Beer Process Simulation

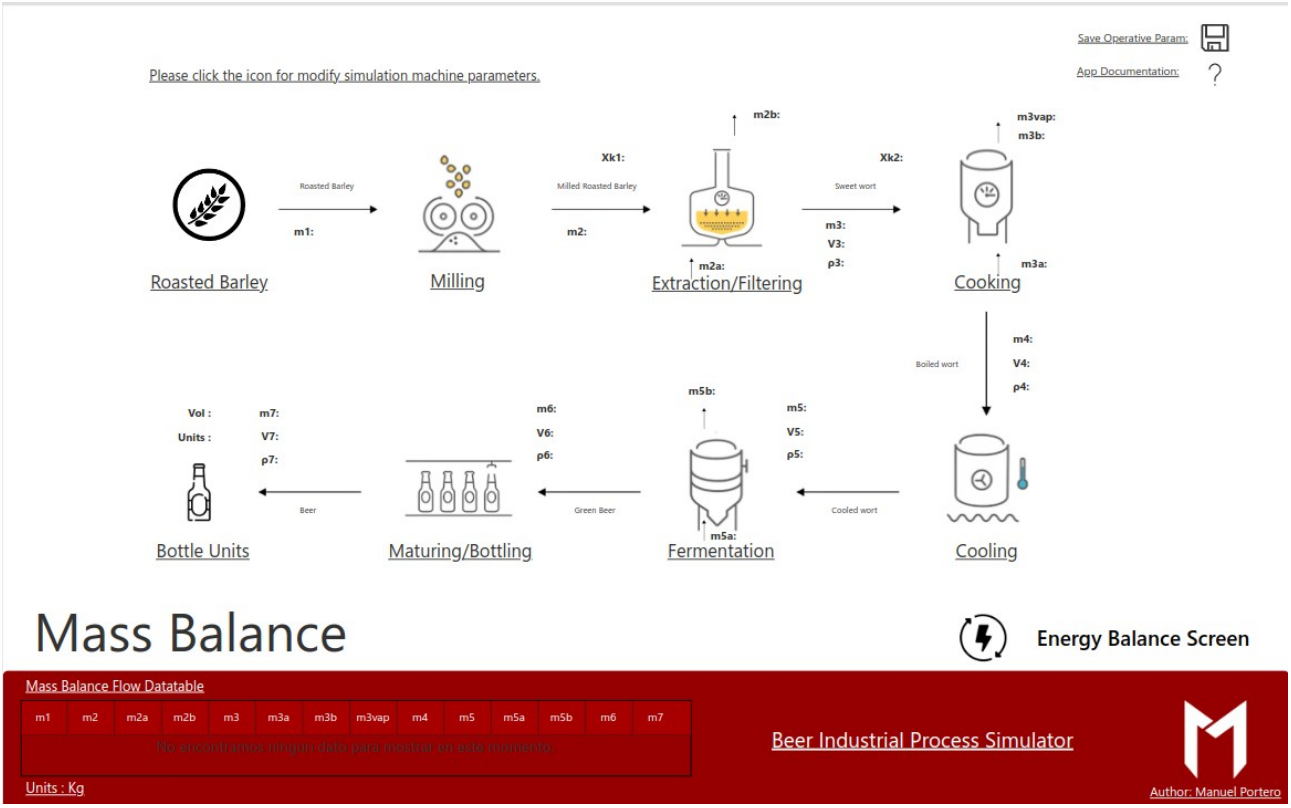
PowerApp Documentation

31/12/2022

Author: Manuel Portero Leiva

Introduction

This document has the purpose to explain the different parts of the SBR Reactor Designing Function App, its code and functionalities, for understanding and replication purposes. The different parts of the architecture solution are show below.

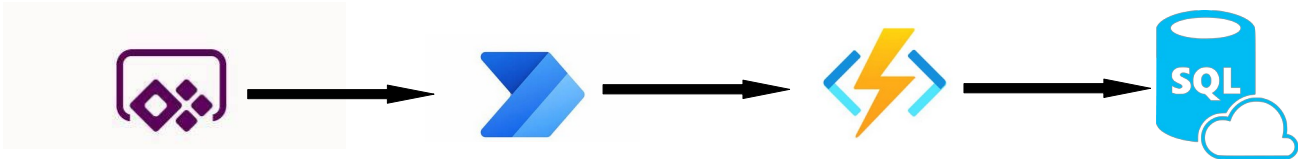


Picture 1: Beer Reactor Simulation Layout

Architecture

The composition of the architecture starts in the PowerApp. Once an equipment design is choosen an equipment design screen is shown, the user modify the equipment and after change the equipment parameters all the process is recalculared via PowerAutomate / Function app.

A full diagram of the solution is shown below.

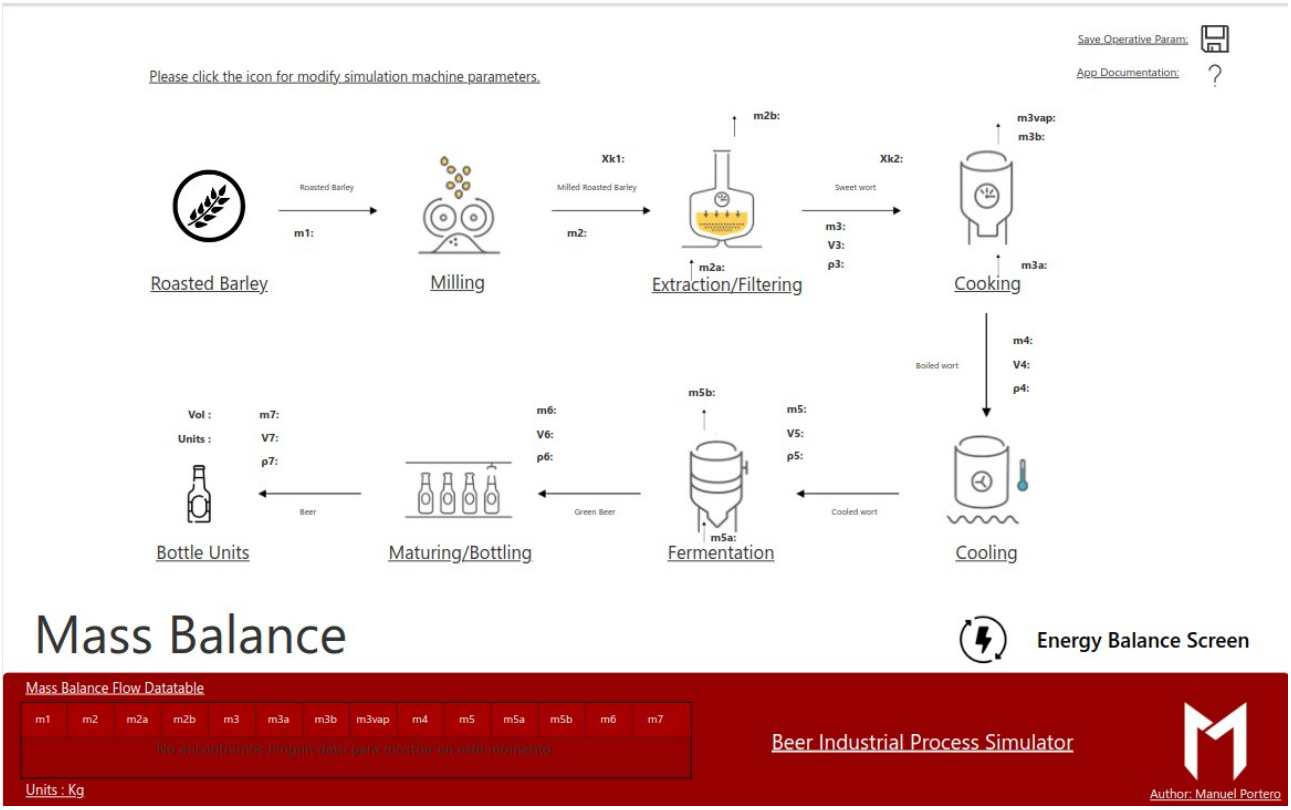


Picture 2: Beer Process Simulation Architecture

PowerApp

Main Screen

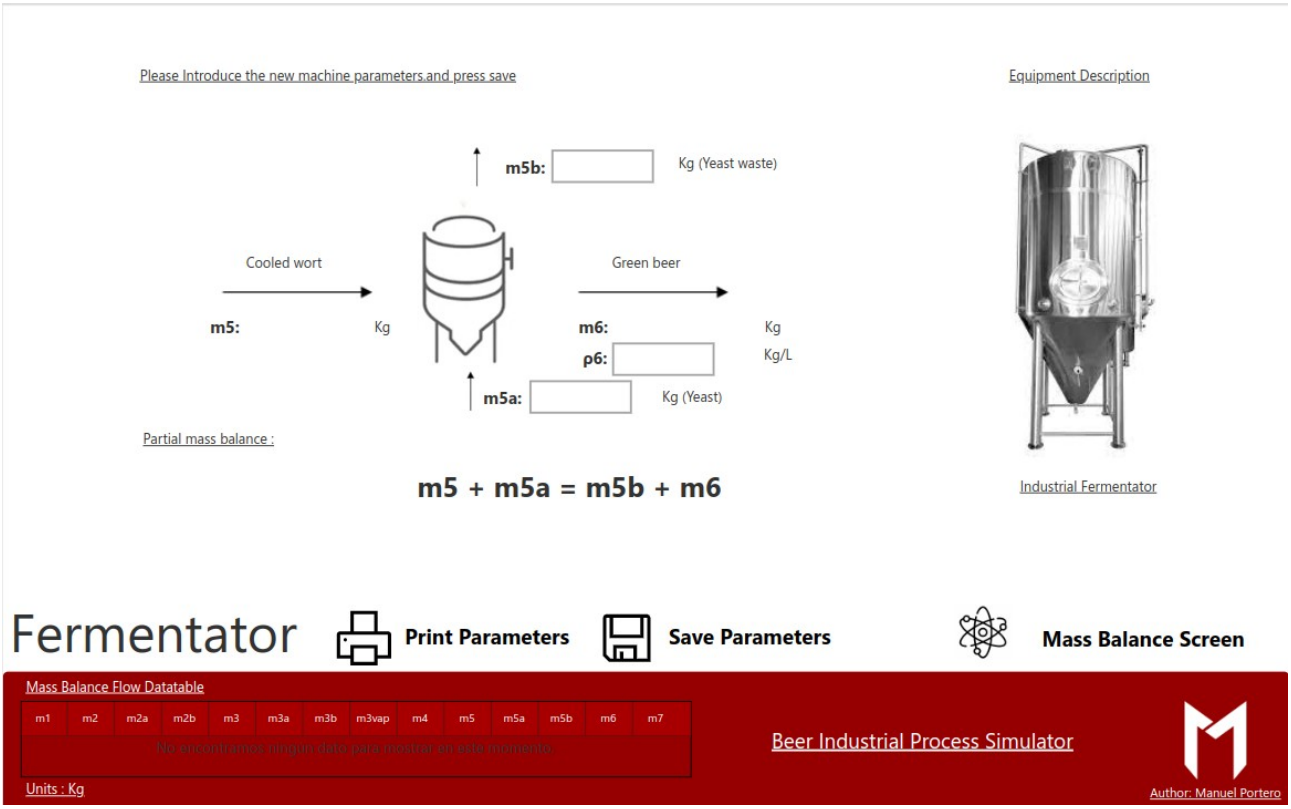
The main screen is composed by the mass balance of the beer process and the navigation buttons to the others screens.



Picture 4: S Beer Process Simulation main Screen

Equipment Screen

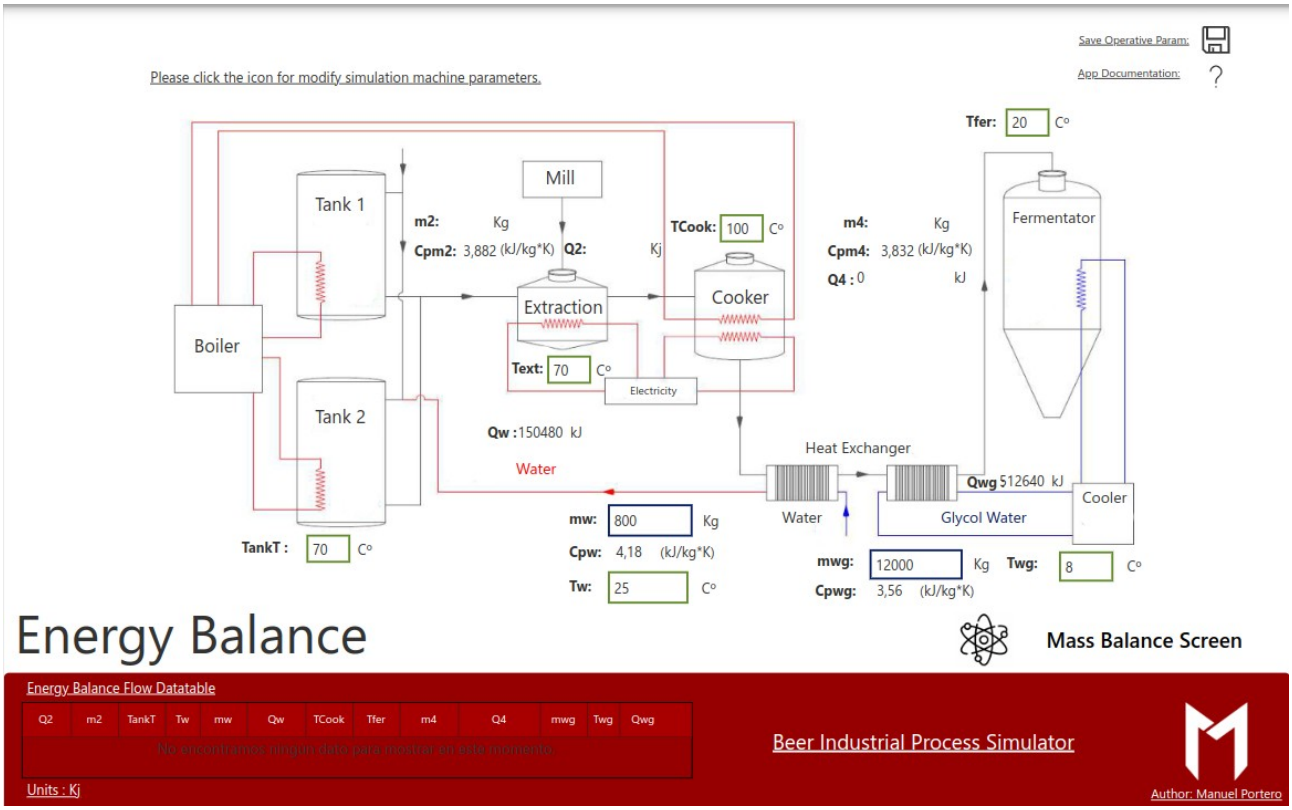
The Equipment Screen is composed by the different equipment parameter's and a Equipment design diagram, we modify the key recalculation parameters in order to optimize the equpment design.



Picture 5: Equpmets Screen

Energy Balance Screen

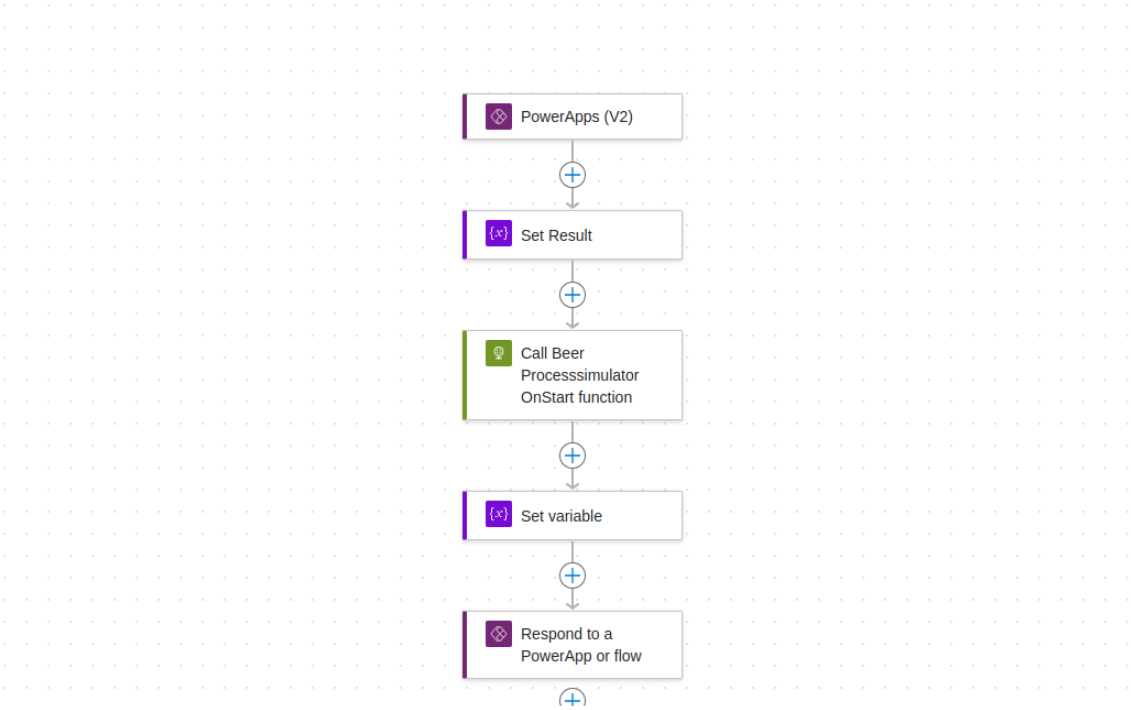
The energy balance screen is composed by the full process energy balance parameters. The user can modify the heat exchange parameters in order to balance the whole energy balance after modifying the mass balance.



Picture 6: Energy Balance Screen

PowerAutomate Flow

The PowerAutomate flow receive the paremeters from the PowerApp clear, the SQL destination table, call the Azure function with an Http Request action and update again the table with the balance outputs parameters



Picture 6: Beer Process Simualtion Flow Sample

Azure Function

The Process Design azure functions will receive the parameters from the PowerAutomate flow and will calculate the Volume and other design parameters of the choosen reactor. The code of each azure functions are shown below :

Beer Process Simulator Simulate

```
import json
import logging
import azure.functions as func
import numpy as np
import pyodbc


def main(req: func.HttpRequest) -> func.HttpResponse:
    logging.info('BeerProcessSimulatorSimulate trigger function processed a request...')

    # SQL database variables declaration

    server = 'beerprocesssimulatorsql.database.windows.net'
    database = 'BeerProcessSimulatorSQL'
    username = 'titansax'
    password = 'SecretoGlasgow11!'
    drivers = [item for item in pyodbc.drivers()]
    driver = drivers[-1]
    logging.info("driver:{}".format(driver))


    #Create a connection string

    cnxn = pyodbc.connect('DRIVER='+driver+';SERVER='+server+';DATABASE='+database+';UID='+username+';PWD='+ password)
    cursor = cnxn.cursor()


    # Query Variables


    try:
        select_query = 'SELECT TOP 1 * FROM [dbo].[MassBalanceSimulated]'
        cursor.execute(select_query)
        data = cursor.fetchall()
        logging.info(data)
```

```
except:
    cnxn.rollback()
finally:
    cnxn.commit()
    cnxn.close()



# Variable declarations


    Xk1 = data[0][0]

    logging.info("Xk1 : " + str(Xk1))

    Xk2 = data[0][1]

    logging.info("Xk2 : " + str(Xk2))

    m1 = data[0][2]

    logging.info("m1 : " + str(m1))

    m2 = data[0][3]

    logging.info("m2 : " + str(m2))

    m2b = data[0][4]

    logging.info("m2b : " + str(m2b))

    m3 = data[0][5]

    logging.info("m3 : " + str(m3))

    V3 = data[0][6]

    logging.info("V3 : " + str(V3))

    rho3 = data[0][7]

    logging.info("rho3 : " + str(rho3))

    m2a = data[0][8]

    logging.info("m2a : " + str(m2a))

    m3vap = data[0][9]

    logging.info("m3vap : " + str(m3vap))
```

```
m3a = data[0][10]
logging.info("m3a : " + str(m3a))
m3b = data[0][11]
logging.info("m3b : " + str(m3b))
m4 = data[0][12]
logging.info("m4 : " + str(m4))
V4 = data[0][13]
logging.info("V4 : " + str(V4))
rho4 = data[0][14]
logging.info("rho4 : " + str(rho4))
m5 = data[0][15]
logging.info("m5 : " + str(m5))
V5 = data[0][16]
logging.info("V5 : " + str(V5))
rho5 = data[0][17]
logging.info("rho5 : " + str(rho5))
m5b = data[0][18]
logging.info("m5b : " + str(m5b))
V5b = data[0][19]
logging.info("V5b : " + str(V5b))
m6 = data[0][20]
logging.info("m6 : "  + str(m6))
V6 = data[0][21]
logging.info("V6 : " + str(V6))
rho6 = data[0][22]
logging.info("rho6 : " + str(rho6))
```

```
m5a = data[0][23]
logging.info("m5b : " + str(m5b))
m7 = data[0][24]
logging.info("m7 : " + str(m7))
V7 = data[0][25]
logging.info("V7 : " + str(V7))
rho7 = data[0][26]
logging.info("rho7 : " + str(rho7))
Vol = data[0][27]
logging.info("Vol : " + str(Vol))

TimeDate = data[0][28]
logging.info("TimeDate : " + str(TimeDate))
# Mass Balance
# Step 1 : Milling
m2 = m1
logging.info("m2 : " + str(m2))
# Step 2 : Extraction

m2b = Xk1 / m2
logging.info("m2b : " + str(m2b))

m3 = m2 + m2a - m2b
logging.info("m3 : " + str(m3))
# Step 3 : Cooking
m3b = m3a * Xk2
logging.info("m3b : " + str(m3b))
m4 = m3 + m3a - m3b - m3vap
```

```
        logging.info("m4 : " + str(m4))

        V3 = m3 / rho3

        logging.info("V3 : " + str(V3))

        V4 = m4 / rho4

        logging.info("V4 : " + str(V4))

        # Cooling

        m5 = m4

        logging.info("m5 : " + str(m5))

        # Fermentation

        m6 = m5 + m5a - m5b

        logging.info("m6 : " + str(m6))

        V5b = m5b / rho5

        logging.info("V5b : " + str(V5b))

        # Maturing and bottling

        m7 = m6

        logging.info("m7 : " + str(m7))

        V7 = m7/rho7

        logging.info("V7 : " + str(V7))


        #Insert Query

        #Create a connection string

                                                                    cnxn            =
pyodbc.connect('DRIVER='+driver+';SERVER='+server+';DATABASE='+databa
se+';UID='+username+';PWD='+ password)
        cursor = cnxn.cursor()


        try:
            delete_query = "DELETE TOP (1) FROM [dbo].[MassBalanceSimulated]"
            cursor.execute(delete_query)
```

```
            insert_query ="INSERT INTO [dbo].[MassBalanceSimulated] ([Xk1],
[Xk2],[m1],[m2],[m2b],[m3],[V3],[rho3],[m2a],[m3vap],[m3a],[m3b],[m4],[V4],
[rho4],[m5],[V5],[rho5],[m5b],[V5b],[m6],[V6],[rho6],[m5a],[m7],[V7],[rho7],
[Vol],[TimeDate]) VALUES (?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)"
             cursor.execute(insert_query,Xk1,Xk2,m1,m2,m2b,m3,V3,rho3,m2a,m3v
ap,m3a,m3b,m4,V4,rho4,m5,V5,rho5,m5b,V5b,m6,V6,rho6,m5a,m7,V7,rho7,Vo
l,TimeDate)
        except:
            cnxn.rollback()
        finally:
            cnxn.commit()
            cnxn.close()


    return func.HttpResponse(
            "BeerProcessSimulatorSimulate function executed successfully.",
            status_code=200
    )
```

Beer Process Simulator - Mass Balance

```
import datetime
import json
import logging
import azure.functions as func
import numpy as np
import pyodbc



def main(mytimer: func.TimerRequest) -> None:
    utc_timestamp = datetime.datetime.utcnow().replace(
        tzinfo=datetime.timezone.utc).isoformat()

 # SQL database variables declaration

    server = 'beerprocesssimulatorsql.database.windows.net'
    database = 'BeerProcessSimulatorSQL'
    username = 'titansax'
    password = 'SecretoGlasgow11!'
    drivers = [item for item in pyodbc.drivers()]
    driver = drivers[-1]
    logging.info("driver:{}".format(driver))


    #Create a connection string

                                                          cnxn                =
pyodbc.connect('DRIVER='+driver+';SERVER='+server+';DATABASE='+databa
se+';UID='+username+';PWD='+ password)
```

```
cursor = cnxn.cursor()


# Query Variables


try:
    select_query = 'SELECT TOP 1 * FROM [dbo].[MassBalance]'
    cursor.execute(select_query)
    data = cursor.fetchall()
    logging.info(data)
except:
    cnxn.rollback()
finally:
    cnxn.commit()
    cnxn.close()




# Variable declarations


    Xk1 = data[0][0]

    logging.info("Xk1 : " + str(Xk1))

    Xk2 = data[0][1]

    logging.info("Xk2 : " + str(Xk2))

    m1 = data[0][2]

    logging.info("m1 : " + str(m1))

    m2 = data[0][3]

    logging.info("m2 : " + str(m2))

    m2b = data[0][4]

    logging.info("m2b : " + str(m2b))

    m3 = data[0][5]

    logging.info("m3 : " + str(m3))

    V3 = data[0][6]

    logging.info("V3 : " + str(V3))

    rho3 = data[0][7]
```

```
logging.info("rho3 : " + str(rho3))
m2a = data[0][8]
logging.info("m2a : " + str(m2a))
m3vap = data[0][9]
logging.info("m3vap : " + str(m3vap))
m3a = data[0][10]
logging.info("m3a : " + str(m3a))
m3b = data[0][11]
logging.info("m3b : " + str(m3b))
m4 = data[0][12]
logging.info("m4 : " + str(m4))
V4 = data[0][13]
logging.info("V4 : " + str(V4))
rho4 = data[0][14]
logging.info("rho4 : " + str(rho4))
m5 = data[0][15]
logging.info("m5 : " + str(m5))
V5 = data[0][16]
logging.info("V5 : " + str(V5))
rho5 = data[0][17]
logging.info("rho5 : " + str(rho5))
m5b = data[0][18]
logging.info("m5b : " + str(m5b))
V5b = data[0][19]
logging.info("V5b : " + str(V5b))
m6 = data[0][20]
```

```
logging.info("m6 : "  + str(m6))

V6 = data[0][21]

logging.info("V6 : " + str(V6))

rho6 = data[0][22]

logging.info("rho6 : " + str(rho6))

m5a = data[0][23]

logging.info("m5b : " + str(m5b))

m7 = data[0][24]

logging.info("m7 : " + str(m7))

V7 = data[0][25]

logging.info("V7 : " + str(V7))

rho7 = data[0][26]

logging.info("rho7 : " + str(rho7))

Vol = data[0][27]

logging.info("Vol : " + str(Vol))

TimeDate = data[0][28]

logging.info("TimeDate : " + str(TimeDate))

# Mass Balance

# Step 1 : Milling

m2 = m1

logging.info("m2 : " + str(m2))

# Step 2 : Extraction

m2b = Xk1 / m2

logging.info("m2b : " + str(m2b))
```

```
m3 = m2 + m2a - m2b

logging.info("m3 : " + str(m3))

# Step 3 : Cooking

m3b = m3a * Xk2

logging.info("m3b : " + str(m3b))

m4 = m3 + m3a - m3b - m3vap

logging.info("m4 : " + str(m4))

V3 = m3 / rho3

logging.info("V3 : " + str(V3))

V4 = m4 / rho4

logging.info("V4 : " + str(V4))

# Cooling

m5 = m4

logging.info("m5 : " + str(m5))

# Fermentation

m6 = m5 + m5a - m5b

logging.info("m6 : " + str(m6))

V5b = m5b / rho5

logging.info("V5b : " + str(V5b))

# Maturing and bottling

m7 = m6

logging.info("m7 : " + str(m7))

V7 = m7/rho7

logging.info("V7 : " + str(V7))


#Insert Query
```

```
        #Create a connection string

                                                                cnxn =
pyodbc.connect('DRIVER='+driver+';SERVER='+server+';DATABASE='+databa
se+';UID='+username+';PWD='+ password)
        cursor = cnxn.cursor()


        try:

                insert_query ="INSERT INTO [dbo].[MassBalance] ([Xk1],[Xk2],[m1],
[m2],[m2b],[m3],[V3],[rho3],[m2a],[m3vap],[m3a],[m3b],[m4],[V4],[rho4],
[m5],[V5],[rho5],[m5b],[V5b],[m6],[V6],[rho6],[m5a],[m7],[V7],[rho7],[Vol],
[TimeDate]) VALUES (?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)"
             cursor.execute(insert_query,Xk1,Xk2,m1,m2,m2b,m3,V3,rho3,m2a,m3v
ap,m3a,m3b,m4,V4,rho4,m5,V5,rho5,m5b,V5b,m6,V6,rho6,m5a,m7,V7,rho7,Vo
l,str(utc_timestamp)[0:19])
        except:
            cnxn.rollback()
        finally:
            cnxn.commit()
            cnxn.close()

    logging.info("dateTime : " + str(utc_timestamp)[0:19])
      logging.info('BeerProcessSimulator function executed successfully at %s',
utc_timestamp)
```

Beer Process Simulator – On start

```
import datetime
import json
import logging
import azure.functions as func
import numpy as np
import pyodbc


def main(req: func.HttpRequest) -> func.HttpResponse:
        logging.info('BeerProcessSimulatorSimulate trigger function processed a
request...')

    # SQL database variables declaration

    server = 'beerprocesssimulatorsql.database.windows.net'
    database = 'BeerProcessSimulatorSQL'
    username = 'titansax'
    password = 'SecretoGlasgow11!'
    drivers = [item for item in pyodbc.drivers()]
    driver = drivers[-1]
    logging.info("driver:{}".format(driver))
```

```
    #Create a connection string

                                                                cnxn                =
pyodbc.connect('DRIVER='+driver+';SERVER='+server+';DATABASE='+databa
se+';UID='+username+';PWD='+ password)
    cursor = cnxn.cursor()


    # Query Variables


    try:
            select_query = 'SELECT TOP 1 * FROM [dbo].[MassBalanceSimulated]
ORDER BY TimeDate DESC'
        cursor.execute(select_query)
        data = cursor.fetchall()
        logging.info(data)
    except:
        cnxn.rollback()
    finally:
        cnxn.commit()
        cnxn.close()




    # Variable declarations


        Xk1 = data[0][0]

        logging.info("Xk1 : " + str(Xk1))

        Xk2 = data[0][1]

        logging.info("Xk2 : " + str(Xk2))

        m1 = data[0][2]

        logging.info("m1 : " + str(m1))

        m2 = data[0][3]

        logging.info("m2 : " + str(m2))

        m2b = data[0][4]

        logging.info("m2b : " + str(m2b))

        m3 = data[0][5]
```

```
logging.info("m3 : " + str(m3))

V3 = data[0][6]

logging.info("V3 : " + str(V3))

rho3 = data[0][7]

logging.info("rho3 : " + str(rho3))

m2a = data[0][8]

logging.info("m2a : " + str(m2a))

m3vap = data[0][9]

logging.info("m3vap : " + str(m3vap))

m3a = data[0][10]

logging.info("m3a : " + str(m3a))

m3b = data[0][11]

logging.info("m3b : " + str(m3b))

m4 = data[0][12]

logging.info("m4 : " + str(m4))

V4 = data[0][13]

logging.info("V4 : " + str(V4))

rho4 = data[0][14]

logging.info("rho4 : " + str(rho4))

m5 = data[0][15]

logging.info("m5 : " + str(m5))

V5 = data[0][16]

logging.info("V5 : " + str(V5))

rho5 = data[0][17]

logging.info("rho5 : " + str(rho5))

m5b = data[0][18]
```

```
        logging.info("m5b : " + str(m5b))
        V5b = data[0][19]
        logging.info("V5b : " + str(V5b))
        m6 = data[0][20]
        logging.info("m6 : "  + str(m6))
        V6 = data[0][21]
        logging.info("V6 : " + str(V6))
        rho6 = data[0][22]
        logging.info("rho6 : " + str(rho6))
        m5a = data[0][23]
        logging.info("m5b : " + str(m5b))
        m7 = data[0][24]
        logging.info("m7 : " + str(m7))
        V7 = data[0][25]
        logging.info("V7 : " + str(V7))
        rho7 = data[0][26]
        logging.info("rho7 : " + str(rho7))
        Vol = data[0][27]
        logging.info("Vol : " + str(Vol))

        TimeDate = data[0][28]
        logging.info("TimeDate : " + str(TimeDate))

                                                                utc_timestamp          =
datetime.datetime.utcnow().replace(tzinfo=datetime.timezone.utc).isoformat()
        logging.info(str(utc_timestamp)[0:19])
        #Insert Query

        #Create a connection string
```

```
                                                                cnxn = pyodbc.connect('DRIVER='+driver+';SERVER='+server+';DATABASE='+database+';UID='+username+';PWD='+ password)
        cursor = cnxn.cursor()


        try:
                insert_query ="INSERT INTO [dbo].[MassBalance] ([Xk1],[Xk2],[m1],[m2],[m2b],[m3],[V3],[rho3],[m2a],[m3vap],[m3a],[m3b],[m4],[V4],[rho4],[m5],[V5],[rho5],[m5b],[V5b],[m6],[V6],[rho6],[m5a],[m7],[V7],[rho7],[Vol],[TimeDate]) VALUES (?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)"
                cursor.execute(insert_query,Xk1,Xk2,m1,m2,m2b,m3,V3,rho3,m2a,m3vap,m3a,m3b,m4,V4,rho4,m5,V5,rho5,m5b,V5b,m6,V6,rho6,m5a,m7,V7,rho7,Vol,str(utc_timestamp)[0:19])
        except:
            cnxn.rollback()
        finally:
            cnxn.commit()
            cnxn.close()


    return func.HttpResponse(
            "BeerProcessSimulatorSaveConfig function executed successfully.",
            status_code=200
    )
```

Beer Process Simulator – On Start

```
import json
import logging
import azure.functions as func
import numpy as np
import pyodbc


def main(req: func.HttpRequest) -> func.HttpResponse:
        logging.info('BeerProcessSimulatorSimulate trigger function processed a request...')

    # SQL database variables declaration

    server = 'beerprocesssimulatorsql.database.windows.net'
    database = 'BeerProcessSimulatorSQL'
    username = 'titansax'
    password = 'SecretoGlasgow11!'
    drivers = [item for item in pyodbc.drivers()]
    driver = drivers[-1]
    logging.info("driver:{}".format(driver))


    #Create a connection string
```

```
                                                                cnxn                =
pyodbc.connect('DRIVER='+driver+';SERVER='+server+';DATABASE='+databa
se+';UID='+username+';PWD='+ password)
    cursor = cnxn.cursor()


    # Query Variables


    try:
            select_query = 'SELECT TOP 1 * FROM [dbo].[MassBalance] ORDER BY
TimeDate DESC'
        cursor.execute(select_query)
        data = cursor.fetchall()
        logging.info(data)
    except:
        cnxn.rollback()
    finally:
        cnxn.commit()
        cnxn.close()




    # Variable declarations


        Xk1 = data[0][0]

        logging.info("Xk1 : " + str(Xk1))

        Xk2 = data[0][1]

        logging.info("Xk2 : " + str(Xk2))

        m1 = data[0][2]

        logging.info("m1 : " + str(m1))

        m2 = data[0][3]

        logging.info("m2 : " + str(m2))

        m2b = data[0][4]

        logging.info("m2b : " + str(m2b))

        m3 = data[0][5]

        logging.info("m3 : " + str(m3))
```

```
V3 = data[0][6]
logging.info("V3 : " + str(V3))
rho3 = data[0][7]
logging.info("rho3 : " + str(rho3))
m2a = data[0][8]
logging.info("m2a : " + str(m2a))
m3vap = data[0][9]
logging.info("m3vap : " + str(m3vap))
m3a = data[0][10]
logging.info("m3a : " + str(m3a))
m3b = data[0][11]
logging.info("m3b : " + str(m3b))
m4 = data[0][12]
logging.info("m4 : " + str(m4))
V4 = data[0][13]
logging.info("V4 : " + str(V4))
rho4 = data[0][14]
logging.info("rho4 : " + str(rho4))
m5 = data[0][15]
logging.info("m5 : " + str(m5))
V5 = data[0][16]
logging.info("V5 : " + str(V5))
rho5 = data[0][17]
logging.info("rho5 : " + str(rho5))
m5b = data[0][18]
logging.info("m5b : " + str(m5b))
```

```
        V5b = data[0][19]
        logging.info("V5b : " + str(V5b))
        m6 = data[0][20]
        logging.info("m6 : "  + str(m6))
        V6 = data[0][21]
        logging.info("V6 : " + str(V6))
        rho6 = data[0][22]
        logging.info("rho6 : " + str(rho6))
        m5a = data[0][23]
        logging.info("m5b : " + str(m5b))
        m7 = data[0][24]
        logging.info("m7 : " + str(m7))
        V7 = data[0][25]
        logging.info("V7 : " + str(V7))
        rho7 = data[0][26]
        logging.info("rho7 : " + str(rho7))
        Vol = data[0][27]
        logging.info("Vol : " + str(Vol))

        TimeDate = data[0][28]
        logging.info("TimeDate : " + str(TimeDate))
        #Insert Query
        #Create a connection string
                                                                cnxn          =
pyodbc.connect('DRIVER='+driver+';SERVER='+server+';DATABASE='+databa
se+';UID='+username+';PWD='+ password)
        cursor = cnxn.cursor()

        try:
```

```
            delete_query = "DELETE TOP (1) FROM [dbo].[MassBalanceSimulated]"
            cursor.execute(delete_query)
                insert_query ="INSERT INTO [dbo].[MassBalanceSimulated] ([Xk1],
[Xk2],[m1],[m2],[m2b],[m3],[V3],[rho3],[m2a],[m3vap],[m3a],[m3b],[m4],[V4],
[rho4],[m5],[V5],[rho5],[m5b],[V5b],[m6],[V6],[rho6],[m5a],[m7],[V7],[rho7],
[Vol],[TimeDate]) VALUES (?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)"
             cursor.execute(insert_query,Xk1,Xk2,m1,m2,m2b,m3,V3,rho3,m2a,m3v
ap,m3a,m3b,m4,V4,rho4,m5,V5,rho5,m5b,V5b,m6,V6,rho6,m5a,m7,V7,rho7,Vo
l,TimeDate)
        except:
            cnxn.rollback()
        finally:
            cnxn.commit()
            cnxn.close()


    return func.HttpResponse(
            "BeerProcessSimulator_OnStartApp function executed successfully.",
            status_code=200
    )
```

SQL database flow :

on_Start – > the function get the first row of the massBalance and add it to the massBalanceSimulated

simulate → the app and flows operate over the massBalanceSimulated table.

on_Save → we save the operational parameters in the massBalance table.

During the process mass balance function will act over the massBalance table replicating the mass balance process

```
CREATE TABLE MassBalance
(
Xk1 float,
Xk2 float,
m1 float,
m2 float,
m2b float,
m3 float,
V3 float,
rho3 float,
m2a float,
m3vap float,
m3a float,
m3b float,
m4 float,
V4 float,
rho4 float,
```

```
m5 float,
V5 float,
rho5 float,
m5b float,
V5b float,
m6 float,
V6 float,
rho6 float,
m5a float,
m7 float,
V7 float,
rho7 float,
Vol float,
timeStamp DateTime
)



CREATE TABLE MassBalanceSimulated
(
Xk1 float,
Xk2 float,
m1 float,
m2 float,
m2b float,
m3 float,
V3 float,
rho3 float,
m2a float,
m3vap float,
m3a float,
m3b float,
m4 float,
V4 float,
rho4 float,
m5 float,
V5 float,
rho5 float,
m5b float,
V5b float,
m6 float,
V6 float,
rho6 float,
m5a float,
m7 float,
V7 float,
rho7 float,
Vol float,
timeStamp DateTime
)
```

```
INSERT INTO [dbo].[MassBalanceSimulated]
(Xk1,Xk2,m1,m2,m2b,m3,V3,rho3,m2a,m3vap,m3a,m3b,m4,V4,rho4,m5,V5,rho5,m5b,V5b
,m6,V6,rho6,m5a,m7,V7,rho7,Vol,TimeDate) VALUES (0.8, 1.041, 250.0, 250.0, 325.0,
1030.0, 1156.0, 1.041, 1279.0, 0.9994, 0.7, 31.44, 1030.0, 983.0, 0.9994, 1030.0, 983.0,
1.048, 31.44, 30.0, 999.4, 1000.0, 0.9994, 0.7, 999.4, 1000.0, 0.9994, 0.06, 2023-09-28)

INSERT INTO [dbo].[MassBalance]
(Xk1,Xk2,m1,m2,m2b,m3,V3,rho3,m2a,m3vap,m3a,m3b,m4,V4,rho4,m5,V5,rho5,m5b,V5b
,m6,V6,rho6,m5a,m7,V7,rho7,Vol,TimeDate) VALUES (0.8, 1.041, 250.0, 250.0, 325.0,
1030.0, 1156.0, 1.041, 1279.0, 0.9994, 0.7, 31.44, 1030.0, 983.0, 0.9994, 1030.0, 983.0,
1.048, 31.44, 30.0, 999.4, 1000.0, 0.9994, 0.7, 999.4, 1000.0, 0.9994, 0.06, 2023-09-28)
```