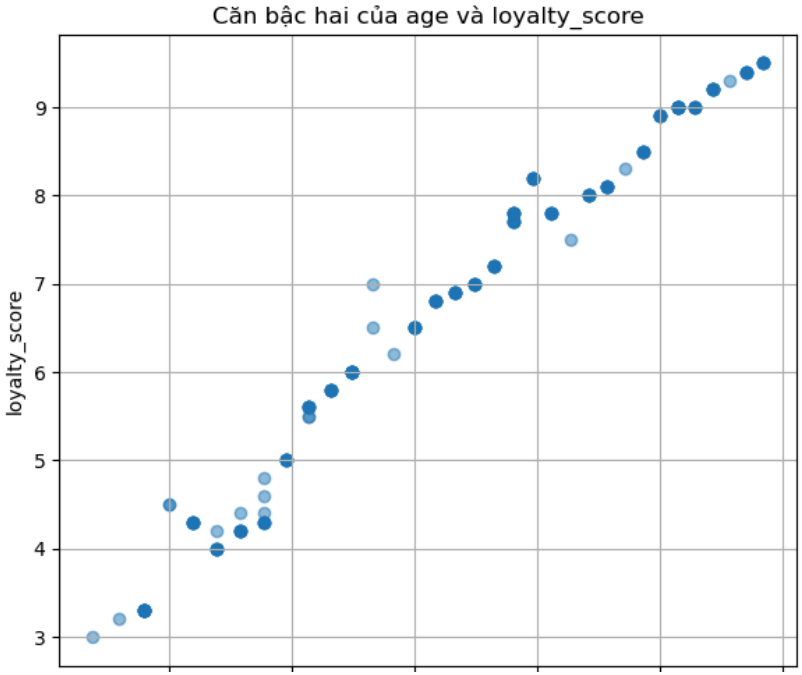
```python
for feature in features:

    feature_data = df[feature].values.tolist()

    if best_methods[feature] == 'Gốc':
        X_best.append(feature_data)

    elif best_methods[feature] == 'Bình phương':
        X_best.append([x ** 2 for x in feature_data])

    elif best_methods[feature] == 'Căn bậc hai':
        X_best.append([math.sqrt(x) if x >= 0 else 0 for x in feature_data])

    elif best_methods[feature] == 'Logarith':
        X_best.append([math.log(x) if x > 0 else 0 for x in feature_data])

    elif best_methods[feature] == 'Lập phương':
        X_best.append([x ** 3 for x in feature_data])

    elif best_methods[feature] == 'Căn bậc ba':
        X_best.append([x ** (1/3) for x in feature_data])


# Chuyển vị
X_best = transpose(X_best)

# Thêm cột hằng số 1 để tính hệ số chặn
X_best_with_intercept = [[1] + row for row in X_best]

# Tính các hệ số
coefficients_best = ols(X_best_with_intercept, y)

# Tính giá trị dự đoán
y_pred_best = predict(X_best, coefficients_best)

# Đánh giá mô hình
residuals_norm_best = evaluate_model(y, y_pred_best)


# In kết quả
print("\nHệ số mô hình:")
print(f"\tHệ số chặn (Intercept): {coefficients_best[0]:.10f}")
```

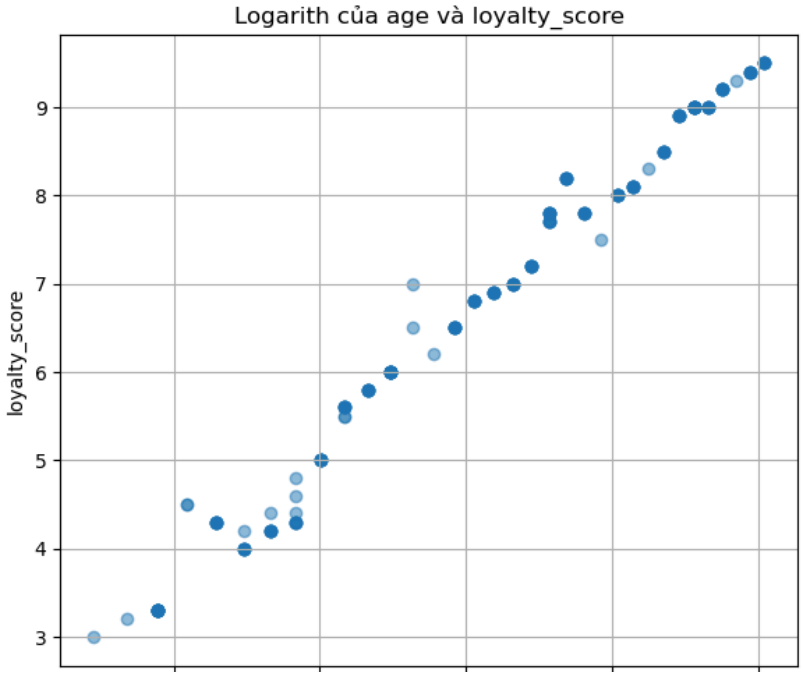
```python
for i, feature in enumerate(features):
    print(f"\tHệ số của {feature}: {coefficients_best[i+1]:.10f}")

# In chỉ số đánh giá mô hình
print("\nĐánh giá mô hình:")
print(f"\tChuẩn vector phần dư của mô hình hiện tại: {residuals_norm_best:.10f}")

# So sánh với các mô hình trước
print(f"\tChuẩn vector phần dư mô hình ban đầu: {residuals_norm:.10f}")
print(f"\tChuẩn vector phần dư mô hình đơn đặc trưng tốt nhất: {best_feature[1]:.10f}")

if residuals_norm_best < residuals_norm and residuals_norm_best < best_feature[1]:
    print("Mô hình hiện tại cho kết quả tốt nhất so với cac mô hình trên.")

# Phương trình hồi quy
print("\nPhương trình hồi quy:")
equation_best = f"loyalty_score = \n{coefficients_best[0]:.10f}"
for i, feature in enumerate(features):
    coefficient = coefficients_best[i+1]
    sign = "+" if coefficient >= 0 else "-"
    equation_best += f"\n{sign} {abs(coefficient):.10f} * {feature} ({best_methods[feature]})"

print(equation_best)
```

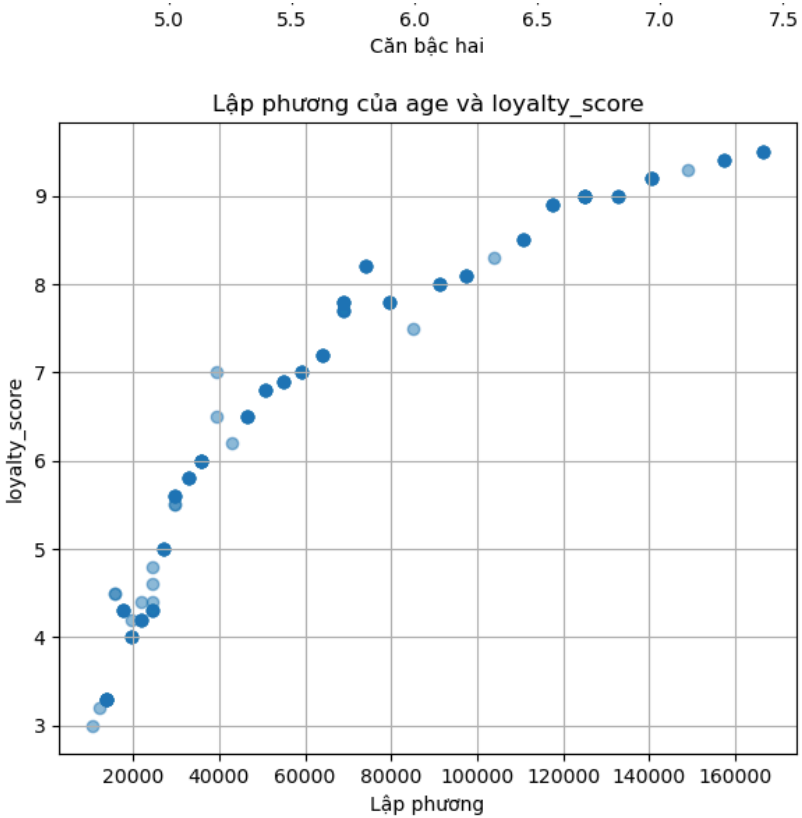
```
Đặc trưng: age
Biến đổi tốt nhất: Logarith, với chuẩn vector của phần dư: 3.69054

Đặc trưng: annual_income
Biến đổi tốt nhất: Bình phương, với chuẩn vector của phần dư: 5.11571

Đặc trưng: purchase_amount
Biến đổi tốt nhất: Gốc, với chuẩn vector của phần dư: 3.15841

Đặc trưng: purchase_frequency
Biến đổi tốt nhất: Gốc, với chuẩn vector của phần dư: 4.70057

Các phương pháp biến đổi tốt nhất cho từng đặc trưng:
        age: Logarith
        annual_income: Bình phương
        purchase_amount: Gốc
        purchase_frequency: Gốc

Hệ số mô hình:
        Hệ số chặn (Intercept): -7.7399237400
        Hệ số của age: 3.1074771436
        Hệ số của annual_income: -0.0000000001
        Hệ số của purchase_amount: 0.0097509272
        Hệ số của purchase_frequency: -0.0261511036

Đánh giá mô hình:
        Chuẩn vector phần dư của mô hình hiện tại: 2.8604340657
        Chuẩn vector phần dư mô hình ban đầu: 2.9664091744
        Chuẩn vector phần dư mô hình đơn đặc trưng tốt nhất: 3.1584141192
Mô hình hiện tại cho kết quả tốt nhất so với cac mô hình trên.

Phương trình hồi quy:
loyalty_score =
-7.7399237400
+ 3.1074771436 * age (Logarith)
- 0.0000000001 * annual_income (Bình phương)
+ 0.0097509272 * purchase_amount (Gốc)
- 0.0261511036 * purchase_frequency (Gốc)
```

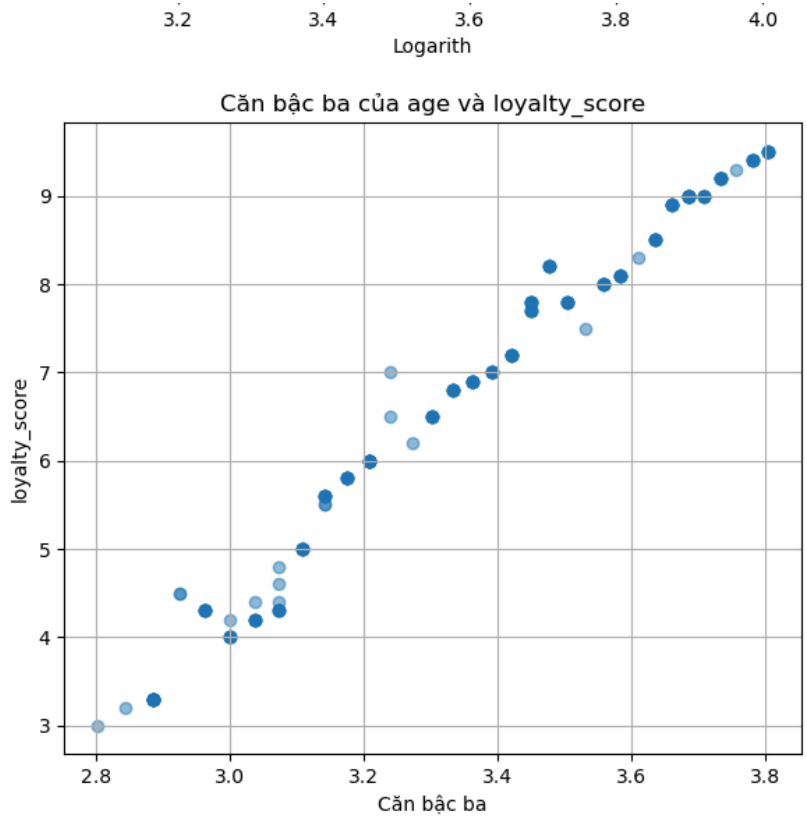

Gốc của age và loyalty_score
loyalty_score
Gốc
3
4
5
6
7
8
9
25
30
35
40
45
50
55

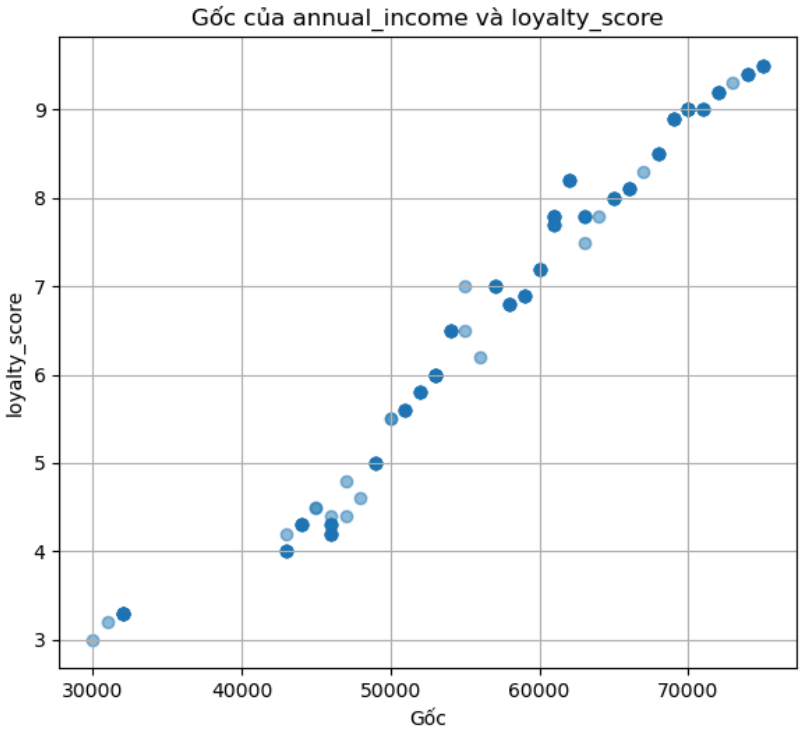

Bình phương của age và loyalty_score
loyalty_score
Bình phương
3
4
5
6
7
8
9
500
1000
1500
2000
2500
3000

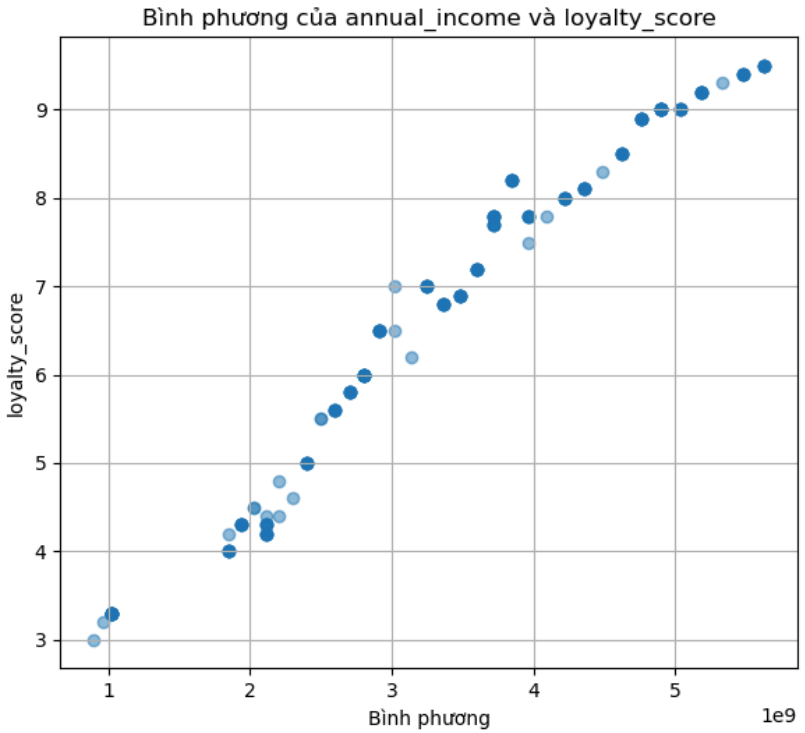

Căn bậc hai của age và loyalty_score
loyalty_score
3
4
5
6
7
8
9

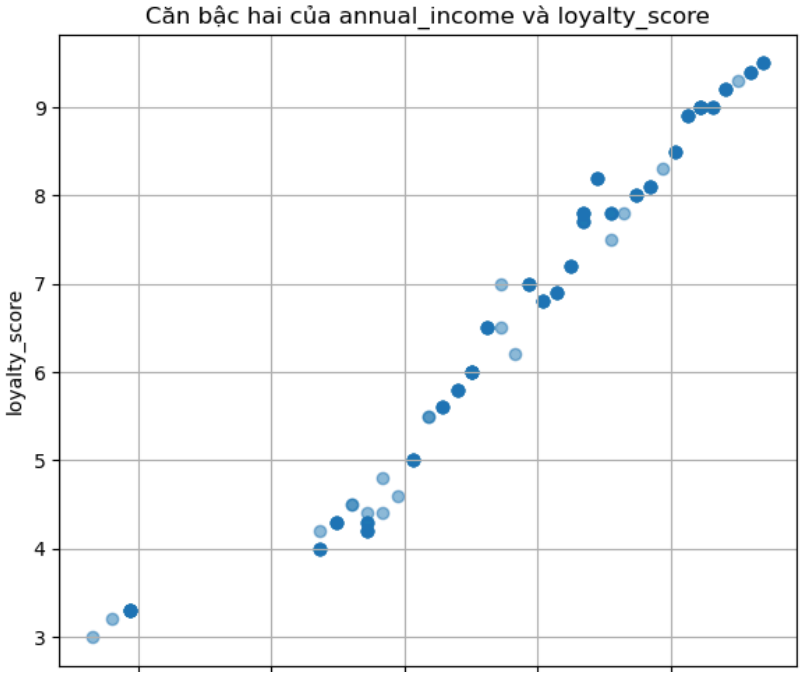

Logarith của age và loyalty_score
loyalty_score
3
4
5
6
7
8
9

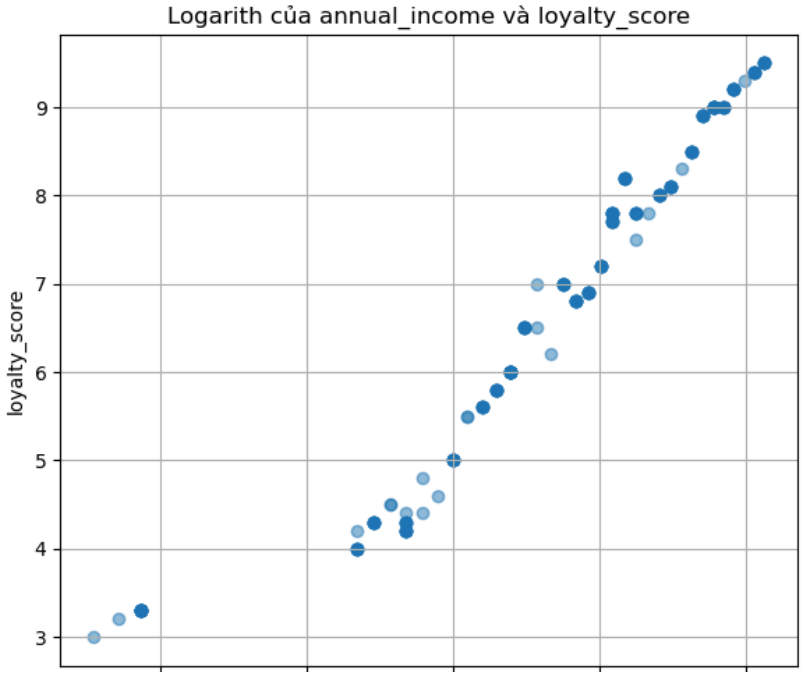

5.0
5.5
6.0
6.5
7.0
7.5
Căn bậc hai
Lập phương của age và loyalty_score
loyalty_score
3
4
5
6
7
8
9
20000
40000
60000
80000
100000
120000
140000
160000
Lập phương

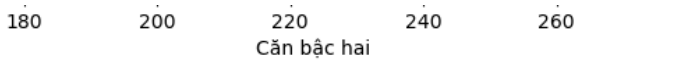

3.2
3.4
3.6
3.8
4.0
Logarith
Căn bậc ba của age và loyalty_score
loyalty_score
3
4
5
6
7
8
9
2.8
3.0
3.2
3.4
3.6
3.8
Căn bậc ba

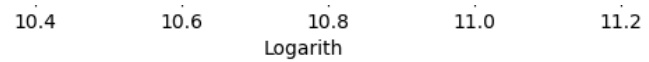

Gốc của annual_income và loyalty_score
loyalty_score
Gốc
30000
40000
50000
60000
70000
3
4
5
6
7
8
9

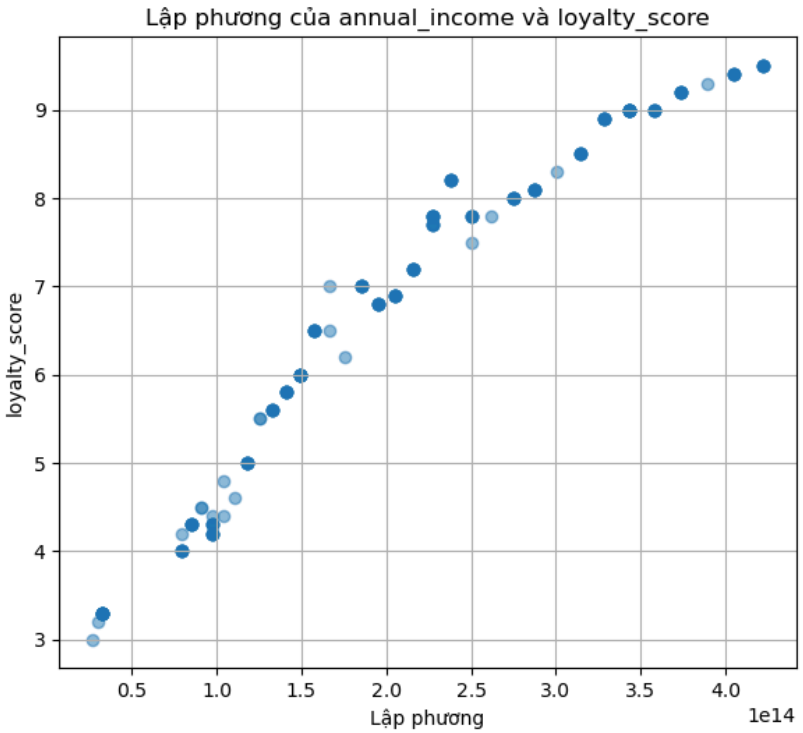

Bình phương của annual_income và loyalty_score
loyalty_score
Bình phương
1e9
1
2
3
4
5
3
4
5
6
7
8
9

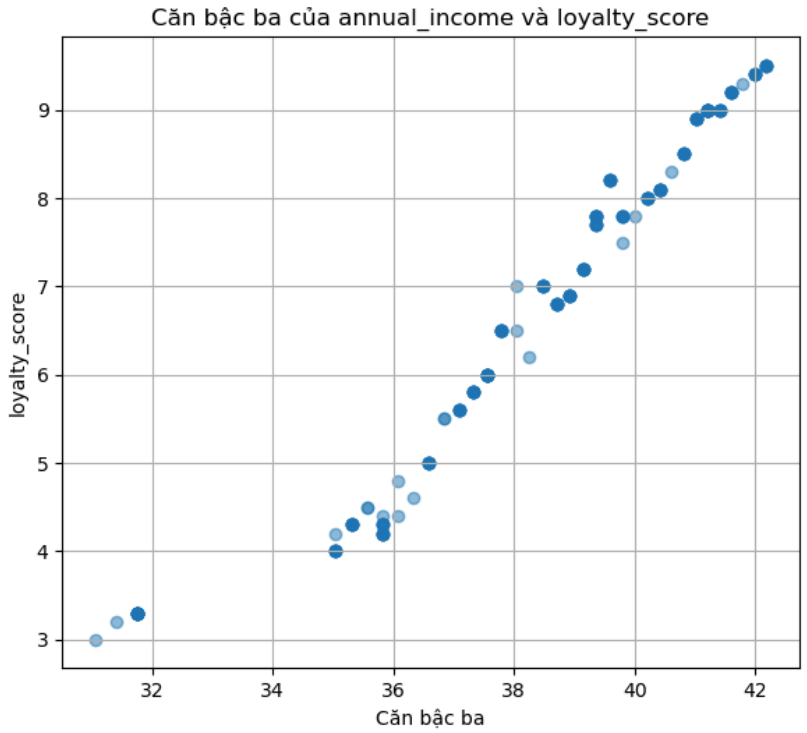

Căn bậc hai của annual_income và loyalty_score
loyalty_score
3
4
5
6
7
8
9

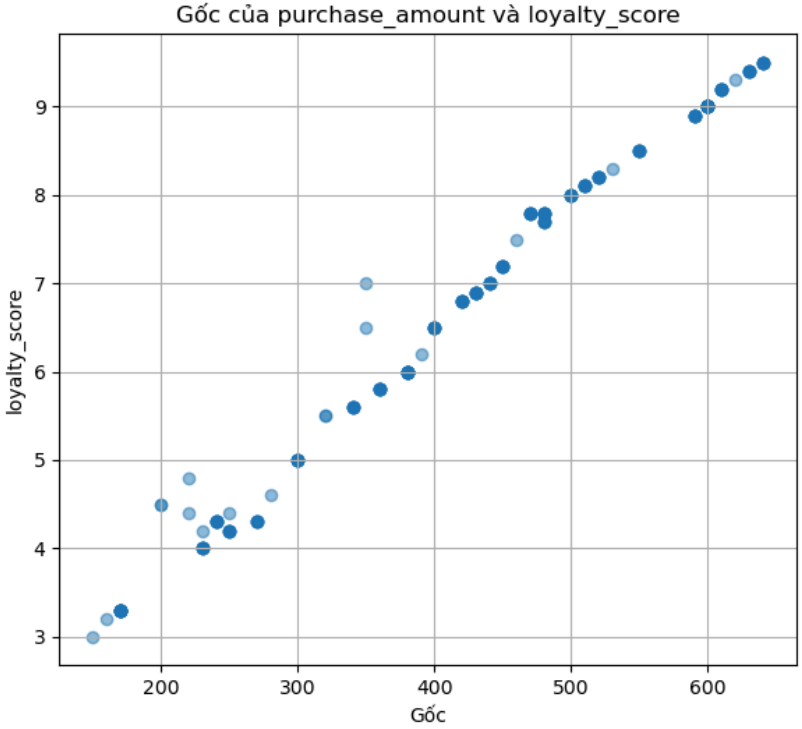

Logarith của annual_income và loyalty_score
loyalty_score
3
4
5
6
7
8
9

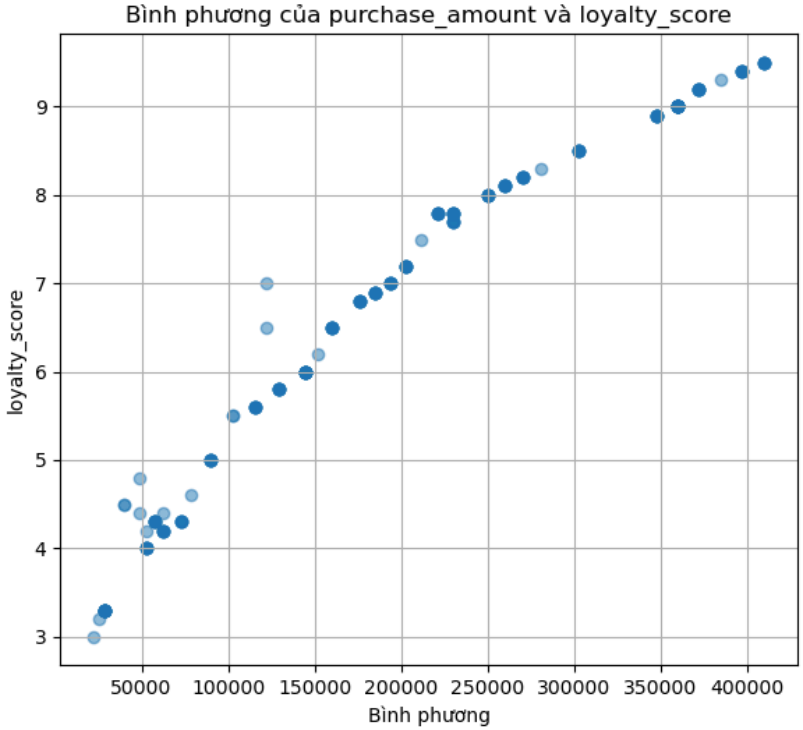

180
200
220
240
260
Căn bậc hai

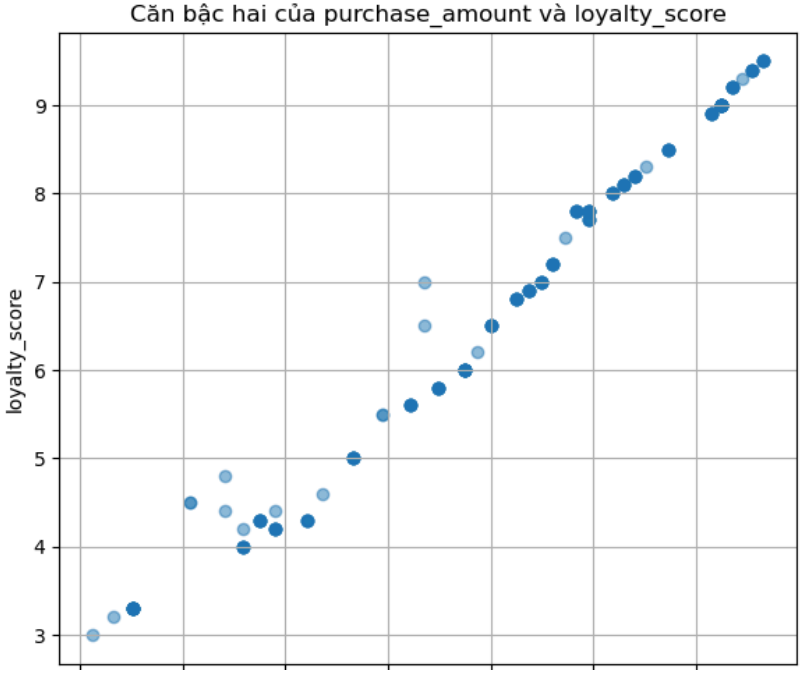

10.4
10.6
10.8
11.0
11.2
Logarith

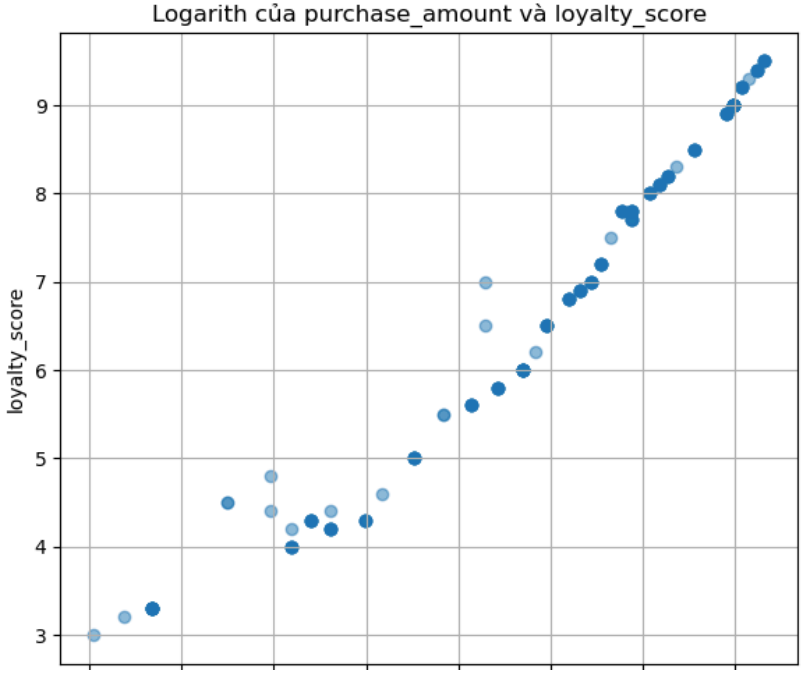

Lập phương của annual_income và loyalty_score
loyalty_score
3
4
5
6
7
8
9
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
1e14
Lập phương

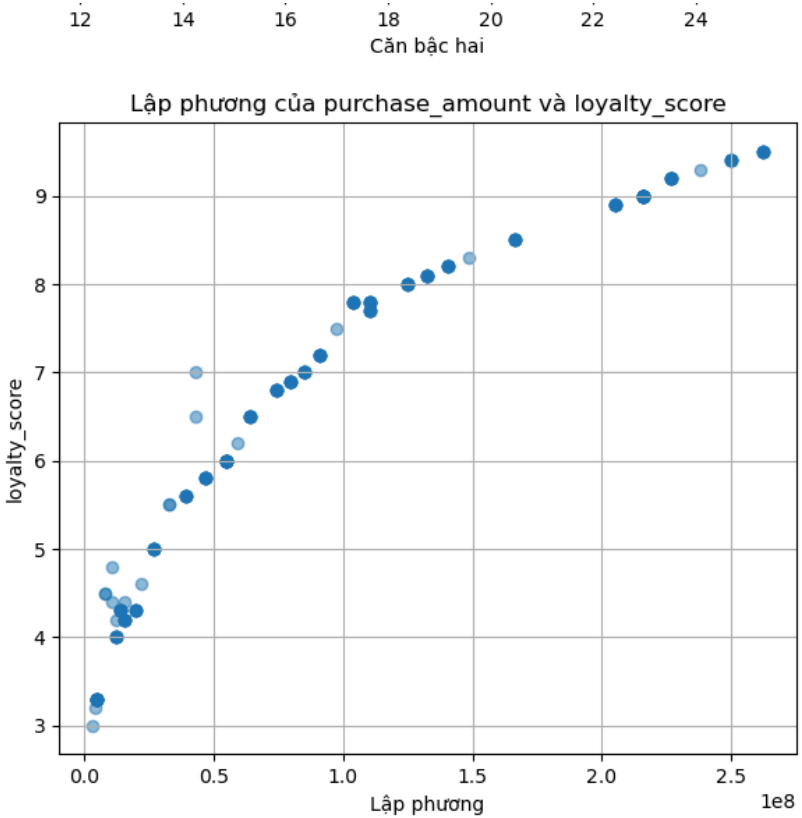

Căn bậc ba của annual_income và loyalty_score
loyalty_score
3
4
5
6
7
8
9
32
34
36
38
40
42
Căn bậc ba

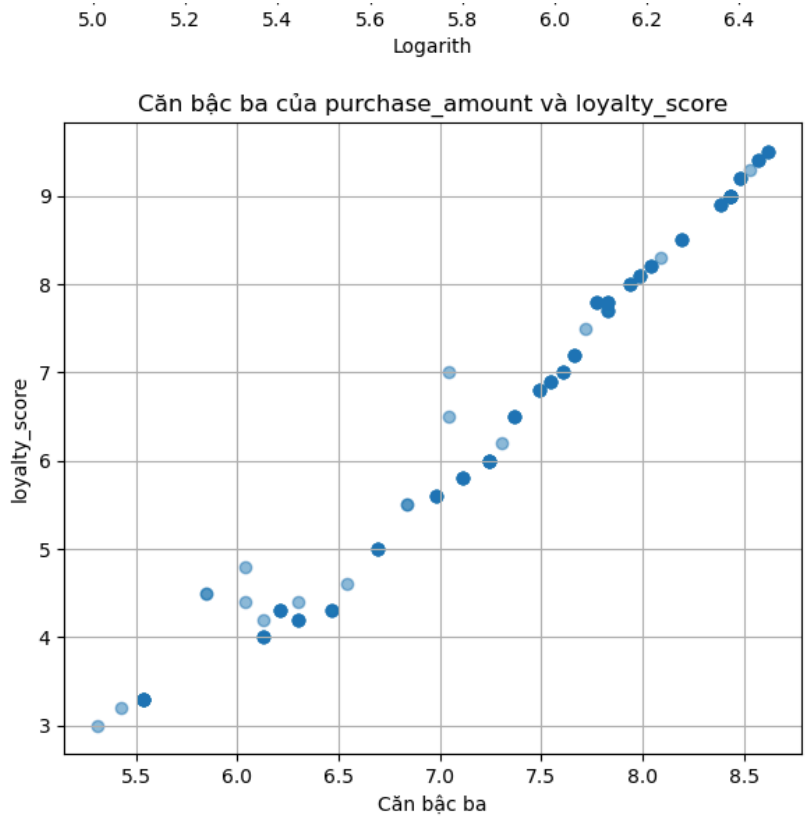

Gốc của purchase_amount và loyalty_score
loyalty_score
Gốc
3
4
5
6
7
8
9
200
300
400
500
600


Bình phương của purchase_amount và loyalty_score
loyalty_score
Bình phương
3
4
5
6
7
8
9
50000
100000
150000
200000
250000
300000
350000
400000


Căn bậc hai của purchase_amount và loyalty_score
loyalty_score
3
4
5
6
7
8
9

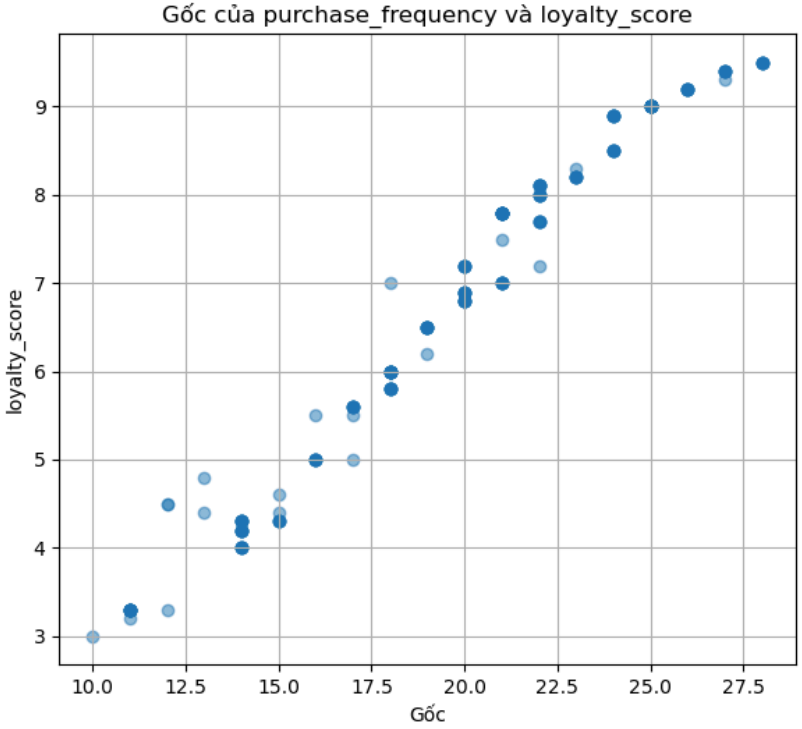

Logarith của purchase_amount và loyalty_score
loyalty_score
3
4
5
6
7
8
9

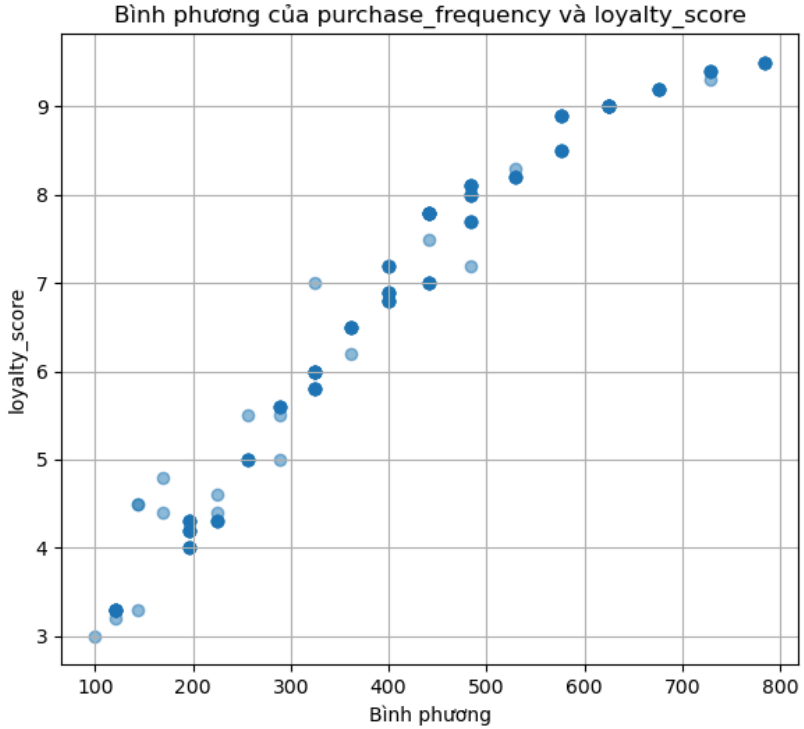

Lập phương của purchase_amount và loyalty_score
loyalty_score
Lập phương
1e8
0.0
0.5
1.0
1.5
2.0
2.5
3
4
5
6
7
8
9

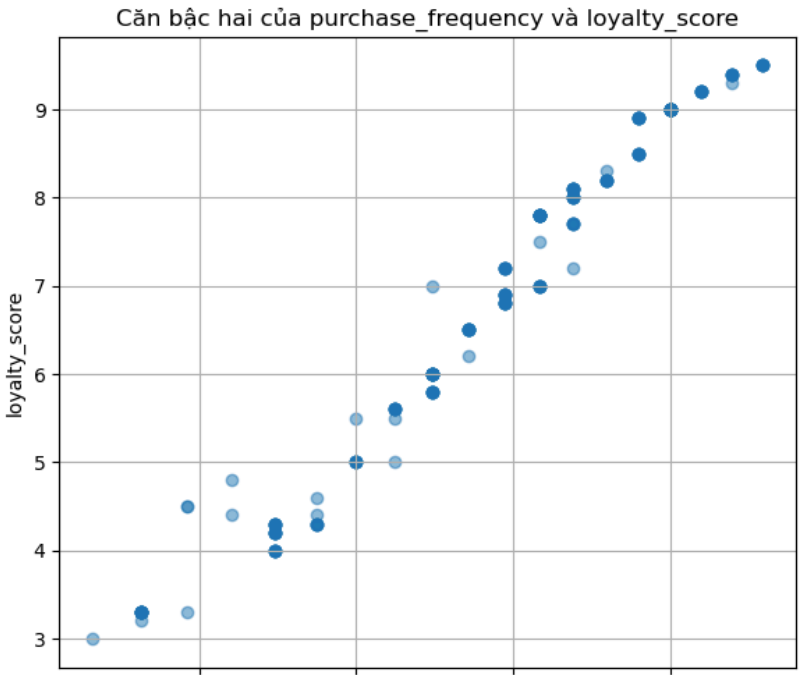

Căn bậc ba của purchase_amount và loyalty_score
loyalty_score
Căn bậc ba
5.5
6.0
6.5
7.0
7.5
8.0
8.5
3
4
5
6
7
8
9

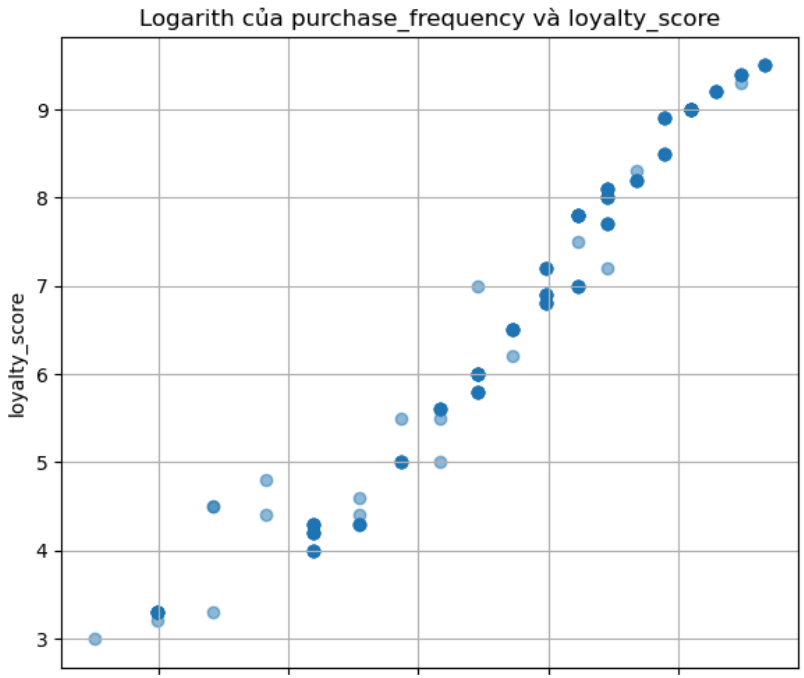
12
14
16
18
20
22
24
Căn bậc hai

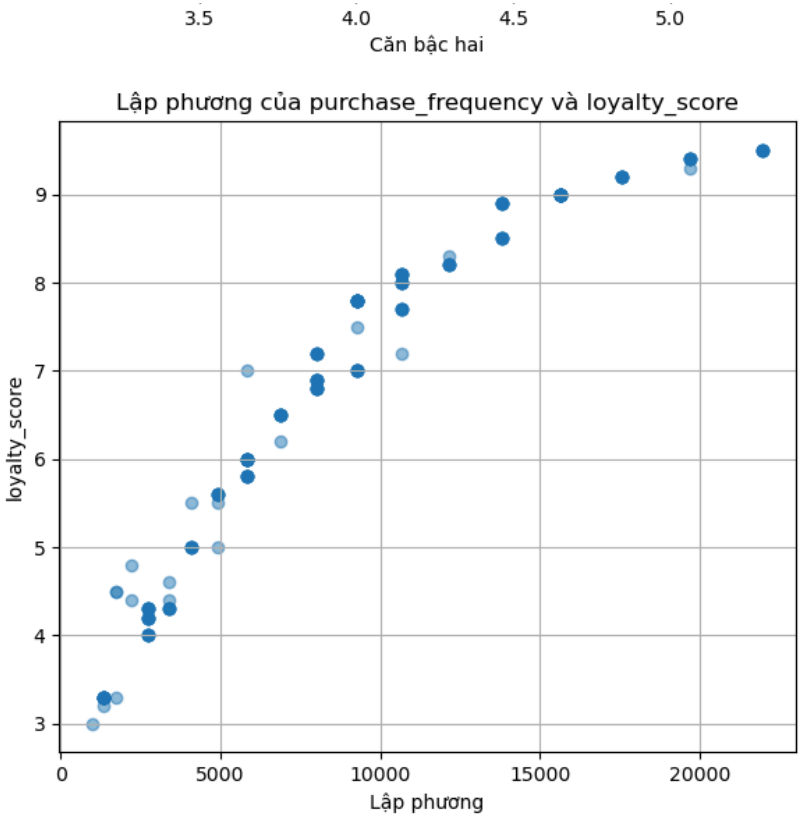
5.0
5.2
5.4
5.6
5.8
6.0
6.2
6.4
Logarith

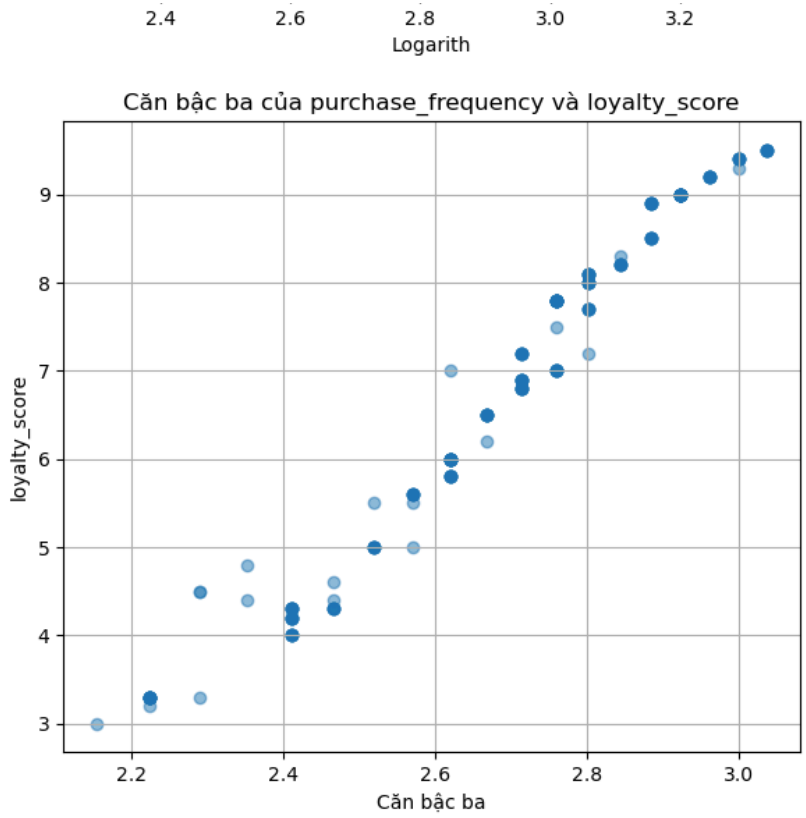

Gốc của purchase_frequency và loyalty_score
loyalty_score
Gốc
3
4
5
6
7
8
9
10.0
12.5
15.0
17.5
20.0
22.5
25.0
27.5


Bình phương của purchase_frequency và loyalty_score
loyalty_score
Bình phương
3
4
5
6
7
8
9
100
200
300
400
500
600
700
800


Căn bậc hai của purchase_frequency và loyalty_score
loyalty_score
3
4
5
6
7
8
9


Logarith của purchase_frequency và loyalty_score
loyalty_score
3
4
5
6
7
8
9


3.5
4.0
4.5
5.0
Căn bậc hai
Lập phương của purchase_frequency và loyalty_score
loyalty_score
3
4
5
6
7
8
9
0
5000
10000
15000
20000
Lập phương


2.4
2.6
2.8
3.0
3.2
Logarith
Căn bậc ba của purchase_frequency và loyalty_score
loyalty_score
3
4
5
6
7
8
9
2.2
2.4
2.6
2.8
3.0
Căn bậc ba