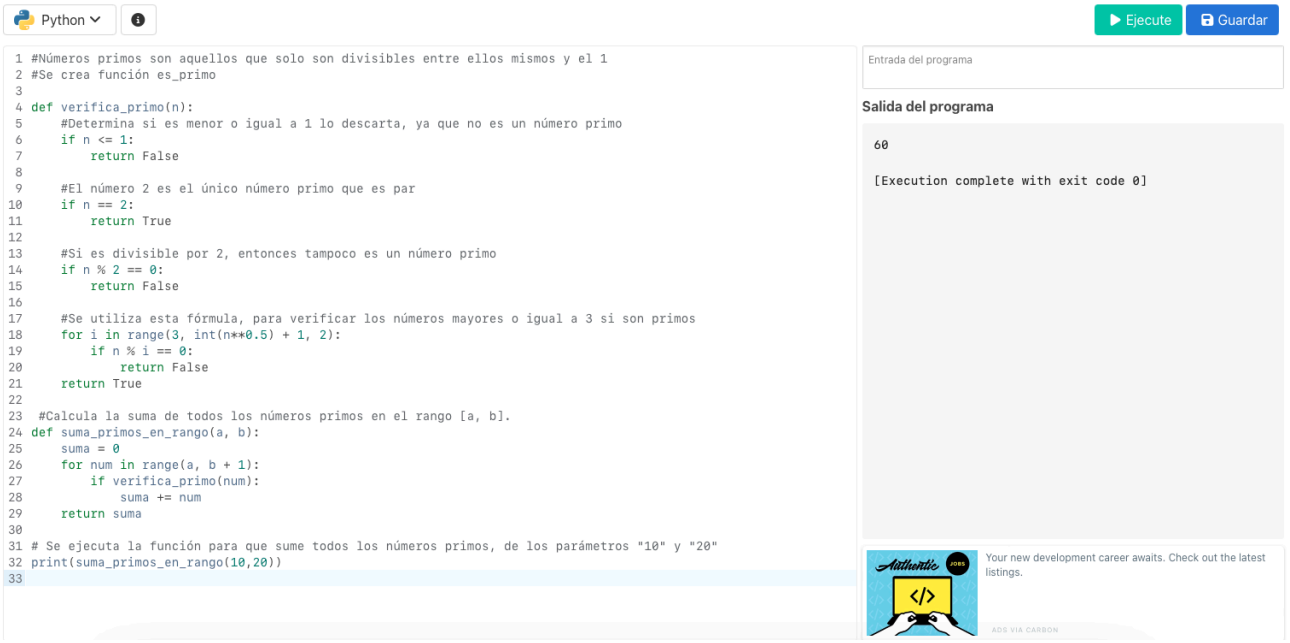
URL GitHUB: https://github.com/luis1021/examen_chn_ingeniero_datos

Problema 1: Resuelve en Python

```
#Números primos son aquellos que solo son divisibles entre ellos mismos y el 1
#Se crea función es_primo

def verifica_primo(n):
    #Determina si es menor o igual a 1 lo descarta, ya que no es un número primo
    if n <= 1:
        return False

    #El número 2 es el único número primo que es par
    if n == 2:
        return True

    #Si es divisible por 2, entonces tampoco es un número primo
    if n % 2 == 0:
        return False

    #Se utiliza esta fórmula, para verificar los números mayores o igual a 3 si son primos
    for i in range(3, int(n**0.5) + 1, 2):
        if n % i == 0:
            return False
    return True

 #Calcula la suma de todos los números primos en el rango [a, b].
def suma_primos_en_rango(a, b):
    suma = 0
    for num in range(a, b + 1):
        if verifica_primo(num):
            suma += num
    return suma

# Se ejecuta la función para que sume todos los números primos, de los parámetros "10" y
"20"
print(suma_primos_en_rango(10,20))
```

Se utiliza la página “https://www.mycompiler.io/es/new/python” para poder probar la función realizada.



Problema 2: Realizar Datamart

```
/*Entidad de Clientes, se agrega campo nombre de Cliente*/
CREATE TABLE Customer_dim (
        customer_dim_id         NUMBER PRIMARY KEY,
        customer_name           VARCHAR2(500),
        customer_age            NUMBER,
        customer_income         NUMBER,
        employment_duration     NUMBER,
        historical_default      CHAR(1),
        cred_hist_length        NUMBER,
        created_date            DATE,
        modified_date           DATE
);
```

```
/*Se propone una entidad de propiedades que tiene un cliente, debido a que podría tener
varias propiedades y dar como garantía alguna si fuere el caso en un préstamo, y el tipo de
propiedad según lo describe en el documento (Alquiler, Propiedad, Hipoteca)*/
CREATE TABLE customer_ownership_dim(
        customer_ownership_dim_id NUMBER PRIMARY KEY,
        customer_dim_id           NUMBER,
        home_address              VARCHAR2(500),
        home_ownership_type       VARCHAR2(50), /*home_ownership*/
        created_date              DATE,
        modified_date             DATE,
        FOREIGN KEY (customer_dim_id) REFERENCES Customer_dim(customer_dim_id)
);

/*Entidad de tasa de interés vigente*/
CREATE TABLE Interest_rate_dim(
        interest_rate_dim_id      NUMBER PRIMARY KEY,
        interest_value            NUMBER, /*loan_int_rate*/
        created_date              DATE,
        modified_date             DATE
);


/*Entidad de préstamos que tiene un cliente*/
CREATE TABLE Loan_fact (
        loan_fact_id              NUMBER PRIMARY KEY,
        customer_dim_id           NUMBER,
        customer_ownership_dim_id NUMBER,
        interest_rate_dim_id      NUMBER,
        loan_intent               VARCHAR2(50),
        loan_grade                VARCHAR2(10),
        loan_amnt                 NUMBER,
        term_years                NUMBER,
        current_loan_status       VARCHAR2(20),
        created_date              DATE,
        modified_date             DATE,
        FOREIGN KEY (customer_dim_id) REFERENCES
Customer_dim(customer_dim_id),
        FOREIGN KEY (customer_ownership_dim_id) REFERENCES
customer_ownership_dim(customer_ownership_dim_id),
        FOREIGN KEY (interest_rate_dim_id) REFERENCES
Interest_rate_dim(interest_rate_dim_id),
);
```

```
/*DataMart Resumen con la información relevante del préstamo asociado a un cliente*/
CREATE TABLE Loan_customer_dm(
        loan_fact_id                NUMBER,
        loan_amnt                   NUMBER,
        loan_intent                 VARCHAR2(50),
        loan_grade                  VARCHAR2(10),
        current_loan_status         VARCHAR2(20),
        term_years                  NUMBER,
        customer_dim_id             NUMBER,
        customer_name               VARCHAR2(500),
        customer_ownership_dim_id NUMBER,
        home_address                VARCHAR2(500),
        home_ownership_type         VARCHAR2(50),
        interest_rate_dim_id        NUMBER,
        interest_value              NUMBER
);
```

Problema 3: Análisis de datos del Covid-19

```
#1
import pandas as pd

# Lee y carga el archivo CSV con la librería importada
DataFrame df = pd.read_csv("country_wise_latest.csv")

#2

# Renombrar algunas columnas para hacerlas más descriptivas
df.rename(columns={'Deaths / 100 Cases': 'Average Deaths', Recovered / 100 Cases:
'Average Recovered', 'Deaths / 100 Recovered': 'Average number of recovered deaths '},
inplace=True)

#3
#Calcular la media de las siguientes columnas
media_confirmed = df['Confirmed'].mean()
media_deaths = df['deaths'].mean()
media_recovered = df['recovered'].mean()
media_active = df['active'].mean()

#Calcular la mediana de las siguientes columnas
mediana_confirmed = df['Confirmed'].median()
mediana_deaths = df['deaths'].median()
mediana_recovered = df['recovered'].median()
mediana_active = df['active'].median()
```

```
#4
#Columna Calculada de cantidad de pacientes vivos
df['Final_active_people'] = df['Confirmed] - df['Deaths'] - df[New Deaths]

#5
#Agrupar por Country y sumatoria de Campos
grouped_data = df.groupby('Country/Region').agg({'Confirmed: 'sum', 'Deaths': 'sum',
'Recovered': 'sum', 'Final_active_people': 'sum'})

#6
#Ordenar los países por el número total de casos confirmados
sorted_data = grouped_data.sort_values(by='Confirmed', ascending=False)

# Mostrar los 10 países con más casos confirmados
top_10_countries = sorted_data.head(10)

#7
# Filtrar los datos para un país específico (por ejemplo, Guatemala)
us_data = df[df['Country/Region'] == 'Guatemala']
# Visualizar la evolución de casos confirmados, muertes y recuperaciones a lo largo del
tiempo

plt.plot(us_data['Date'], us_data['Confirmed], label='Confirmed Cases')
plt.plot(us_data['Date'], us_data['Deaths'], label='Confirmed Deaths')
plt.plot(us_data['Date'], us_data['Recovered'], label=Confirmed Recovered')
plt.xlabel('Date')
plt.ylabel('Number of Cases')
plt.title('COVID-19 Cases in Guatemala')
plt.legend() plt.xticks(rotation=45)
plt.show()
```

Problema 4: Consultas SQL para un banco con cuentas monetarias, cheques y préstamos.

```
/*Query 1 */
Select  a.nombre,
        a.direccion,
        a.telefono,
        b.id_cuenta,
        b.saldo
from    clientes a,
        cuentas b
where  a.id_cliente = b.id_cliente
order   by a.nombre asc;
```

```
/*Query 2 */
Select sum(saldo)
from   cuentas;

/*Query 3*/
Select *
from    (
          Select  a.nombre,
                  b.id_cuenta,
                  sum(b.saldo) saldo
          from    clientes a,
                  cuentas b
          where  a.id_cliente = b.id_cliente
          group   by  a.nombre, b.id_cuenta
          )
where  rownum <= 5
order   by saldo desc;

/*Query 4*/
Select  a.id_cheque,
          c.nombre,
          a.monto,
          a.fecha_emision,
          a.beneficiario
from     cheques a,
          cuentas b
          clientes c
where  a.id_cuenta = b.id_cuenta
and      b.id_cliente = c.id_cliente
and      a.fecha_emision = trunc(sysdate,'MM')
order   by a.fecha_emision desc;

/*Query 5*/
Select  c.nombre,
          sum(a.monto) monto_total_cheques
from     cheques a,
          cuentas b
          clientes c
where  a.id_cuenta = b.id_cuenta
and      b.id_cliente = c.id_cliente
and      a.fecha_emision = trunc(sysdate,'MM')
group  by c.nombre
order   by 2 desc;
```

```
/*Query 6*/
Select sum(a.monto) monto_total
from    prestamos a
where EXTRACT(YEAR FROM a.fecha_otorgamiento) = EXTRACT(YEAR FROM
SYSDATE);

/*Query 7*/
Select *
from    (
          Select  b.nombre,
                  sum(a.monto) monto_total
          from    prestamos a
                  clientes b
          where  a.id_cliente = b.id_cliente
          and     a.fecha_otorgamiento between trunc(sysdate, 'YEAR')
          and    add_months(trunc(sysdate, 'YEAR'), 4) - 1
          group  by b.nombre
          )
where rownum <=10
order  by monto_total desc
```