Examen Consultor Sr. Parte Teórica

Creado por: Edgar Gerardo Alarcón González

SQL

a)

Este código borra una tabla llamada `ws_envd_analitica.env_movimientos` borrando toda la estructura e información que contenía. Posteriormente la vuelve crearla estableciendo los tipos de datos que cada variable va a contener, así como los formatos en como se está presentando la información para esta tabla. Finalmente del directorio indicado se cargan datos ya preexistentes que tienen el formato indicado en la creación de la tabla.

`DROP` y `PURGE` son funciones destinadas para eliminar la tabla. `CREATE` la crea. otros como `INT`, `STRING`, `DATE`, etc. son tipos de datos para las variables y `FORMAT` cómo vienen presentados estos tipos de datos. `LOAD DATA` e `INTO TABLE` cargan y guardan en una table de un directorio.

```
DROP TABLE ws_envd_analitica.env_movimientos PURGE;

CREATE TABLE ws_envd_analitica.env_movimientos (
id_master       INT,
id_cliente      STRING,
Nombre          STRING,
Ap_paterno      STRING,
Ap_materno      STRING,
anio            INT,
txns            INT,
monto           BIGINT,
fecha           DATE,
prod_subprod    MAP<STRING,STRING>,
coordenada      ARRAY<INT>,
estadisticas    STRUCT<CECO:INT, ACC:INT, MATRICULA:INT, REGISTRO:INT>
)
PARTITION BY RANGE(year)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '|'
collection items terminated by ','
MAP KEYS TERMINATED BY ':';

LOAD DATA INPATH '/user/agguadarrama/CrystalData/BaseTransaccional.txt'
INTO TABLE ws_envd_analitica.env_movimientos;
```

b)

Este código borra una tabla (`ws_pgsd_analitica.tmp_pgs_969077_od`) en caso de existir y crea otra (`ws_envd_analitica.env_txns`) que parece ser la actualización de un histórico. La creación de esta última tabla se hace como un resumen utilizando funciones de agregación de una consulta realizada a la base de datos.

Algunas de las funciones presentes ya fueron explicadas anteriormente. Otras importantes son:

- `AS`: que funciona para llamar al objeto de la izquierda con otro nombre a manera de que sea práctica su manipulación.
- `count`, `sum` y `GROUP BY` : son funciones de agregación para conteo y suma respectivamente. Mientras que la última indica con base en qué variable se ejecutarán las funciones de agregación.
- `select`, `from`, `where` y `on`: son funciones que permiten de forma básica la selección, cruce y filtración de tablas a través de condiciones de las variables propias en las tablas.
- `IF`, `CASE`, `WHEN`, `ELSE`, `END`: es un condicional que en caso de ser verdadera o falsa brinda alternativas. Van de la mano con operadores lógicos tales como `AND`, `NOT IN`, etc.
- `cast`: transforma el tipo de dato vía cohesión
- `regexp_replace`: busca y reemplaza texto a través de expresiones regulares.
- `year(to_date(now()))`: extrae el presente año.
- `LEFT ANTI JOIN`: se queda con todo lo de la tabla izquierda quitando lo que se encuentra en la tabla derecha (en la intersección).
- `BETWEEN`: sirve para filtrar dado intervalos sobre una variable.

```
drop table if exists ws_pgsd_analitica.tmp_pgs_969077_odp purge;

create table
    ws_envd_analitica.env_txns stored as parquet as
SELECT
    id_master
    ,num_p
    ,count(*)    as txns
    ,sum(monto)  as monto
FROM(
    select
         *
        ,year(min(fec_reg_ps))*100+month(min(fec_reg_ps)) as num_p
    from(
        SELECT
             IF(B.id_master is null,-1*A.ID_CLIENTE,B.id_master)
            ,IF(FCCLIENTEID = 'N/A', NULL, cast(regexp_replace(FCCLIENTEID, "[^0-9]","") as BIGINT) )
            ,concat(fcctenombre,fcctepaterno,fcctematerno,'-',from_timestamp(fcctefechanacimiento,'yy-M
            ,CASE
                WHEN prod_subprod['prod'] NOT IN ('2','4') AND prod_subprod['subprod'] = ''    estadisti
                WHEN prod_subprod['prod']     IN ('2','4') AND prod_subprod['subprod'] = 'INT' estadisti
                ELSE estadisticas.ACC END
            ,IF( anio BETWEEN '2022-01-01' AND  year(to_date(now())), 'anio actual','anios_pasados')
            ,row_number() over (partition by id_master,fecha order by fecha desc)
        FROM ws_envd_analitica.env_movimientos as A
        LEFT ANTI JOIN
```

```
            (
            SELECT
                 CAST(id_cliente AS BIGINT) as id_cliente_num
                ,min(id_master) as id_master
            FROM cd_baz_bdclientes.cd_cte_master
            WHERE cod_tipo_cliente = 'CLIENTE_UNICO_0'
            GROUP BY id_cliente_num
            ) AS B
        ON A.ID_CLIENTE = B.id_cliente_num
        WHERE TO_DATE(fecha) BETWEEN '2010-01-02' AND  '2020-11-09'
        ) AS C
    WHERE row1 = 1

    UNION ALL

    SELECT * FROM ws_envd_analitica.env_movimientos_hist

    ) AS D
GROUP BY id_master, num_p
;
```

c)

Similar al anterior, esta consulta quita una tabla si existe y crea una nueva a partir de una consulta realizada a la base de datos. La manera en que se hace esta consulta es devolviendo sumas condicionadas sobre ciertos valores y haciendo un cruce que mantenga las sumas de la consulta izquierda aunque no se encuentren en la tabla de la derecha, la cual también se obtiene bajo ciertos filtros.

Además de algunas de las funciones que ya comentamos vemos:

- `LEFT JOIN`: realiza un cruce manteniendo todos los datos de la tabla izquierda aunque no se encuentren con datos de la de la derecha.
- `DISTINCT`: es un operador para filtrar valores distintos de algo.
- `OVER`: indica sobre qué se aplicará una función de agregación.
- `PARTITION BY`: solicita una variable sobre la cuál se hará una partición/separación de los resultados.

```
drop table if exists ws_envd_analitica.env_txns_hist purge;

create table
    ws_envd_analitica.env_txns_hist
stored as parquet as
SELECT
     id_master
    ,persona
    ,num_periodo_mes
    ,txns
    ,SUM(txns) OVER (PARTITION BY id_master ORDER BY num_periodo_mes ROWS BETWEEN  2 PRECEDING AND CURR
    ,SUM(txns) OVER (PARTITION BY id_master ORDER BY num_periodo_mes ROWS BETWEEN 11 PRECEDING AND CURR
    ,monto
    ,SUM(monto)       OVER (PARTITION BY id_master ORDER BY num_periodo_mes ROWS BETWEEN  2 PRECEDING A
    ,SUM(monto)       OVER (PARTITION BY id_master ORDER BY num_periodo_mes ROWS BETWEEN 11 PRECEDING A
FROM(
    SELECT
          id_master
         ,persona
        ,num_periodo_mes
        ,SUM(IF(num_periodo_mes = num_p, txns,0))  AS txns
        ,SUM(IF(num_periodo_mes = num_p, monto,0)) AS monto
    FROM
        ws_envd_analitica.env_txns AS A

    LEFT JOIN
        (
        SELECT
           DISTINCT num_periodo_mes
        FROM
            cd_baz_bdclientes.cd_gen_fechas_cat
        WHERE
            num_periodo_mes BETWEEN 201801 AND year(date_add(now(),-1))*100+month(date_add(now(),-1))
        ) AS B
    ON A.num_p <= B.num_periodo_mes
    GROUP BY id_master,num_periodo_mes
    ) AS C
;
```

R

a)

Esto es la creación de una función que parece que lo que busca es leer archivos en formato UTF-8. Esta función solicita una ruta, el separador de columnas, indicar si el archivo a leer tiene encabezados y si se desea quitarlos.

Principalmente se utilizan estructuras de control para evitar errores, o bien, para dar opciones a la persona que utiliza la función. Por ejemplo, si no se indica una ruta, la función `choose.files()` te permitirá buscar el archivo que deseas cargar. Por otro lado si el archivo que se está tratando de abrir, no resulta ser algo convertible a data frame por no coincidir en el número de columnas, entonces ocurrirá un error, etc. Todo esto es filtrado por casos (if, if-else).

Además de estas, se utilizando funciones tales como `readLines` y `readLines` para manipular los datos leídos y acomodarlos de acuerdo al separador. También se utilizan funciones básicas como `paste` y `print` para dar formato a los nombres de las columnas o bien indicar algún error que haya acontecido. Esto igual puede ser mejorado utilizando funciones como `warning`.

```
readTxtDelim<-function(path=NULL,sep=",",HeaderFirstLine=TRUE,quitHeaders=FALSE){
  if(is.null(path)){
    tab<-readLines(choose.files(),encoding = "UTF-8")
  }else{
    tab<-readLines(path,encoding = "UTF-8")
  }
  tab<-str_split(tab,sep)
  tab<-do.call(rbind.data.frame, tab)
  names(tab)<-paste("V",1:dim(tab)[2],sep = "")
  if(HeaderFirstLine==TRUE){
    if(quitHeaders==TRUE){
      tab<-tab[2:dim(tab)[1],]
    }else{
      namesTab=as.character(c(tab[1,]))
      if(length(namesTab)==dim(tab)[2]){
        tab<-tab[2:dim(tab)[1],]
        names(tab)<-namesTab
      }else{
        print("The length of titles does not correspond to the number of columns")
      }
    }
  }
  return(tab)
}
```

b)

Esta es la definición de una función, que parece ocupar un data.frame/table o similares para darle cierto formato dados algunos campos especificados por el usuario. Lo cual hace a esto en general como parte del proceso T de un ETL para dar un formato adecuado a los datos que se están manipulando.

Se utilizan funciones derivadas de expresiones regulares como `gsub` y `substr` para realizar minería de textos. La mayor parte de esta función consiste en el entendimiento de cómo manipular data.frames/tables en R con el objetivo de dar formato y filtrar columnas y los nombres de las mismas.

La función `setkey` sirve para ordenar una columna de un data.table y que la manipulación a través de ella sea más eficiente en términos computacionales (tiempo).

```
asignacion_dt_V1109<-function(tab,mt,Campo_estados,Campo_CP,Campos_out_main,estado_mal,estado_bien){

  campos_originales<-paste(names(tab),collapse="-")
  columnas_originales<-paste(names(tab))
  tab[,Estado2 := eval(Campo_estados) ]
  tab[,Estado2:=toupper(stri_trans_general(Estado2,"Latin-ASCII"))]
  if(length(estado_mal)!=length(estado_bien)){
    print("Vectores mal")
  }else{
    for (i in 1:length(estado_mal)) {
      tab[Estado2==estado_mal[i], Estado2 := estado_bien[i]]
    }
  }

  tab[,CP2 := eval(Campo_CP)]
  tab[,CP2:=as.character(as.numeric(trimws(CP2)))]
  tab[nchar(CP2)==4, CP2 := paste("0",CP2,sep="")]

  tab[,Llave:=gsub(" ","_",paste(Estado2,CP2,sep="_")) ]
  main<-data.table(unique(read.csv(mt,encoding ='UTF-8',header = TRUE, stringsAsFactors = FALSE)))
  estados_main<-unique(substr(main$Origen,1,nchar(main$Origen)-6))
  estados_tab<-unique(gsub(" ","_",tab[,Estado2] ))

  setkey(tab,Llave)
  setkey(main,Origen)
  RES<-main[tab]
  resultados<-list()
  nombres<-c()

  if( dim(RES[is.na(BAZ_1)])[1]==0 ){
    resultados[[1]]<-0
    nombres[2]<-"EXITO - Se encontraron todos los CP"
  }else{
    resultados[[1]]<-data.table(table(RES[is.na(BAZ_1),Origen]))[order(-N)]
    nombres[2]<-"CP"
  }

  resultados[[2]]<-RES[,c(columnas_originales,Campos_out_main),with=F]
  nombres[2]<-"Archivo"

  names(resultados)<-nombres
  return(resultados)}
```

c)

Este es un ejemplo de una función que trabaja en paralelo. Esto significa que está haciendo uso de los núcleos/cores que utiliza la computadora con el objetivo de realizar algún trabajo de una manera más eficiente. Las funciones que delatan el uso de esta metodología de programación son `detectCores`, `makeCluster` y `clusterEvalQ` donde su tarea es buscar y establecer la cantidad de núcleos a utilizar y en el caso de la última indicar cuáles librerías se utilizarán bajo este proceso.

Posteriormente se crea una función, la cuál funciona principalmente a través de dos listas/data.frames (con cualquiera sirve siempre que existan las entradas indicadas) que está buscando cierta condición en el objeto `b` y en caso de que no lo encuentre simplemente dirá que no hay resultados. Esta función parece buscar y cierta información que será concatenada a los resultados.

Posteriormente se leen dos tablas con `fread` donde también se especifica el encoding para leerlas correctamente y a estas se les aplica la función creada pero en paralelo, esto para que sea más eficiente su ejecución como ya antes se mencionó.

```
numCores <- detectCores()
clus <- makeCluster(numCores)

clusterEvalQ(clus, {
  library(sp)
  library(rgdal)
  library(data.table)
})

intersect_function <- function(a,b){

  for ( k in 1:dim(b)[1] ) {
    if( b[k,] == "TRUE"   ){
      resultados <- paste(a$llave,b$CVEGEO[k],sep="|")
      break
    }else{
      resultados <- NA
    }
  }
}

dt_a <- fread(choose.files(),colClasses = "character",encoding = 'Latin1')
dt_b <- fread(choose.files(),colClasses = "character",encoding = 'Latin1')

rest <- parApply( cl=clus, dt_a, 1, intersect_function, b=dt_b  )
```

Python

a)

Al principio se importan unas librerías, luego se define una función que lo que hace es plantear una ecuación lineal salvo un error aleatorio controlado por una semilla. Luego, se crea un vector del 0 al 20 en incrementos de 0.5 para aplicarles la función anterior, se hace un gráfico con título donde se presumiblemnte se ve una recta salvo ligeras variaciones. Luego, a este conjunto de datos se le aplica un modelo de regresión lineal donde después se muestran los coeficientes (pendiente e intercepto respectivamente), los cuales deberían cercanos a lo que se definió en la función ya que así fueron creados. Finalmente también utiliza funciones para pronosticar el valor del número 5 dado el modelo ajustado.

Algunos de los comandos no triviales que podemos ver son:

- `from`, `import` y `as`: sirven para invocar una librería o bien una función dentro de esta y renombrarla para facilitar su manipulación.
- `def` y `return`: para definir una función y (opcionalmente) regrese un valor.
- `np.random.seed()`, `np.random.randn`: sirven para generar y controlar números pseudoaleatorios.
- `np.arange()`: crea un vector como lo describimos anteriormente.
- `LinearRegression()`: es el comando que permite realizar una regresión lineal. También derivado de este se puede ajustar un modelo y predecir dado el modelo.
- `print()` y `str()`: en este caso fueron utilizados para mostrar los coeficientes y concatenarlos.

```
import numpy as np
import matplotlib.pyplot as plt
#%matplotlib inline
from sklearn.linear_model import LinearRegression
def f(x):
    np.random.seed(42)
    y = 0.1*x + 1.25 + 0.2*np.random.randn(x.shape[0])
    return y

x = np.arange(0, 20, 0.5)
y = f(x)
plt.scatter(x,y,label='data', color='blue')
plt.title('Ejercicio para comentar')
regresion_lineal = LinearRegression()
regresion_lineal.fit(x.reshape(-1,1), y)
print('w = ' + str(regresion_lineal.coef_) + ', b = ' +
      str(regresion_lineal.intercept_))
n_x = np.array([5])
p1 = regresion_lineal.predict(n_x.reshape(-1,1))
print(p1)
```

b)

Parece que se busca hacer la configuración de una página en formato html de manera automatizada, al principio parece que se está extrayendo el número de seamana con respecto a la fecha actual, posteriormente se utiliza código python para dar formato al cpodigo html que busca desplegar información de acuerdo a la semana dada.

Principalmente se utilizan cilcos a lo largo de rangos de listas, pero lo más importante es el conocimiento del lenguaje html para dar formato al código.

```
from datetime import date,timedelta

fi=date(2021,1,4)
today=date.today()

for s in list(range(0, 371,7)):
    if (fi+timedelta(days=s)<=today and today<fi+timedelta(days=s+7)):
      s_p=int(s/7+1)

print(str(s_p-1))

for i in range(s_p):
    if i==0 or i==1:
        continue
    sem_b=i
    if ( len(str(sem_b))==1 ):
        semana_b="""str(0)"""+str(sem_b)
    else:
        semana_b=str(sem_b)
    semana_num=i

def df_in_h(titulo,df_t):
    df_t=df_t.to_numpy()
    tab="""
    <h3><strong>"""+titulo+"""</strong></h3>
    <table border="3">
    <tbody>
    <tr>
    <td style="width: 200px; text-align: center; background-color: #e5e3e2;"><strong>Cuentas</strong></
    </tr>
    """
    for row in df_t:
        tab=tab+"""<tr>
    """
        for elem in row:
            tab=tab+"""<td style="text-align: center;"> """+str(elem)+"""</td>
    """
        tab=tab+"""</tr>
    """
    tab=tab+"""</tbody>
    </table>
    <p></p>"""
    return tab

tab1=df_in_h("""Resultados de la semana """+str(s_p)+""":""",df)
print(tab1)
```

c)

Similar al anterior, en este caso parece que se busca extraer información de una base de datos a través de lenguaje sql y enviar los resultados por correo electrónico. Lo correspondiente a la función `HiveContext` está relacionado con el lenguaje sql, de nuevo este código parece utilizar principalmente cadenas de texto, funciones y conocimiento de lenguaje html.

```
conf = SparkConf().setAppName('Portafolio_envio_email')
sc = SparkContext.getOrCreate(conf=conf)
hiveContext = HiveContext(sc)

periodo=str(hiveContext.sql("""SELECT MAX(num_periodo_mes) from gobiernoprogsocial.cd_dig_cte_hist""").

hiveContext.sql("""INSERT INTO gobiernoprogsocial.digital_universo_ps
    select *
    from gobiernoprogsocial.bienestardexalta2020021801
""")

df1 =hiveContext.sql("""SELECT * from gobiernoprogsocial.digital_universo_ps""").collect()

def data_frame_in_html(t,d):
    tab="""
    <h3><strong>"""+titulo+"""</strong></h3>
    <table border="3">
    <tbody>
    <tr>
    <td style="width: 200px; text-align: center; background-color: #e5e3e2;"><strong>Semana</strong></t
    <td style="width: 200px; text-align: center; background-color: #e5e3e2;"><strong>Id_programa</strong
    <td style="width: 200px; text-align: center; background-color: #e5e3e2;"><strong>Porcentaje de usua
    <td style="width: 200px; text-align: center; background-color: #e5e3e2;"><strong>Usuarios con activ
    <td style="width: 200px; text-align: center; background-color: #e5e3e2;"><strong>Porcentaje de usua
    </tr>
    """
    for row in df2:
        tab=tab+"""<tr>
    """
        for elem in row:
            tab=tab+"""<td style="text-align: center;"> """+str(elem)+"""</td>
    """
        tab=tab+"""</tr>
    """
    tab=tab+"""</tbody>
    </table>
    <p></p>"""
    return tab

adena="""
<body>
<h1 style="color:black; text-align: center;"><strong>Programas Sociales - Actividad Digital</strong></h
<h2 style="text-align: center;"><strong>Reporte semanal de los usuarios en el programa Bienestar que ya
"""
```

```
######### Configurar y mandar correo

fromaddr = "Programas Sociales <oozie@ektrh7cdhhn2.sie.ad.elektra.com.mx>"

to = ['angel.guadarrama@bancoazteca.com.mx']
cc= ['angel.guadarrama@bancoazteca.com.mx']

addresses = to+cc

msg = MIMEMultipart('alternative')
msg['From'] = fromaddr
msg['To'] = ", ".join(to)
msg['CC'] = ", ".join(cc)
msg['Subject'] = "Actividad digital en Programas Sociales"
msg.attach(MIMEText(cadena,'html'))
text = msg.as_string()
print("body")
server = smtplib.SMTP(host='localhost',port=25)
server.sendmail (fromaddr,addresses,text)
server.quit()
```