```
import pandas as pd
import statsmodels.api as sm
import numpy as np
from scipy.stats import norm

# 数据读取和参数设置
df = pd.read_excel('aaaa.xls')
df['const'] = 1

CPC = 0.673
price_com = 8.187
comment_com = 0

# 定义变量名集合
dummy_vars = [f'dummy_{i}' for i in list(range(2, 60)) + list(range(61, 125)) + list(range(126, 151))]
base_vars = ['const', 'cost', 'cost2', 'display_quantity', 'phrase', 'exact', 'price_com', 'comment_com']

# 定义回归变量集
xm_vars = base_vars + ['jiaohu1', 'jiaohu2', 'jiaohu3', 'jiaohu4'] + dummy_vars
my_vars = base_vars + ['click_number', 'jiaohu5', 'jiaohu6'] + dummy_vars
xy_vars = base_vars + dummy_vars

def calculate_effects(xm_params, my_params, xy_params):
    """计算各种效应"""
    effect_x_on_m = (xm_params['cost'] + xm_params['cost2'] * CPC +
                     xm_params['jiaohu3'] * price_com + xm_params['jiaohu4'] * price_com * CPC +
                     xm_params['jiaohu1'] * comment_com + xm_params['jiaohu2'] * comment_com * CPC)

    effect_m_on_y = (my_params['click_number'] + my_params['jiaohu5'] * price_com +
                     my_params['jiaohu6'] * comment_com)

    direct_effect = my_params['cost'] + my_params['cost2'] * CPC

    indirect_effect = effect_x_on_m * effect_m_on_y

    total_effect = xy_params['cost'] + xy_params['cost2'] * CPC

    return effect_x_on_m, effect_m_on_y, direct_effect, indirect_effect, total_effect

def run_regressions(data):
    """运行三个回归模型"""
    med = data['click_number']
    y = data['sales_volume']

    # 运行回归
    xm_result = sm.OLS(med, data[xm_vars]).fit(cov_type='HC3')
    my_result = sm.OLS(y, data[my_vars]).fit(cov_type='HC3')
    xy_result = sm.OLS(y, data[xy_vars]).fit(cov_type='HC3')
```

```
    return calculate_effects(xm_result.params, my_result.params, xy_result.params)

def bootstrap_analysis(n_bootstrap=1000):
    """Bootstrap 分析"""
    print("开始 Bootstrap 分析...")

    # 原始样本计算
    original_effects = run_regressions(df)
    effect_x_on_m, effect_m_on_y, direct_effect, indirect_effect, total_effect = original_effects

    # Bootstrap 采样
    bootstrap_results = {
        'effect_x_on_m': [],
        'effect_m_on_y': [],
        'direct_effect': [],
        'indirect_effect': [],
        'total_effect': []
    }

    for i in range(n_bootstrap):
        if (i + 1) % 100 == 0:
            print(f"Bootstrap 进度: {i + 1}/{n_bootstrap}")

        # 重采样
        boot_data = df.sample(n=len(df), replace=True)
        boot_effects = run_regressions(boot_data)

        # 存储结果
        for j, key in enumerate(bootstrap_results.keys()):
            bootstrap_results[key].append(boot_effects[j])

    # 计算统计量
    results = {}
    effect_names = ['effect_x_on_m', 'effect_m_on_y', 'direct_effect', 'indirect_effect', 'total_effect']
    original_values = [effect_x_on_m, effect_m_on_y, direct_effect, indirect_effect, total_effect]

    for i, (name, original_val) in enumerate(zip(effect_names, original_values)):
        boot_values = np.array(bootstrap_results[name])

        # 计算统计量
        std_err = np.std(boot_values)
        z_value = original_val / std_err if std_err != 0 else 0
        p_value = 2 * (1 - norm.cdf(abs(z_value)))

        # 计算置信区间
        ci_lower = np.percentile(boot_values, 2.5)
        ci_upper = np.percentile(boot_values, 97.5)

        results[name] = {
```

```
                'estimate': original_val,
                'std_err': std_err,
                'z_value': z_value,
                'p_value': p_value,
                'ci_lower': ci_lower,
                'ci_upper': ci_upper
            }

    return results

def print_results(results):
    """打印格式化结果"""
    print("\n" + "="*80)
    print("Bootstrap 中介效应分析结果")
    print("="*80)

    effect_labels = {
        'effect_x_on_m': 'X 对 M 的效应',
        'effect_m_on_y': 'M 对 Y 的效应',
        'direct_effect': '直接效应',
        'indirect_effect': '间接效应',
        'total_effect': '总效应'
    }

    for name, label in effect_labels.items():
        result = results[name]
        print(f"\n{label}:")
        print(f"  估计值: {result['estimate']:.6f}")
        print(f"  标准误: {result['std_err']:.6f}")
        print(f"  Z 值: {result['z_value']:.4f}")
        print(f"  P 值: {result['p_value']:.4f}")
        print(f"  95%置信区间: [{result['ci_lower']:.6f}, {result['ci_upper']:.6f}]")

        # 显著性标记
        if result['p_value'] < 0.001:
            sig = "***"
        elif result['p_value'] < 0.01:
            sig = "**"
        elif result['p_value'] < 0.05:
            sig = "*"
        else:
            sig = ""
        print(f"  显著性: {sig}")

# 运行分析
if __name__ == "__main__":
    results = bootstrap_analysis(1000)
    print_results(results)
```