PROCESO GESTIÓN DEL TALENTO HUMANO

FORMATO INFORME MENSUAL EJECUCIÓN CONTRACTUAL

Barranquilla, Mayo de 2022

Señor (a)
MANUEL GREGORIO HORMECHEA LANCE
SUPERVISOR(A) CONTRATO No. **3298199**
Coordinador Sede TIC
TIC Y Energía
Barranquilla

Asunto: Informe mensual de ejecución contractual Mes Mayo del año 2022

Referencia: No 3298199 del año 2022

LILIOLA PAOLA PRINS BALLESTAS, identificado con la cédula de ciudadanía No. 32.780.488 de Barranquilla, en mi calidad de Contratista del SENA, en TIC Y Energía, en cumplimiento del Contrato de Prestación de Servicios de la referencia, a continuación presento el Informe de actividades realizadas en el mes objeto de cobro.

Valor y forma de Pago: e). un pago correspondiente al mes de Mayo de 2022 por valor de (3.477.822). Los honorarios serán pagados por el SENA al contratista de acuerdo al cronograma definido por la Dirección Administrativa y Financiera de la Dirección General, en la cuenta de Ahorros No. 620280107 BBVA, cuyo titular es el Contratista.

Plazo: Será hasta el 17 de Diciembre de 2022.

OBJETO:
Prestar los servicios profesionales como Instructor para impartir formación en Programación de Software de acuerdo con la planeación de la coordinación de TIC y Energía del Centro Nacional Colombo Alemán

Obligaciones Especificas: *(Trascriba las obligaciones especificas del contrato, dentro del siguiente cuadro)*

No	Obligaciones	Acciones realizadas	Evidencias
1	Impartir formación a las fichas asignadas	Seleccionar las estrategias de enseñanza, aprendizaje y evaluación pertinentes para el grupo. Orientar el proceso de aprendizaje de acuerdo con el diseño curricular vigente. Reportar la información académica y administrativa que se generen en el proceso.	Se realizó solicitud de creación de las fichas.

2	Impartir formación a las fichas asignadas	Sesiones de orientación mediadas por tecnología (Google Meeting - WhatsApp)	Capturas de pantalla tomadas de las sesiones
3	Impartir formación a las fichas asignadas	Espacios Virtuales disponibles en la plataforma institucional	Materiales y proyectos publicados en la plataforma

A continuación relaciono los desplazamientos que realicé previo a la presentación de este informe. Una vez finalizado cada desplazamiento presenté al ordenador del gasto el informe en el Formato Informe Legalización Desplazamiento Contratista GTH-F-087, en el que se describieron las actividades desarrolladas y los resultados de cada desplazamiento. Cada informe cuenta con el visto bueno del Supervisor.

Se lista a continuación el soporte de la legalización de los desplazamientos realizados, los cuales forman parte integral del presente informe de ejecución contractual.

ITEM	No DE LA ORDEN DE VIAJE	LUGAR DE DESPLAZAMIENTO	FECHAS DE DESPLAZAMIENTOS	
			INICIO	FINALIZACIÓN

Nota 1: *Por cada desplazamiento que haya realizado el contratista, adjuntará el respectivo informe que la soporte. En caso de haber realizado el desplazamiento en fecha posterior a la presentación del informe de ejecución contractual, deberá reportarlo en el siguiente informe de ejecución contractual.*

Para el trámite de la cuenta me permito adjuntar: Documentos electrónicos enunciados como evidencias del cumplimiento de las obligaciones contractuales y los desplazamientos realizados y el No. 7834783888 de la planilla, operador y periodo. (Decreto Ley 2106 de 2019 – " *Decreto Ley Antitrámites"*)

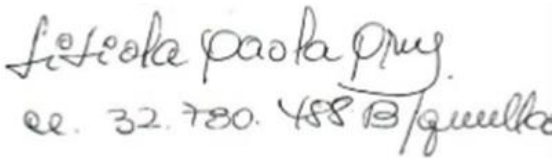
Declaración bajo la gravedad de juramento.

Me permito declarar **bajo la gravedad de juramento** que durante el periodo objeto de cobro SI____NO_X__ he suscrito otro u otros contratos con entidade(s) del Estado el (los) cual(es) se encuentra(n) en ejecución; la anterior declaración de acuerdo con lo establecido en el numeral 5º de la Directiva Presidencial No. 001 de 2022, a fin de identificar potenciales conflictos de interés y hacer la gestión del riesgos para la toma de decisiones dentro del principio de la debida diligencia frente a la ejecución de los contratos de prestación de servicios profesionales o de apoyo a la gestión.

Nota: En el evento de señalar **SI,** el análisis efectuado por el supervisor y ordenador de pago deberá quedar documentado y publicado en el SECOP II en los documentos relacionados en el archivo Gestión Contractual GC_ del periodo objeto de cobro.

Evidencias en () folios

Cordialmente,



LILIOLA PAOLA PRINS BALLESTAS
Contratista
C.C. No. 32.780.488

Recibí a satisfacción:

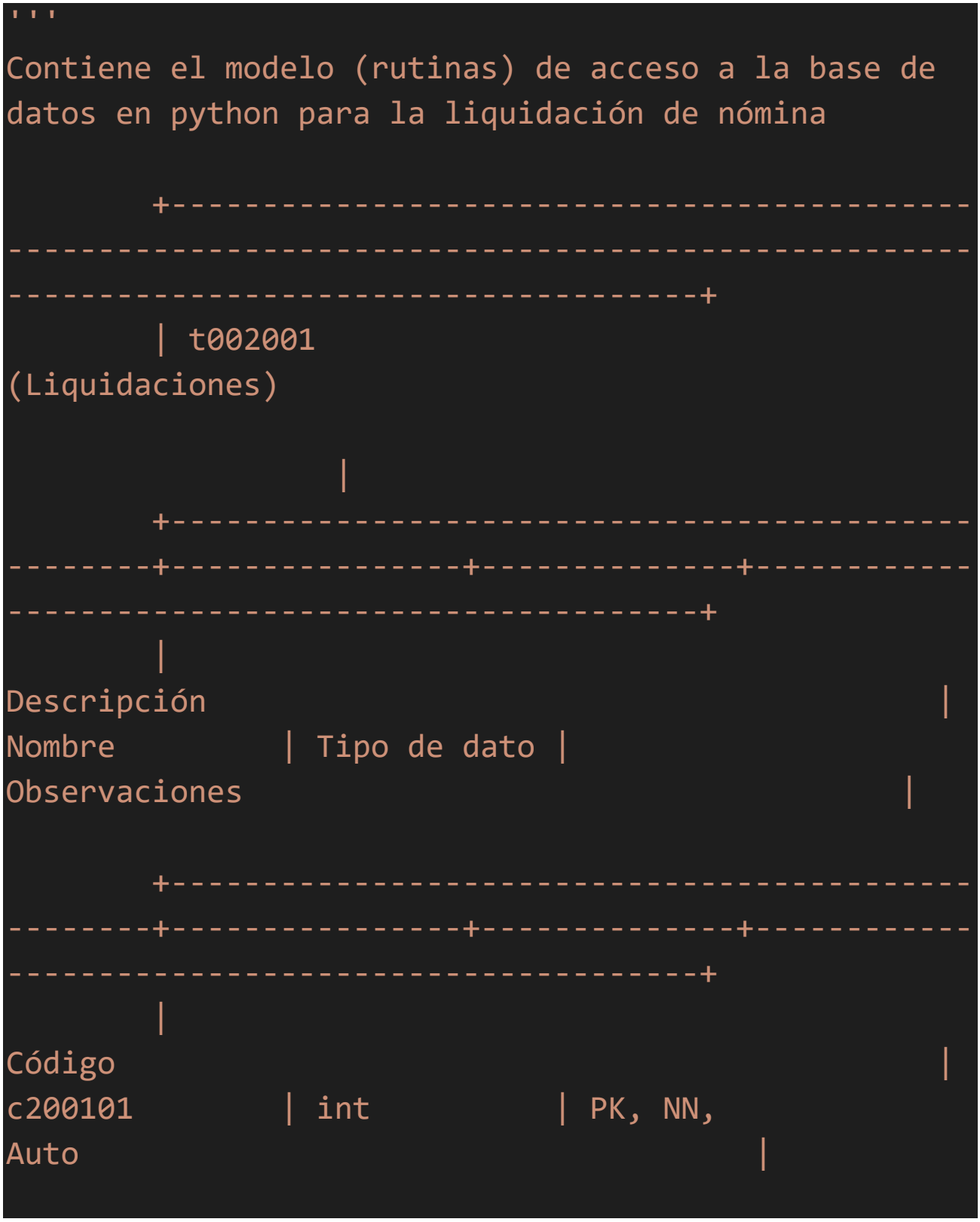
MANUEL GREGORIO HORMECHEA LANCE
Supervisor(a) Contrato 3298199 de 2022
Coordinador Sede TIC

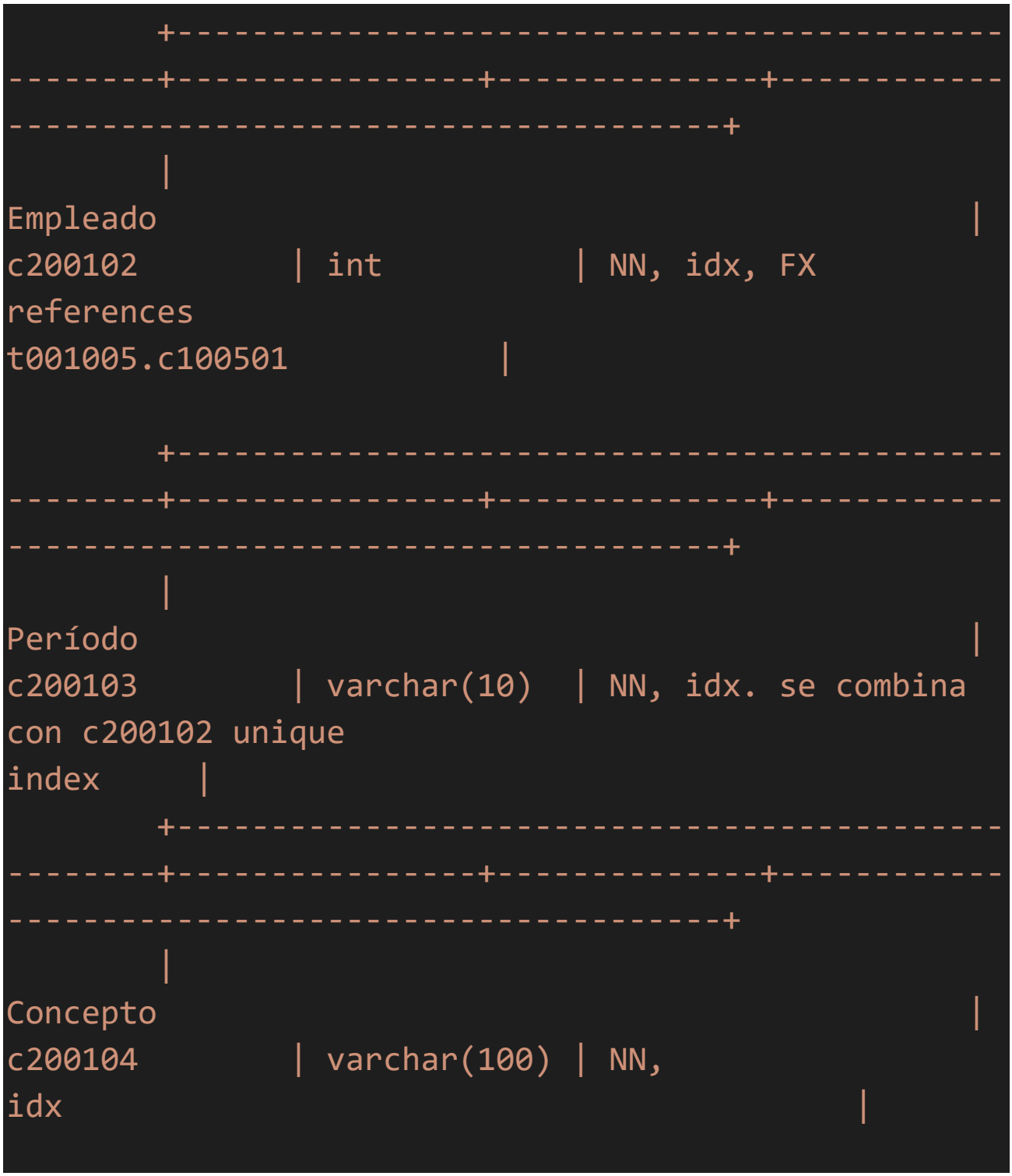
EVIDENCIAS

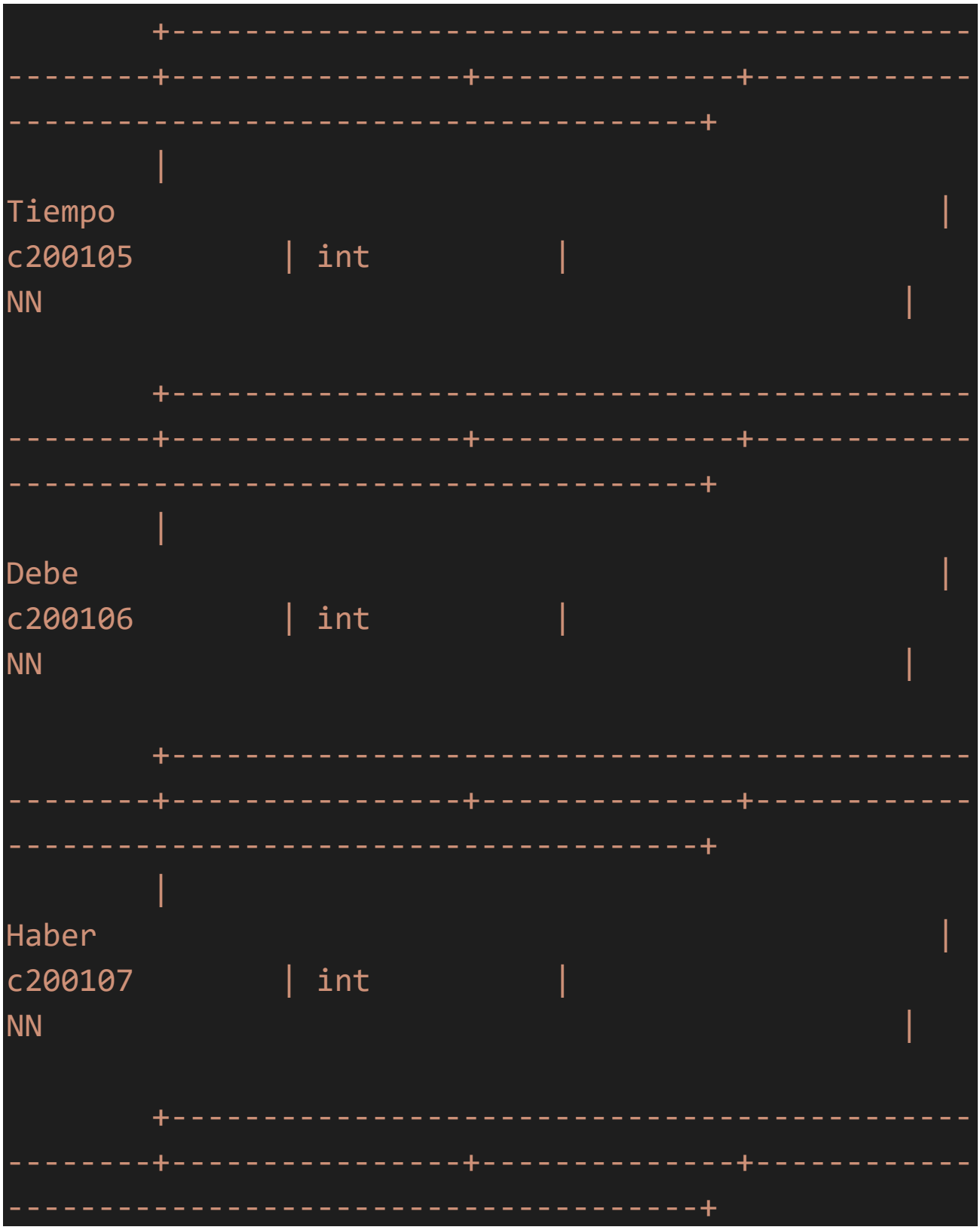
```
'''
Nota : Para la solución, se asume una velocidad uniforme TL
'''
import sys
try:
    # Se leen los datos, puede hacerle lectura múltiple o simple
    # Se opta por lectura multiple
    dato = input().strip().split()
    # Los datos quedan en una lista, se pasan a variables
    dist = int(dato[0].strip())
    maxv = int(dato[1].strip())
    tiem = int(dato[2].strip())
    # Se realizan las validaciones
    if dist<=0 or maxv<=0 or tiem<=0:
        print('ERROR')
    else:
        velo = dist / tiem * 3.6 # Se convierte mt/seg en km/hr
        desv = velo / maxv       # Se calcula el porcentaje
        if desv<=1:
            print('OK')
        elif desv<1.2:
            print('MULTA')
        else:
            print('CURSO SENSIBILIZACION')
except Exception:
    print(f'ERROR: {sys.exc_info()[1]}')
```

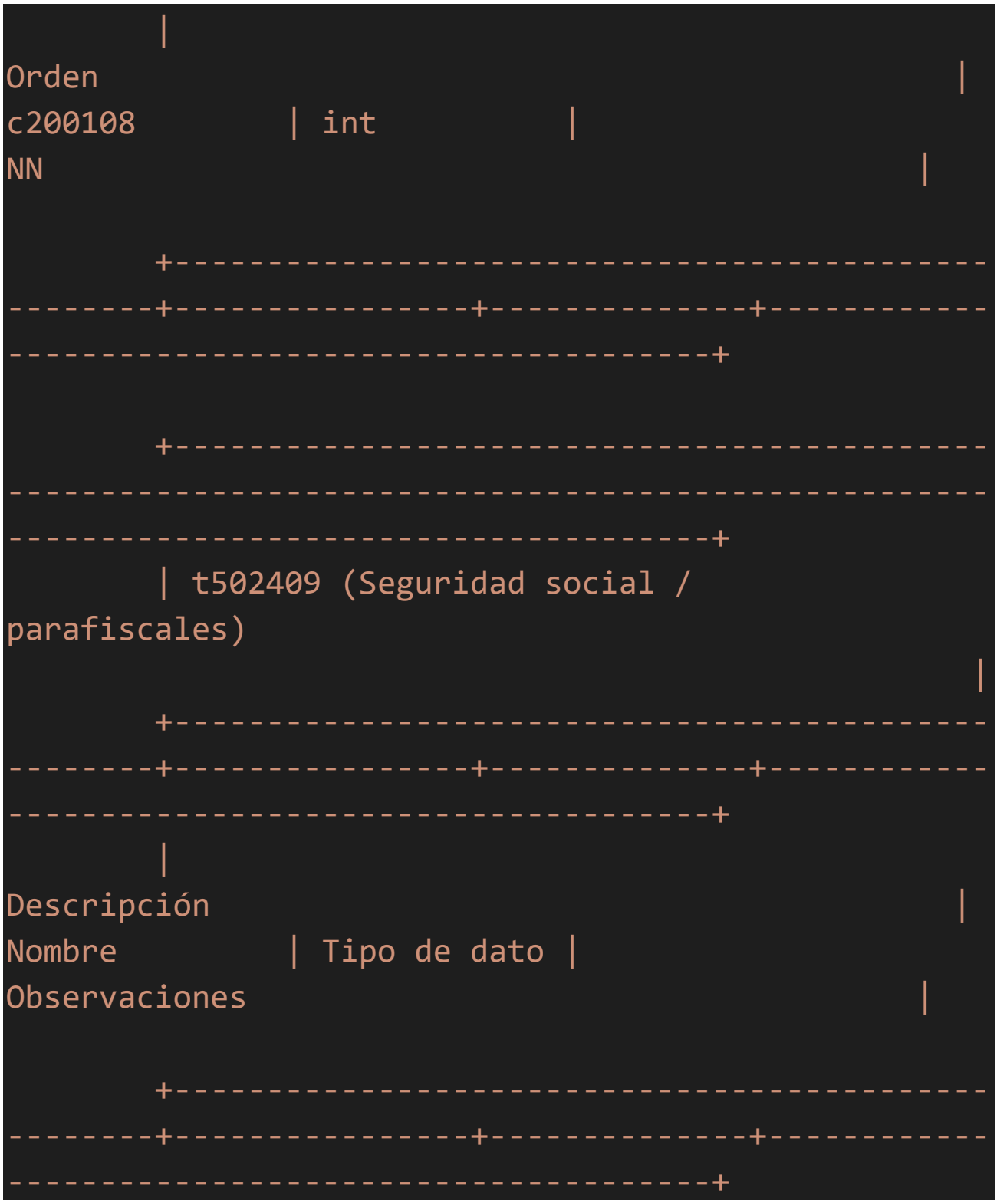
```
'''
Nota : En la solución se utilizarán list slices [Sliding Window] y funciones de listas
'''
import sys
try:
    # Se lee la primera línea representa la cantidad de examenes
    # y la cantidad que puede recordar
    data = input().strip().split()
    nexa = int(data[0].strip())
    nrec = int(data[1].strip())
    # Se realizan las primeras validaciones
    if 1<=nexa<=1000 and 1<=nrec<=1000:
        # Se procede a leer los examenes
        data = input().strip().split()
        # Validación sobre la cantidad de examenes
        if nexa!=len(data):
            print('ERROR')
        else:
            data = [int(w) for w in data] # Se convierten a enteros
            tcop = tdet = 0
            proc = []
            # Se crean los slices y se cuentan los repetidos
            slice = []
            for idx in range(1,nexa):
                ele = data[idx]
                if len(slice)>=nrec:
                    slice.pop(0)
```

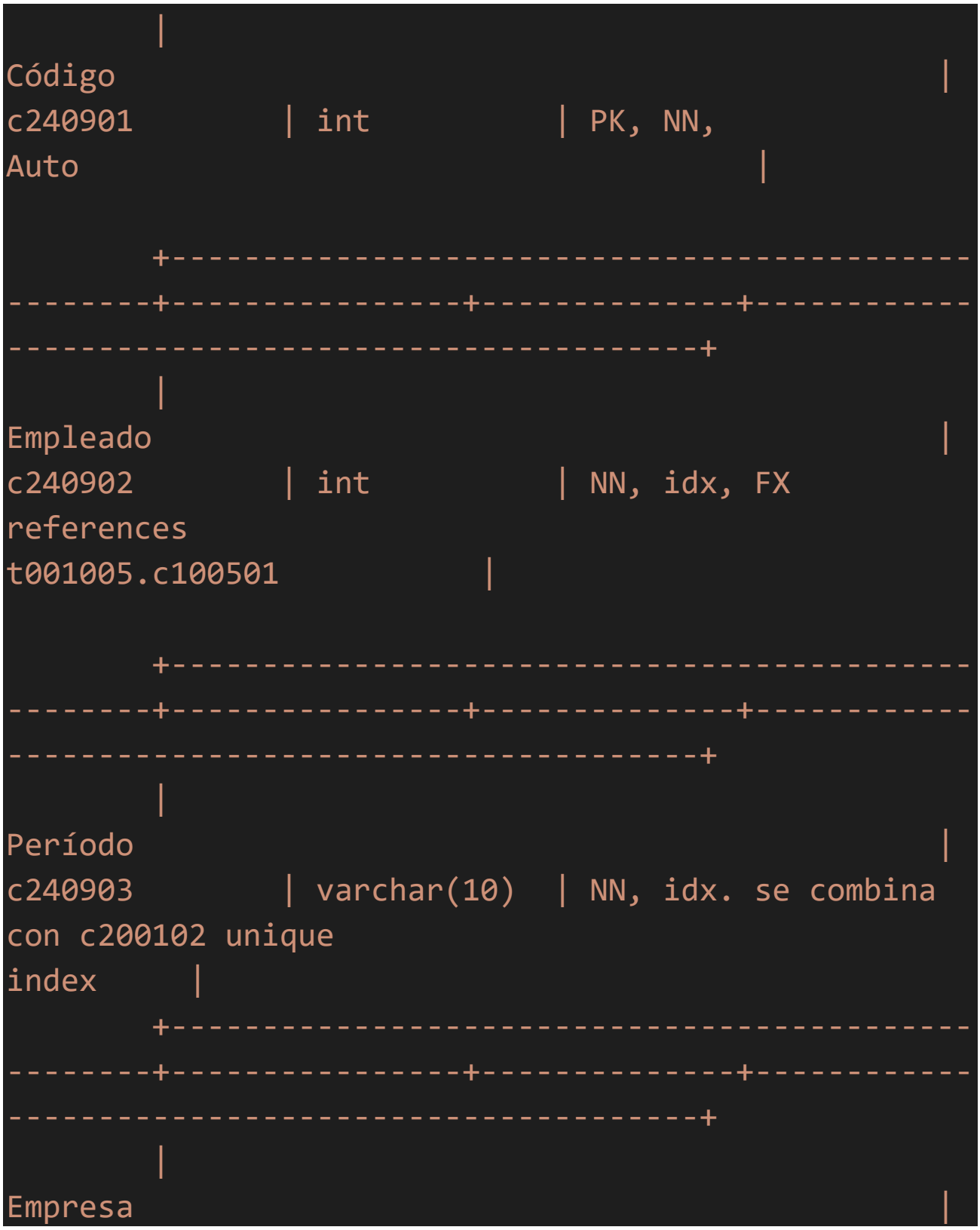
```
            slice.append(data[idx-1])
            if ele not in proc:
                proc.append(ele)
                tcop += data.count(ele)-1
            if ele in slice:
                tdet += 1
        print(f'{tcop} {tdet}')
    else:
        print('ERROR')
except Exception:
    print(f'ERROR: {sys.exc_info()[1]}')
```

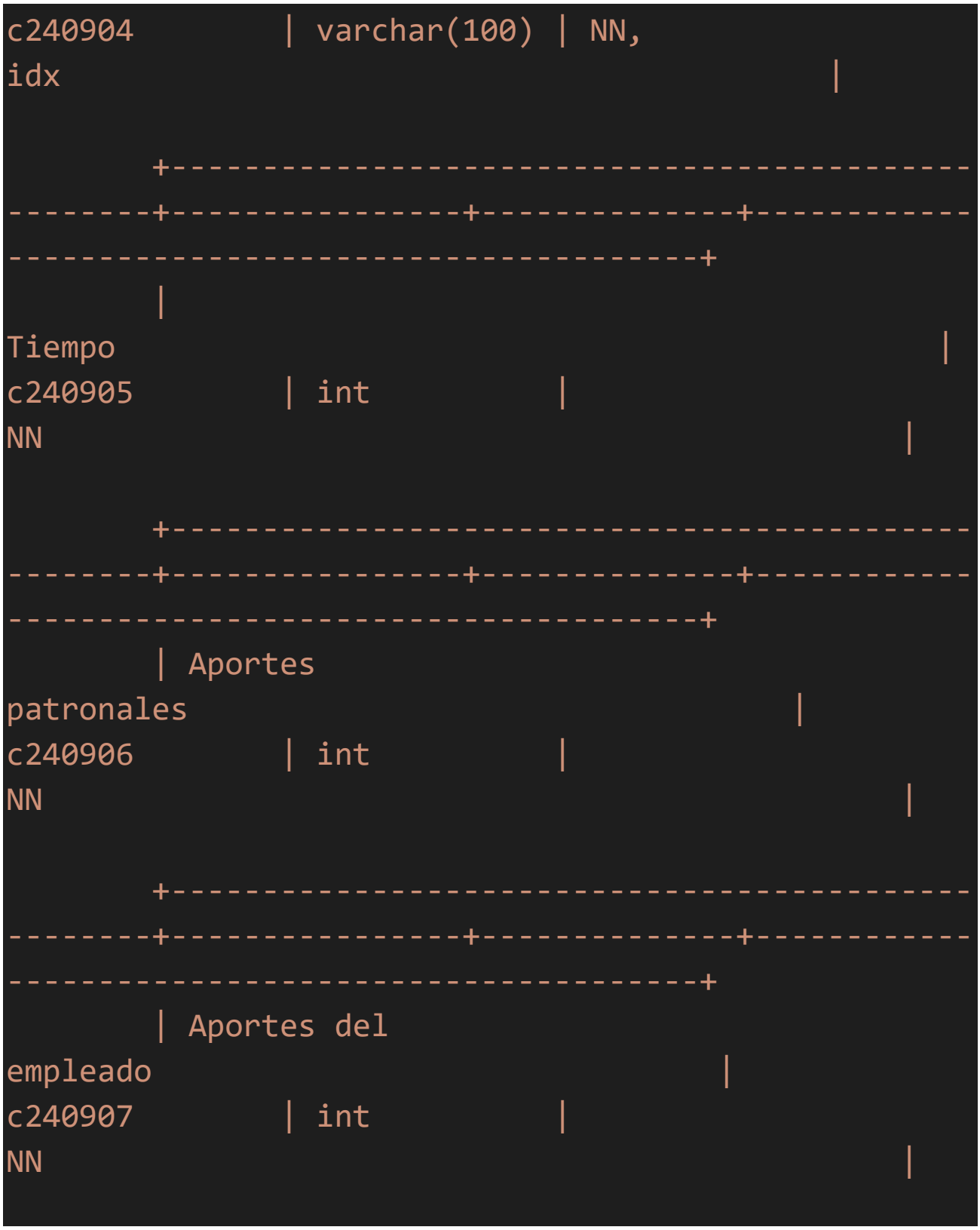

'''
Contiene el modelo (rutinas) de acceso a la base de datos en python para la liquidación de nómina
| t002001 (Liquidaciones)
| Descripción | Nombre | Tipo de dato | Observaciones |
| Código | c200101 | int | PK, NN, Auto |


+----------+----------+----------+----------+
Empleado	c200102	int	NN, idx, FX references t001005.c100501
Período	c200103	varchar(10)	NN, idx. se combina con c200102 unique index
Concepto	c200104	varchar(100)	NN, idx


Tiempo
c200105 | int |
NN
Debe
c200106 | int |
NN
Haber
c200107 | int |
NN


| Orden | Descripción | Nombre | Tipo de dato | Observaciones |
| c200108 | int | NN |
| t502409 (Seguridad social / parafiscales) |


Código	c240901	int	PK, NN, Auto
Empleado	c240902	int	NN, idx, FX references t001005.c100501
Período	c240903	varchar(10)	NN, idx. se combina con c200102 unique index
Empresa


c240904 | varchar(100) | NN,
idx
Tiempo
c240905 | int |
NN
Aportes patronales
c240906 | int |
NN
Aportes del empleado
c240907 | int |
NN

```
        +-----------------------------------------
--------+---------------+-------------+-----------
-----------------------------------+
        | Otros
aportes                                          |
c240908         | int           |
NN                                                 |

        +-----------------------------------------
--------+---------------+-------------+-----------
-----------------------------------+
        |
Concepto                                           |
c240909         | varchar(100) | NN,
idx                                                |

        +-----------------------------------------
--------+---------------+-------------+-----------
-----------------------------------+

'''
from utils.db import seleccion, accion
from utils.globals import HOR_LEGAL, REC_DIU,
REC_NOC, REC_DOM, AUX_TRA, SAL_MIN, CAJA_COM,
ADMIN_RIESGO, NIVEL_RIESGO
import sys
```

```
class payroll():
    ''' Rutina para liquidación de nómina '''
    def __init__(self, per=None):
        self.periodo=per

    def liquidarnomina(self) -> tuple:
        try:
            res = True
            msg = 'INFO: La nómina fue liquidada
satisfactoriamente'
            # Recuperar los empleados a liquidar
            sql = f"SELECT c100501, c100502||'
'||c100503 AS 'nombre', c100504||'-'||c100505 AS
'ide', c100506, c100507, c100508, c290804, c290805,
c290806, c290807, c290808 FROM t001005 LEFT JOIN
t502908 ON c290802=c100501 AND
c290803='{self.periodo}' WHERE C100509='A'"
            dat = seleccion(sql)
            if len(dat)==0:
                msg = f'INFO: No existen registros
para liquidar en el período {self.periodo}'
                res = False
            else:
                # Se realiza la liquidación de nómina
                accion('DELETE FROM t002001 WHERE
c200103=?', [self.periodo])
```



```
                accion('DELETE FROM t502409 WHERE
c240903=?', [self.periodo])
                s2001 = 'INSERT INTO t002001(c200102,
c200103, c200104, c200105, c200106, c200107, c200108)
VALUES(?, ?, ?, ?, ?, ?, ?)'
                s2409 = 'INSERT INTO t502409(c240902,
c240903, c240904, c240905, c240906, c240907, c240908,
c240909, c240910) VALUES(?, ?, ?, ?, ?, ?, ?, ?, ?)'
                for ele in dat:
                    # Se recuperan los datos
                    cod = ele[0]
                    nom = ele[1]
                    ide = ele[2]
                    sal = ele[3]
                    eps = ele[4]
                    afp = ele[5]
                    tra = ele[6]
                    aus = ele[7]
                    edi = ele[8]
                    eno = ele[9]
                    dom = ele[10]
                    # Se realizan los calculos base
                    if tra==0:
                        tra=30
                    vho = sal/HOR_LEGAL
                    SAL = (int)(sal*(tra-aus)/30)
                    TRA = (int)((tra-aus)*AUX_TRA/30)
```

```
                    EDI = (int)(edi*vho*REC_DIU)
                    ENO = (int)(eno*vho*REC_NOC)
                    DOM = (int)(dom*vho*REC_DOM)
                    AUS = (int)(aus*sal/30)
                    IBC = SAL+EDI+ENO+DOM-AUS
                    EPS =
(int)(round(0.04*IBC/100)*100)
                    FSP =
(int)(round(0.001*IBC/100)*100)
                    TOT = SAL
                    # Se imprime, graban los
resultados
                    print(f'{nom:<50s} {ide:<30s}')
                    txt = f'SALARIO BASE({sal:10,})
(+)'
                    print(f'{txt:<50} {tra-aus:2d}
{SAL:15,}')
                    accion(s2001, (cod, self.periodo,
txt, tra-aus, SAL, 0, 0))
                    if SAL<=2*SAL_MIN:
                        txt = 'AUXILIO DE TRANSPORTE
(+)'
                        print(f'{txt:<50} {tra-
aus:2d} {TRA:15,}')
                        accion(s2001, (cod,
self.periodo, txt, tra-aus, TRA, 0, 1))
                        TOT += TRA
```

```
                    if EDI>0:
                        txt = f'EXTRAS DIURNAS (+)'
                        print(f'{txt:<50} {edi:2d}
{EDI:15,}')
                        accion(s2001, (cod,
self.periodo, txt, edi, EDI, 0, 2))
                        TOT += EDI
                    if ENO>0:
                        txt = f'EXTRAS NOCTURNAS (+)'
                        print(f'{txt:<50} {eno:2d}
{ENO:15,}')
                        accion(s2001, (cod,
self.periodo, txt, eno, ENO, 0, 3))
                        TOT += ENO
                    if DOM>0:
                        txt = f'DOMINICALES Y
FESTIVOS (+)'
                        print(f'{txt:<50} {dom:2d}
{DOM:15,}')
                        accion(s2001, (cod,
self.periodo, txt, dom, DOM, 0, 4))
                        TOT += DOM
                    if AUS>0:
                        txt = f'AUSENCIAS({sal:10,})
(-)'
                        print(f'{txt:<50} {aus:2d}
{AUS:32,}')
```

```
                        accion(s2001, (cod,
self.periodo, txt, aus, 0, AUS, 10))
                        TOT -= AUS
                    txt = f'Salud({eps}) (-)'
                    print(f'{txt:<50} {tra:2d}
{EPS:32,}')
                    accion(s2001, (cod, self.periodo,
txt, tra-aus, 0, EPS, 11))
                    txt = f'Pensión({afp}) (-)'
                    print(f'{txt:<50} {tra:2d}
{EPS:32,}')
                    accion(s2001, (cod, self.periodo,
txt, tra-aus, 0, EPS, 12))
                    TOT -= 2*EPS
                    if IBC>4*SAL_MIN:
                        txt = f'Fondo de solidaridad
pensional({afp}) (-)'
                        print(f'{txt:<50} {tra:2d}
{FSP:32,}')
                        accion(s2001, (cod,
self.periodo, txt, tra-aus, 0, FSP, 13))
                        TOT -= FSP
                    txt = 'Total devengado (=)'
                    accion(s2001, (cod, self.periodo,
txt, tra-aus, TOT, 0, 20))
                    # Salud
```

```
                    accion(s2409, (cod, self.periodo,
eps, tra-aus, (int)(round(2.125*EPS/100)*100), EPS,
0, 'SALUD', IBC))
                    # Pensión
                    accion(s2409, (cod, self.periodo,
afp, tra-aus, 3*EPS, EPS, FSP, 'PENSIÓN', IBC))
                    # Caja de Compensación
                    accion(s2409, (cod, self.periodo,
CAJA_COM, tra-aus, EPS, 0, 0, 'CAJA', IBC))
                    # Riesgo laboral
                    accion(s2409, (cod, self.periodo,
ADMIN_RIESGO, tra-aus,
(int)(round(NIVEL_RIESGO*IBC/100)*100), 0, 0, 'ARL',
IBC))
                    # SENA
                    accion(s2409, (cod, self.periodo,
'SENA', tra-aus, (int)(round(0.02*IBC/100)*100), 0,
0, 'SENA', IBC))
                    # ICBF
                    accion(s2409, (cod, self.periodo,
'ICBF', tra-aus, (int)(round(0.03*IBC/100)*100), 0,
0, 'ICBF', IBC))
                    print(f'{txt:<50} {tra:2d}
{TOT:59,}')
                    print()
        except Exception:
            res = False
```

```
            msg = f"ERROR: al liquidar nómina
{sys.exc_info()[1]}"
        return (res, msg)

    def imprimirnomina(self,per) -> tuple:
        try:
            res = False
            msg = 'INFO: Nómina impresa
satisfactoriamente'
            nom = []
            sql = f"SELECT c200102, c200104, c200105,
c200106, c200107 FROM t002001 WHERE
c200103='{self.periodo}' ORDER BY c200102, c200108"
            dat = seleccion(sql)
            if len(dat)==0:
                msg = f'INFO: No existe liquidación
de nómina para el período {per}'
            else:
                res = True
                for ele in dat:
                    emp = ele[0]
                    con = ele[1]
                    tie = ele[2]
                    deb = ele[3]
                    if deb==0:
                        deb=''
                    else:
```

```
                        deb = f'{deb:15,}'
                    hab = ele[4]
                    if hab==0:
                        hab=''
                    else:
                        hab = f'{hab:15,}'
                    nom.append([emp, con, tie, deb,
hab])
        except Exception:
            res = False
            msg = f"ERROR: al imprimir nómina
{sys.exc_info()[1]}"
        return (res, msg, nom)

    def imprimirvolante(self,cod,per) -> tuple:
        try:
            res = False
            msg = 'INFO: Volante consultado
satisfactoriamente'
            nom = []
            sql = f"SELECT c200102, c200104, c200105,
c200106, c200107 FROM t002001 WHERE c200102={cod} AND
c200103='{self.periodo}' ORDER BY c200108"
            dat = seleccion(sql)
            if len(dat)==0:
                msg = f'INFO: No existe liquidación
de nómina para el período {per}'
```

```
            else:
                res = True
                for ele in dat:
                    emp = ele[0]
                    con = ele[1]
                    tie = ele[2]
                    deb = ele[3]
                    if deb==0:
                        deb=''
                    else:
                        deb = f'{deb:15,}'
                    hab = ele[4]
                    if hab==0:
                        hab=''
                    else:
                        hab = f'{hab:15,}'
                    nom.append([emp, con, tie, deb,
hab])
        except Exception:
            res = False
            msg = f"ERROR: al imprimir nómina
{sys.exc_info()[1]}"
        return (res, msg, nom)

    def imprimirss(self,ent,con,per) -> tuple:
        try:
            res = False
```

```
            msg = 'INFO: No se puede generar el
reporte'
            sda = []
            sql = f"SELECT c100502||' '||c100503 AS
'nombre', c100503||'-'||c100504 AS 'identificacion',
c240905, c240907, c240906, c240908, c240910 FROM
t502409 INNER JOIN t001005 ON c100501=c240902 WHERE
c240904='{ent}' AND c240903='{per}' AND
c240909='{con}' ORDER BY nombre"
            dat = seleccion(sql)
            if len(dat)==0:
                msg = f'INFO: No existe liquidación
de nómina para el período {per}'
            else:
                res = True
                for ele in dat:
                    nom = ele[0]
                    ide = ele[1]
                    dia = ele[2]
                    emp = ele[3]
                    pat = ele[4]
                    otr = ele[5]
                    ibc = ele[6]
                    tot = emp+pat+otr
                    sda.append([nom, ide, dia, ibc,
emp, pat, otr, tot])
        except Exception:
```

```
            res = False
            msg = f"ERROR: al generar listado SS
{sys.exc_info()[1]}"
        return (res, msg, sda)

    def imprimircertificado(self,cod) -> tuple:
        ''' TASK: Implementar este módulo '''
        try:
            res = True
            msg = 'INFO: Implementar'
        except Exception:
            res = False
            msg = f"ERROR: al generar certificado
{sys.exc_info()[1]}"
        return (res, msg)
```