Weekly Staff Caseload Report Generator for Behavior Health Outpatient Clinic

For each staff member identified in email list generate a Word Document according to client specifications showing performance metrics and outcomes between the start and end dates. Example performance outcomes include the number of individual and group therapy sessions delivered, the number of clients on their caseload, the number of No Shows/Cancellations, etc.

The script archives the reports to a folder location and emails the custom generated report to each staff member showing only their metrics, and also emails their supervisors.

In [1]:
```
from IPython.display import display, HTML
display(HTML("<style>.container { width:100% !important; }</style>"))
import pandas as pd
from datetime import datetime
import io
import os
import errno
import pyodbc
import warnings
import win32com.client as win32
warnings.simplefilter("ignore", UserWarning)
pd.options.mode.chained_assignment = None  # default='warn'
```

In [2]:
```
#parameters
services_start_date = datetime(2023,8,20,0,0,0)
services_end_date = datetime(2023,8,26,23,59,59)
next_scheduled_service_days = 14.0
last_service_days = 14.0
reportpath = (r'C:\Users\mbeccaria\St. Josephs Rehabilitation Center, Inc\Outpatient Project Management - Reporting\Ad-I
```

In [3]:
```
#SQL Group Assignments - found Groups > Group > Information > General Information under "Schedules"
server = str(os.getenv('report_server'))
database = 'evolv_cs'
username = os.getenv('report_user')
password = os.getenv('report_pass')
cnxn = pyodbc.connect('DRIVER={SQL Server};SERVER='+server+';DATABASE='+database+';UID='+username+';PWD='+ password)
cursor = cnxn.cursor()
query = """SELECT concat(sv.last_name,', ',sv.first_name) as staff_name
            ,Count(*) as count
          FROM [evolv_cs].[dbo].[schedules] s
          left join staff_view sv
          on default_staff_id = sv.staff_id
          where s.is_active = 1
          and sv.last_name is not null
          group by last_name, first_name
          order by sv.last_name, sv.first_name"""
group_assignments = pd.read_sql(query, cnxn)
```

In [4]:
```
#SQL services
server = str(os.getenv('report_server'))
database = 'evolv_cs'
username = os.getenv('report_user')
password = os.getenv('report_pass')
cnxn = pyodbc.connect('DRIVER={SQL Server};SERVER='+server+';DATABASE='+database+';UID='+username+';PWD='+ password)
cursor = cnxn.cursor()
query = """SELECT
   sev.[people_id]
  ,prv.full_name
  ,sev.[id_no]
  ,sev.[program_name]
  ,fv.[profile_name] facility
  ,rsv.[staff_name]
  ,rsv.[email_address]
  ,sev.[actual_date]
  ,sev.[event_log_id]
  ,sev.[event_name] as service
  ,sev.[is_noshow]
  ,sev.[duration] duration_num
  ,sev.[reason_for_no_show]
  FROM [evolv_cs].[dbo].[service_events_view] sev
  LEFT JOIN rpt_staff_view rsv
    ON sev.staff_id = rsv.staff_id
  LEFT JOIN people_reports_view prv
    ON sev.people_id = prv.people_id
  LEFT JOIN facility_view fv
    ON sev.[site_providing_service] = fv.group_profile_id
where (CONVERT(Date,sev.actual_date) >= '""" + str(services_start_date.date()) + """')
and (CONVERT(Date,sev.actual_date) <= '""" + str(services_end_date.date()) + """')
and prv.full_name is not null
and sev.is_deleted = 0
"""
services = pd.read_sql(query, cnxn,parse_dates = ['actual_date'])
```

In [5]:
```
#SQL Caseloads
server = str(os.getenv('report_server'))
database = 'evolv_cs'
username = os.getenv('report_user')
password = os.getenv('report_pass')
cnxn = pyodbc.connect('DRIVER={SQL Server};SERVER='+server+';DATABASE='+database+';UID='+username+';PWD='+ password)
cursor = cnxn.cursor()
query = """
SELECT distinct
p.full_name name
,p.id_no
,w.[program_name]
,w.[full_name] worker_name
FROM [evolv_cs].[dbo].[worker_assignment_expanded_view] w
left join all_people_clients_view p
on w.people_id = p.people_id
left join event_view ev
on w.event_log_id = ev.event_log_id
where
p.id_no in (select p.id_no from all_people_clients_view p where p.id_no != 'N/A' group by p.id_no)
and program_end_date is NULL
and w.end_date is NULL
order by program_name, name
"""
caseloads = pd.read_sql(query, cnxn)
```

In [7]:
```
#SQL client's latest service and earliest scheduled service dates who are on caseload
server = str(os.getenv('report_server'))
database = 'evolv_cs'
username = os.getenv('report_user')
password = os.getenv('report_pass')
cnxn = pyodbc.connect('DRIVER={SQL Server};SERVER='+server+';DATABASE='+database+';UID='+username+';PWD='+ password)
cursor = cnxn.cursor()
query = """
SELECT distinct
p.id_no
,p.full_name
,min(nsev.scheduled_date) next_scheduled_date
,max(sev.actual_date) last_service_date
,datediff(day,getdate(),min(nsev.scheduled_date)) next_scheduled_date_days
,datediff(day,max(sev.actual_date),getdate()) last_service_date_days
FROM [evolv_cs].[dbo].[worker_assignment_expanded_view] w
left join people_reports_view p
on w.people_id = p.people_id
```

```
left join next_scheduled_event_view nsev
on w.people_id = nsev.people_id
left join service_events_view sev
on w.people_id = sev.people_id
where
w.program_end_date is NULL
group by p.id_no,p.full_name
"""
clients_last_service_earliest_scheduled = pd.read_sql(query, cnxn)
```

In [8]:

```
#SQL counselor calendars of STAFF EVENTS ONLY
server = str(os.getenv('report_server'))
database = 'evolv_cs'
username = os.getenv('report_user')
password = os.getenv('report_pass')
cnxn = pyodbc.connect('DRIVER={SQL Server};SERVER='+server+';DATABASE='+database+';UID='+username+';PWD='+ password)
cursor = cnxn.cursor()
query = """
SELECT
      [staff_name]
      ,[event_name]
      ,[scheduled_date]
      ,[scheduled_end_date]
      ,[event_code]
      ,[category_code]
      ,[is_staff_event]
      ,DATEDIFF(MINUTE,[scheduled_date],[scheduled_end_date])/60.0 diff
FROM [evolv_cs].[dbo].[calendar_events2do_view]
WHERE (CONVERT(Date,scheduled_date) >= '""" + str(services_start_date.date()) + """')
AND (CONVERT(Date,scheduled_date) <= '""" + str(services_end_date.date()) + """')
and is_staff_event = 1
ORDER BY calendar_date_real desc
"""
schedules = pd.read_sql(query, cnxn)
```

In [10]:

```
#SQL staff_view
server = str(os.getenv('report_server'))
database = 'evolv_cs'
username = os.getenv('report_user')
password = os.getenv('report_pass')
cnxn = pyodbc.connect('DRIVER={SQL Server};SERVER='+server+';DATABASE='+database+';UID='+username+';PWD='+ password)
cursor = cnxn.cursor()
query = """
SELECT
```

```
      [last_name]
      ,[first_name]
      ,[job_title]
      ,[email_address]
      ,[email_staff]
      ,[start_date]
      ,[end_date]
      ,[security_scheme]
      ,[is_active]
      ,[is_active_staff]
  FROM [evolv_cs].[dbo].[staff_view]
  where is_active_staff = 1
  order by last_name, first_name
"""
staff = pd.read_sql(query, cnxn)
```

In [12]:
```
caseloads.drop_duplicates(keep = 'first', inplace = True)
```

In [13]:
```
#some clean up

#remove leading and trailing spaces
services = services.replace(r"^ +| +$", r"", regex=True)
caseloads = caseloads.replace(r"^ +| +$", r"", regex=True)
clients_last_service_earliest_scheduled = clients_last_service_earliest_scheduled.replace(r"^ +| +$", r"", regex=True)
schedules = schedules.replace(r"^ +| +$", r"", regex=True)

#create daily date
services['date'] = services['actual_date'].dt.date
clients_last_service_earliest_scheduled['next_scheduled_date'] = clients_last_service_earliest_scheduled['next_schedule
clients_last_service_earliest_scheduled['last_service_date'] = clients_last_service_earliest_scheduled['last_service_da

#Replace reason for no-show null values
services[['reason_for_no_show']] = services[['reason_for_no_show']].fillna(value = 'No Reason Given')
```

In [14]:
```
#generate staff names based on client services list
staff_names = services[
                    (services['actual_date'] >= services_start_date)
                    & (services['actual_date'] <= services_end_date)
                    & (services['program_name'].isin(['CCBHC','Outpatient','Intake /  Registration','Outpatient - Conti
                    ]
staff_names = staff_names.staff_name.unique()
```

In [15]:
```
services['date'] = pd.to_datetime(services['actual_date']).dt.normalize()
```

```
In [16]: #Caseload Continuing Care
def get_caseload_continuing_care_clients(caseloads, staff_name):
    result = caseloads[
                        (caseloads['worker_name'] == staff_name)
                        &
                        (caseloads['program_name'] == 'Outpatient - Continuing Care')
                        ].groupby(['id_no','worker_name'])['id_no'].nunique().sum()
    return str(result)
```

```
In [17]: #Caseload SUD
def get_caseload_sud_clients(caseloads, staff_name):
    result = caseloads[
                        (caseloads['worker_name'] == staff_name)
                        &
                        (caseloads['program_name'] == 'Outpatient')
                        ].groupby(['id_no','worker_name'])['id_no'].nunique().sum()
    return str(result)
```

```
In [18]: #Caseload CCBHC
def get_caseload_ccbhc_clients(caseloads, staff_name):

    result = caseloads[
                        (caseloads['worker_name'] == staff_name)
                        &
                        (caseloads['program_name'] == 'CCBHC')
                        ].groupby(['id_no','worker_name'])['id_no'].nunique().sum()
    return str(result)
```

```
In [19]: #Caseload Active Group Assignments
def get_active_group_assignments(group_assignments, staff_name):

    result = group_assignments[
                        (group_assignments['staff_name'] == staff_name)
                        ]
    if result.empty:
        return str(0)
    else:
        return str(result['count'].item())
```

```
In [20]: #Individual sessions sud count
def get_individual_sessions_op(staff_name, services, caseloads, services_start_date, services_end_date):
    result = services[
                            services['service'].isin
```

```
                                       (["* Individual Therapy","* Brief Treatment"])
                                        &
                                        (services['is_noshow'] == False)
                                        &
                                        (services['duration_num'] > 0 )
                                        &
                                        (services['staff_name'] == staff_name)
                                        &
                                        (services['actual_date'] >= services_start_date)
                                        &
                                        (services['actual_date'] <= services_end_date)
                                        &
                                        (services['program_name'] == 'Outpatient')
                                        &
                                        (services['facility'].isin(["Elizabethtown Outpatient","Keeseville Outpatient","Mal
                          ].groupby(['actual_date','full_name','facility','people_id'])['people_id'].nunique().sum()
    return str(result)
```

In [21]:
```
#Individual sessions CCBHC count
def get_individual_sessions_ccbhc(staff_name, services, caseloads, services_start_date, services_end_date):
    result = services[
                        services['service'].isin
                               (["CCBHC - Individual Therapy Note"])
                                &
                                (services['is_noshow'] == False)
                                &
                                (services['duration_num'] > 0 )
                                &
                                (services['staff_name'] == staff_name)
                                &
                                (services['actual_date'] >= services_start_date)
                                &
                                (services['actual_date'] <= services_end_date)
                                &
                                (services['program_name'] == 'CCBHC')
                                &
                                (services['facility'].isin(["Elizabethtown Outpatient","Keeseville Outpatient","Mal
                  ].groupby(['actual_date','full_name','facility','people_id'])['people_id'].nunique().sum()
    return str(result)
```

In [22]:
```
#Individual sessions OPCC count
def get_individual_sessions_opcc(staff_name, services, caseloads, services_start_date, services_end_date):
    result = services[
                        services['service'].isin
```

```
                                      (["* Individual Therapy"])
                                       &
                                       (services['is_noshow'] == False)
                                       &
                                       (services['duration_num'] > 0 )
                                       &
                                       (services['staff_name'] == staff_name)
                                       &
                                       (services['actual_date'] >= services_start_date)
                                       &
                                       (services['actual_date'] <= services_end_date)
                                       &
                                       (services['program_name'] == 'Outpatient - Continuing Care')
                                       &
                                       (services['facility'].isin(["Elizabethtown Outpatient","Keeseville Outpatient","Mal
                          ].groupby(['actual_date','full_name','facility','people_id'])['people_id'].nunique().sum()
    return str(result)
```

In [23]:
```
#group session count
#only coun them if they are more than 1 person, count once per person attended
def get_group_sessions(staff_name, services, caseloads, services_start_date, services_end_date):
    result = services[
                            services['service'].isin
                                      (["Group Therapy",
                                       "Group Therapy - Multi Family"])
                                       &
                                       (services['is_noshow'] == False)
                                       &
                                       (services['duration_num'] >= 60 )
                                       &
                                       (services['staff_name'] == staff_name)
                                       &
                                       (services['actual_date'] >= services_start_date)
                                       &
                                       (services['actual_date'] <= services_end_date)
                                       &
                                       (services['program_name'].isin(['Outpatient']))
                                       &
                                       (services['facility'].isin(["Elizabethtown Outpatient","Keeseville Outpatient","Mal
                          ].groupby(['actual_date','service','staff_name','facility','people_id'])['actual_date'].ngroups
    if result > 1:
        return str(result)
    else:
        return ''
```

```
In [24]: #SUD Assessments
         def get_sud_assessments(staff_name, services, caseloads, services_start_date, services_end_date):
             result = services[
                                     services['service'].isin
                                            (["1A CCHBC Client Adult Assessment",
                                              "1B CCBHC CHILD ASSESSMENT",
                                             "NOMS Assessment"])
                                             &
                                             (services['is_noshow'] == False)
                                             &
                                             (services['staff_name'] == staff_name)
                                             &
                                             (services['actual_date'] >= services_start_date)
                                             &
                                             (services['actual_date'] <= services_end_date)
                                             &
                                             services['program_name'].isin(['CCBHC','Outpatient','Intake /  Registration','Outpa

                                 ].groupby(['actual_date','service','staff_name','facility'])['actual_date'].ngroups
             return str(result)
```

```
In [25]: #CCBHC Assessments
         def get_ccbhc_assessments(staff_name, services, caseloads, services_start_date, services_end_date):
             result= services[
                                     services['service'].isin
                                            (["2 CCBHC BMI Assessment and Follow-Up",
                                             "3 CCBHC - Child Weight Assessment",
                                             "4 CCBHC Tobacco Counseling",
                                             "5A CCBHC Adult  Discharge Assessment",
                                             "5B CCBHC CHILD Discharge Assessment",
                                             "6 CCBHC Adult Suicide Risk",
                                             "7 CCBHC - Depression Screening",
                                             "9 CCBHC Child Suicide Risk Assessment",
                                             "10 A CCBHC Adult Reassessment W/Client",
                                             "10 B CCBHC CHILD REASSESSMENT w/Client",
                                             "11 CCBHC Alcohol Screen/Counseling",
                                              "~ 12 CCBHC - PHQ-9",
                                              "~ CCBHC - Mental Health Brief Assessment",
                                              "Mental Status Exam - CCBHC",
                                              "~ GAD 7"
                                             ])
                                             &
                                             (services['is_noshow'] == False)
                                             &
```

```
                                        (services['staff_name'] == staff_name)
                                        &
                                        (services['actual_date'] >= services_start_date)
                                        &
                                        (services['actual_date'] <= services_end_date)
                                        &
                                        services['program_name'].isin(['CCBHC','Outpatient','Intake /  Registration','Outpa

                            ].groupby(['actual_date','service','staff_name','people_id'])['people_id'].ngroups
    return str(result)
```

In [26]:

```
#SDV count
def get_same_day_visits(staff_name, services, caseloads, services_start_date, services_end_date):
    result = services[
                                services['service'].isin
                                       (["* Brief Intervention",
                                         "* Brief Treatment",
                                         "* Crisis Intervention",
                                         "* Family Collateral (Client Not Present)",
                                         "* Family Counseling - Client Present",
                                         "* Family Therapy - Client Present Telehealth",
                                         "* Individual Therapy",
                                         "* Injection - (only)",
                                         "* Medication Induction",
                                         "* Medication Management",
                                         "Contact - Individual less than 25 minutes",
                                         "Group Therapy","Initial Psychiatric Evaluation",
                                         "Intensive Outpatient Therapy - Group","Nurses Note",
                                         "Physicians Note","Psychiatry Follow-up Note"])
                                        &
                                        (services['is_noshow'] == False)
                                        &
                                        (services['duration_num'] > 0 )
                                        &
                                        (services['staff_name'] == staff_name)
                                        &
                                        (services['actual_date'] >= services_start_date)
                                        &
                                        (services['actual_date'] <= services_end_date)
                                        &
                                        (services['program_name'] == 'Outpatient')
                                        &
                                        (services['facility'].isin(["Elizabethtown Outpatient","Keeseville Outpatient","Mal
                            ].groupby(['full_name','date','facility'])['people_id'].count().pipe(lambda  dfx: dfx.loc[dfx>1
    return str(result)
```

```
In [27]: #Contacts
def get_client_contacts(staff_name, services, caseloads, services_start_date, services_end_date):
    result = services[
                        services['service'].isin
                               (["Client Communication Note","CCBHC - Client Communication Note"]
                               )
                                &
                                (services['is_noshow'] == False)
                                &
                                (services['staff_name'] == staff_name)
                                &
                                (services['actual_date'] >= services_start_date)
                                &
                                (services['actual_date'] <= services_end_date)
                                &
                                services['program_name'].isin(['CCBHC','Outpatient','Intake /  Registration','Outpa
                                &
                                (services['facility'].isin(["Elizabethtown Outpatient","Keeseville Outpatient","Mal
                    ].groupby(['actual_date','facility','staff_name'])['people_id'].ngroups
    return str(result)
```

```
In [28]: #Brief Intervention count
def get_brief_visits(staff_name, services, caseloads, services_start_date, services_end_date):
    result = services[
                        services['service'].isin
                               (["* Brief Intervention"]
                               )
                                &
                                (services['is_noshow'] == False)
                                &
                                (services['duration_num'] > 0 )
                                &
                                (services['staff_name'] == staff_name)
                                &
                                (services['actual_date'] >= services_start_date)
                                &
                                (services['actual_date'] <= services_end_date)
                                &
                                (services['program_name'].isin(['Outpatient','Intake /  Registration']))
                                &
                                (services['facility'].isin(["Elizabethtown Outpatient","Keeseville Outpatient","Mal
                    ].groupby(['actual_date','facility','staff_name','people_id'])['people_id'].ngroups
    return str(result)
```

```
In [29]: #CCBHC No-Shows
def get_ccbhc_noshows(staff_name, services, caseloads, services_start_date, services_end_date):
    result = services[
                                   (services['is_noshow'] == True)
                                   &
                                   (services['staff_name'] == staff_name)
                                   &
                                   (services['actual_date'] >= services_start_date)
                                   &
                                   (services['actual_date'] <= services_end_date)
                                   &
                                   (services['program_name'].isin(['CCBHC']))
                                   &
                                   (services['facility'].isin(["Elizabethtown Outpatient","Keeseville Outpatient","Mal
                        ].groupby(['actual_date','facility','staff_name','service','full_name'])['people_id'].ngroups
    return str(result)
```

```
In [30]: #OP No-Shows
def get_op_noshows(staff_name, services, caseloads, services_start_date, services_end_date):
    result = services[
                                   (services['is_noshow'] == True)
                                   &
                                   (services['staff_name'] == staff_name)
                                   &
                                   (services['actual_date'] >= services_start_date)
                                   &
                                   (services['actual_date'] <= services_end_date)
                                   &
                                   (services['program_name'].isin(['Outpatient']))
                                   &
                                   (services['facility'].isin(["Elizabethtown Outpatient","Keeseville Outpatient","Mal
                        ].groupby(['actual_date','facility','staff_name','service','full_name'])['people_id'].ngroups
    return str(result)
```

```
In [31]: #Other No-Shows
def get_other_noshows(staff_name, services, caseloads, services_start_date, services_end_date):
    result = services[
                                   (services['is_noshow'] == True)
                                   &
                                   (services['staff_name'] == staff_name)
                                   &
                                   (services['actual_date'] >= services_start_date)
                                   &
                                   (services['actual_date'] <= services_end_date)
```

```
                                    &
                                    (~services['program_name'].isin(['Outpatient','CCBHC']))
                                    &
                                    (services['facility'].isin(["Elizabethtown Outpatient","Keeseville Outpatient","Mal
                        ].groupby(['actual_date','facility','staff_name','service','full_name'])['people_id'].ngroups
    return str(result)
```

In [32]:
```
#All No-Show Reasons
def get_all_noshow_reasons(staff_name, services, caseloads, services_start_date, services_end_date):
    result = services[
                                    (services['is_noshow'] == True)
                                    &
                                    (services['staff_name'] == staff_name)
                                    &
                                    (services['actual_date'] >= services_start_date)
                                    &
                                    (services['actual_date'] <= services_end_date)
                                    &
                                    (services['program_name'].isin(['Outpatient','Intake /  Registration','CCBHC','Outp
                                    &
                                    (services['facility'].isin(["Elizabethtown Outpatient","Keeseville Outpatient","Mal
                        ].groupby(['reason_for_no_show'])['people_id'].count()
    return_str_list = []
    for key, value in result.items():
        return_str = ':'.join([key,str(value)])
        return_str_list.append(return_str)
    return_str = ', '.join(return_str_list)

    return return_str
```

In [33]:
```
#No client services in last x days
def get_clients_no_services_or_scheduled(staff_name, services, caseloads, clients_last_service_earliest_scheduled,next_
    #get results of staff worker caseload and merge with latest service date and soonest scheduled date data
    result = caseloads[['id_no','worker_name']]
    result.drop_duplicates(inplace=True)
    result = result[
                            (result['worker_name'] == staff_name)
                    ].merge(clients_last_service_earliest_scheduled, on='id_no', how='left')

    #filter down users who don't have a next scheduled visit or been seen recently
    result = result[
                              (
                                (result['next_scheduled_date_days'] > next_scheduled_service_days)
                                |
                                (result['next_scheduled_date_days']).isnull()
                              )
                                &
                               (
                                (result['last_service_date_days'] > last_service_days)
                                |
                                (result['last_service_date_days']).isnull()
                               )


                            ]
    return str(len(result))
```

In [34]:
```
#Staff Event Hours
def get_staff_event_hours(staff_name, schedules):
    result = schedules[

                                    (schedules['staff_name'] == staff_name)
                                    &
                                    (schedules['scheduled_date'] >= services_start_date)
                                    &
                                    (schedules['scheduled_date'] <= services_end_date)
    ].groupby(['event_name'])['diff'].sum().round(decimals = 1)
    return_str_list = []
    for key, value in result.items():
        return_str = ':'.join([key,str(value)])
        return_str_list.append(return_str)
    return_str = ', '.join(return_str_list)
    return return_str
```

In [35]:
```
#countable sessions
def get_countable_sessions(staff_name, services, caseloads, services_start_date, services_end_date):
    result = services[
                            services['service'].isin
                                   (["* Brief Intervention",
                                     "* Brief Treatment",
                                     "* Crisis Intervention",
                                     "* Family Collateral (Client Not Present)",
                                     "* Family Counseling - Client Present",
                                     "* Family Therapy - Client Present Telehealth",
                                     "* Individual Therapy",
                                     "* Injection - (only)",
                                     "* Medication Induction",
                                     "* Medication Management",
                                     "Contact - Individual less than 25 minutes",
                                     "Group Therapy",
                                     "Initial Psychiatric Evaluation",
                                     "Intensive Outpatient Therapy - Group",
                                     "Nurses Note","Physicians Note",
                                     "Psychiatry Follow-up Note"]
                                   )
                                    &
                                    (services['is_noshow'] == False)
                                    &
                                    (services['duration_num'] > 0 )
                                     &
                                    (services['staff_name'] == staff_name)
                                    &
                                    (services['actual_date'] >= services_start_date)
                                    &
                                    (services['actual_date'] <= services_end_date)
                                    &
                                    (services['program_name'] == 'Outpatient')
                                    &
                                    (services['facility'].isin(["Elizabethtown Outpatient","Keeseville Outpatient","Mal
                        ].groupby(['actual_date','full_name','facility','people_id'])['people_id'].nunique().sum()
    return str(result)
```

In [36]:
```
#billable sessions
def get_billable_sessions(staff_name, services, caseloads, services_start_date, services_end_date):
    result = services[
                        services['service'].isin
                               (["* Brief Intervention",
                                 "* Brief Treatment",
                                 "* Crisis Intervention",
                                 "* DSS Contact With Client",
                                 "* Family Collateral (Client Not Present)",
                                 "* Family Counseling - Client Present",
                                 "* Family Therapy - Client Present Telehealth",
                                 "* Individual Therapy",
                                 "* Injection - (only)",
                                 "* Medication Induction",
                                 "* Medication Management",
                                 "* Multi-Family Group",
                                 "* Peer Advocate",
                                 "* Pre-Assessment Progress Note",
                                 "Group Therapy"]
                               )
                                &
```

```
                                    (services['is_noshow'] == False)
                                    &
                                    (services['duration_num'] > 0 )
                                    &
                                    (services['staff_name'] == staff_name)
                                    &
                                    (services['actual_date'] >= services_start_date)
                                    &
                                    (services['actual_date'] <= services_end_date)
                                    &
                                    services['program_name'].isin(['CCBHC','Outpatient','Intake /  Registration','Outpa
                                    &
                                    (services['facility'].isin(["Elizabethtown Outpatient","Keeseville Outpatient","Mal
                        ].groupby(['actual_date','full_name','facility'])['people_id'].ngroups
    return str(result)
```

In [37]:

```
#CCBHC Sessions
def get_ccbhc_sessions(staff_name, services, caseloads, services_start_date, services_end_date):
    result = services[

                                    (services['is_noshow'] == False)
                                    &
                                    (services['duration_num'] > 0 )
                                     &
                                    (services['staff_name'] == staff_name)
                                    &
                                    (services['actual_date'] >= services_start_date)
                                    &
                                    (services['actual_date'] <= services_end_date)
                                    &
                                    (services['program_name'] == 'CCBHC')
                                    &
                                    (services['facility'].isin(["Elizabethtown Outpatient","Keeseville Outpatient","Mal
                        ].groupby(['actual_date','staff_name','facility'])['people_id'].ngroups
    return str(result)
```

In [38]:

```
def creat_word_document(staff_name, services, caseload, schedules, services_start_date, services_end_date,next_schedule
    #create word document
    from docx import Document
    from docx.shared import Cm, Pt
    from docx.enum.text import WD_ALIGN_PARAGRAPH
    from docx.enum.style import WD_STYLE_TYPE
```

```
wd = Document()

styles = wd.styles
style = styles.add_style('smaller',WD_STYLE_TYPE.PARAGRAPH )
style.base_style = styles['Normal']
style.paragraph_format.space_before = Pt(0)
style.paragraph_format.space_after = Pt(0)
font = style.font
font.size = Pt(8)

styles['Heading 3'].paragraph_format.space_before = Pt(0)
styles['Heading 3'].paragraph_format.space_after = Pt(0)


section = wd.sections[0]
footer = section.footer
footer.add_paragraph('Creation Date: ' + datetime.now().strftime("%m/%d/%Y %H:%M"), style = 'smaller')

table = wd.add_table(0,3)
table.style = 'TableGrid'

#first row header
table.add_row()
row = table.rows[0]
row.cells[0].text = 'Caseload Report'
row.cells[0].paragraphs[0].alignment = WD_ALIGN_PARAGRAPH.CENTER
table.cell(0,0).merge(table.cell(0,2))

#Report Dates
table.add_row()
row = table.rows[1]
row.cells[0].text = 'Reported Service Dates: ' + services_start_date.strftime("%Y-%m-%d") + ' through ' + services_
row.cells[0].paragraphs[0].alignment = WD_ALIGN_PARAGRAPH.CENTER
table.cell(1,0).merge(table.cell(1,2))

#Report Dates
table.add_row()
row = table.rows[2]
row.cells[0].text = 'Counselor Name'
row.cells[1].text = staff_name
row.cells[2].text = 'Date:'

#---------------#
#blank and merged
table.add_row()
```

```python
    row = table.rows[3]
    table.cell(3,0).merge(table.cell(3,2))
    #---------------#

    #caseload - merged
    table.add_row()
    row = table.rows[4]
    row.cells[0].text = 'Caseload as of ' + datetime.now().strftime("%m/%d/%Y %H:%M")
    table.cell(4,0).merge(table.cell(4,2))

    #Continuing Care Clients
    table.add_row()
    row = table.rows[5]
    row.cells[0].text = 'Continuing Care Clients'
    row.cells[1].text = get_caseload_continuing_care_clients(caseloads, staff_name)
    table.cell(5,1).merge(table.cell(5,2))

    #SUD Clients
    table.add_row()
    row = table.rows[6]
    row.cells[0].text = 'SUD Clients'
    row.cells[1].text = get_caseload_sud_clients(caseloads, staff_name)
    table.cell(6,1).merge(table.cell(6,2))


    #CCBHC Care Clients
    table.add_row()
    row = table.rows[7]
    row.cells[0].text = 'CCBHC Clients'
    row.cells[1].text = get_caseload_ccbhc_clients(caseloads, staff_name)
    table.cell(7,1).merge(table.cell(7,2))

    #Active Group Assignments
    table.add_row()
    row = table.rows[8]
    row.cells[0].text = 'Active Group Assignments'
    row.cells[1].text = get_active_group_assignments(group_assignments, staff_name)
    table.cell(8,1).merge(table.cell(8,2))

    #---------------#
    #blank and merged
    table.add_row()
    row = table.rows[9]
    table.cell(9,0).merge(table.cell(9,2))
    #---------------#
```

```
#Individual Sessions
table.add_row()
row = table.rows[10]
row.cells[0].text = 'Individual Sessions'
row.cells[1].text = 'OP: ' + get_individual_sessions_op(staff_name, services, caseloads, services_start_date, servi
table.cell(10,1).merge(table.cell(10,2))

#Group Sessions
table.add_row()
row = table.rows[11]
row.cells[0].text = 'Group Sessions'
row.cells[1].text = get_group_sessions(staff_name, services, caseloads, services_start_date, services_end_date)
table.cell(11,1).merge(table.cell(11,2))

#SUD Assessments
table.add_row()
row = table.rows[12]
row.cells[0].text = 'SUD Assessments'
row.cells[1].text = get_sud_assessments(staff_name, services, caseloads, services_start_date, services_end_date)
table.cell(12,1).merge(table.cell(12,2))

#CCBHC Assessments
table.add_row()
row = table.rows[13]
row.cells[0].text = 'CCBHC Assessments'
row.cells[1].text = get_ccbhc_assessments(staff_name, services, caseloads, services_start_date, services_end_date)
table.cell(13,1).merge(table.cell(13,2))

#Same Day Visits
table.add_row()
row = table.rows[14]
row.cells[0].text = 'Same Day Visits'
row.cells[1].text = get_same_day_visits(staff_name, services, caseloads, services_start_date, services_end_date)
table.cell(14,1).merge(table.cell(14,2))

#Client Contacts
table.add_row()
row = table.rows[15]
row.cells[0].text = 'Client Contacts'
row.cells[1].text = get_client_contacts(staff_name, services, caseloads, services_start_date, services_end_date)
table.cell(15,1).merge(table.cell(15,2))

#Briefs
table.add_row()
```

```
    row = table.rows[16]
    row.cells[0].text = 'Briefs'
    row.cells[1].text = get_brief_visits(staff_name, services, caseloads, services_start_date, services_end_date)
    table.cell(16,1).merge(table.cell(16,2))

    #No Shows
    table.add_row()
    row = table.rows[17]
    row.cells[0].text = 'No Shows'
    row.cells[1].text = 'CCBHC: ' + str(get_ccbhc_noshows(staff_name, services, caseloads, services_start_date, service
    table.cell(17,1).merge(table.cell(17,2))

    #No Show Reasons
    table.add_row()
    row = table.rows[18]
    row.cells[0].text = 'No Show Reasons'
    row.cells[1].text = get_all_noshow_reasons(staff_name, services, caseloads, services_start_date, services_end_date)
    table.cell(18,1).merge(table.cell(18,2))

    #Client no follow up 14 days
    table.add_row()
    row = table.rows[19]
    row.cells[0].text = '# clients no actual/scheduled services in last/next 14 days'
    row.cells[1].text = get_clients_no_services_or_scheduled(staff_name, services, caseloads, clients_last_service_earl
    table.cell(19,1).merge(table.cell(19,2))

    #Staff Event Hours
    table.add_row()
    row = table.rows[20]
    row.cells[0].text = 'Staff Event Hours - Overlapping Events Included'
    row.cells[1].text = get_staff_event_hours(staff_name, schedules)
    table.cell(20,1).merge(table.cell(20,2))


    #---------------#
    #rows - blank and merged

    #Blank
    table.add_row()
    row = table.rows[21]
    table.cell(21,0).merge(table.cell(21,2))
    #---------------#

    #Total # of Countable Sessions
    table.add_row()
    row = table.rows[22]
    row.cells[1].text = 'Total Countable Sessions'
    row.cells[2].text = get_countable_sessions(staff_name, services, caseloads, services_start_date, services_end_date)

    #Total # of Billable Sessions
    table.add_row()
    row = table.rows[23]
    row.cells[1].text = 'Total Billable Sessions'
    row.cells[2].text = get_billable_sessions(staff_name, services, caseloads, services_start_date, services_end_date)

    #Total # of CCBHC Sessions
    table.add_row()
    row = table.rows[24]
    row.cells[1].text = 'Total CCBHC Sessions'
    row.cells[2].text = get_ccbhc_sessions(staff_name, services, caseloads, services_start_date, services_end_date)

    wd.add_paragraph('\r\nCounselor Signature: __________________________________________________________________\r\n')
    wd.add_paragraph('Supervisor Signature: __________________________________________________________________')

    #2nd page - adding definitions page
    wd.add_page_break()
    wd.add_heading('Definitions', level=1)

    #Caseload
    wd.add_heading('Continuing Care\SUD\CCBHC\Group Clients', level=3)
    wd.add_paragraph('Clients assigned to you from Outpatient, Outpatient - Continuing Care\CCBHC programs at time of re

    #Services
    wd.add_heading('Individual Sessions', level=3)
    paragraph = wd.add_paragraph('OP = Services = * Individual Therapy,* Brief Treatment: CCBHC = CCBHC - Individual The

    wd.add_heading('Group Sessions', level=3)
    paragraph = wd.add_paragraph('Services = Group Therapy, Group Therapy - Multi Family;NoShow = False; Duration >=60;

    wd.add_heading('SUD Assessments', level=3)
    paragraph = wd.add_paragraph('Services = 1A CCHBC Client Adult Assessment,1B CCBHC CHILD ASSESSMENT,NOMS Assessment

    wd.add_heading('CCBHC Assessments', level=3)
    paragraph = wd.add_paragraph('Services = 2 CCBHC BMI Assessment and Follow-Up,3 CCBHC - Child Weight Assessment,4 CC

    wd.add_heading('Same Day Visits', level=3)
    paragraph = wd.add_paragraph('Services = * Brief Intervention,* Brief Treatment* Crisis Intervention,* Family Colla

    wd.add_heading('Client Contacts', level=3)
    paragraph = wd.add_paragraph('Services = Client Communication Note,CCBHC - Client Communication Note; NoShow = False
    wd.add_heading('Briefs', level=3)
    paragraph = wd.add_paragraph('Services = * Brief Intervention; NoShow = False; Duration > 0; Program = Outpatient o

    wd.add_heading('No Shows', level=3)
    paragraph = wd.add_paragraph('Services = Any; NoShow = True; Facility = Any OP facility', style = 'smaller')

    wd.add_heading('No Show Reasons', level=3)
    paragraph = wd.add_paragraph('Services = Any; NoShow = True; Facility = Any OP facility; Program = Outpatient,Intake

    wd.add_heading('# clients no actual/scheduled services in last 14 days', level=3)
    paragraph = wd.add_paragraph('# of clients on caseload that have not had a service in the last 14 days or do not hav

    wd.add_heading('Staff Event Hours', level=3)
    paragraph = wd.add_paragraph('Staff event time scheduled in clinician''s calendar by category. All events, even ove

    # clients no actual/scheduled services in last 14 days

    #Totals
    wd.add_heading('Total Countable Sessions', level=3)
    paragraph = wd.add_paragraph('Services = Same as SDV above, all visits (even same day) count; NoShow = False; Durat

    wd.add_heading('Total Billable Sessions', level=3)
    paragraph = wd.add_paragraph('Services = Services starting with *, Group Therapy; NoShow = False; Duration > 0; Fac

    wd.add_heading('Total CCBHC Sessions', level=3)
    paragraph = wd.add_paragraph('Services = any; NoShow = False, Duration > 0; Program = CCBHC; Facility = any OP faci

    wd.save(reportpath + '\\' + staff_name + '.docx')
    return reportpath + '\\' + staff_name + '.docx'
```

In [39]:
```
# Make the folder if it doesn't exist
try:
    os.makedirs(reportpath)
except OSError as e:
    if e.errno != errno.EEXIST:
        raise
```

In [40]:
```
# List of staff members who management wants reports created for
staff_names_for_email = {
    ##'Staff Name':'Staff email address'

}
```

```
In [ ]: for staff_name in staff_names:
            if staff_name in staff_names_for_email.keys():
                print(staff_name + " " + staff_names_for_email[staff_name])
                report_path = creat_word_document(staff_name, services, caseloads, schedules, services_start_date, services_end
                outlook = win32.Dispatch('outlook.application')
                mail = outlook.CreateItem(0)
                mail.To = staff_names_for_email[staff_name]
                mail.CC = ''
                mail.Subject = staff_name + ' Workload Report for ' + services_start_date.strftime("%Y-%m-%d") + ' through ' +
                mail.HtmlBody = ('Please find attached a report that summarizes the following for dates ' + services_start_date
        '&emsp;a. The number of persons assigned to you in each layer of Netsmart<br>'
        '&emsp;b. Number of services you provided in the different layers in Netsmart<br>'
        '&emsp;c. Number of no shows from your scheduled services<br>'
        '&emsp;d. Number of meetings, trainings, non-service events that you attended<br>'
        'Please bring this report to your supervision meeting.<br><br>'
        '<br><br>'
        'Mike Beccaria<br>Sr. Director of Business Intelligence<br>he/him/his<br>'
        'St. Joseph’s Addiction Treatment & Recovery Center<br>'
        '159 Glenwood Dr., PO Box 470, Saranac Lake, NY 12983<br>'
        '518.637.4314<br>www.stjoestreatment.org<br>'
                                    )
                mail.Attachments.Add(Source = report_path)
                #mail.Save()
                mail.Send()
```

```
In [ ]: for staff_name_email in staff_names_for_email.keys():
            if staff_name_email not in staff_names:
                print(staff_name_email)
```