Vaccines Given to Patients Under 19 Years Old in 2021

-This Notebook was created using bulk Healthcare Data regarding vaccines given that has had all PHI *(*Protected Health Information) removed. This data was sourced from a SQL Server using a SQL query (that will be commented below) and exported into an Excel file to then have all PHI removed. The file was then saved as a CSV file for import and use in this Python Jupyter Notebook.-

SQL Query

SELECT DISTINCT e3_Vaccines_Administered.ImmunizationId, e1_Master_Patients.e1MasterPatients_ID, e3_Vaccines_Administered.Imm_Pat_ID, e3_Vaccines_Administered.Imm_Pat_ID2, e3_Vaccines_Administered.Immun_Enc_ID, e3_Vaccines_Administered.Enc_Claim_ID, e3_Vaccines_Administered.Enc_ClaimID_is_0, e3_Vaccines_Administered.Claim_Ins_Group, e1_Master_Patients.e1Pri_Ins_Grp, e3_Vaccines_Administered.Ins_Group, e3_Vaccines_Administered.Ins_CHIP, (Int(DateDiff("d",e1_Master_Patients.e1Pat_DOB,e3_Vaccines_Administered.Imm_GivenDate)/365)) AS Age_at_Imm_GivenDate, e3_Vaccines_Administered.Pat_Older_19, e3_Vaccines_Administered.Age_Group, e3_Vaccines_Administered.Enc_Date, e3_Vaccines_Administered.Enc_Date_YYMM, e3_Vaccines_Administered.Enc_Service_Location, e3_Vaccines_Administered.Imm_ItemID, e3_Vaccines_Administered.Imm_Item_Name, e3_Vaccines_Administered.Imm_VFC_AOV, e3_Vaccines_Administered.Imm_VFC_AOV_Reclass, e3_Vaccines_Administered.Imm_VFC_Report_Column, e3_Vaccines_Administered.Imm_USIIS_Desc, e3_Vaccines_Administered.Imm_GivenDate, e3_Vaccines_Administered.Imm_Given_Date_YYMM, Format(e3_Vaccines_Administered.Imm_GivenDate,"yyyy") & "/Q" & Format(e3_Vaccines_Administered.Imm_GivenDate,"q") AS Quarter, e3_Vaccines_Administered.Given_Date_Seen_by_CHC, e3_Vaccines_Administered.Given_by_CHC_derived, e3_Vaccines_Administered.Imm_LotID, e3_Vaccines_Administered.Imm_LotNumber, e3_Vaccines_Administered.Imm_Dose, e3_Vaccines_Administered.Imm_VaccineManufacturer, e3_Vaccines_Administered.Imm_Facility, e3_Vaccines_Administered.e1Service_Location, e3_Vaccines_Administered.Imm_CPTcvx, e3_Vaccines_Administered.Imm_VaccineName, e3_Vaccines_Administered.Imm_Mvx_Code, e3_Vaccines_Administered.Imm_Cvx_Code, e3_Vaccines_Administered.Imm_VFC_Code, e3_Vaccines_Administered.Imm_Action, e3_Vaccines_Administered.Imm_decDoses, e3_Vaccines_Administered.Imm_Route, e3_Vaccines_Administered.QA_Facility_Match, e3_Vaccines_Administered.QA_Lot_Number_Match, e3_Vaccines_Administered.Units, e3_Vaccines_Administered.Vaccine_Billed_on_Claim, e3_Vaccines_Administered.Vaccine_Billed_Amount INTO (Vax_Admin_Prt1 for 2021 no PHI) FROM e3_Vaccines_Administered LEFT JOIN e1_Master_Patients ON (e3_Vaccines_Administered.Imm_Pat_ID2 = e1_Master*Patients.e1Pat#*_2) AND (e3_Vaccines_Administered.Imm_Pat_ID = e1_Master*Patients.e1Pat#*) WHERE e3_Vaccines_Administered.Imm_Given_Date_YYMM > '2020/12' AND e3_Vaccines_Administered.Imm_Given_Date_YYMM < '2022/01';

Import Data, Packages and Libraries Needed

In [1]:
```
import numpy as np
import pandas as pd
```

In [2]:
```
# import and create DataFrame from CSV
vax_data = pd.read_csv('Vax_Admin_Prt1 for 2021 no PHI.csv')
```

In [3]:
```
# Visualize top 5 Rows to make sure data was imported correctly
vax_data.head()
```

Out[3]:

	ImmunizationId	e1MasterPatients_ID	Imm_Pat_ID	Imm_Pat_ID2	Immun_Enc_ID	Enc_Claim_ID	Enc_ClaimID_is_0	Claim_Ins_Group	e1Pri_Ins_Grp	Ins_Group	...	Imm_Cvx_Code	Imm_VFC_Code	Imm_
0	406327	131207	1074560	1074560	3494445	1723886.0	N	1)Self Pay	9)Private Ins	Non-Insured	...	150	V07	
1	406328	209653	1051974	1051974	3632143	1724411.0	N	9)Private Ins	9)Private Ins	Insured	...	130	V01	
2	406329	209653	1051974	1051974	3632143	1724411.0	N	9)Private Ins	9)Private Ins	Insured	...	94	V01	
3	406330	160779	1127359	1127359	3605232	1723957.0	N	3)Mcaid HMO	3)Mcaid HMO	Medicaid	...	133	V02	
4	406331	219842	1124666	1124666	3649693	1723909.0	N	3)Mcaid HMO	3)Mcaid HMO	Medicaid	...	48	V02	

5 rows × 47 columns

In [4]:
```
# explore data, find out how many missing values, etc...
vax_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 64673 entries, 0 to 64672
Data columns (total 47 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   ImmunizationId           64673 non-null  int64
 1   e1MasterPatients_ID      64673 non-null  int64
 2   Imm_Pat_ID               64673 non-null  int64
 3   Imm_Pat_ID2              64673 non-null  object
 4   Immun_Enc_ID             64673 non-null  int64
 5   Enc_Claim_ID             52755 non-null  float64
 6   Enc_ClaimID_is_0         64673 non-null  object
 7   Claim_Ins_Group          52503 non-null  object
 8   e1Pri_Ins_Grp            39233 non-null  object
 9   Ins_Group                64673 non-null  object
 10  Ins_CHIP                 64673 non-null  object
 11  Age_at_Imm_GivenDate     64673 non-null  int64
 12  Pat_Older_19             64673 non-null  object
 13  Age_Group                64673 non-null  object
 14  Enc_Date                 52755 non-null  object
 15  Enc_Date_YYMM            52755 non-null  object
 16  Enc_Service_Location     52755 non-null  object
 17  Imm_ItemID               64673 non-null  int64
 18  Imm_Item_Name            64672 non-null  object
 19  Imm_VFC_AOV              64673 non-null  object
 20  Imm_VFC_AOV_Reclass      64673 non-null  object
 21  Imm_VFC_Report_Column    31755 non-null  object
 22  Imm_USIIS_Desc           31990 non-null  object
 23  Imm_GivenDate            64673 non-null  object
 24  Imm_Given_Date_YYMM      64673 non-null  object
 25  Quarter                  64673 non-null  object
 26  Given_Date_Seen_by_CHC   64673 non-null  object
 27  Given_by_CHC_derived     62504 non-null  object
 28  Imm_LotID                53872 non-null  float64
 29  Imm_LotNumber            62465 non-null  object
 30  Imm_Dose                 64473 non-null  object
 31  Imm_VaccineManufacturer  62227 non-null  object
 32  Imm_Facility             63481 non-null  object
 33  e1Service_Location       52742 non-null  object
 34  Imm_CPTcvx               64661 non-null  object
 35  Imm_VaccineName          64661 non-null  object
 36  Imm_Mvx_Code             49548 non-null  object
 37  Imm_Cvx_Code             64647 non-null  object
 38  Imm_VFC_Code             51897 non-null  object
 39  Imm_Action               53872 non-null  object
 40  Imm_decDoses             64673 non-null  int64
 41  Imm_Route                51929 non-null  object
 42  QA_Facility_Match        64673 non-null  object
 43  QA_Lot_Number_Match      64673 non-null  object
 44  Units                    64673 non-null  int64
 45  Vaccine_Billed_on_Claim  64673 non-null  object
 46  Vaccine_Billed_Amount    31265 non-null  object
dtypes: float64(2), int64(8), object(37)
memory usage: 23.2+ MB
```

In [5]:
```
# lets see summary statistics
vax_data.describe()
```

Out[5]:

	ImmunizationId	e1MasterPatients_ID	Imm_Pat_ID	Immun_Enc_ID	Enc_Claim_ID	Age_at_Imm_GivenDate	Imm_ItemID	Imm_LotID	Imm_decDoses	Units
count	64673.000000	64673.000000	6.467300e+04	6.467300e+04	5.275500e+04	64673.000000	64673.000000	53872.000000	64673.000000	64673.0
mean	440424.877801	144379.688015	8.826000e+05	3.118431e+06	1.807008e+06	24.870255	401630.093146	8178.246325	0.982821	1.0
std	19879.515334	64219.848012	3.887205e+05	1.486026e+06	1.214959e+05	24.189748	99686.393255	1694.057132	0.129938	0.0
min	406327.000000	77.000000	1.665260e+05	0.000000e+00	0.000000e+00	0.000000	-1.000000	0.000000	0.000000	1.0
25%	423324.000000	94394.000000	1.001715e+06	3.657480e+06	1.767370e+06	1.000000	282408.000000	8162.000000	1.000000	1.0
50%	440323.000000	161790.000000	1.093289e+06	3.779761e+06	1.819274e+06	17.000000	443622.000000	8471.000000	1.000000	1.0
75%	457133.000000	198897.000000	1.134199e+06	3.910336e+06	1.858228e+06	46.000000	493969.000000	8873.000000	1.000000	1.0
max	506722.000000	228556.000000	1.153654e+06	4.026727e+06	1.908845e+06	96.000000	503393.000000	9266.000000	1.000000	1.0

*interesting to see the youngest person to receive a vaccine was 0 and the oldest was 96 with the average age being almmost 25 (24.87)

In [6]:
```
# we will be comparing the field 'Vaccine_Billed_Amount' later and thus need to change from $ currency to
# just a float without the dollar sign or any commas

vax_data['Vaccine_Billed_Amount'] = vax_data['Vaccine_Billed_Amount'].str.replace(',','')
vax_data['Vaccine_Billed_Amount'] = vax_data['Vaccine_Billed_Amount'].str.replace('$','')
vax_data['Vaccine_Billed_Amount'] = vax_data['Vaccine_Billed_Amount'].astype(float)
```

```
C:\Users\cflink\AppData\Local\Temp\ipykernel_21008\814292187.py:5: FutureWarning: The default value of regex will change from True to False in a future version. In addition, single character regular expressions will *not* be treated as literal strings when regex=True.
  vax_data['Vaccine_Billed_Amount'] = vax_data['Vaccine_Billed_Amount'].str.replace('$','')
```

Find out if we have any duplicates

In [7]:
```
duplicate_vax = vax_data[vax_data.duplicated('ImmunizationId')]

print("Duplicate Rows based on ImmunizationId :")
duplicate_vax
```

```
Duplicate Rows based on ImmunizationId :
```

Out[7]:

	ImmunizationId	e1MasterPatients_ID	Imm_Pat_ID	Imm_Pat_ID2	Immun_Enc_ID	Enc_Claim_ID	Enc_ClaimID_is_0	Claim_Ins_Group	e1Pri_Ins_Grp	Ins_Group	...	Imm_Cvx_Code	Imm_VFC_Code
1190	407619	201168	227597	21535.1	3648090	1727893.0	N	3)Mcaid HMO	3)Mcaid HMO	Medicaid	...	158	V02
1388	407825	49192	222628	188329.4	3663592	1728455.0	N	1)Self Pay	2)Medicaid	Non-Insured	...	150	V01
1866	408332	207264	1131306	1131306	3641511	1730083.0	N	1)Self Pay	9)Private Ins	Non-Insured	...	150	V03
3427	409979	171156	1117083	1117083	3672280	1736565.0	N	3)Mcaid HMO	3)Mcaid HMO	Medicaid	...	83	V02
4803	411475	49192	222628	188329.4	3666779	1741511.0	N	1)Self Pay	2)Medicaid	Non-Insured	...	115	V01
...	...	...	...	...	...	...	...	...	...	...	...	...	...
61268	470239	13474	183394	183785	4012892	1887081.0	N	6)Medicare	6)Medicare	Insured	...	207	V01
61269	470239	13475	183394	183785	4012892	1887081.0	N	6)Medicare	6)Medicare	Insured	...	207	V01
63326	474430	101849	237495	37571.1	0	NaN	N	NaN	3)Mcaid HMO	Medicaid	...	208	NaN
63328	474431	101849	237495	37571.1	0	NaN	N	NaN	3)Mcaid HMO	Medicaid	...	208	NaN
63365	474986	165643	1133164	1133164	0	NaN	N	NaN	3)Mcaid HMO	Medicaid	...	207	NaN

135 rows × 47 columns

Drop Duplicates

In [8]:
```
vax_data.drop_duplicates(subset=['ImmunizationId'], keep='first', inplace=True)
# dropped dups and saved df 'inplace' instead of a new df, check df.info to see how many rows were dropped
# df had 64673 entries to begin with, we found there to be 135 duplicates, after dropping those we ended up with 135 less
# entries at a total of 64538
vax_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 64538 entries, 0 to 64672
Data columns (total 47 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   ImmunizationId          64538 non-null  int64
 1   e1MasterPatients_ID     64538 non-null  int64
 2   Imm_Pat_ID              64538 non-null  int64
 3   Imm_Pat_ID2             64538 non-null  object
 4   Immun_Enc_ID            64538 non-null  int64
 5   Enc_Claim_ID            52650 non-null  float64
 6   Enc_ClaimID_is_0        64538 non-null  object
 7   Claim_Ins_Group         52398 non-null  object
 8   e1Pri_Ins_Grp           39099 non-null  object
 9   Ins_Group               64538 non-null  object
 10  Ins_CHIP                64538 non-null  object
 11  Age_at_Imm_GivenDate    64538 non-null  int64
 12  Pat_Older_19            64538 non-null  object
 13  Age_Group               64538 non-null  object
 14  Enc_Date                52650 non-null  object
 15  Enc_Date_YYMM           52650 non-null  object
 16  Enc_Service_Location    52650 non-null  object
 17  Imm_ItemID              64538 non-null  int64
 18  Imm_Item_Name           64537 non-null  object
 19  Imm_VFC_AOV             64538 non-null  object
 20  Imm_VFC_AOV_Reclass     64538 non-null  object
 21  Imm_VFC_Report_Column   31705 non-null  object
 22  Imm_USIIS_Desc          31940 non-null  object
 23  Imm_GivenDate           64538 non-null  object
 24  Imm_Given_Date_YYMM     64538 non-null  object
 25  Quarter                 64538 non-null  object
 26  Given_Date_Seen_by_CHC  64538 non-null  object
 27  Given_by_CHC_derived    62377 non-null  object
 28  Imm_LotID               53762 non-null  float64
 29  Imm_LotNumber           62338 non-null  object
 30  Imm_Dose                64338 non-null  object
 31  Imm_VaccineManufacturer 62100 non-null  object
 32  Imm_Facility            63351 non-null  object
 33  e1Service_Location      52637 non-null  object
 34  Imm_CPTcvx              64526 non-null  object
 35  Imm_VaccineName         64526 non-null  object
 36  Imm_Mvx_Code            49450 non-null  object
 37  Imm_Cvx_Code            64512 non-null  object
 38  Imm_VFC_Code            51791 non-null  object
 39  Imm_Action              53762 non-null  object
 40  Imm_decDoses            64538 non-null  int64
 41  Imm_Route               51823 non-null  object
 42  QA_Facility_Match       64538 non-null  object
 43  QA_Lot_Number_Match     64538 non-null  object
 44  Units                   64538 non-null  int64
 45  Vaccine_Billed_on_Claim 64538 non-null  object
 46  Vaccine_Billed_Amount   31215 non-null  float64
dtypes: float64(3), int64(8), object(36)
memory usage: 23.6+ MB
```

In [9]:
```
# Lets see if there are any duplicates now
duplicate_vax2 = vax_data[vax_data.duplicated('ImmunizationId')]

print("Duplicate Rows based on ImmunizationId :")
duplicate_vax2
```

```
Duplicate Rows based on ImmunizationId :
```

Out[9]:

ImmunizationId	e1MasterPatients_ID	Imm_Pat_ID	Imm_Pat_ID2	Immun_Enc_ID	Enc_Claim_ID	Enc_ClaimID_is_0	Claim_Ins_Group	e1Pri_Ins_Grp	Ins_Group	...	Imm_Cvx_Code	Imm_VFC_Code	Imm_A

0 rows × 47 columns

looks like we removed all the duplicate rows based on 'ImmunizationId' which was the primary key in the database we pulled from

Remove Non-Vaccine Injections and Vaccines Not Given by Organization

create a couple of lists:

one with all non-vaccine injections & one with locations that are part of organization when 'Enc_Claim_ID' is NaN

In [10]:
```
non_vax = ['Ketorolac Tromethamine', 'triamcinalone', 'Depo 150 Mg Unins (Supply + Discount Fee)', 'Ceftriaxone', 'Cyanocobalamin(Vitamin B12)', 'DEPO 150 mg uninsured DO
in_org = ['OQUIRRH VIEW CHC', 'CENTRAL CITY CHC', 'STEPHEN D RATCLIFFE', '72ND STREET CHC', 'NEIGHBORHOOD CHC', 'ELLIS SHIPP CHC']
```

In [11]:
```
#drop all vax not given in an organization facility
enc_claim_na = vax_data[vax_data['Enc_Claim_ID'].isna()]

non_org = enc_claim_na[~enc_claim_na['Imm_Facility'].isin(in_org)]

non_org_idx = enc_claim_na[~enc_claim_na['Imm_Facility'].isin(in_org)].index

vax_data.drop(non_org_idx, inplace=True)
```

In [12]:
```
#drop all non-vax injections
vax_data = vax_data[~vax_data['Imm_Item_Name'].isin(non_vax)]
```

Missing Values

Some of the missing values are needed to analyze the data properly and filter for our needs

In [13]:
```
# Claim_Ins_Group & e1Pri_Ins_Grp have 'null' values that need to be imputed from the other fields

vax_data['e1Pri_Ins_Grp']  = np.where((vax_data['e1Pri_Ins_Grp'].isna()) &
                                      (vax_data['Claim_Ins_Group'].isna()) &
                                      (vax_data['Ins_Group'] == 'Non-Insured'), '1)Self Pay', vax_data['e1Pri_Ins_Grp'])

vax_data['Claim_Ins_Group'] = np.where((vax_data['Claim_Ins_Group'].isna()),
                                       vax_data['e1Pri_Ins_Grp'], vax_data['Claim_Ins_Group'])

vax_data['e1Pri_Ins_Grp'] = np.where((vax_data['e1Pri_Ins_Grp'].isna()),
                                       vax_data['Claim_Ins_Group'], vax_data['e1Pri_Ins_Grp'])
```

In [14]:
```
# 'Imm_VFC_Report_Column' & 'Imm_USIIS_Desc' have 'null'/missing values that need to be filled in
# This data is all categorized by 'Imm_ItemID' field, which has a reference table that I will import.
# Then we can join the two tables and fill the missing values

vax_ref_data = pd.read_csv('Vax_ref_Imm_table.csv')

vax_data = vax_data.merge(vax_ref_data, how='left')
# drop the columns that are replaced with 'm1VFC_Report_Column' & 'USIIS_Desc' from the reference table
vax_data = vax_data.drop(['Imm_VFC_Report_Column', 'Imm_USIIS_Desc'], axis=1)
```

In [15]:
```
# reorder columns to put 'm1VFC_Report_Column' & 'USIIS_Desc' where 'Imm_VFC_Report_Column' & 'Imm_USIIS_Desc' where
# and place the new 'VFC/AOV' column next to the columns with similar data for comparison sake

vax_data = vax_data[['ImmunizationId','e1MasterPatients_ID','Imm_Pat_ID','Imm_Pat_ID2','Immun_Enc_ID',
 'Enc_Claim_ID','Enc_ClaimID_is_0','Claim_Ins_Group','e1Pri_Ins_Grp','Ins_Group','Ins_CHIP','Age_at_Imm_GivenDate',
 'Pat_Older_19','Age_Group','Enc_Date','Enc_Date_YYMM','Enc_Service_Location','Imm_ItemID','Imm_Item_Name',
 'Imm_VFC_AOV','Imm_VFC_AOV_Reclass','VFC/AOV','m1VFC_Report_Column','USIIS_Desc','Imm_GivenDate','Imm_Given_Date_YYMM',
 'Quarter','Given_Date_Seen_by_CHC','Given_by_CHC_derived','Imm_LotID','Imm_LotNumber','Imm_Dose',
 'Imm_VaccineManufacturer','Imm_Facility','e1Service_Location','Imm_CPTcvx','Imm_VaccineName','Imm_Mvx_Code',
 'Imm_Cvx_Code','Imm_VFC_Code','Imm_Action','Imm_decDoses','Imm_Route','QA_Facility_Match','QA_Lot_Number_Match',
 'Units','Vaccine_Billed_on_Claim','Vaccine_Billed_Amount']]
```

In [16]:
```
# reset index after multiple changes
vax_data = vax_data.reset_index(drop=True)
```

Last Step - Update 'Imm_VFC_AOV_Reclass' Column

when a vaccine was given to a child (less than 19) but was billed it is not considered "VFC" but "AOV"

In [17]:
```
# lets use a for loop with some if statements

for i in vax_data.index:
    if vax_data.at[i, 'Age_at_Imm_GivenDate'] > 19:
        vax_data.at[i, 'Imm_VFC_AOV_Reclass'] = 'AOV'
    else:
        if vax_data.at[i, 'Vaccine_Billed_Amount'] == 0:
            vax_data.at[i, 'Imm_VFC_AOV_Reclass'] = 'VFC'
        elif vax_data.at[i, 'Vaccine_Billed_Amount'] > 0:
            vax_data.at[i, 'Imm_VFC_AOV_Reclass'] = 'AOV'
        else:
            vax_data.at[i, 'Imm_VFC_AOV_Reclass'] = vax_data.at[i, 'Imm_VFC_AOV']
```

Answering the Question: How Many Vaccines Were Given to Patients Under 19 Years Old in 2021

In [18]:
```
vax_given_under19 = len(vax_data[vax_data['Age_at_Imm_GivenDate'] < 19])

print("There were " + str(vax_given_under19) + " vaccines given to patients under 19 years old in 2021.")
```

```
There were 31899 vaccines given to patients under 19 years old in 2021.
```

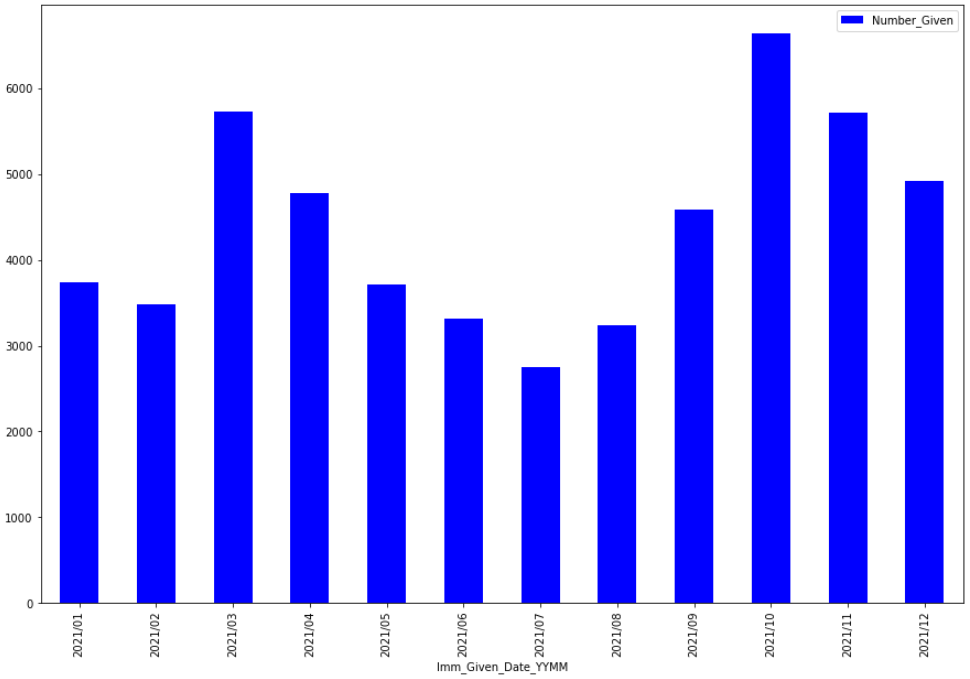
Lets Visualize Some of this Data

In [19]:
```
month_occ = vax_data.groupby(['Imm_Given_Date_YYMM']).count()
```

```
month_occ = month_occ.reset_index()
month_occ = month_occ[['Imm_Given_Date_YYMM', 'ImmunizationId']]
month_occ.columns = ['Imm_Given_Date_YYMM', 'Number_Given']
month_occ_plot = month_occ.plot.bar(x='Imm_Given_Date_YYMM', y='Number_Given', color = 'blue', figsize = (15,10))
month_occ_plot
```

Out[19]:

```
<AxesSubplot:xlabel='Imm_Given_Date_YYMM'>
```

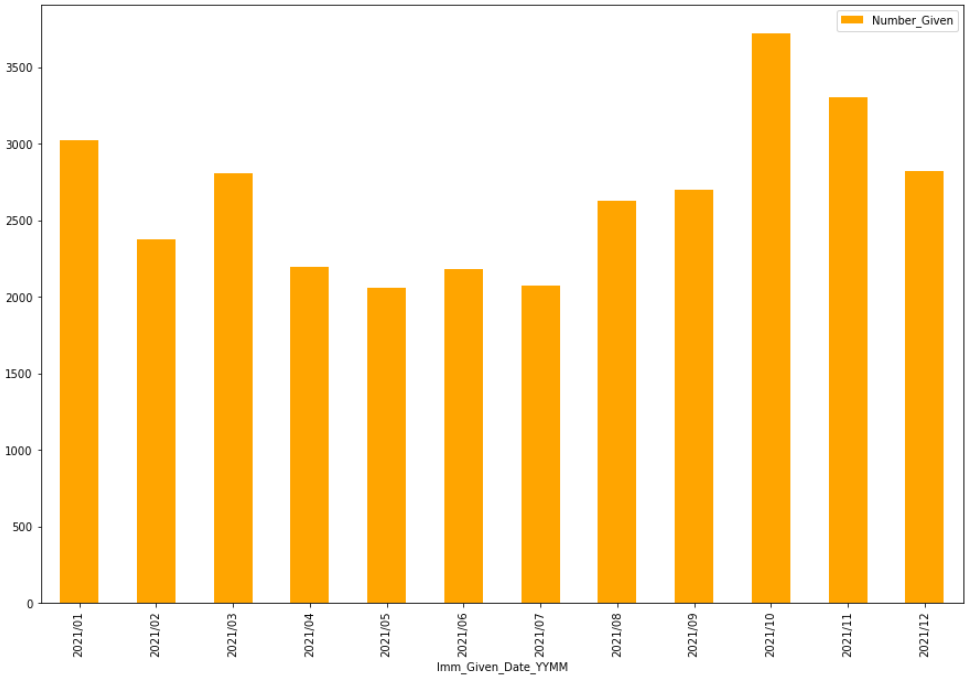


In [20]:

```
under19 = vax_data[vax_data['Age_at_Imm_GivenDate'] < 19]
month_occ_under19 = under19.groupby(['Imm_Given_Date_YYMM']).count()
month_occ_under19 = month_occ_under19.reset_index()
month_occ_under19 = month_occ_under19[['Imm_Given_Date_YYMM', 'ImmunizationId']]
month_occ_under19.columns = ['Imm_Given_Date_YYMM', 'Number_Given']
month_occ_under19plot = month_occ_under19.plot.bar(x='Imm_Given_Date_YYMM',
                                                   y='Number_Given', color='orange', figsize = (15,10))
month_occ_under19plot
```

Out[20]:

```
<AxesSubplot:xlabel='Imm_Given_Date_YYMM'>
```



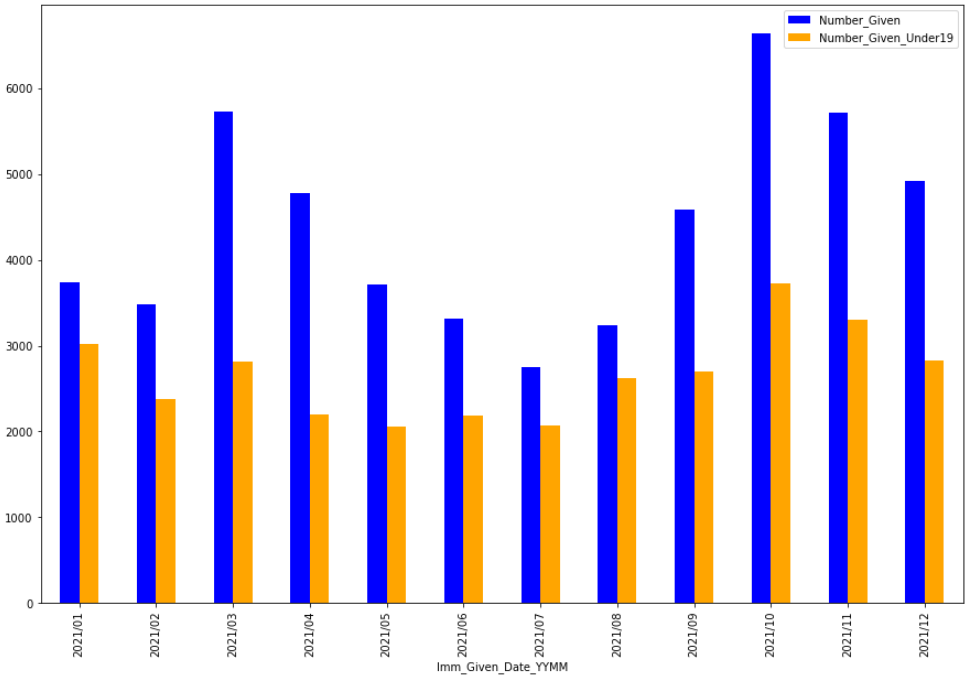
In [21]:

```
under19_given = month_occ_under19["Number_Given"]

month_occ.insert(2,"Number_Given_Under19", under19_given)
month_occ_plot_compare = month_occ.plot.bar(x='Imm_Given_Date_YYMM',
                                            color = {'Number_Given':'blue', 'Number_Given_Under19':'orange'},
                                            figsize = (15,10))
month_occ_plot_compare
```

Out[21]:

```
<AxesSubplot:xlabel='Imm_Given_Date_YYMM'>
```


Number_Given
Number_Given_Under19
6000
5000
4000
3000
2000
1000
0
2021/01
2021/02
2021/03
2021/04
2021/05
2021/06
2021/07
2021/08
2021/09
2021/10
2021/11
2021/12
Imm_Given_Date_YYMM

In [ ]: