Boeing Mesa Chemical Consumption

Sohrab Rajabi, Arunima Roy, Tamlyn Tamura

OMSBA 5510-01 Capstone

August 26, 2022

Table of Contents

I. Executive Summary

At the beginning of our internship, we were given two objectives as the Boeing Mesa Chemical Consumption Team. The first was to break down chemical consumption data by Boeing organizations and the second was to predict or forecast chemical consumption by organization for the next 5 years, which we initially intended to do through a classification algorithm. However, due to some factors outside of ours and Boeing's control, our priorities were re-evaluated mid-project, and our focus shifted to forecasting chemical consumption instead of breaking it down by department. We used two different time series models – the Autoregressive Integrated Moving Average (ARIMA) and the Seasonal Autoregressive Integrated Moving Average (SARIMA). We conclude that the SARIMA model is a better predictor of future chemical usage than the ARIMA model because it provides a model that best fits that data that we were given. The overall trend that we see is that the usage of chemicals will increase overtime, possibly to around 87,000 gallons of chemicals by the year 2025.

II. Introduction

The Boeing Mesa Chemical Consumption team oversees handling products across various divisions and business units, including Commercial, Defense, and Services. Boeing has recently dedicated a part of its Mesa location to product development and enhancement by implementing advanced composites that are lighter substitutes for the original materials. Composites are a combination of two or more materials that are layered on top of one another to create more optimal substitutes, such as making an object lighter, stronger and/or resistant to electricity. For Boeing's

purpose, this can make the aircraft lighter and more maneuverable and, thus, more fuel-efficient.

Our team, which is comprised of 1 project manager and 2 analysts, will integrate historical data collected by Boeing as it relates to chemical consumption and hazardous waste. After the data cleansing and preprocessing, the goal was then to apply machine learning classification algorithms to be able to break down chemical consumption by organization. However, due to a missing critical file that Boeing was unable to provide, it was impossible to connect our chemical consumption data with the organization data seamlessly. This meant that instead of a classification algorithm, our team decided to use a different form of predictive analytics, known as forecasting analysis.

We are committed to assisting Boeing in their ownership of corporate social responsibility, especially as it relates to their environmental impact. Even though our initial agenda was slightly altered, we collectively found a way to work with what we have and still produce the results Boeing had requested to a degree. By forecasting Boeing's chemical consumption over the next five years, it will now have the opportunity to streamline solutions for maintaining its compliance status will local, state and federal laws as it relates to air emissions.

III. Problem Statement

The problem tasked by Boeing is to utilize its past data surrounding chemical consumption and to break down chemical consumption at the organization and business unit level. Additionally, we will have to implement a forecasting time series analysis to predict future chemical consumptions and to determine whether Boeing will

surpass its compliance limit within the next five years. To address these problems, our team's first goal is to create a forecasting model represented by a visualization that breaks down and forecasts environmental consumable data by organization. If this first objective is met, then the secondary objective is to create a dashboard of this. Since Boeing does not currently have access to PowerBI, we have worked with Boeing and come to a consensus that the final dashboard will be created in Tableau.

IV. Literature Review

"Predictive Analytics vs Machine Learning: What's the Difference?"

This article provides some basic information on what predictive analytics is and how it is related to machine learning. At a high level, predictive analytics can be considered an older, more methodical and rigid approach to determining outcomes, while machine learning is a newer and more adaptable approach that can solve the same, if not more complex, types of problems. In the article, Johnson quotes, "To Predictive Analysts, machine learning is an extension of their practice, another tool in their toolbox, that helps them to do their job better. Using ML, predictive anaysts can:

- Provide answers, with confidence, to more complex problems
- Offer real-time answers to questions that persist through time
- Explore entirely new kinds of problems" (Johnson, 2020).

One limitation of predictive analytics, especially as it relates to forecasting, is that it is typically only used when trying to work with and predict numerical data. Meanwhile, machine learning can be used to predict qualitative variables in addition to quantitative ones.

As for machine learning, Johnson explains that machine learning is different from predictive analytics because “machine learning has less to do with reporting than it does to do with the modeling itself” (Johnson, 2020). The learning capabilities in machine learning allow many of the parameters and nodes to be optimized over time using the algorithm’s computations. However, this can be both a good and a bad thing, since statisticians will often reference this as a “black box” due to the lack of visibility or control over how those nodes and parameters are optimized. Circling back to how predictive analytics and machine learning are related, one can come to the conclusion that predictive analytics *as it was once known* (i.e. forecasting) is a more traditional version of predictive modeling, but machine learning does fall under the category of predictive analytics. Thus, it is simply a tool that people within the predictive analytics community can utilize as another method to determine certain outcomes.

“Time Series Forecasting with ARIMA, SARIMA, and SARIMAX”:

This article is focused on providing a comprehensive guide to understanding ARIMA, SARIMA and SARIMAX models. For the purpose of this project, we will be focusing on creating the ARIMA and SARIMA models. The acronym ARIMA stands for “Auto-Regressive Integrated Moving Average”. The ARIMA model stems AR, I and MA models.

The AR model takes the parameter *p* which will determine “the number of lagged series that we use” (Artley, 2022). When the *p* parameter is set to 0, then the model will return just the white noise, but when the parameter is set to 1, then it will return the previous time stamp adjusted by a multiplier and adding white noise. While increasing the parameter allows us to go further back in adjusting time stamps by their multipliers,

adding an additional parameter, known as the MA model, will be more beneficial. The MA (moving average) model takes the parameter *q, "*the number of lagged forecasting error terms in the prediction" (Artley, 2022). When the parameter *q* is set to 1, then MA(1) will be a constant term plus the previous white noise term times a multiplier, added with the current white noise term" (Artley, 2022). The I in ARIMA is known as the difference order and takes parameter *d,* with *d* equating to the number of transformations necessary to allow the data to be stationary. While ARIMA is a great forecasting model choice for beginning analysts, ARIMA assumes the data is stationary and therefore, does not account for seasonality or exogenous variables, so a different model needs to be used for that scenario.

SARIMA is an extension of ARIMA that accounts for seasonality, and SARIMAX stands for Seasonal Auto-Regressive Integrated Moving Average with Exogenous factors. By the looks of the name, it is apparent that SARIMAX addresses two of the pitfalls of an ARIMA model—seasonality and exogenous variables. The SARIMAX model conveniently enables one to difference data by seasonal frequency while also enabling the model to respond quickly to the effect of exogenous/external data as opposed to just relying on the influence of lagging terms. Below is the equation for the SARIMAX model:

$$d_t = c + \sum_{n=1}^{p} \alpha_n d_{t-n} + \sum_{n=1}^{q} \theta_n \epsilon_{t-n} + \sum_{n=1}^{r} \beta_n x_{n_t} + \sum_{n=1}^{P} \phi_n d_{t-sn} + \sum_{n=1}^{Q} \eta_n \epsilon_{t-sn} + \epsilon_t$$

(Artley, 2022)

"ARIMA Model – Complete Guide to Time Series Forecasting in Python":
This is an article that touches on SARIMA and SARIMAX, but the focus is on ARIMA models and model interpretation. It provides a different way of thinking about the ARIMA model interpretation: "An ARIMA model is characterized by 3 terms: *p, d, q,* where *p* is

the order of the AR term, *q* is the order of the MA term, [and] *d* is the number of differencing required to make the time series stationary" (Prabhakaran, 2021). It is also explained here that it is essential to make sure that the data is stationary because the term "Auto Regressive" in ARIMA means that it is a linear regression model, and it is widely known that a property of linear regression models is that they generally tend to perform best when the predictors are not correlated, which is why they are better known as independent variables. So, if the data is not yet stationary, then in order to apply an ARIMA model, the article recommends that the most common approach is to difference it, a.k.a. taking the difference of the current value and the previous value, taking into account that more complex models may require more than one differencing. Prabhakaran continues, "The value of *d,* therefore, is the minimum number of differencing needed to make the series stationary, [and] if the time series is already stationary, then *d* = 0 (Prabhakaran, 2021).

Next, the article explains a bit more about how to get the appropriate *p* and *q* variables. To fulfill the AR aspect of ARIMA, *p* can be understood as the amount of lags that will be used as predictors. Meanwhile, in order to fulfill the MA aspect of ARIMA, *q* can be understood as the "number of lagged forecast errors that should go into the ARIMA model" (Prabhakaran, 2021). After the three variable terms have been defined, then we can apply the ARIMA model on top of the data and get a summary of the results. It is important to compare the p-value as well as the terms in the AR and MA models. If the term is too small/insignificant and the P>|z| is too large aka larger than p-value = 0.05 then it is best to rerun the model without that specific term (example: if MA(2) = 0.000 and its P>|z| is 0.998 should be excluded when the model is run again).

“Complete Guide to SARIMAX in Python for Time Series Modeling”:

This article focuses on implementing, understanding and interpreting a SARIMAX time series model with Python. This article once again reinforces that whether you’re applying an ARIMA or a SARIMAX model, it is imperative to check the data for any seasonality. In addition to looking at the data and noticing patterns that indicate seasonality, another way to test whether the data is stationary or not is using the Augmented Dickey-Fuller test; this test is based on hypothesis testing, meaning that when the p-value is greater than 0.05, we can assume the data contains seasonality, and when the p-value is less than 0.05, we can assume the data is stationary.

When the data contains seasonality, there are two options to go about this: The first would be to remove the seasonality from the data to make it stationary and apply an ARIMA forecasting model, and the second option would be to leave the seasonality in the data and instead use a SARIMA or SARIMAX forecasting model. If one decides to continue use with the ARIMA model, then it is required to apply a rolling mean differencing to remove the seasonality. Graphing and visualizing the data should prove how much of the seasonality has been removed, and re-applying the ADfuller test and checking for the p-value should determine whether or not the data has become stationary. Below is the SARIMAX equation and the key for each variable:

$$\phi_p(L)\tilde{\phi}_P(L^s)\Delta^d\Delta_s^D y_t = A(t) + \theta_q(L)\tilde{\theta}_Q(L^s)\epsilon_t$$

Where:

- $\phi_p(L)$ is the non-seasonal autoregressive lag polynomial
- $\bar{\phi}_P(L^s)$ is the seasonal autoregressive lag polynomial
- $\Delta^d \Delta_s^D y_t$ is the time series, differenced d times, and seasonally differenced D times.
- $A(t)$ is the trend polynomial (including the intercept)
- $\theta_q(L)$ is the non-seasonal moving average lag polynomial
- $\bar{\theta}_Q(L^s)$ is the seasonal moving average lag polynomial

(Verma, 2021)

"Create Your First Dashboard in Tableau":

This article walks through a very basic tutorial on creating a dashboard within Tableau. Dashboards are especially helpful when it comes to presenting to executives or other business-related stakeholders. Not only are dashboards a great way to convey data analysis findings in a more visual way, but often stakeholders such as C-level executives often do not have the time to be sifting through multiple worksheets or tabs of a report to find the significant takeaways. Thus, having a single-page dashboard is the most optimal way to concisely explain and highlight key takeaways from the data.

The best way to go about creating a Tableau dashboard is to first create all the graphs in Tableau worksheets, then select the Dashboard icon and choose the worksheets that should be included from there. In addition to covering the basics of building a dashboard, this article does a great job at introducing some of the visual nuances to be aware of, such as strategically minimizing white space.

V. Description of Data

Parquet files:

A parquet file is an open-source, column-oriented data storage file type created by Apache and is part of the Apache Hadoop family (Levy, 2022). Parquet files are useful for holding large amounts of data. This is useful because tools like Excel are not strong enough to handle large data sets, and thus, typically cannot analyze more than ~10,000 rows of data without the application becoming faulty.

2016_2022dw_All_Transactions_Gallons_Total.parquet:

	Unnamed: 0	ID	FROM_LOCATION	TO_LOCATION	Part	TRANSACTION	TRANSACTION_DATE	Kit_Quantity	Unit	KITS	UPDATED_BY
0	381.0	898678.0	539-F4	531-71	RM016244	540619.0	2022-04-28	150.0	FT	NaN	Michael Johnson
1	905.0	898679.0	539-F4	531-71	RM016244	540285.0	2022-04-25	125.0	FT	NaN	Michael Johnson
2	906.0	898680.0	539-F4	531-71	RM016244	538746.0	2022-04-04	115.0	FT	NaN	Michael Johnson
3	907.0	898681.0	539-F4	531-71	RM016244	536590.0	2022-03-03	150.0	FT	NaN	Michael Johnson
4	908.0	898682.0	539-F4	531-71	RM016244	535575.0	2022-02-17	108.0	FT	NaN	Michael Johnson
...	...	...	...	...	...	...	...	...	...	...	...
4	141164.0	1039842.0	539-F4	531-65	NP1578	400795.0	2016-07-05	5.0	EA	5.0	B299575
5	141165.0	1039843.0	539-F4	520-192	NP1578	400023.0	2016-06-24	2.0	EA	2.0	B1497693
6	141166.0	1039844.0	539-F4	560-TC500	NP1578	400007.0	2016-06-23	3.0	EA	3.0	B1497693
7	141167.0	1039845.0	539-F4	560-TC500	NP1578	399989.0	2016-06-22	2.0	EA	2.0	B1497693
8	141168.0	1039846.0	539-F4	560-TC500	NP1578	395852.0	2016-04-22	5.0	EA	5.0	B2164270

213473 rows × 24 columns

TRADE_NAME	KIT_PART	SDS_Quantity	SDS_Number	SPEC_MATERIAL_TYPE	SPEC_MATERIAL_CLASS	DOCUMENT_SPEC	Year	Conversion	DivideBySG	Specific_gravity	Usage_gallons	Part_specific
HEXPLY M73 PREPREG	N	150.0	129805.0	1	None	MMS5025	2022.0	0.097962	1.0	1.4	10.495889	NaN
HEXPLY M73 PREPREG	N	125.0	129805.0	1	None	MMS5025	2022.0	0.097962	1.0	1.4	8.746574	NaN
HEXPLY M73 PREPREG	N	115.0	129805.0	1	None	MMS5025	2022.0	0.097962	1.0	1.4	8.046848	NaN
HEXPLY M73 PREPREG	N	150.0	129805.0	1	None	MMS5025	2022.0	0.097962	1.0	1.4	10.495889	NaN
HEXPLY M73 PREPREG	N	108.0	129805.0	1	None	MMS5025	2022.0	0.097962	1.0	1.4	7.557040	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...
DUSTING GAS/FREEZE SPRAY	N	5.0	83377.0	None	None	None	2016.0	0.980000	NaN	1.0	4.900000	1.0
DUSTING GAS/FREEZE SPRAY	N	2.0	83377.0	None	None	None	2016.0	0.980000	NaN	1.0	1.960000	1.0
DUSTING GAS/FREEZE SPRAY	N	3.0	83377.0	None	None	None	2016.0	0.980000	NaN	1.0	2.940000	1.0
DUSTING GAS/FREEZE SPRAY	N	2.0	83377.0	None	None	None	2016.0	0.980000	NaN	1.0	1.960000	1.0
DUSTING GAS/FREEZE SPRAY	N	5.0	83377.0	None	None	None	2016.0	0.980000	NaN	1.0	4.900000	1.0

This data file provides documentation on different chemical transactions, such as the transaction date, chemical usage in gallons, respective location for chemical usage, and SDS number, which stands for Safety Data Sheet and is a unique identifier and “includes information such as the properties of each chemical; the physical, health, and environmental health hazards; protective measures; and safety precautions for handling, storing, and transporting the chemical” (“Safety Data Sheets,” n.d.). This is useful because we are able to see how much of what chemical was purchased and consumed, and what the properties of those chemicals are, which is essential for forecasting what compliance laws Boeing may or may not be at risk of violating within the next 5 years.

2016_2022dw_CAS_Totals.parquet:

	Part	TRANSACTION_DATE	TRADE_NAME	SDS_Number	TO_LOCATION	Specific_gravity	Usage_gallons	CAS_Number	Norm_Conc	Max_Conc	
0	RM016244	2022-04-28	HEXPLY M73 PREPREG	129805.0	531-71	1.4	10.495889	106-89-8	0.0	0.01	
1	RM016244	2022-04-28	HEXPLY M73 PREPREG	129805.0	531-71	1.4	10.495889	123-31-9	0.0	0.01	
2	RM016244	2022-04-28	HEXPLY M73 PREPREG	129805.0	531-71	1.4	10.495889	5026-74-4	5.0	10.00	Oxiran
3	RM016244	2022-04-28	HEXPLY M73 PREPREG	129805.0	531-71	1.4	10.495889	67-64-1	1.0	3.00	
4	RM016244	2022-04-28	HEXPLY M73 PREPREG	129805.0	531-71	1.4	10.495889	7440-44-0	0.0	100.00	
...	...	...	...	...	...	...	...	...	...	...	
4	NP1578	2016-07-05	DUSTING GAS/FREEZE SPRAY	83377.0	531-65	1.0	4.900000	None	NaN	NaN	
5	NP1578	2016-06-24	DUSTING GAS/FREEZE SPRAY	83377.0	520-192	1.0	1.960000	None	NaN	NaN	
6	NP1578	2016-06-23	DUSTING GAS/FREEZE SPRAY	83377.0	560-TC500	1.0	2.940000	None	NaN	NaN	
7	NP1578	2016-06-22	DUSTING GAS/FREEZE SPRAY	83377.0	560-TC500	1.0	1.960000	None	NaN	NaN	
8	NP1578	2016-04-22	DUSTING GAS/FREEZE SPRAY	83377.0	560-TC500	1.0	4.900000	None	NaN	NaN	

1173758 rows × 32 columns

	Chem_name	SPEC_MATERIAL_TYPE	SPEC_MATERIAL_CLASS	DOCUMENT_SPEC	CAS_content_lb_gal	CAS_lbs	CAS_tons	Month	Year	Unnamed: 0
	Epichlorohydrin	1	None	MMS5025	0.00000	0.0000	0.000000	4.0	2022.0	NaN
	Hydroquinone	1	None	MMS5025	0.00000	0.0000	0.000000	4.0	2022.0	NaN
	Oxiranemethanamine,...	1	None	MMS5025	0.58380	6.1275	0.003064	4.0	2022.0	NaN
	Acetone	1	None	MMS5025	0.11676	1.2255	0.000613	4.0	2022.0	NaN
	Carbon	1	None	MMS5025	0.00000	0.0000	0.000000	4.0	2022.0	NaN
	...	...	...	...	...	...	...	...	...	...
	None	None	None	None	NaN	NaN	NaN	NaN	2016.0	141164.0
	None	None	None	None	NaN	NaN	NaN	NaN	2016.0	141165.0
	None	None	None	None	NaN	NaN	NaN	NaN	2016.0	141166.0
	None	None	None	None	NaN	NaN	NaN	NaN	2016.0	141167.0
	None	None	None	None	NaN	NaN	NaN	NaN	2016.0	141168.0

ID	FROM_LOCATION	TRANSACTION	Kit_Quantity	Unit	KITS	UPDATED_BY	KIT_PART	SDS_Quantity	Conversion	DivideBySG	Part_specific
NaN	None	NaN	NaN	None	NaN	None	None	NaN	NaN	NaN	NaN
NaN	None	NaN	NaN	None	NaN	None	None	NaN	NaN	NaN	NaN
NaN	None	NaN	NaN	None	NaN	None	None	NaN	NaN	NaN	NaN
NaN	None	NaN	NaN	None	NaN	None	None	NaN	NaN	NaN	NaN
NaN	None	NaN	NaN	None	NaN	None	None	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...
1039842.0	539-F4	400795.0	5.0	EA	5.0	B299575	N	5.0	0.98	NaN	1.0
1039843.0	539-F4	400023.0	2.0	EA	2.0	B1497693	N	2.0	0.98	NaN	1.0
1039844.0	539-F4	400007.0	3.0	EA	3.0	B1497693	N	3.0	0.98	NaN	1.0
1039845.0	539-F4	399989.0	2.0	EA	2.0	B1497693	N	2.0	0.98	NaN	1.0
1039846.0	539-F4	395852.0	5.0	EA	5.0	B2164270	N	5.0	0.98	NaN	1.0

This data file provides information such as the CAS number, which stands for Chemical Abstracts Services and can be described as a “unique numeric identifier that can contain up to 10 digits, divided by hyphens into three parts” (“CAS REGISTER And CAS Registry Numbers FAQ”, n.d.). Each CAS number ties back to a specific type of chemical, like caffeine, for example.

Excel/CSV files:

Manager_Mesa_Site_exportedResults_29-6-2022_11_44_09.csv:

BEMSID	Name	Phone	Location	Company	Business Unit	Assigned HR Department	Accounting Department	Mail Code	Org Structure Code	isManager	Email	Mob Pho
2260450	Riley, Lauren V	+1 (480) 8910786	Mesa, AZ	The Boeing Company	BDS	6053003	HM-01-D71A	M510-A101	B9FRA	Y	lauren.v.riley@boeing.com	+1 (48 34340
2247549	Ahmed, Zadeed N	+1 (480) 8911264	Mesa, AZ	The Boeing Company	CORP	1230060	MX-ME-3510	M530-B340	BE9KEC	Y	zadeed.n.ahmed@boeing.com	+1 (48 20897
781384	Johns, Christopher	NaN	Mesa, AZ	The Boeing Company	BDS	3004845	MX-ME-4Q21	NaN	B9GADF	Y	christopher.johns2@boeing.com	Na
2305258	Haver, Christopher S	+1 (480) 8911089	Mesa, AZ	The Boeing Company	CORP	1230060	MX-ME-3S50	M530-B340	BE9KEC	Y	christopher.s.haver@boeing.com	+1 (48 21706
...	...	...	...	...	...	...	...	...	...	...	...	
336479	Mason, Robert C	+1 (480) 8912673	Mesa, AZ	The Boeing Company	BDS	3002206	MX-ME-46A0	M543-D152	B9GBKDBA	Y	robert.c.mason@boeing.com	+1 (60 56191
341965	Vander Molen, Jack D	+1 (562) 6734573	Mesa, AZ	The Boeing Company	BGS	1965912	76-PS-5KA8	841-SA01	B3IBD	Y	jack.d.vandermolen@boeing.com	+1 (56 67345
3479534	Phillips, Lauri	+1 (480) 2866764	Mesa, AZ	The Boeing Company	BDS	1968014	GG-GG-SDC2	NaN	B9AAB	Y	lauri.phillips@boeing.com	+1 (48 28667
3490963	Myers, Lamar M	NaN	Mesa, AZ	The Boeing Company	BDS	3001741	MX-CR-MD01	NaN	B9GDLG	Y	lamar.m.myers@boeing.com	Na
2140018	Robles, Manuel A	+1 (480) 4449503	Mesa, AZ	The Boeing Company	CORP	6030918	MX-ME-3F21	M530-B336	BE9KEA	Y	manuel.a.robles2@boeing.com	Na

387 rows × 1 columns

This data file contains basic information, such as contact information for managers at Boeing. A caveat of the data, as emphasized by our Boeing contact, Nicholas Polich, is that managers can change so often at Boeing—whether it is moving to a new department, turnover, or otherwise, using manager information is not the best way to break down the company by organization. Rather, the valuable piece of information in this data table is the Accounting Department code for each hiring manager entry. While managers may come and go, Accounting Departments tend to stay relatively the same over time.

This data set was going to be particularly useful for breaking down chemical consumption by organization (using the Accounting Department code), however, there was a missing field between accounting department code and the chemicals used by a

certain organization. By the time this concern was brought up with Boeing, an additional Excel file to join this file with actual chemical consumption data was too late and was not utilized in our final forecasting model.

MasterOrgFile_OrgLevel.xlsx:

	SetID	DeptID	OrgLvlCD	FunCd	MgrID	HRRepID	DirectHC	IndirectHC	OrgCD	DeptDesc	effdte	Nodenum	NodeEnd	TreeLvl	parent
0	BCAGX	1880000	0	1	2889897	354380	2	65927	40	Boeing Commercial Airplanes	2019-05-06	0	0	1	1233950
1	BCAGX	1231123	43	1	96093	598304	2	6752	43	737 Program	2018-09-21	0	0	2	1880000
2	BCAGX	1234310	43B	10	110607	598304	5	5693	43A	737 Manufacturing	2019-01-22	0	0	3	1231123
3	BCAGX	1234313	43BB	10	212674	215546	5	1753	43AB	737 Wings	2018-04-27	0	0	4	1234310
4	BCAGX	6040163	43BBSB	10	459996	215546	4	221	43ABWM	737 Wings Sys Inst 2&3rd shift	2019-02-15	0	0	6	6044146
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
6403	BCAGX	6031950	ZZPDCB	10	88759	3060789	11	11	4J7CBB	PSD 747/767/777 Struts	2019-05-03	0	0	6	6030272
6404	BCAGX	6045018	ZZPDDD	10	406159	3060789	10	10	4J7CB	PSD IE	2019-05-03	0	0	6	1232097
6405	BCAGX	1232094	ZZPDF	10	1790109	3060789	4	68	4J7C	PSD RVST	2019-05-03	0	0	5	1232090
6406	BCAGX	6043355	ZZPF	12	267304	2760653	3	11	4J1G	SC Strategy Propulsion	2019-04-26	0	0	4	1840000
6407	BCAGX	6030324	ZZPGD	3	1170534	2760653	4	4	4J7	BCA Contracts	2019-02-19	0	0	5	6033300

6408 rows × 35 columns

sugosc	cmnt_cd	DIC	EnrDirectHC	EnrIndirectHC	MgrEnr	MgrAsnd	health	tips	HRREP	MANAGER	mgrwkloc	mgrflsa	Function	Comnt	mgracntg	Dept_Desc	LOF	lofid	LOFMAIL
NaN	NaN	0	1	65226	True	True	0.406426	3473	Van der leer Monica C	MCALLISTER KEVIN G	WA016	E	Program Mgmt	NaN	66-C6-1001	NaN	Z_UNKNOWN	1	NaN
NaN	NaN	0	0	5931	False	True	0.450766	251	Dunning Becky A	LINDBLAD WARREN ERIC	WA016	E	Program Mgmt	NaN	66-CE-1006	NaN	Z_UNKNOWN	1	NaN
NaN	NaN	0	0	5134	False	True	0.509915	173	Dunning Becky A	MCCULLEY GARY KEVIN	WA016	E	Operations	NaN	66-CR-4000	NaN	Z_UNKNOWN	1	NaN
NaN	NaN	0	1	1656	False	True	0.846939	15	Holling-sato Heidi J	DELANEY MICHAEL RICHARD	WA016	E	Operations	NaN	66-CR-W202	NaN	Z_UNKNOWN	1	NaN
NaN	NaN	0	2	215	False	True	1.000000	0	Holling-sato Heidi J	BAKKI LANCE R	WA016	M	Operations	NaN	66-CR-W550	NaN	Z_UNKNOWN	1	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
NaN	7.0	0	10	10	False	True	0.000000	1	Duckworth Salyse M	SALO DANNY DEAN	WA004	M	Operations	OSC tip	66-CP-5E00	NaN	Z_UNKNOWN	1	NaN
NaN	NaN	0	0	0	False	True	1.000000	0	Duckworth Salyse M	SMITH PATRICK LAWRENCE	WA004	M	Operations	NaN	66-CP-5E01	NaN	Z_UNKNOWN	1	NaN
NaN	NaN	0	2	63	False	True	0.250000	3	Duckworth Salyse M	SCHLIPF ANDY C	WA016	M	Operations	NaN	66-CP-5R01	NaN	Z_UNKNOWN	1	NaN
NaN	7.0	0	0	1	False	True	0.000000	2	Foraker Megan N	FOLKERT WENDI MARIE	WA004	E	SM&P	OSC tip	66-CP-1105	NaN	Z_UNKNOWN	1	NaN
NaN	NaN	0	0	0	False	True	1.000000	0	Foraker Megan N	YAMASHITA MARLENE	WA004	E	Contracts	NaN	66-CP-1108	NaN	Z_UNKNOWN	1	NaN

This data file contains some basic information on how the Boeing organization is structured. While the data is very relative and not all of it is useful, the column “mgractng” is important for joining with the Accounting Department from the above data file. This is also a data file that did not end up getting utilized as a catalyst of missing a critical file that was not provided by Boeing.

be_SDS_Chemicals.xlsx:

	Unnamed: 0	Min_Conc	Max_Conc	Norm_Conc	US EPA SNUR?	CAS_Number	Chem_name	URL	SDS_Number	RevisionDate	IUPAC_name
0	40133	0.2	0.2	0.20	NaN	7696-12-0	(1,3-dioxo-4,5,6,7-tetrahydroisoindol-2-yl)met...	NaN	53848	1900-01-01	(1,3-dioxo-4,5,6,7-tetrahydroisoindol-2-yl)met...
1	44810	0.1	1.0	0.55	NaN	7696-12-0	(1,3-dioxo-4,5,6,7-tetrahydroisoindol-2-yl)met...	NaN	97031	1900-01-01	(1,3-dioxo-4,5,6,7-tetrahydroisoindol-2-yl)met...
2	45182	0.1	1.0	0.55	NaN	7696-12-0	(1,3-dioxo-4,5,6,7-tetrahydroisoindol-2-yl)met...	NaN	99068	1900-01-01	(1,3-dioxo-4,5,6,7-tetrahydroisoindol-2-yl)met...
3	45494	0.1	1.0	0.55	NaN	7696-12-0	(1,3-dioxo-4,5,6,7-tetrahydroisoindol-2-yl)met...	NaN	99717	1900-01-01	(1,3-dioxo-4,5,6,7-tetrahydroisoindol-2-yl)met...
4	50468	0.1	1.0	0.55	NaN	7696-12-0	(1,3-dioxo-4,5,6,7-tetrahydroisoindol-2-yl)met...	NaN	105521	1900-01-01	(1,3-dioxo-4,5,6,7-tetrahydroisoindol-2-yl)met...
...	...	...	...	...	...	...	...	...	...	...	...
157915	70516	0.0	5.0	2.50	NaN	12136-78-6	NaN	NaN	302037	1900-01-01	NaN
157916	71583	0.0	5.0	2.50	NaN	12597-71-6	NaN	NaN	158221	1900-01-01	NaN
157917	76441	100.0	100.0	100.00	NaN	1330-44-5	NaN	NaN	65657	1900-01-01	NaN
157918	78418	0.0	0.1	0.05	NaN	1303-28-2	NaN	NaN	154341	1900-01-01	NaN
157919	80804	0.0	100.0	50.00	NaN	38959-35-2	NaN	NaN	153169	1900-01-01	NaN

157920 rows × 11 columns

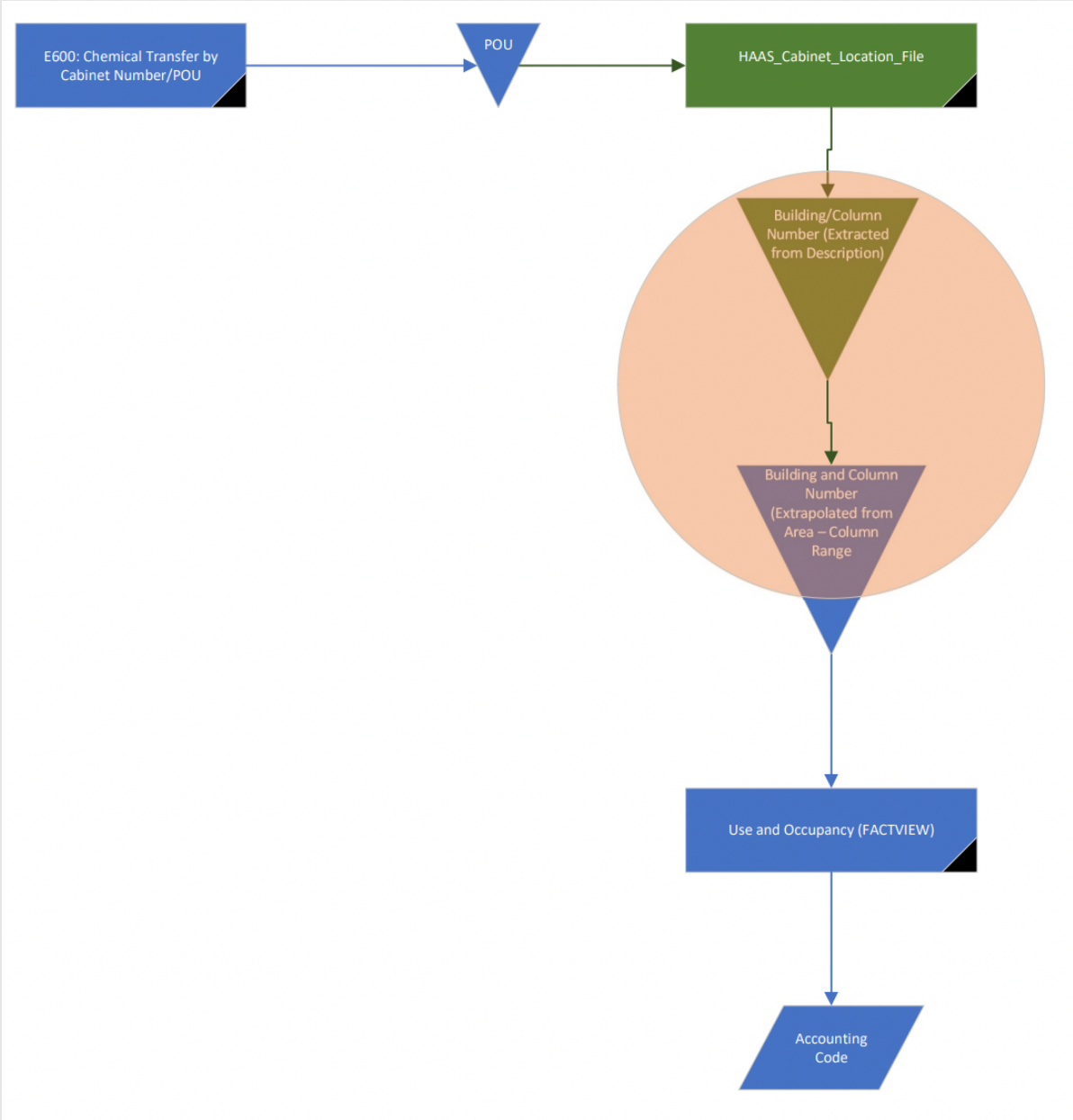
This data file contains general information on the chemicals purchased and consumed at Boeing such as the SDS number, CAS number, chemical name, and the minimum, maximum and normal concentration of each type of chemical. This is important for both analysis and forecasting purposes because, as the data implies, this is going to be where we get information on what chemicals are being used and their respective concentration levels.

2022.7.2_Mesa_POU_Locations.xlsx

	Storage Area	Storage Area Name	Area	Building	Floor	Room	Location Column	Location Section	Location Detail
0	1052598	23206 STRAP - HAZ-MAT FLAM POU 543-206 COL H13	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	1053942	22213 - HAZ-MAT FLAM POU 520-213 2ND FLOOR COL...	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	1066983	26031 - HAZ-MAT FLAM POU 560-31 TOOL CRIB 500	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	1077081	23038 - HAZ-MAT FLAM POU 543-38 COL-M7	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	1102413	22206 - HAZ-MAT FLAM POU 2ND FLOOR 520-206 COL...	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...
193	5923309	21071 - HAZ-MAT FREEZER 531-71 CLEAN ROOM	NaN	NaN	NaN	NaN	NaN	NaN	NaN
194	5980594	23152 - HAZ-MAT COR POU 543-152 COL-L9	NaN	NaN	NaN	NaN	NaN	NaN	NaN
195	6291845	26201 - HAZ-MAT FLAM POU 562-1 FLIGHT LINE FUE...	NaN	NaN	NaN	NaN	NaN	NaN	NaN
196	6317625	22214 - HAZ-MAT FLAM REFRIG POU 520-214 COL F1...	NaN	NaN	NaN	NaN	NaN	NaN	NaN
197	6346061	23051 - HAZ-MAT COR POU 543-51 COL J2	NaN	NaN	NaN	NaN	NaN	NaN	NaN

198 rows × 9 columns

While this dataset was not used in the final analysis, this was supposed to be one of the critical elements to link chemical consumption to organization data. If we were given the last file to create the link, this particular dataset would have required data extraction in order to separate the Building and Column numbers, which are located inside “Storage Area Name”. The following diagram was provided by Boeing representative Nicholas Polich as our map to link the two categories of data we were provided:



By looking at the diagram, it is shown that the one file we were not able to obtain was HAAS_Cabinet_Location_File.

2021.5 Mesa Square Footage (Use and Occupancy Code).xls

	Region	Site	Property	Property Name	Bldg #	Bldg Name	Bldg Address 1	Bldg Address 2	City	State	Zip Code	County	Occupancy Pool	Occupancy Pool Name	Floor
3	SOUTHWEST	ARIZONA MESA	MESA	Mesa Helicopters	50-502	GUARD GATE 2	5000 E McDowell Road	NaN	Mesa	Arizona	85215-9707	Maricopa	MO	MESA OCCUPANCY	01
4	SOUTHWEST	ARIZONA MESA	MESA	Mesa Helicopters	50-503	GUARD GATE GATE 3	5000 E McDowell Road	NaN	Mesa	Arizona	85215-9707	Maricopa	MO	MESA OCCUPANCY	01
5	SOUTHWEST	ARIZONA MESA	MESA	Mesa Helicopters	50-504	GUARD GATE GATE 4	5000 E McDowell Road	NaN	Mesa	Arizona	85215-9707	Maricopa	MO	MESA OCCUPANCY	01
6	SOUTHWEST	ARIZONA MESA	MESA	Mesa Helicopters	50-505	GUARD GATE GATE 20	5000 E McDowell Road	NaN	Mesa	Arizona	85215-9707	Maricopa	MO	MESA OCCUPANCY	01
7	SOUTHWEST	ARIZONA MESA	MESA	Mesa Helicopters	50-510	Administration Building	5000 E McDowell Road	NaN	Mesa	Arizona	85215-9707	Maricopa	MO	MESA OCCUPANCY	01
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
6915	Subloc/Room (a) = Sublocation / Room	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
6916	Class (b) = Class Code	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
6917	Occ/Vac (c) = Occupied / Vacant	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
6918	SecCirc SqFt (d) = Secondary Circulation Squar...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
6919	All numbers displayed are the sum of Sublocati...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

6917 rows × 56 columns

Floor Name	Floor Area	Floor Area Name	Floor Area Type	Long Term Vacancy	Subloc/Room (a)	Subloc/Room Name	Space Type	Space Type Name	Class (b)	Class Name	Class Type	Occ/Vac (c)	Acctg Org	Acctg Org Name	
FIRST FLOOR	AREA01.1	NaN	UTILITY_MISC	NaN	11A1	Guard Gate 2	DS	Assignable	06	Support	SiteSupport	O	GG-GG-SWMG	Mesa General Security	
FIRST FLOOR	AREA1.1	NaN	UTILITY_MISC	NaN	11A1	Guard Gate 3	DS	Assignable	06	Support	SiteSupport	V	GG-GG-SWMG	Mesa General Security	
FIRST FLOOR	AREA1.1	NaN	UTILITY_MISC	NaN	11A1	Guard Gate 4	DS	Assignable	06	Support	SiteSupport	V	GG-GG-SWMG	Mesa General Security	
FIRST FLOOR	AREA01.1	NaN	UTILITY_MISC	NaN	11A1	Guard Gate 20	DS	Assignable	06	Support	SiteSupport	V	GG-GG-SWMG	Mesa General Security	
FIRST FLOOR	AREA1.1	Area 1 - 1st Floor	BUSINESS	NaN	1001	NaN	BC	Building Common	07	Non-Assignable	PriCirc	O	NaN	NaN	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

Subloc / Room SqFt	Sec Circ SqFt	Total SqFt	Subloc/Room Attributes	BEMSID-Name	BEMSID	Name	OvHd Alloc	OvHd Alloc Name	Pool Accounting	Division Code	Division Name	Subdivision Code	Subdivision Name	Resource Subcategory
86.25	0	86.25	NaN	NaN	NaN	NaN	S2	SW S&F Protection	NaN	S**	NaN	SM*	NaN	95
43.46	0	43.46	NaN	NaN	NaN	NaN	S2	SW S&F Protection	NaN	S**	NaN	SM*	NaN	95
26.44	0	26.44	NaN	NaN	NaN	NaN	S2	SW S&F Protection	NaN	S**	NaN	SM*	NaN	95
31.11	0	31.11	NaN	NaN	NaN	NaN	S2	SW S&F Protection	NaN	S**	NaN	SM*	NaN	95
1,033.02	0	1,033.02	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Production Flag	Labor Code	Pool-OBA	Finance Site Code	Finance Sub Site Code	Work Order	Loan Pool	SubPool	Business Unit	Location Code	Department
N	XX	NaN	NaN	NaN	NaN	NaN	NaN	GG	GG	SWMG
N	XX	NaN	NaN	NaN	NaN	NaN	NaN	GG	GG	SWMG
N	XX	NaN	NaN	NaN	NaN	NaN	NaN	GG	GG	SWMG
N	XX	NaN	NaN	NaN	NaN	NaN	NaN	GG	GG	SWMG
NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...
NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

This dataset was also not used in the final analysis, but we felt it was important to include because, given the “Use and Occupancy” in the above diagram provided by Boeing, this was a table we could use to cross-reference and, therefore, validate what part of the “Storage Area Name” from the Mesa_POU_Locations.xlsx was the Building Number (displayed as Bldg #) and potentially the Column Number, in case it was labeled as something different here.

VI. Data Preprocessing

As mentioned in the introduction, our original objective for the project was to breakdown chemical consumption by organization. The data files given to us by the Boeing representatives included the parquet files containing information related to

chemical consumption and the excel files containing information related to organizations.

In our initial analysis of the two parquet files (we will shorten the file names and call them CAS and GAL), we found that most of the variables were identical. With the parquet file that included extra variables, we confirmed with the Boeing Representatives if they were important to include - they were not. Therefore, we identified 3 different scenarios that we could use. We verified by Boeing on which single scenario to focus on to ensure we have enough data points to forecast.

Scenario 1: Our first step in cleaning the data was to identify the duplicates and remove them from the parquet files, CAS and GAL. Then we picked out the common columns in both – this ended up being 24 columns. The CAS file has 1,024,753 observations and the GAL file had 193,907 observations, and this is after removing duplicates. After this, we appended the two files together, dropped duplicates again to ensure there was no overlap among the data, coming to a total of 1,218,660 observations. We further trimmed the total number of variables to select that we deemed as important. Our last check for the dataset was to look at the nulls. We were seeing that for some of the variables, close to 990,000 of the observations had null values. Since null values are not productive in our analysis, we dropped the nulls and ended with a total of 1,177,217 observations and 9 variables. The variables were the following:

1. TO_LOCATION
2. Part
3. TRANSACTION_DATE

4. TRADE_NAME
5. SDS_Number
6. DOCUMENT_SPEC
7. Year
8. Specific_gravity
9. Usage_gallons

Scenario 2: Like scenario 1, we trimmed the CAS parquet file to match the number and header of columns as the GAS parquet file, coming to 24 columns. The GAL file was left alone. Then we appended the two files, dropped the duplicates and the total before evaluating the nulls came out to be 1,387,231 observations. We looked at the variables that had the least number of null values. The list is the same as the one above in Scenario 1. After removing the null values, our data set had 1,139,776 observations and 9 variables.

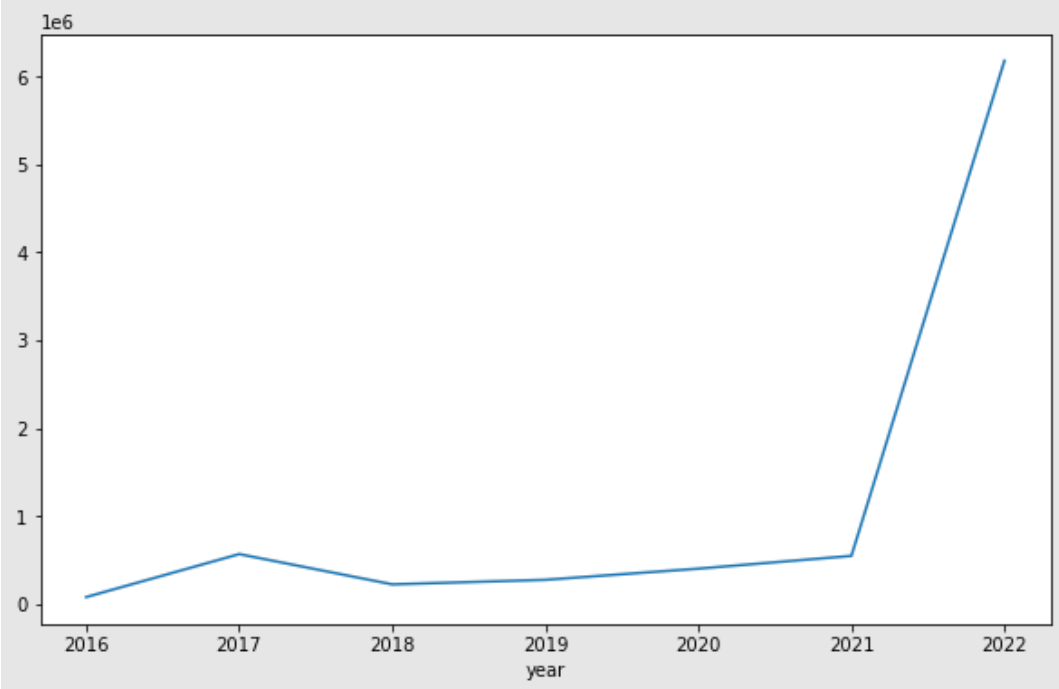
Scenario 3: Since the data between the two files are very similar, we suggested to the Boeing team that we could simply use the larger file that was provided. We dropped the nulls and kept the same 9 variables, listed above since those had the most information and the least number of null values. The total number of observations if we went with this scenario gave us 1,173,758 observations.

Among the 3 scenarios, Scenario 1 was the best choice moving forward. This file integrated both parquet files, giving us a good number of observations to do our analysis. We also felt that the steps we took to process the data gave us a more accurate final dataset.

After obtaining the final dataset from scenario 1, we did a deep dive into the data to better understand what information we were working with and underwent an exploratory data analysis to get some initial findings. Our final data frame to work with had a total of 9 variables and 1,177,217 observations. Since we would be using a time series model eventually, we created 3 new variables - year, month, and date from the "TRANSACTION_DATE" variable.

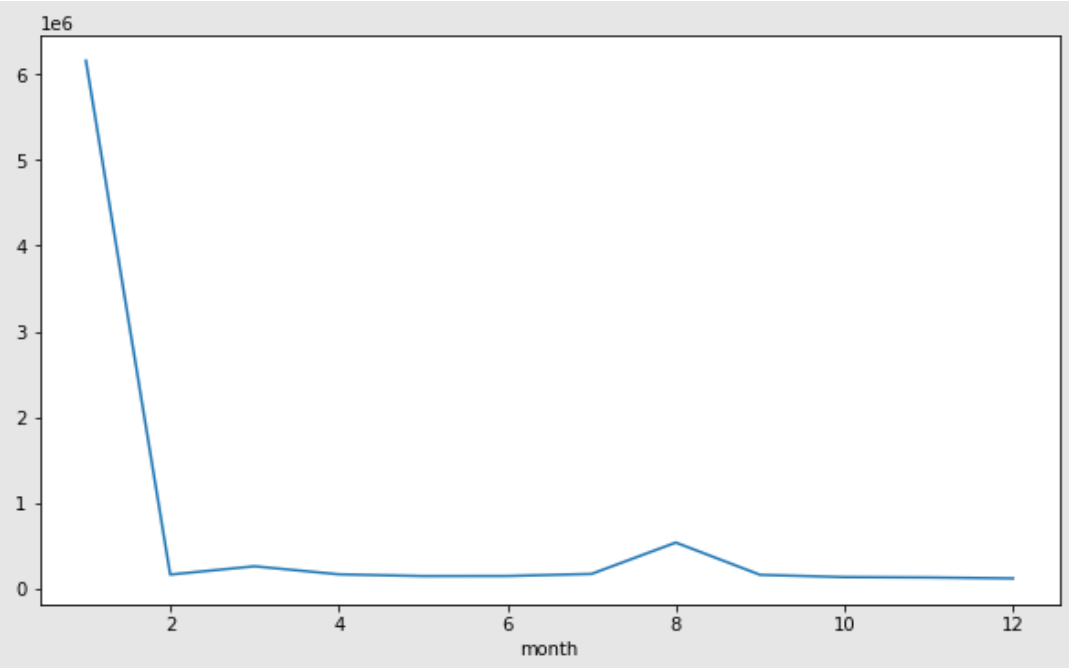
One interesting finding was the relationship between "year" and "Usage_gallons". The first insight that we get from this is that our data ranges between the years 2016 – 2022. This gives us an idea that the range of years of data that we have is from 2016 to 2022. The other thing to point out here is that there was a significant increase in usage of chemicals (in gallons) in 2021. In doing some current event research, this lines up with the news that mentions that Boeing had "deliveries of new places surge" (Josephs, 2021). Second to the increased surge, Boeing is working on their sustainability efforts when they select materials for their products by identifying "opportunities for eliminating, capturing and recycling solvents and chemicals while developing alternative materials, primers and coatings". (Maull, McElroy, Wingbermuehle, 2021).

Usage Gallons per Year:



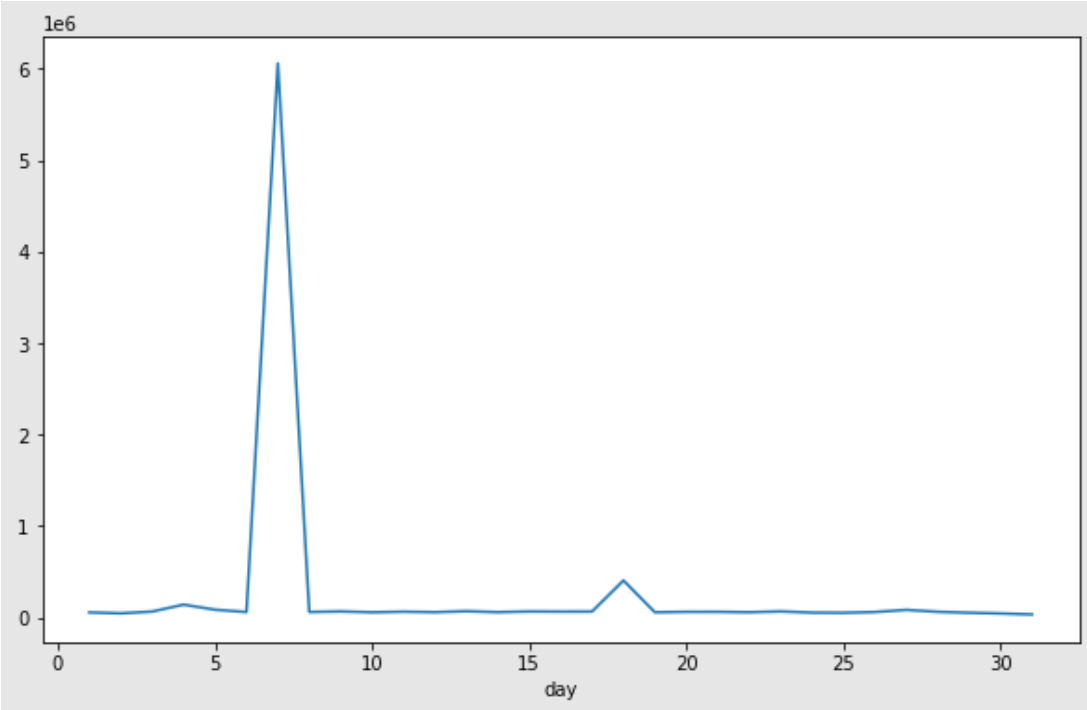
When looking at the usage gallons grouped by month visualization below, the fact to note about this relationship is that the usage in gallons most commonly increases in the beginning of the year in January. Then it stabilizes and stays constant until August, when it spikes up again, not nearly to the extent as it is in the beginning of the year but still an increase. It then flattens out until the end of the calendar year.

Usage Gallons Grouped by Month:



Very similar to the Usage Gallons Grouped by Month, the visualization for Usage Gallons Grouped by Day indicates what day of the month is more likely to have a transaction for when a chemical was used. By this, we can see that there was a spike between days 6-8 and then again between days 16-19, approximately.

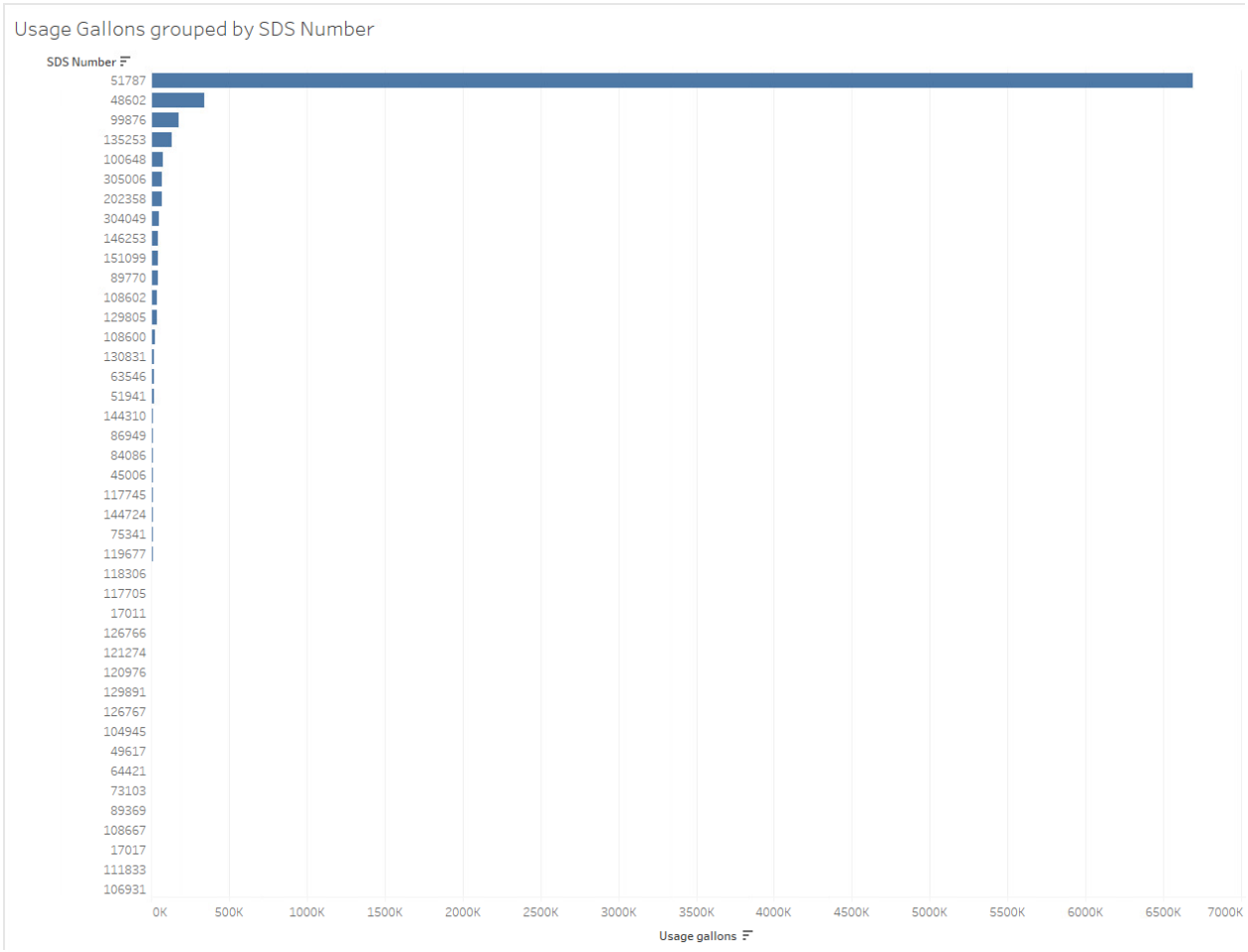
Usage Gallons Grouped by Day:



One of the main items that was asked of us was to understand the chemical and part usage by organization. Unfortunately, the organization data was not something that could be implemented as part of the project due to a missing field that would've allowed the join between organization and chemical usage. However, we've identified a few key insights that will help get a better bigger picture of what is most commonly used and is not.

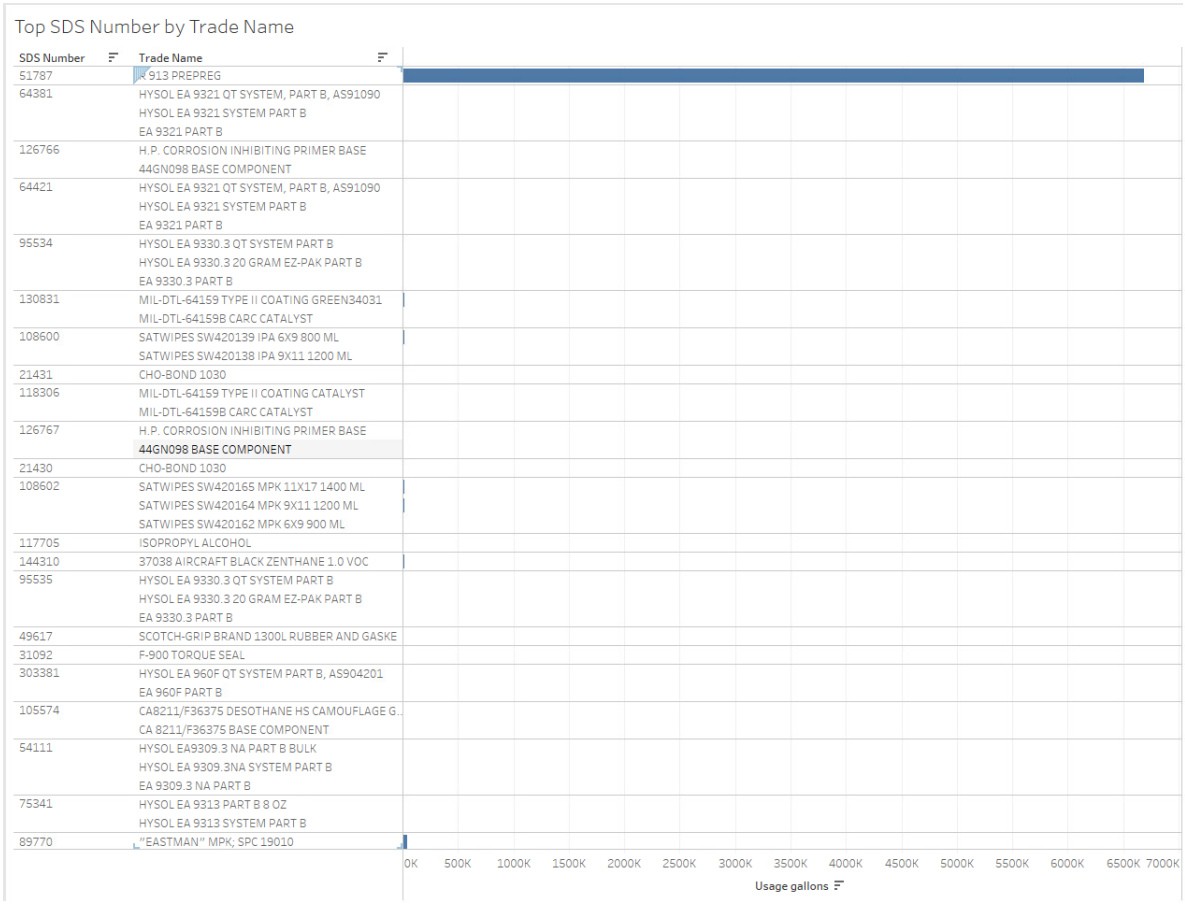
At a basic level, we can see in the visualization below that shows the most commonly used chemical. It has the SDS number '51787'. The total number of usage gallons for this SDS Number is 6,690,098 units.

Usage Gallons grouped by SDS Number:



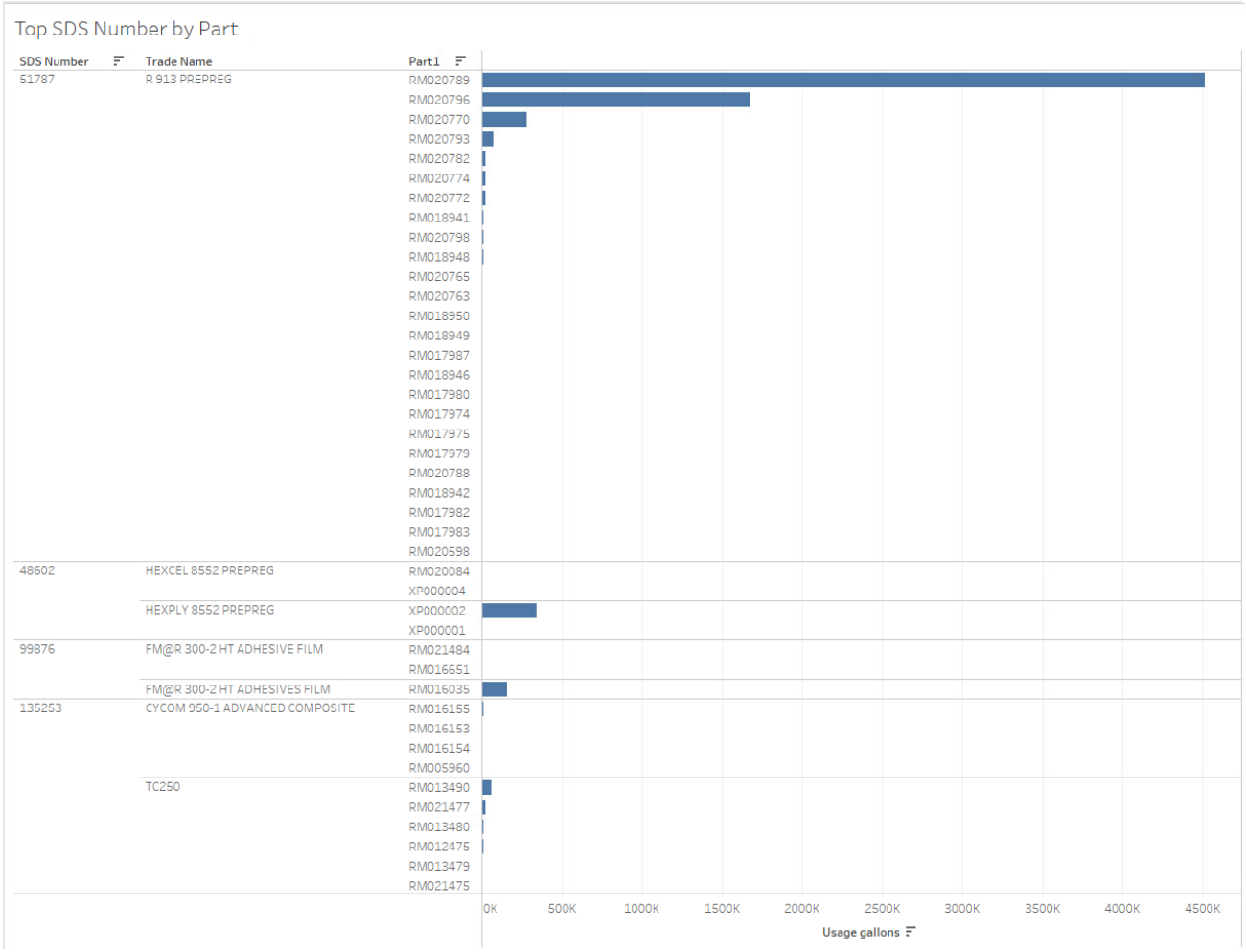
This is handy to know but drilling down and adding additional fields will give more complex insights. The first insight here is understanding what Trade Name is associated with SDS Number 51787 - in our case, the trade name associated with SDS Number 51787 is the ‘R 913 PREPREG’.

Top SDS Number by Trade Name:



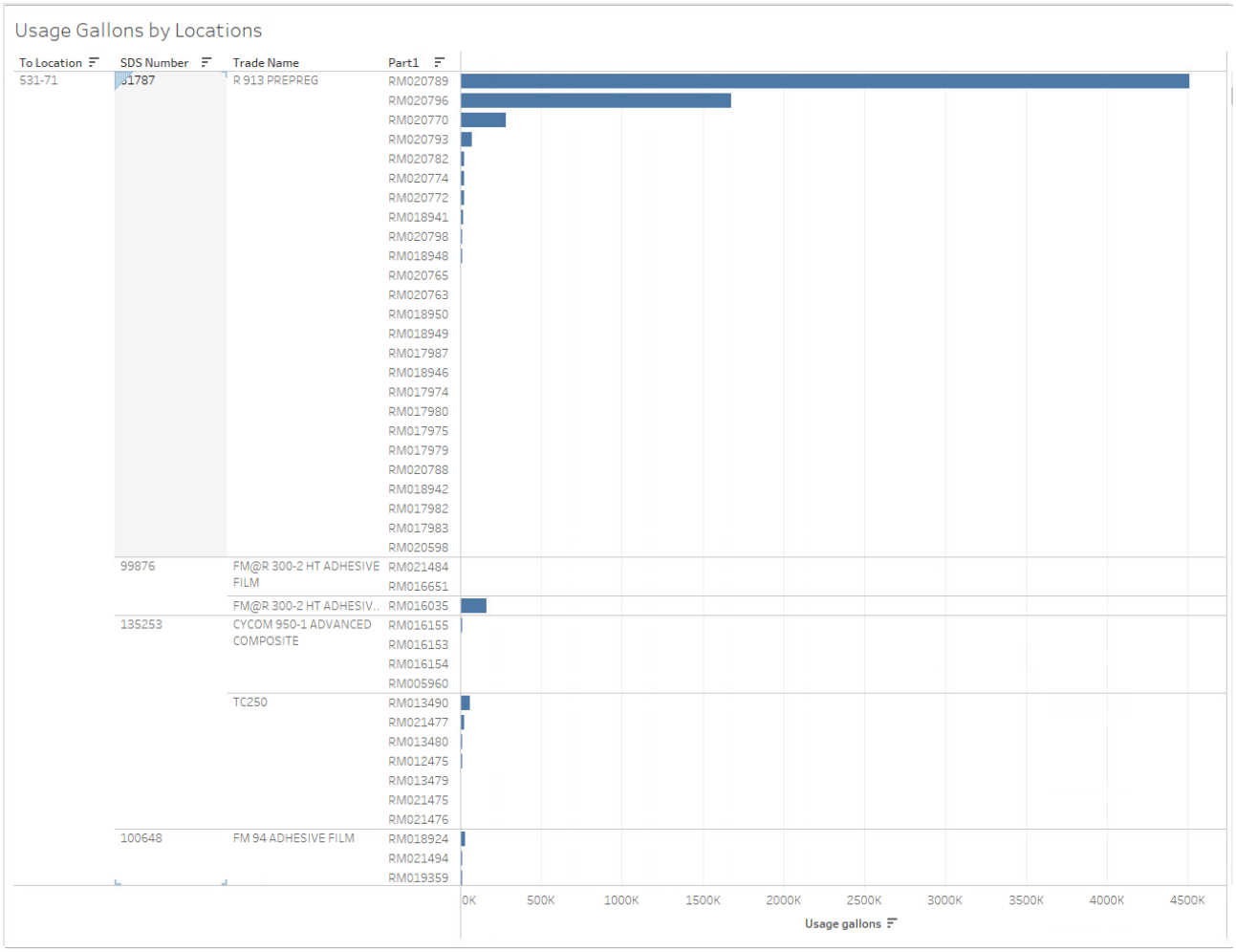
Adding an additional field here, specifically the Part1 field will give us the complete picture since there can be many parts related to the SDS number. Ideally, we would have shown the Chemical Abstracts Service number, or CAS number, but this was information that we did not have. The CAS number is a unique identifier that is assigned to every chemical substance. In substitution, we used the Part1 field. Since we do not have access to all of Boeing databases regarding this information, an assumption here is that the Boeing team will be able to identify this part number to a CAS number.

Top SDS Number by Trade Name and Part:



Another instance where having the organization data would be helpful is connecting it to the Location data that is already identified. The closest we can get to breaking down by “organization” is by showing the location of where the chemical was sent to. For example, we know that the most identified part number is “RM020789”, under the Trade Name “R 913 PREPREG” and SDS Number 51787. The Location where this part number was sent to is “531-71”.

Usage Gallons by Locations:



Even though we did not end up using the excel files with the organization information, it is important to go through the steps taken to show case what could be done for the future. Knowing that the ideal situation here would be to connect all the data together, one suggestion that we have for the Boeing team is to invest in a relational database to better query the data that exists and then to additionally reduce the manual effort of using R, like we did, to join the tables and create a final dataset.

Our Boeing representatives confirmed to us that the ‘Accounting Department’ variable would be the appropriate and unique variable to use to join the excel files together. However, among the different extracts, 'Accounting Department’ was available

but at times named differently. For example, in the ‘MasterOrgFile’ the accounting department field called ‘mractng’ and in the ‘Manager_Mesa_Site’ file, the field was actually called ‘Accounting Department’ contained the same data in the correct format. This was easy to join together but format is important to note because this was not the same in other files. In the ‘Accounting Departments from Craig Huebner’ file, we had to concatenate 'Acctg Bus Unit Cd' + 'Acctg Location Cd' + 'Acctg Dept Cd' to create the ‘Account Department’ field.

VII. Models and Evaluations

As discussed, the two models chosen for this project are ARIMA (Auto Regressive Integration Moving Average) and SARIMAX (Seasonal Auto-Regressive Integrated Moving Average with Exogenous factors). Both models are commonly used as forecasting models. From our Basic EDA script, we extracted a file to use for both forecasting models.

Starting with ARIMA, there are a few parameters that need to be defined for the model to work – *p* is the number of lag observations included in the model, also known as the lag order; *d* is the number of times that the raw observations are differenced, also called the degree of differencing; and 1 is the size of the moving average window called the order of moving average.

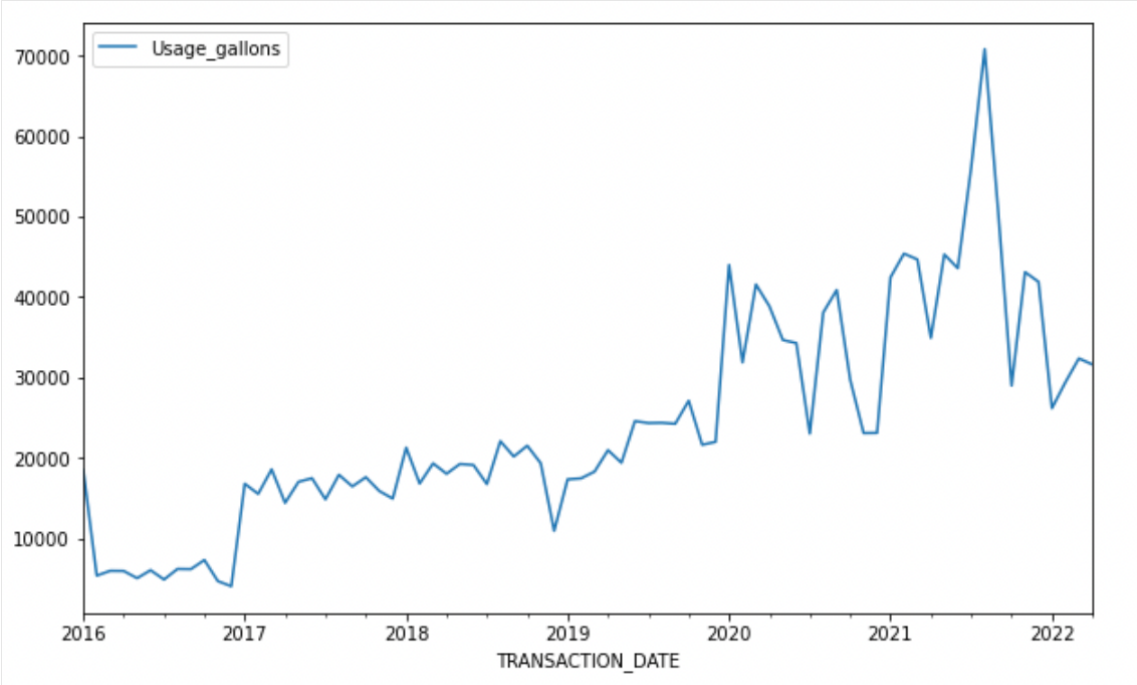
We utilized a few Python Libraries. For ARIMA we used a package called “statsmodels.tsa.arima.model” and imported the “ARIMA” package to utilize the premade function to help with forecasting. To check whether the data we were using was stationary, we used a library called ‘statsmodels.tsa.stattools’ and imported ‘adfuller’. For SARIMAX, we used statsmodels.tsa.statespace.sarimax to import the

‘SARIMAX’ function. Lastly, we used statsmodels.graphics.tsaplots, the ‘plot_acf’, ‘plot_pacf’ to help us determine our p, q, and d values to find the best model to use.

ARIMA

Upon implementing the ARIMA and SARIMAX models, we first needed to identify the transaction date variable as the datetime variable to use for forecasting. We initially noticed that the last month in the dataset, May 2022, was not fully complete, so we decided to filter out that month from our forecasting model. Since forecasting models tend to perform better monthly and reduce a lot of the noise that is associated with weekly transactions, we then decided to group the transactions by month to get total gallons consumed each month to allow the data to be more digestible and interpretable. After this initial round of data manipulation, our plot looked like this:

Tracking Usage Gallons by Transaction Date – Initial Plot:



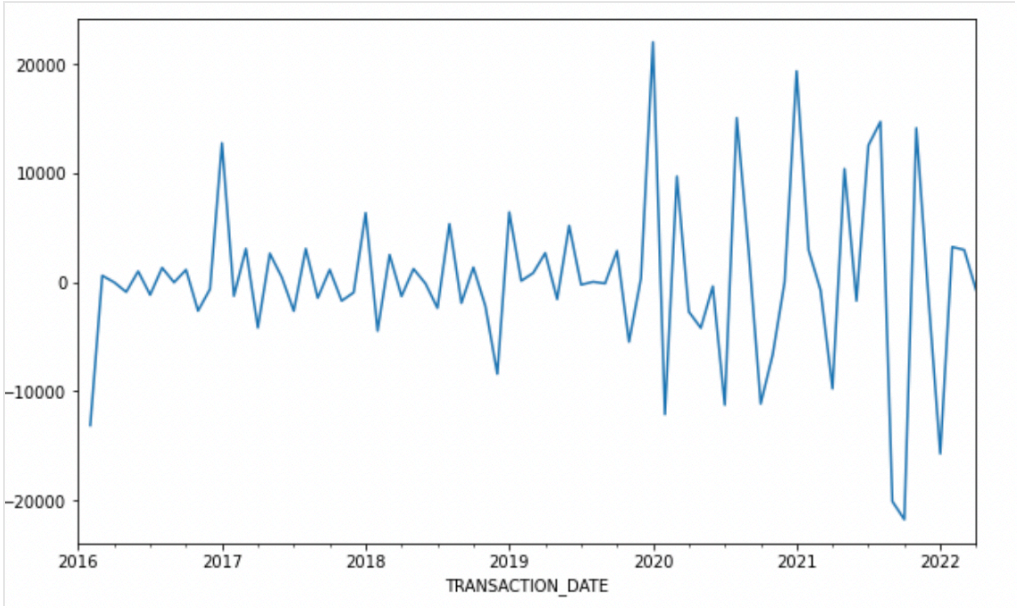
We then ran an Augmented Dickey-Fuller test to determine whether seasonality is present in the data, or if we are dealing with an already stationary dataset. As discussed in the Literature Review, the ADfuller test is based on hypothesis testing, so if the p-value is less than 0.05, we can assume that the data is already stationary (Verma, 2021). The results for the ADfuller test are as follows:

ADfuller Test – Round 1:

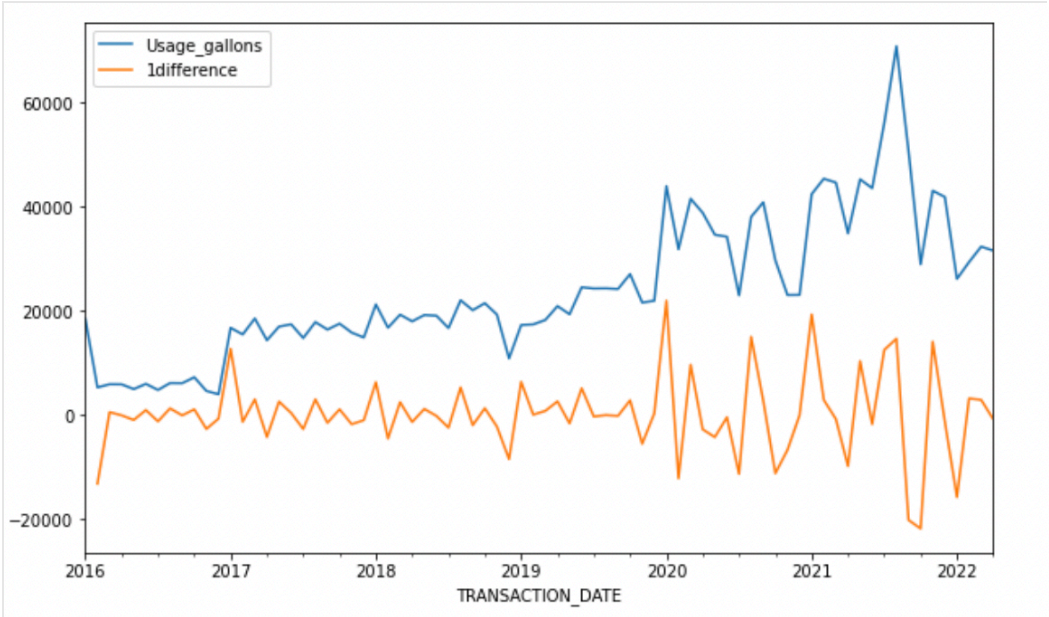
```
{'adf': -1.6736608458992983,
 'pvalue': 0.4447317279527293,
 'usedlag': 2,
 'nobs': 73,
 'criticalvalues': {'1%': -3.5232835753964475,
  '5%': -2.902030597326081,
  '10%': -2.5883710883843123},
 'icbest': 1303.8129598480937}
```

Looking at the p-value score of 0.44, we can determine that we cannot reject the null hypothesis and that we need to remove the seasonality from our dataset using differencing before we can proceed with the full implementation of the ARIMA model. As discussed in the Literature Review, the differencing variable *d* can be understood as the minimum number of differencing rounds needed to make the data stationary (Prabhakaran, 2022). Differencing is performed by subtracting the previous observation from the current observation. Below is what our differencing graph looks like, followed by a graph depicting our initial plot with the differencing plot overlayed:

Differencing Plot:



Original Plot with Differencing Plot Overlayed:

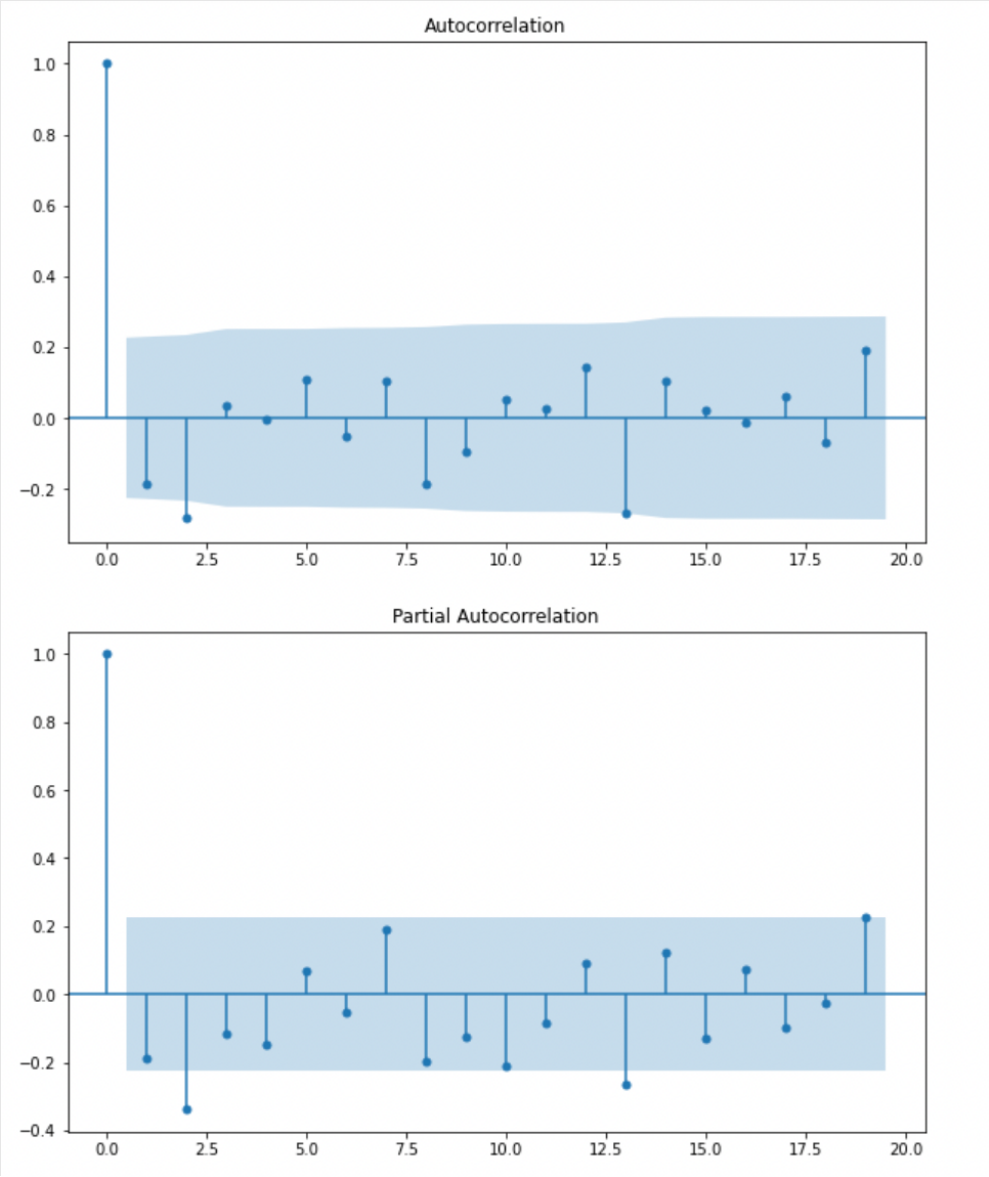


Here are the results once we re-apply the ADfuller test:

ADfuller Test – Round 2:

```
{'adf': -4.193589166184938,
 'pvalue': 0.0006755592722814878,
 'usedlag': 12,
 'nobs': 62,
 'criticalvalues': {'1%': -3.540522678829176,
  '5%': -2.9094272025108254,
  '10%': -2.5923136524453696},
 'icbest': 1281.6118533940491}
```

Since the p-value is now less than 0.05, we can reject the null hypothesis and conclude that our *d* = 1 as well as that the data is stationary. Now that the data is stationary, we then decided to look at the Autocorrelation (ACF) and Partial Autocorrelation (PACF) plots. Since most of the values are within the shaded region, then it can be assumed that most of the peaks and valleys present in the data are not statistically significant:

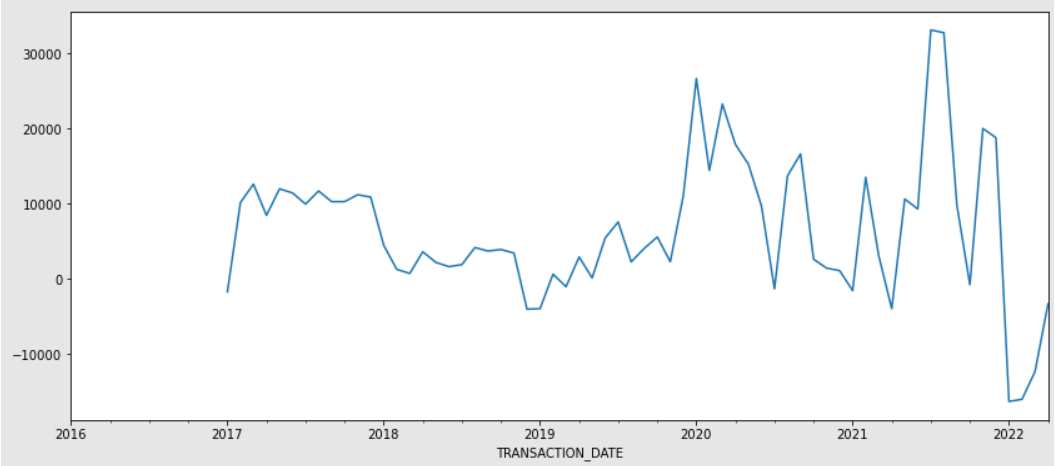


SARIMAX

For the SARIMAX model, we followed a similar process to the ARIMA model. We also chose to create our prediction model using the variables 'TRANSACTION_DATE' and 'Usage_gallons'. The same exact process was taken until the second round of ADF Testing in the ARIMA model for the SARIMAX model. After determining that the data model is stationary, we checked the model for seasonal difference. The seasonal difference can be computed by shifting the data by the number of rows per season (in

our example 12 months per year) and subtracting them from the previous season. This is not the first seasonal difference. If we get that the seasonal difference is stationary, then the D value will be 0. If not, then we will compute the seasonal first difference.

Seasonal Difference Factor for SARIMAX model:



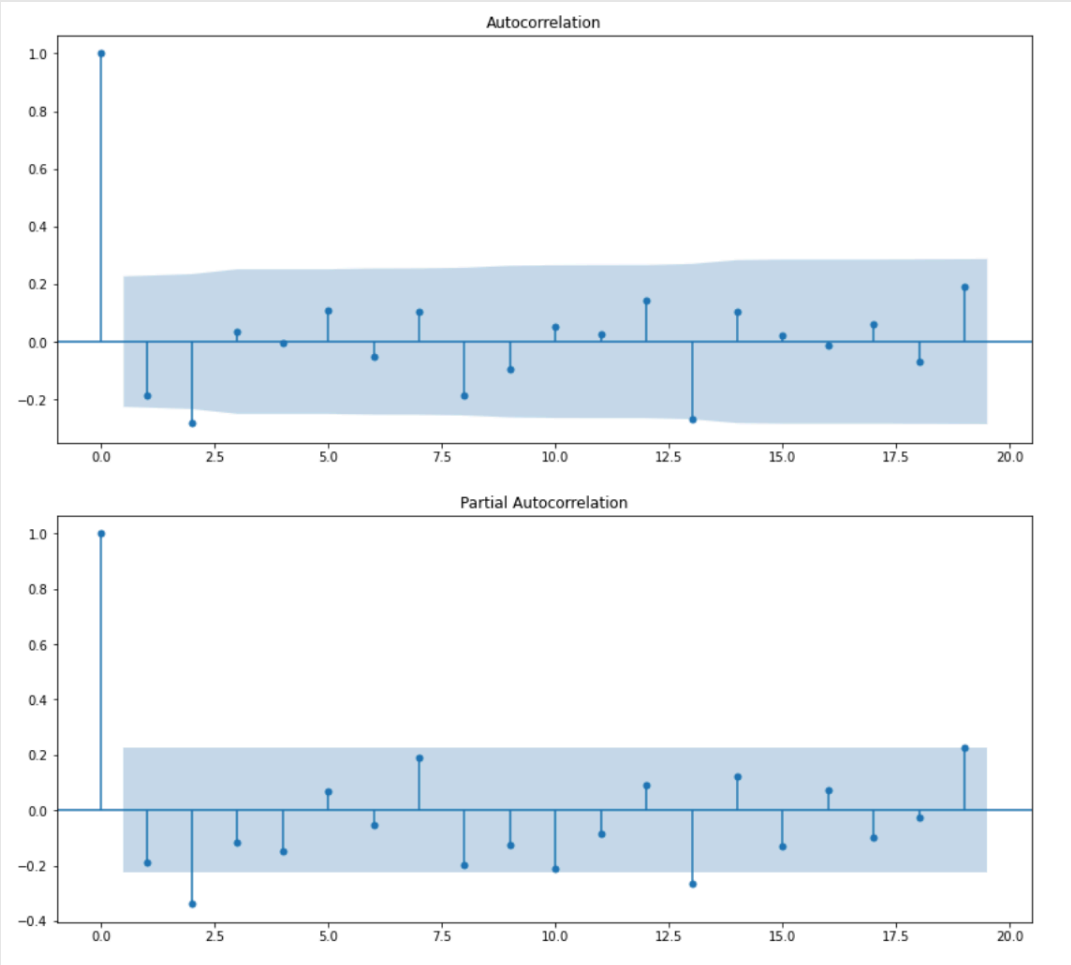
After taking the seasonal difference, we want to ensure again that the model continues to be stationary. We ran a third round of ADF testing with the result indicating a p-value less than 0.5 meaning that we still maintained a stationary model. The ADF results are as follows:

ADfuller Test – Round 3:

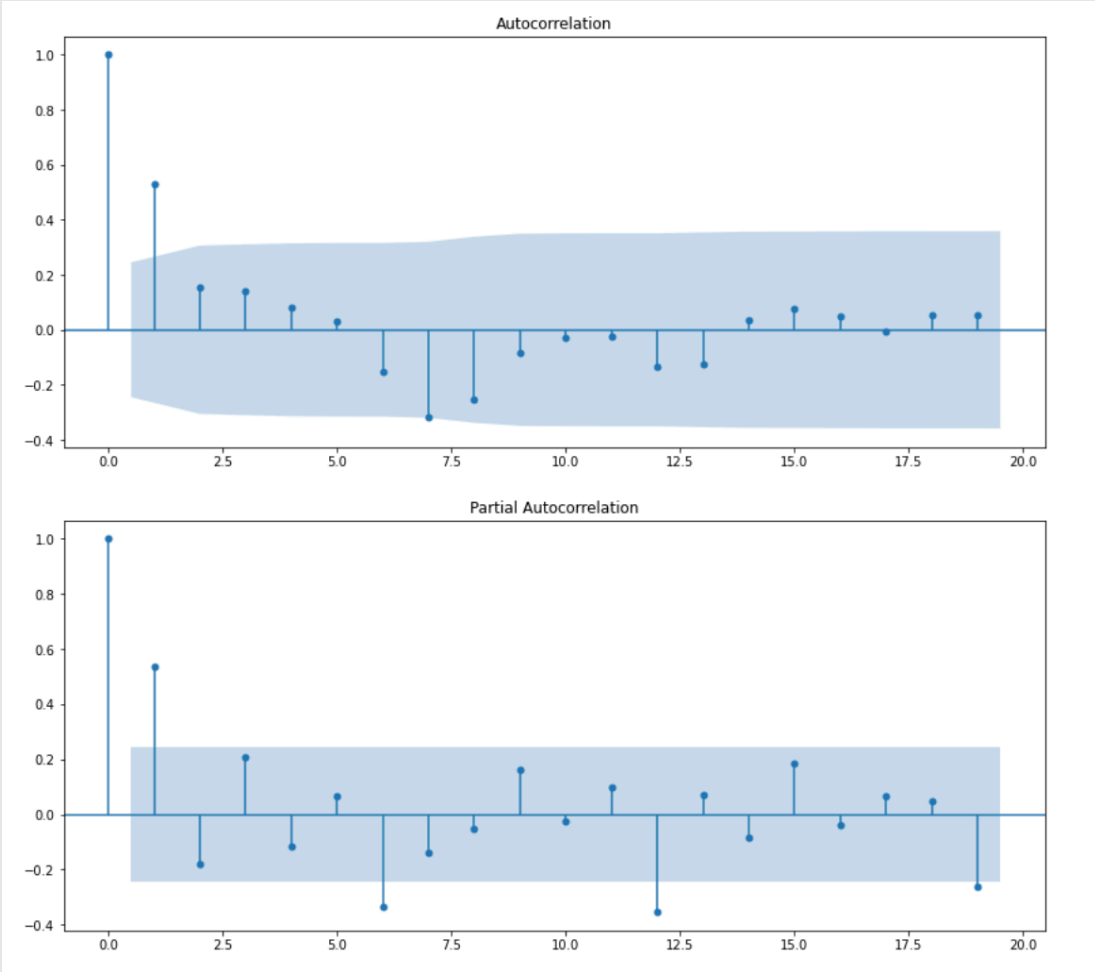
```
{'adf': -3.4221391202075835,
 'pvalue': 0.010226909181380008,
 'usedlag': 6,
 'nobs': 57,
 'criticalvalues': {'1%': -3.5506699942762414,
  '5%': -2.913766394626147,
  '10%': -2.5946240473991997},
 'icbest': 1091.4797304043595}
```

Since the p-value is still less than 0.05, we can reject the null hypothesis and conclude that our *d* = 1 as well as that the data is stationary. Now that the data is stationary, we then decided to look at the Autocorrelation (ACF) and Partial Autocorrelation (PACF) plots. Since most of the values are within the shaded region, then it can be assumed that most of the peaks and valleys present in the data are not statistically significant.

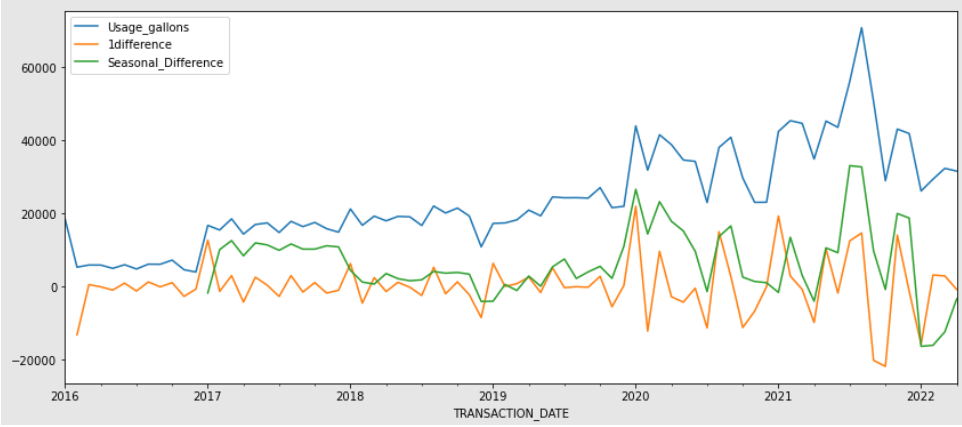
ACF and PACF with 1 Difference:



ACF and PACF with Seasonal Difference:



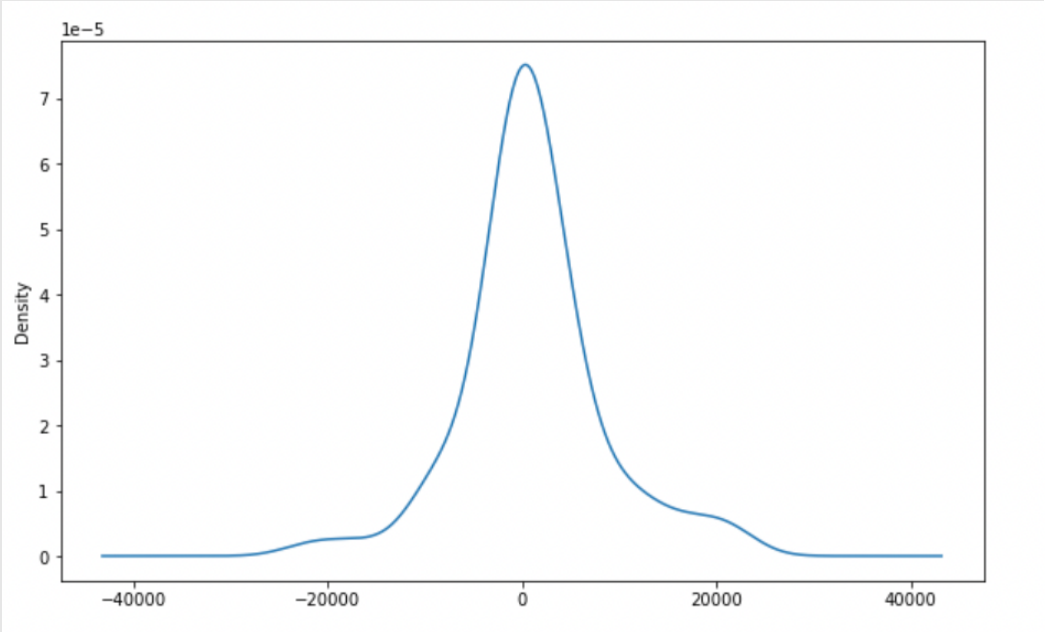
All Difference Plots in a Single Graph:



VIII. Interpretations and Analysis

ARIMA(1, 1, 1):

First, we ran an ARIMA(1, 1, 1) model. We can also see it has a normal distribution:



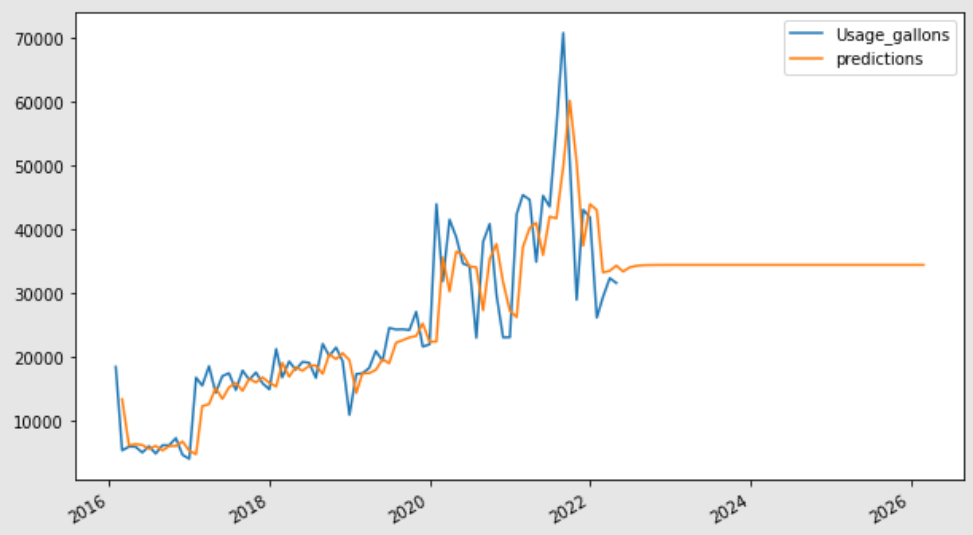
SARIMAX Results

Dep. Variable:	Usage_gallons	No. Observations:	76
Model:	ARIMA(1, 1, 1)	Log Likelihood	-769.831
Date:	Tue, 23 Aug 2022	AIC	1545.661
Time:	21:30:02	BIC	1552.614
Sample:	01-31-2016	HQIC	1548.438
	- 04-30-2022		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.3666	0.159	2.305	0.021	0.055	0.678
ma.L1	-0.7618	0.099	-7.668	0.000	-0.956	-0.567
sigma2	5.194e+07	1.65e-09	3.14e+16	0.000	5.19e+07	5.19e+07

Ljung-Box (L1) (Q):	0.14	Jarque-Bera (JB):	18.62
Prob(Q):	0.71	Prob(JB):	0.00
Heteroskedasticity (H):	7.32	Skew:	0.18
Prob(H) (two-sided):	0.00	Kurtosis:	5.41

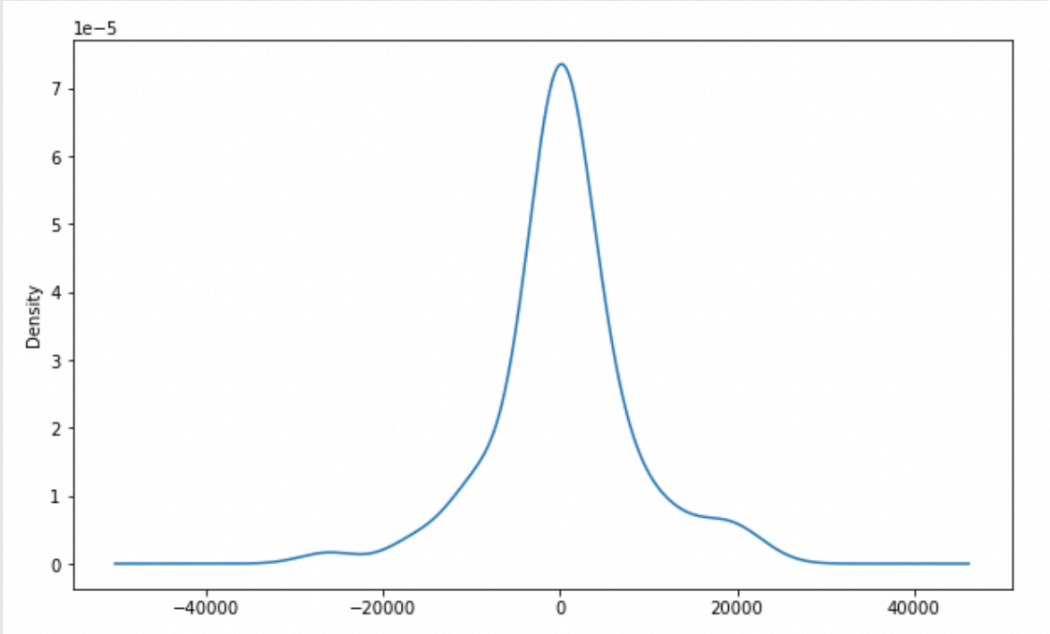
Looking at the results of ARIMA(1, 1, 1), an important variable to consider is the Akaike Information Criterion (AIC). The AIC determines how well the model fits the data it was created with. Typically, the lower the AIC, the better the model is at prediction. In this model, we have an AIC of 1545.661. The forecasting model can be seen below:



An important note on ARIMA models is that because it looks at the previous value, the forecasting models do not look as complex as SARIMA, which we will get into in the SARIMA model prediction. For the purpose of comparing ARIMA models, we are strictly comparing AIC values to choose the best model prediction.

ARIMA(1, 1, 0):

Here is our ARIMA model with parameters set to p = 1, q = 1, and *d* = 0. The data still has a normal distribution, which is good. The AIC value is 1552.475, meaning that ARIMA(1, 1, 1) is still the better model.

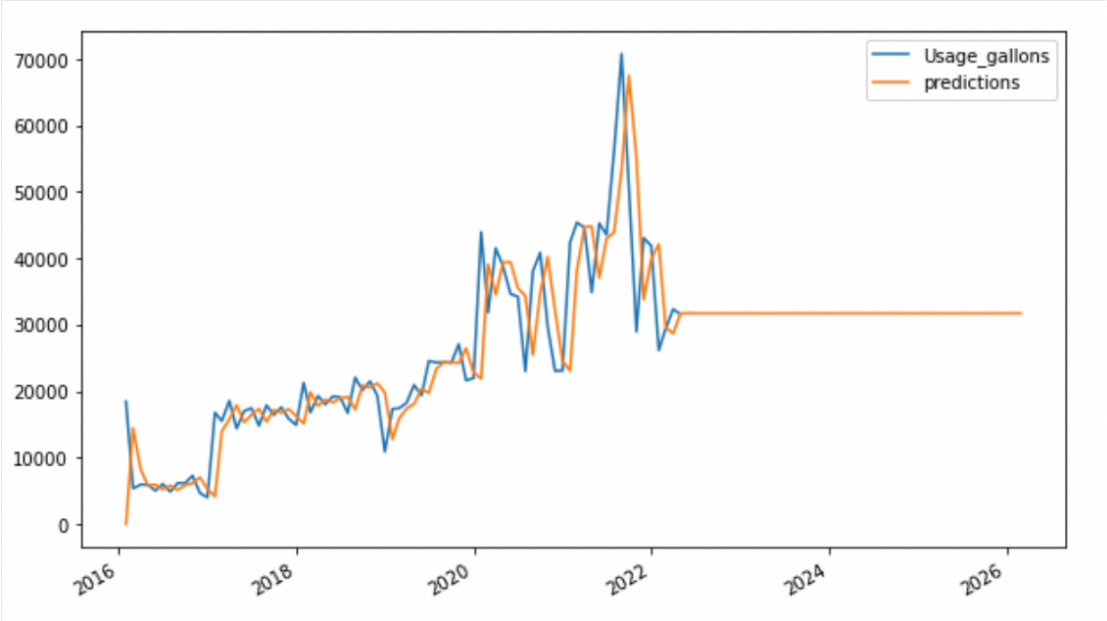


SARIMAX Results

Dep. Variable:	Usage_gallons	No. Observations:				76
Model:	ARIMA(1, 1, 0)	Log Likelihood				-774.237
Date:	Tue, 23 Aug 2022	AIC				1552.475
Time:	21:30:03	BIC				1557.110
Sample:	01-31-2016	HQIC				1554.325
	- 04-30-2022					
Covariance Type:	opg					

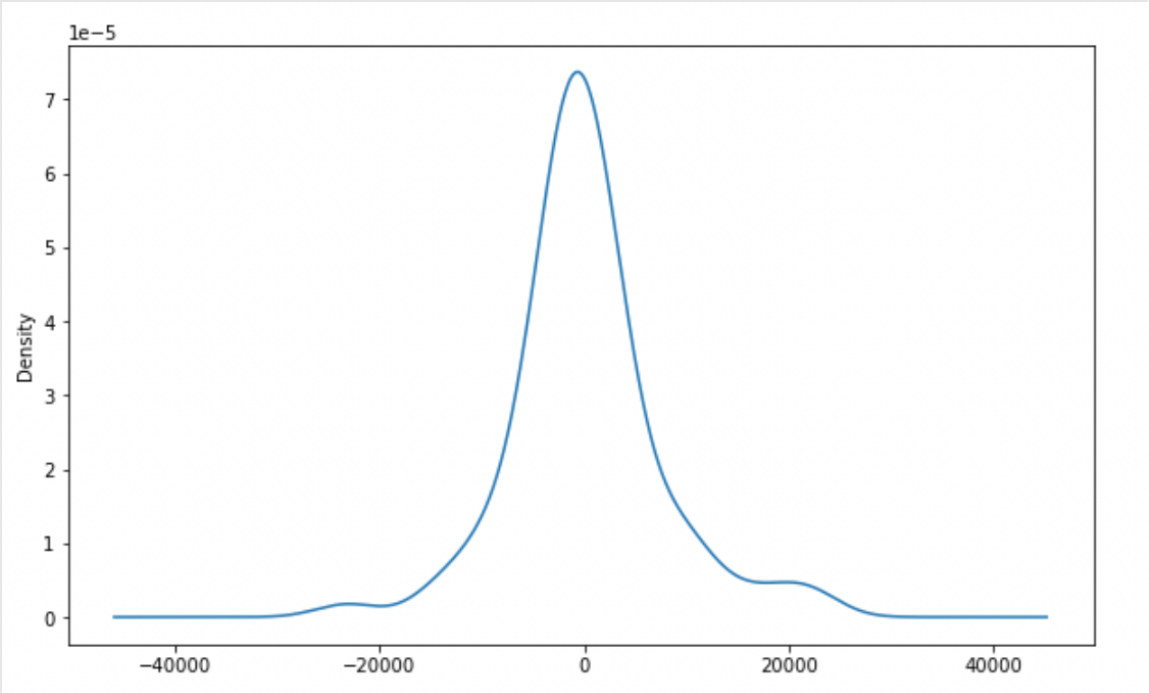
	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	-0.2227	0.072	-3.102	0.002	-0.363	-0.082
sigma2	5.375e+07	3.71e-10	1.45e+17	0.000	5.37e+07	5.37e+07

Ljung-Box (L1) (Q):	0.12	Jarque-Bera (JB):	22.83
Prob(Q):	0.73	Prob(JB):	0.00
Heteroskedasticity (H):	8.36	Skew:	-0.15
Prob(H) (two-sided):	0.00	Kurtosis:	5.69

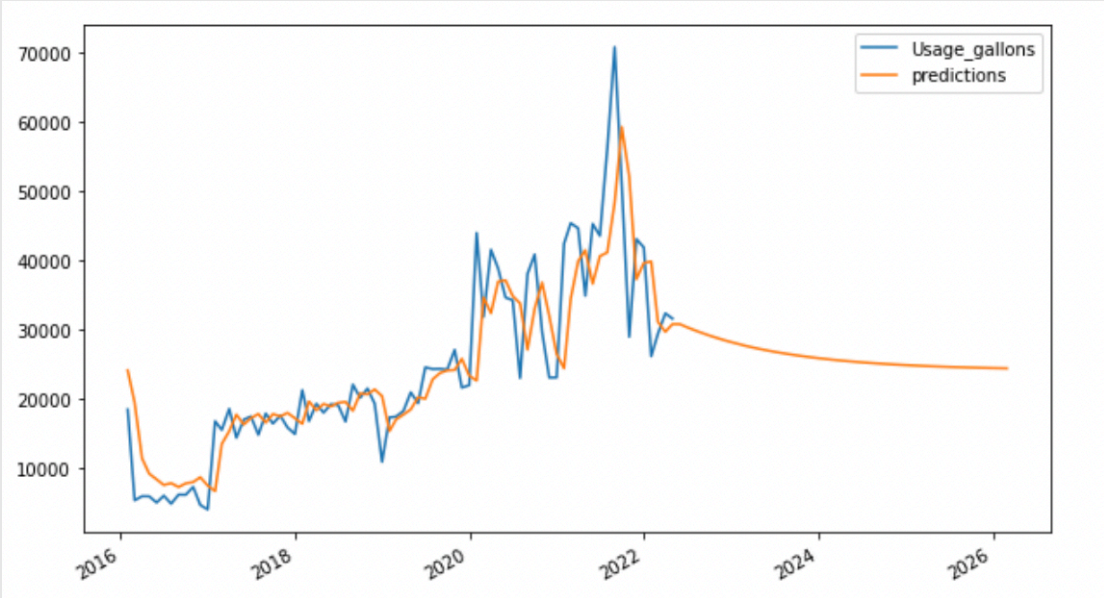


ARIMA(1, 0, 1):

In this model, we can once again see the normal distribution of data. The AIC value here is 1572.594, which is higher than the previous two models, so it appears to be the worst model thus far.

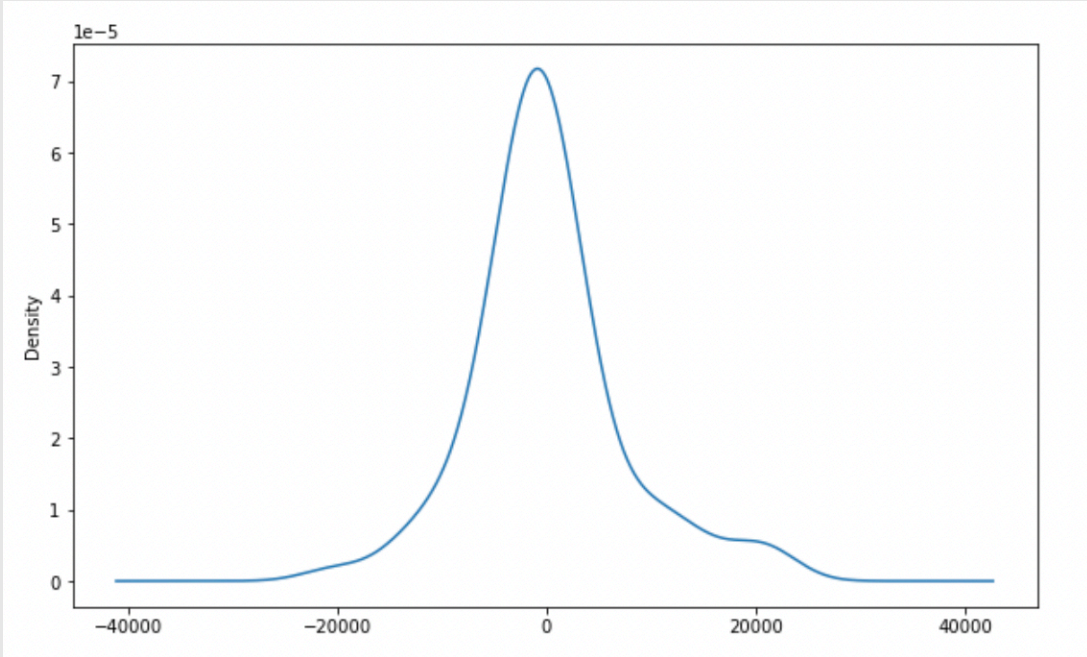


```
                               SARIMAX Results
==========================================================================================
Dep. Variable:                 Usage_gallons   No. Observations:                   76
Model:                 ARIMA(1, 0, 1)   Log Likelihood                -782.297
Date:                Tue, 23 Aug 2022   AIC                           1572.594
Time:                        21:30:04   BIC                           1581.916
Sample:                    01-31-2016   HQIC                          1576.319
                         - 04-30-2022
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const       2.413e+04   8839.908      2.729      0.006    6802.252    4.15e+04
ar.L1          0.9324      0.042     21.952      0.000       0.849       1.016
ma.L1         -0.3735      0.117     -3.204      0.001      -0.602      -0.145
sigma2      5.078e+07      5.099   9.96e+06      0.000    5.08e+07    5.08e+07
===================================================================================
Ljung-Box (L1) (Q):                   0.85   Jarque-Bera (JB):                22.89
Prob(Q):                              0.36   Prob(JB):                         0.00
Heteroskedasticity (H):               5.68   Skew:                             0.43
Prob(H) (two-sided):                  0.00   Kurtosis:                         5.55
===================================================================================
```



ARIMA(2, 0, 0):

For the next model, the parameters are set to ARIMA(2, 0, 0). The AIC value is even higher here, at 1574.619, so this will not be a contender in our best ARIMA model selection. Given all of the different ARIMA models, we can confirm that the optimal parameters are set to ARIMA(1, 1, 1).

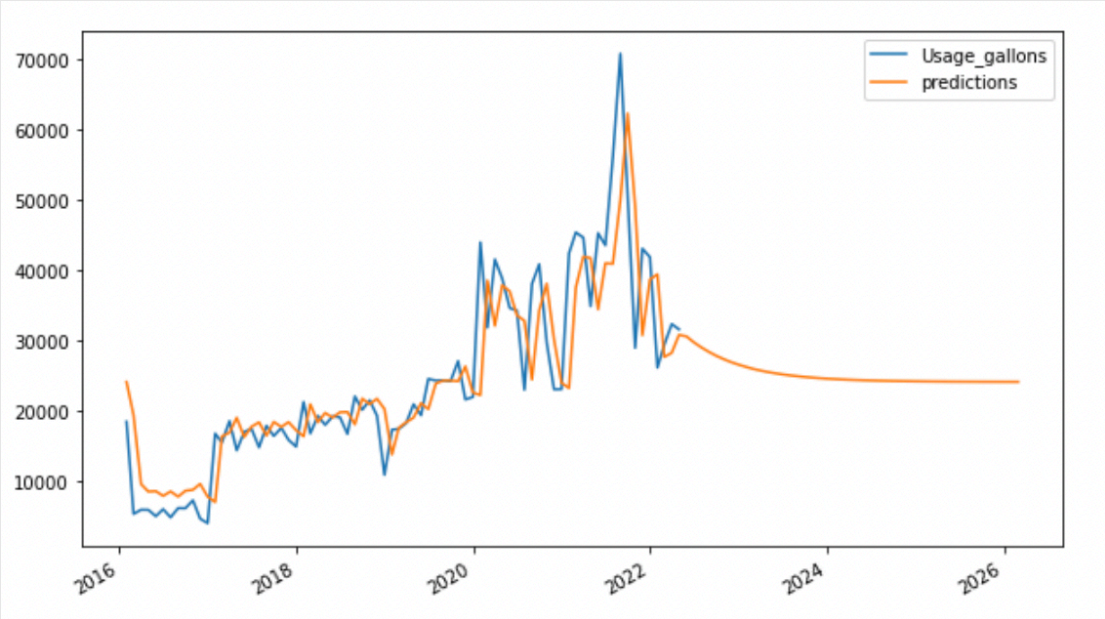


SARIMAX Results

Dep. Variable:	Usage_gallons		No. Observations:			76
Model:	ARIMA(2, 0, 0)		Log Likelihood			-783.309
Date:	Tue, 23 Aug 2022		AIC			1574.619
Time:	21:30:04		BIC			1583.942
Sample:	01-31-2016		HQIC			1578.345
	- 04-30-2022					
Covariance Type:	opg					

	coef	std err	z	P>|z|	[0.025	0.975]
const	2.413e+04	6580.414	3.667	0.000	1.12e+04	3.7e+04
ar.L1	0.7391	0.084	8.819	0.000	0.575	0.903
ar.L2	0.1151	0.077	1.497	0.134	-0.036	0.266
sigma2	5.023e+07	3.612	1.39e+07	0.000	5.02e+07	5.02e+07

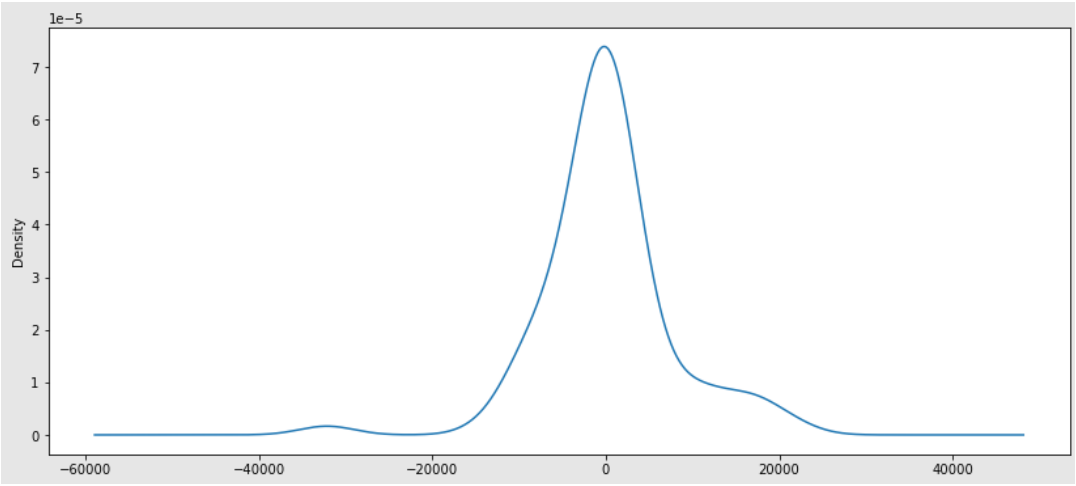
Ljung-Box (L1) (Q):	0.02	Jarque-Bera (JB):	15.29
Prob(Q):	0.88	Prob(JB):	0.00
Heteroskedasticity (H):	5.48	Skew:	0.64
Prob(H) (two-sided):	0.00	Kurtosis:	4.79



SARIMAX (p = 1, d = 1, and q = 1, P = 1, D = 1, Q = 0, and m = 12):

Based on the ACF and the PACF we were able the determine the ideal values for p, d, and q. Then we can build our forecasting models to determine if the model is normal and what our predictions for the next 5 years look like. For the first SARIMAX model, we set p = 1, d = 1, and q = 1, P = 1, D = 1, Q = 0, and m = 12. We ended up getting a normal distribution for our model.

Data Distribution for SARIMAX Model:



Looking at the results of the model, the measure to use to determine the best fit model for our data is the AIC value. Akaike information criterion (AIC) is a mathematical method for evaluating how well a model fits the data it was generated from for the best fit. A lower AIC score is better. More parameters also affect the AIC score but in our case, we only have 2 variables – TRANSACTION_DATE and 'Usage_gallons'. The AIC score is 1316.386, as determined by the most ideal parameters for p, q, and d.

SARIMAX Model 1 Results:

SARIMAX Results

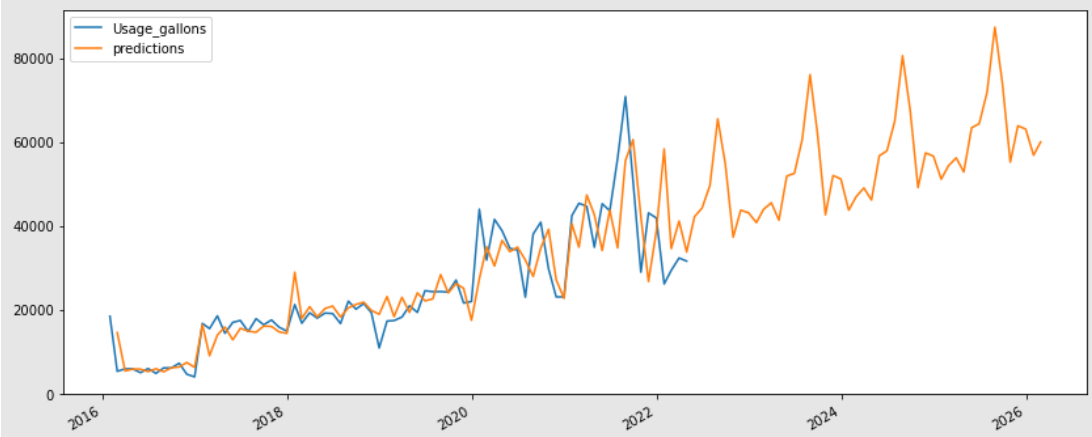
Dep. Variable:	Usage_gallons	No. Observations:	76
Model:	SARIMAX(1, 1, 1)x(1, 1, [], 12)	Log Likelihood	-654.193
Date:	Tue, 23 Aug 2022	AIC	1316.386
Time:	21:32:50	BIC	1324.959
Sample:	01-31-2016	HQIC	1319.758
	- 04-30-2022		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.5681	0.099	5.737	0.000	0.374	0.762
ma.L1	-1.0000	0.112	-8.960	0.000	-1.219	-0.781
ar.S.L12	-0.3797	0.159	-2.384	0.017	-0.692	-0.068
sigma2	5.93e+07	1.88e-09	3.15e+16	0.000	5.93e+07	5.93e+07

Ljung-Box (L1) (Q):	0.20	Jarque-Bera (JB):	46.43
Prob(Q):	0.65	Prob(JB):	0.00
Heteroskedasticity (H):	17.91	Skew:	-0.47
Prob(H) (two-sided):	0.00	Kurtosis:	7.10

For the predictions graph, we can see the seasonal spikes that may occur in the upcoming years. The overall trend shows that usage gallons will steadily increase in the next 5 years. Like how the historical data shows us spikes in usage gallons throughout a year, we can see how in the future, there too will be seasonal spikes in where the usage gallon will increase.

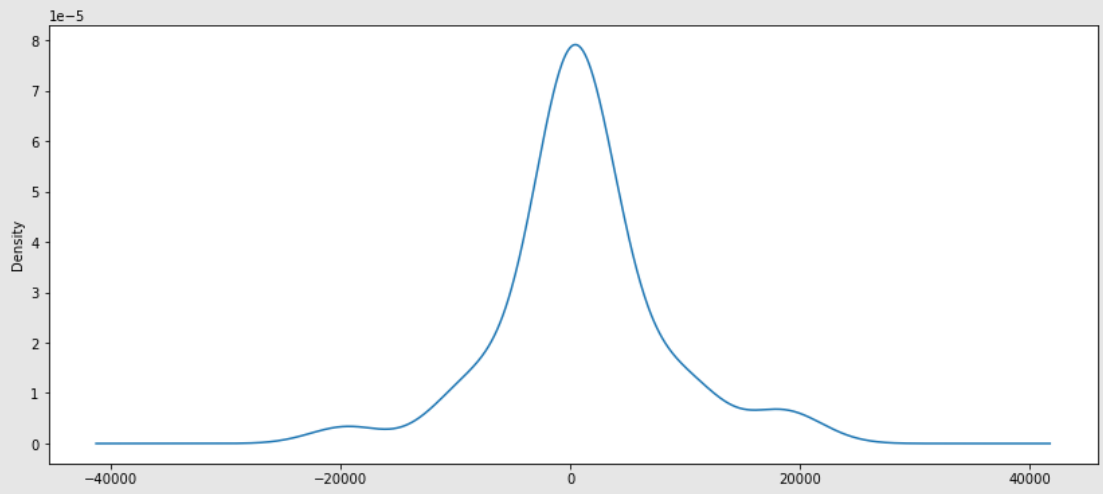
Usage Gallons Predictions Model 1:



SARIMAX (p = 1, d = 1, and q = 1, P = 1, D = 0, Q = 0, and m = 12):

For the second SARIMAX model, we set p = 1, d = 1, and q = 1, P = 1, **D = 0**, Q = 0, and m = 12. We still ended up getting a normal distribution for our model.

Data Distribution for SARIMAX Model 2:



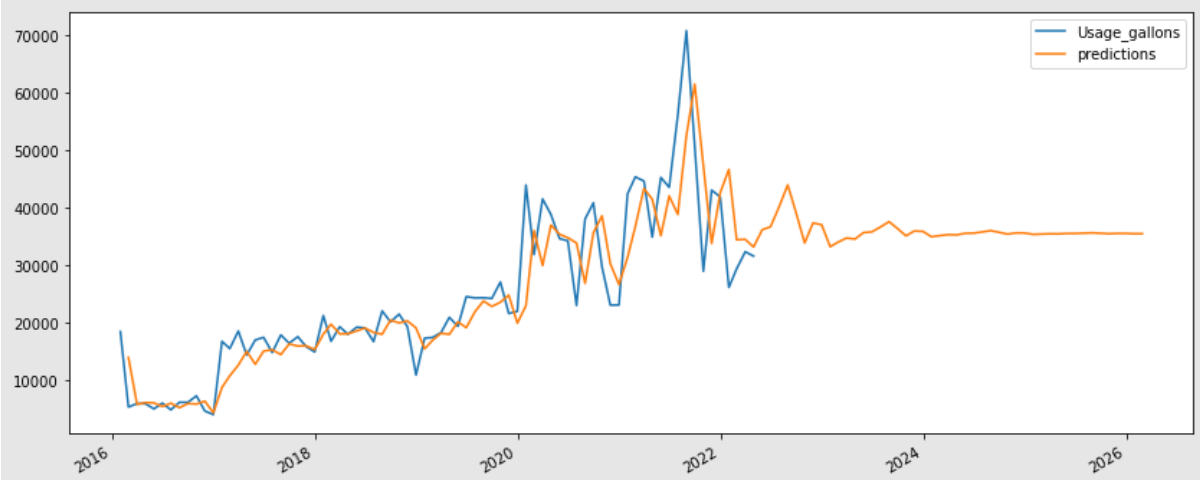
The AIC score, however, in this second model has increased drastically. The AIC score from Model 1 to Model 2 has increased by 228.35. This would not be the best model to use moving forward.

SARIMAX Model 2 Results:

```
                                SARIMAX Results
==========================================================================================
Dep. Variable:                      Usage_gallons   No. Observations:                   76
Model:             SARIMAX(1, 1, 1)x(1, 0, [], 12)  Log Likelihood                -768.368
Date:                             Tue, 23 Aug 2022  AIC                           1544.736
Time:                                     21:32:51  BIC                           1554.006
Sample:                                 01-31-2016  HQIC                          1548.437
                                      - 04-30-2022
Covariance Type:                              opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.4572      0.148      3.092      0.002       0.167       0.747
ma.L1         -0.8233      0.093     -8.898      0.000      -1.005      -0.642
ar.S.L12       0.2444      0.094      2.589      0.010       0.059       0.429
sigma2      5.194e+07   1.09e-09   4.78e+16      0.000    5.19e+07    5.19e+07
===================================================================================
Ljung-Box (L1) (Q):                   0.24   Jarque-Bera (JB):                16.96
Prob(Q):                              0.62   Prob(JB):                         0.00
Heteroskedasticity (H):               9.52   Skew:                             0.03
Prob(H) (two-sided):                  0.00   Kurtosis:                         5.33
===================================================================================
```

We can also see how the predictions graph has changed as well. It doesn't make sense that the usage gallons would taper off and be stagnant between 30,000 and 40,000 gallons over the next 5 years.

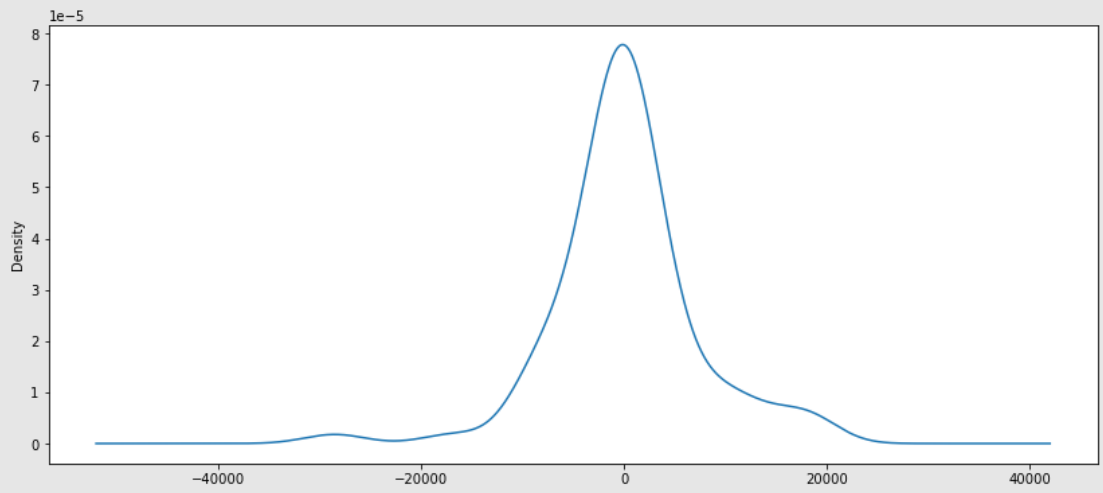
Usage Gallons Predictions Model 2:



SARIMAX (p = 1, d = 1, and q = 1, P = 1, D = 1, Q = 1, and m = 12):

For the third SARIMAX model, we set p = 1, d = 1, and q = 1, P = 1, **D = 1**, **Q = 1**, and m = 12. We still ended up getting a normal distribution for our model.

Data Distribution for SARIMAX Model 3:



The AIC score in the third model is very close to the first model. The AIC score for Model 3 is 1316.351, which is lower than Model 1 by 0.035. This model is a better model to use to predict future usage gallon consumption.

SARIMAX Model 3 Results:

SARIMAX Results

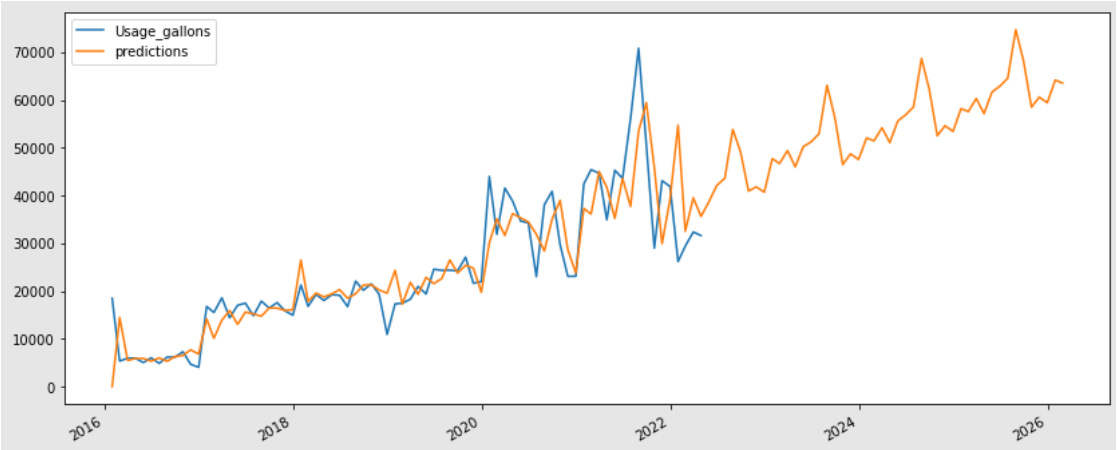
Dep. Variable:	Usage_gallons	No. Observations:	76
Model:	SARIMAX(1, 1, 1)x(1, 1, 1, 12)	Log Likelihood	-653.175
Date:	Tue, 23 Aug 2022	AIC	1316.351
Time:	21:32:52	BIC	1327.067
Sample:	01-31-2016	HQIC	1320.565
	- 04-30-2022		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.5579	0.219	2.546	0.011	0.128	0.987
ma.L1	-0.9646	0.128	-7.550	0.000	-1.215	-0.714
ar.S.L12	-0.1102	0.594	-0.186	0.853	-1.274	1.053
ma.S.L12	-0.6084	0.494	-1.230	0.219	-1.577	0.361
sigma2	8.079e+07	1.63e-08	4.94e+15	0.000	8.08e+07	8.08e+07

Ljung-Box (L1) (Q):	0.18	Jarque-Bera (JB):	35.63
Prob(Q):	0.67	Prob(JB):	0.00
Heteroskedasticity (H):	23.55	Skew:	-0.53
Prob(H) (two-sided):	0.00	Kurtosis:	6.53

The results for the prediction graph estimate a less steeper usage gallon consumption over the course of the next 5 years than model 1 predicted. The peak usage for model 1 would reach close to 80,000 where here it is showing close to 70,000.

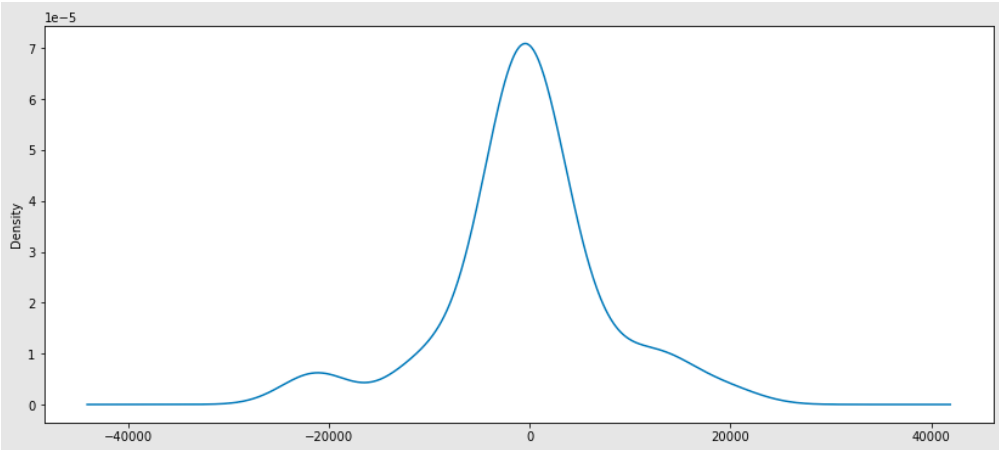
Usage Gallons Predictions Model 3:



SARIMAX (p = 1, d = 2, and q = 1, P = 1, D = 0, Q = 0, and m = 12):

For the final SARIMAX model, we set p = 1, **d = 2**, and q = 1, P = 1, **D = 0**, **Q = 0**, and m = 12. We still ended up getting a normal distribution for our model.

Data Distribution for SARIMAX Model 4:



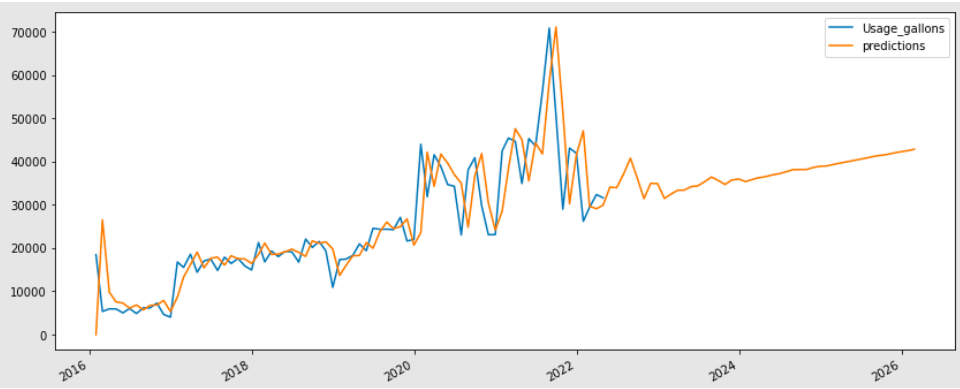
The AIC score in the model 4 is similar to the model 2. The AIC score for Model 4 is 1540.994, which is the second worst model, among the four that we have tested out with SARIMAX.

SARIMAX Model 4 Results:

```
                                SARIMAX Results
==========================================================================================
Dep. Variable:                      Usage_gallons   No. Observations:                   76
Model:             SARIMAX(1, 2, 1)x(1, 0, [], 12)   Log Likelihood                -766.497
Date:                             Tue, 23 Aug 2022   AIC                           1540.994
Time:                                     21:32:53   BIC                           1550.210
Sample:                                 01-31-2016   HQIC                          1544.671
                                      - 04-30-2022
Covariance Type:                              opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.1310      0.126     -1.042      0.298      -0.377       0.115
ma.L1         -0.9863      0.134     -7.364      0.000      -1.249      -0.724
ar.S.L12       0.2337      0.133      1.763      0.078      -0.026       0.493
sigma2      7.877e+07   1.36e-09   5.79e+16      0.000    7.88e+07    7.88e+07
===================================================================================
Ljung-Box (L1) (Q):                   0.40   Jarque-Bera (JB):                15.76
Prob(Q):                              0.53   Prob(JB):                         0.00
Heteroskedasticity (H):              16.61   Skew:                            -0.34
Prob(H) (two-sided):                  0.00   Kurtosis:                         5.15
===================================================================================
```

This prediction graph is again very similar to model 2 that we saw previously. The straight line as a prediction over the next 5 years would not make sense given how oscillating the historical data is.

Usage Gallons Predictions Model 4:



Our overall interpretation of these two models concludes that the SARIMA model is the better model to use over the ARIMA model. Simply the process of our entire project has changed as we had to shift our efforts towards forecasting. The original goal for our project was to breakdown chemical consumption by organization which changed to forecasting due to unforeseen missing data.

Some limitations that we felt for this project was the fact that we didn't have the extra dataset that would've enabled us to join the data together. Our Boeing representative was out of office and by the time we discussed what was missing and the solution to get back on track, it was week 6. Unfortunately, we had to move on to our analysis with the data that we were given relating to our forecasting objective. However, in our previous section relating to data preprocessing, we explain how to combine the excel files that were given and what variables to join on. This would be helpful for a future implementation of the project.

Another limitation that we felt was time. Because we changed objectives in week 6, we didn't have the time to go back and further preprocess the data to cater it towards the forecasting objective. The next step that we would take here if we had the time would be to eliminate extra variables that we didn't use in the end to have a cleaner dataset to use and also potentially run another model. We would do the SARIMAX model which would take into account any exogenous variables that could give us an even more accurate forecasting model.

IX. Discussion

As mentioned before, we strongly believe that the SARIMA model is more effective and accurate at forecasting than the ARIMA model. We believe this because of the process

we took. We started by evaluating the validity of the dataset. Looking at the trends of the data sets gave us a datapoint to determine if the dataset was stationary or not. To validate stationarity, we ran the Augmented Dickey-Fuller (ADF) test and repeated the process of differencing and checking with ADF until we reached stationarity. This helped us choose the parameters to use in the SARIMA function. And then lastly, the ACF and PACF tests also gave the confidence on which parameters to use when running the model. The entire process was crucial and across the ARIMA and SARIMA models, the first model we ran gave us the best fit model.

Across the two models, one measure that we relied on was the Akaike Information Criterion, or AIC score. This score helps with model selection. The way that we evaluate the number is that the lower the number, the better the model. Not only did we hypothesize that after evaluating the best parameters to use would give us the best AIC score, but it was also proven. The AIC score for ARIMA was 1545.661 and for SARIMA it was 1316.386 confirming that the SARIMA model was a better predictor of chemical usage.

X. Conclusion

All things considered, the time series model does have limitations in dealing with the type of problem we were tasked to solve. While time series modeling is a great tool for problems related to customer demand analysis from a sales perspective, it is not an optimal model for tackling problems related to internal inventory management. Rather, an inventory management method would be a better option to analyze material usage because these models are specifically built upon operation plans, materials and stock/inventory.

As a team, we still believe that a classification algorithm would have been more effective at predicting chemical consumption. However, since we were not provided the full data, we were unable to actually break down consumption at the organizational level. Thus, we had to get scrappy and work with what we had—and our only option was to use a forecasting model to predict chemical consumption to some degree. Given external factors such as our Boeing rep's paternity leave, we had a major time constraint in how long we had to determine we needed to use a forecasting model. Even if we had more time to explore forecasting analyses further, we would love to experiment with exogenous variables and see if the SARIMAX model would outperform the SARIMA model, in the same way that the SARIMA model outperformed the ARIMA model.

XI. References

Artley, B. (2022, June 27). *Time Series Forecasting with ARIMA, SARIMA and SARIMAX*. Medium. Retrieved August 14, 2022, from https://towardsdatascience.com/time-series-forecasting-with-arima-sarima-and-sarimax-ee61099e78f6

Effrosynidis, D. (2020, April 11). *Forecasting wars: Classical forecasting methods vs machine learning*. Medium. Retrieved August 22, 2022, from https://towardsdatascience.com/forecasting-wars-classical-forecasting-methods-vs-machine-learning-4fd5d2ceb716

Johnson, J. (2020, September 18). *Predictive Analytics vs Machine Learning: What's the Difference?* BMC Blogs. Retrieved August 21, 2022, from https://www.bmc.com/blogs/machine-learning-vs-predictive-analytics/

Josephs, L. (2022, January 11). *Boeing 2021 airplane deliveries surged, led by return of 737 MAX, but were still behind AirbusLe*. CNBC. Retrieved August 21, 2022, from https://www.cnbc.com/2022/01/11/boeing-2021-deliveries-surge-led-by-return-of-737-max-but-still-behind-airbus.html

Levy, E. (2022, July 15). *What is the Parquet File Format and Why You Should Use it*. Upsolver. Retrieved August 8, 2022, from https://www.upsolver.com/blog/apache-parquet-why-use

Maull, L., McElroy, P., & Wingbermuehle, J. (2021, April 16). *Sustainability is Built In: Boeing's Product Life Cycle, Every Step of the Way*. Boeing. Retrieved August 22, 2022, from https://www.boeing.com/features/innovation-quarterly/2021/04/boeing-product-sustainability.page

Prabhakaran, S. (2022, March 8). *ARIMA Model - Complete Guide to Time Series Forecasting in Python: ML+*. Machine Learning Plus. Retrieved August 17, 2022, from https://www.machinelearningplus.com/time-series/arima-model-time-series-forecasting-python/

Safety Data Sheets (SDSS). Safety Data Sheets (SDSs) | Environmental Health and Safety. (n.d.). Retrieved August 9, 2022, from https://ehs.research.uiowa.edu/chemical/safety-data-sheets-sdss

Saxena, S. (2021, March 15). *Dashboard in tableau: Create your first dashboard in tableau*. Analytics Vidhya. Retrieved August 21, 2022, from https://www.analyticsvidhya.com/blog/2021/03/create-your-first-dashboard-in-tableau/

Verma, Y. (2021, July 29). *Complete Guide to SARIMAX in Python for Time Series Modeling*. Analytics India Magazine. Retrieved August 17, 2022, from https://analyticsindiamag.com/complete-guide-to-sarimax-in-python-for-time-series-modeling/

XII. Appendix

Data Preparation

1- Build different functions to read different data types and convert them to Pandas DataFrame

In [2]:
```
def read_parquet(file):
    '''Transform '.parquet' to pandas data frame.
represent basics information of data set'''

    df = pd.DataFrame(pd.read_parquet(file, engine='auto'))

    print('Size of dataframe: ', df.size, '\n')
print('# Dimentions of dataframe: ',df.ndim, '\n')
print('Main variables of dataframe: ',df.columns, '\n')
print('Sum of nulls: \n', df.isnull().sum(), '\n')
print('# Unique observations: ',df.ndim, '\n')
    print('General information of dataframe: ',df.info(), '\n')

return df
```

In [3]:
```
def read_excel(file):
    '''Transform '.xlsx' to pandas data frame.
represent basics information of data set'''

    df = pd.DataFrame(pd.read_excel(file))


    print('Size of dataframe: ', df.size, '\n')
print('# Dimentions of dataframe: ',df.ndim, '\n')
print('Main variables of dataframe: ',df.columns, '\n')
print('Sum of nulls: \n', df.isnull().sum(), '\n')
print('# Unique observations: ',df.ndim, '\n')
    print('General information of dataframe: ',df.info(), '\n')

return df
```

In [4]:
```
def read_csv(file):
    '''Transform '.csv' to pandas data frame.
represent basics information of data set'''

    df = pd.DataFrame(pd.read_csv(file))

    print('Size of dataframe: ', df.size, '\n')
print('# Dimentions of dataframe: ',df.ndim, '\n')
print('Main variables of dataframe: ',df.columns, '\n')
print('Sum of nulls: \n', df.isnull().sum(), '\n')
print('# Unique observations: ',df.ndim, '\n')
    print('General information of dataframe: ',df.info(), '\n')

return df
```

```
def visualization_parquet(file, attrib):
    desc = read_parquet_file(file)[attrib].describe()

    #create a subplot without frame
    plot = plt.subplot(111, frame_on=False)

    #remove axis
    plot.xaxis.set_visible(False)
plot.yaxis.set_visible(False)

    #create the table plot and position it in the upper left corner
table(plot, desc,loc='upper right')

    #save the plot as a png file
plt.savefig('desc_plot.png')
```

In [5]:

2- Read data and Build Pandas DataFrames

Ther are two .parquet files:

- "2016_2022dw_All_Transactions_Gallons_Total.parquet" which is smaller file
- "2016_2022dw_CAS_Totals.parquet" which is larger file

There are three scenarios to concatenate .parquet files:

- Scenario I: Remove duplicates in each file seprately then choose similar columns and finally concatenate files. And, Remove NAs.
- Scenario II: Choose similar columns between two data frames then concatenate files and finally Remove duplicates. And, Remove NAs.
- Scenario III: Use just larger file.

In [6]:
```
gallons_total =
read_parquet('C:/Users/sohra/Downloads/2016_2022dw_All_Transacti
```

```
Size of dataframe:  5123352

# Dimentions of dataframe:  2

Main variables of dataframe:  Index(['Unnamed: 0', 'ID',
'FROM_LOCATION', 'TO_LO

CATION', 'Part',

       'TRANSACTION', 'TRANSACTION_DATE', 'Kit_Quantity', 'Unit',
'KITS',

       'UPDATED_BY', 'TRADE_NAME', 'KIT_PART', 'SDS_Quantity',
'SDS_Number',

       'SPEC_MATERIAL_TYPE', 'SPEC_MATERIAL_CLASS',
'DOCUMENT_SPEC', 'Year',

       'Conversion', 'DivideBySG', 'Specific_gravity',
'Usage_gallons',

       'Part_specific'],
dtype='object')

Sum of nulls:

 Unnamed: 0                 0

ID                         0

FROM_LOCATION              0

TO_LOCATION                0

Part                       0

TRANSACTION                0
```

```
TRANSACTION_DATE            0

Kit_Quantity                0

Unit                        0

KITS                    13703

UPDATED_BY                  0

TRADE_NAME                  0

KIT_PART                    0

SDS_Quantity                0

SDS_Number                  0

SPEC_MATERIAL_TYPE     119015

SPEC_MATERIAL_CLASS    125510

DOCUMENT_SPEC            6295

Year                        0

Conversion                 26

DivideBySG                 52

Specific_gravity         3668

Usage_gallons
167 Part_specific
213447 dtype: int64

# Unique observations:  2

<class 'pandas.core.frame.DataFrame'>
Int64Index: 213473 entries, 0 to 8

Data columns (total 24 columns):

 #   Column              Non-Null Count
Dtype         ---  ------              -
-------------    -----               0
Unnamed: 0             213473 non-null
float64                     1       ID
213473 non-null   float64           2
FROM_LOCATION          213473 non-null
object                3     TO_LOCATION
213473 non-null  object
4   Part                 213473 non-null  object
5   TRANSACTION          213473 non-null  float64
    6   TRANSACTION_DATE     213473 non-null
    datetime64[ns]  7   Kit_Quantity         213473 non-
    null  float64         8   Unit                213473
    non-null  object
9   KITS                 199770 non-null  float64
```

```
10  UPDATED_BY           213473 non-null  object
    11  TRADE_NAME           213473 non-null  object
    12  KIT_PART             213473 non-null  object
    13  SDS_Quantity         213473 non-null  float64
14  SDS_Number           213473 non-null  float64
15  SPEC_MATERIAL_TYPE   94458 non-null   object
    16  SPEC_MATERIAL_CLASS  87963 non-null   object
    17  DOCUMENT_SPEC        207178 non-null  object
    18  Year                 213473 non-null  float64
19  Conversion           213447 non-null  float64
20  DivideBySG           213421 non-null  float64
    21  Specific_gravity     209805 non-null  float64
    22  Usage_gallons        213306 non-null  float64
    23  Part_specific        26 non-null      float64
    dtypes: datetime64[ns](1), float64(13), object(10)
memory usage: 40.7+ MB

General information of dataframe:  None
```

In [7]:
```
cas_total = read_parquet('C:/Users/sohra/Desktop/Untitled
Folder/2016_2022dw_CAS
```

```
Size of dataframe:  37560256

# Dimentions of dataframe:  2

Main variables of dataframe:  Index(['Part', 'TRANSACTION_DATE',
'TRADE_NAME',

'SDS_Number', 'TO_LOCATION',

       'Specific_gravity', 'Usage_gallons', 'CAS_Number',
'Norm_Conc',

       'Max_Conc', 'Chem_name', 'SPEC_MATERIAL_TYPE',
'SPEC_MATERIAL_CLASS',

       'DOCUMENT_SPEC', 'CAS_content_lb_gal', 'CAS_lbs',
'CAS_tons', 'Month',

       'Year', 'Unnamed: 0', 'ID', 'FROM_LOCATION', 'TRANSACTION',

       'Kit_Quantity', 'Unit', 'KITS', 'UPDATED_BY', 'KIT_PART',
'SDS_Quantity', 'Conversion', 'DivideBySG', 'Part_specific'],
dtype='object')

Sum of nulls:

 Part                          0

TRANSACTION_DATE              0

TRADE_NAME                    0

SDS_Number                    0

TO_LOCATION                   0
```

```
Specific_gravity          5645
Usage_gallons              390
CAS_Number               33886
Norm_Conc                34299
Max_Conc                 34299
Chem_name               721719
SPEC_MATERIAL_TYPE      586467
SPEC_MATERIAL_CLASS     668152
DOCUMENT_SPEC            28385
CAS_content_lb_gal       36257
CAS_lbs                  36503
CAS_tons                 36503
Month                    30375
Year                         0
Unnamed: 0             1143383
ID                     1143383
FROM_LOCATION          1143383
TRANSACTION            1143383
Kit_Quantity           1143383
Unit                   1143383
KITS                   1144855
UPDATED_BY             1143383
KIT_PART               1143383
SDS_Quantity           1143383
Conversion             1143383
DivideBySG
1143392 Part_specific
1173749 dtype: int64

# Unique observations:  2

<class 'pandas.core.frame.DataFrame'>
Int64Index: 1173758 entries, 0 to 8

Data columns (total 32 columns):
 #   Column               Non-Null Count    Dtype
```

```
---  ------                --------------    -----

0    Part                  1173758 non-null  object
1    TRANSACTION_DATE      1173758 non-null  datetime64[ns]
     2    TRADE_NAME           1173758 non-null  object
     3    SDS_Number           1173758 non-null  float64
     4    TO_LOCATION          1173758 non-null  object
5    Specific_gravity      1168113 non-null  float64
6    Usage_gallons         1173368 non-null  float64
     7    CAS_Number           1139872 non-null  object
     8    Norm_Conc            1139459 non-null  float64
     9    Max_Conc             1139459 non-null  float64
10   Chem_name             452039 non-null   object
11   SPEC_MATERIAL_TYPE    587291 non-null   object
     12   SPEC_MATERIAL_CLASS  505606 non-null   object
     13   DOCUMENT_SPEC        1145373 non-null  object
     14   CAS_content_lb_gal   1137501 non-null
     float64          15  CAS_lbs               1137255
     non-null  float64          16  CAS_tons
     1137255 non-null  float64          17  Month
     1143383 non-null  float64          18  Year
     1173758 non-null  float64          19  Unnamed: 0
     30375 non-null    float64          20  ID
     30375 non-null    float64          21
     FROM_LOCATION         30375 non-null    object
22   TRANSACTION           30375 non-null    float64
23   Kit_Quantity          30375 non-null    float64
     24   Unit                 30375 non-null    object
     25   KITS                 28903 non-null    float64
     26   UPDATED_BY           30375 non-null    object
27   KIT_PART              30375 non-null    object
28   SDS_Quantity          30375 non-null    float64
     29   Conversion           30375 non-null    float64
 30  DivideBySG            30366 non-null
float64          31  Part_specific         9
non-null         float64         dtypes:
datetime64[ns](1), float64(19), object(12)
memory usage: 295.5+ MB

General information of dataframe:  None
Scenarios I:
```

```
In [8]: s1_cas = cas_total.drop_duplicates() s1_cas = s1_cas[['Unnamed:
        0', 'ID', 'FROM_LOCATION', 'TO_LOCATION', 'Part', 'TR
        'Kit_Quantity', 'Unit', 'KITS', 'UPDATED_BY', 'TRADE_NAME', 'KI
        'SPEC_MATERIAL_TYPE', 'SPEC_MATERIAL_CLASS', 'DOCUMENT_SPEC', '
                'Specific_gravity', 'Usage_gallons', 'Part_specific']]
s1_cas.shape
```

```
Out[8]: (1024753, 24)
```

```
In [9]: s1_gal = gallons_total.drop_duplicates()
         s1_gal.shape
```

Out[9]: (193907, 24)

```
In [10]: s1_trans_concat = pd.concat([s1_cas,
          s1_gal])
          s1_trans_concat.drop_duplicates()
          s1_trans_concat.shape
```

Out[10]:

```
print(f'scenario #1 shape is {s1_trans_concat.shape}')
print('Sum of nulls: \n', s1_trans_concat.isnull().sum(), '\n')
s1_trans_concat = s1_trans_concat[['TO_LOCATION', 'Part', 'TRANSACTION_DATE',
'T                                    'DOCUMENT_SPEC', 'Year',
'Specific_gravity', s1_trans_concat = s1_trans_concat.dropna() print(f'Sum of
nulls: {s1_trans_concat.isnull().sum()} \n scenario #1 final shap
```

(1218660,

24) In [11]:

```
scenario #1 shape is (1218660, 24)
Sum of nulls:
 Unnamed: 0              994378
ID                       994378
FROM_LOCATION            994378
TO_LOCATION                   0
Part                          0
TRANSACTION              994378
TRANSACTION_DATE              0
Kit_Quantity             994378
Unit                     994378
KITS                    1009545
UPDATED_BY               994378
TRADE_NAME                    0
KIT_PART                 994378
SDS_Quantity             994378
```

```
SDS_Number                   0
SPEC_MATERIAL_TYPE      660022
SPEC_MATERIAL_CLASS     690783
DOCUMENT_SPEC            32287
Year                         0
Conversion              994397
DivideBySG              994432
Specific_gravity          9236
Usage_gallons
478 Part_specific
1218625 dtype: int64

Sum of nulls: TO_LOCATION         0
Part                 0
TRANSACTION_DATE     0
TRADE_NAME           0
SDS_Number           0
DOCUMENT_SPEC        0
Year                 0
Specific_gravity     0
Usage_gallons        0
dtype: int64   scenario #1
final shape: (1177217, 9)
```

In [ ]:

Scenarios I[I]:

```
In [12]: s2_cas = cas_total[['Unnamed: 0', 'ID', 'FROM_LOCATION',
         'TO_LOCATION', 'Part',                  'Kit_Quantity',
         'Unit', 'KITS', 'UPDATED_BY', 'TRADE_NAME', 'KI
         'SPEC_MATERIAL_TYPE', 'SPEC_MATERIAL_CLASS', 'DOCUMENT_SPEC',
         '
                'Specific_gravity', 'Usage_gallons', 'Part_specific']]
```

```
In [13]: s2_gal = gallons_total[[]]
```

```
In [14]: s2_trans_concat = pd.concat([s2_cas,
          s2_gal])
          s2_trans_concat.drop_duplicates()
          s2_trans_concat.shape
```

Out[14]: (1387231, 24)

```
In [15]: print(f'scenario #1 shape is {s2_trans_concat.shape}')
          print('Sum of nulls: \n', s2_trans_concat.isnull().sum(),
          '\n') s2_trans_concat = s2_trans_concat[['TO_LOCATION',
          'Part', 'TRANSACTION_DATE', 'T
          'DOCUMENT_SPEC', 'Year', 'Specific_gravity',

s2_trans_concat = s2_trans_concat.dropna() print(f'Sum of nulls:
{s2_trans_concat.isnull().sum()} \n scenario #2 final shap
```

```
scenario #1 shape is (1387231, 24)
Sum of nulls:
 Unnamed: 0             1356856
ID                      1356856
FROM_LOCATION           1356856
TO_LOCATION              213473
Part                     213473
TRANSACTION             1356856
TRANSACTION_DATE         213473
Kit_Quantity            1356856
Unit                    1356856
KITS                    1358328
UPDATED_BY              1356856
TRADE_NAME               213473
KIT_PART                1356856
SDS_Quantity            1356856
SDS_Number               213473
SPEC_MATERIAL_TYPE       799940
SPEC_MATERIAL_CLASS      881625
DOCUMENT_SPEC            241858
Year                     213473
Conversion              1356856
DivideBySG              1356865
Specific_gravity         219118
```

```
Usage_gallons
213863 Part_specific
1387222 dtype: int64

Sum of nulls: TO_LOCATION         0

Part                0

TRANSACTION_DATE    0

TRADE_NAME          0

SDS_Number          0

DOCUMENT_SPEC       0

Year                0

Specific_gravity    0
Usage_gallons       0
dtype: int64   scenario #2
final shape: (1139776, 9)
```

In [16]:
```
s2_trans_concat.head(100)
```

Out[16]:

	TO_LOCATION	Part	TRANSACTION_DATE	TRADE_NAME	SDS_Number	DOCUMENT_SP
0	531-71 MMS50	RM016244	2022-04-28	HEXPLY M73 PREPREG	129805.0	
1	531-71 MMS50	RM016244	2022-04-28	HEXPLY M73 PREPREG	129805.0	
2	531-71 MMS50	RM016244	2022-04-28	HEXPLY M73 PREPREG	129805.0	
3	531-71 MMS50	RM016244	2022-04-28	HEXPLY M73 PREPREG	129805.0	
4	531-71 MMS50	RM016244	2022-04-28	HEXPLY M73 PREPREG	129805.0	
...	...	...	...	...	...	
95	531-71 MMS3	RM016255	2022-03-03	FM 300-1 FM 300 AND ADHESIVE	100293.0	

				FILMS	
				FM 300 AND	
96 MMS3	531-71 RM016255	2022-03-03	FM 300-1		100293.0
				ADHESIVE FILMS	
				FM 300 AND	
97 MMS3	531-71 RM016255	2022-03-03	FM 300-1		100293.0

A

DHESIVEFILMSTO_LOCATION Part TRANSACTION_DATE TRADE_

				NAME	SDS_Number	DOCUMENT_SP
98	531-71	RM016255	2022-03-03	FM 300 AND FM 300-1 ADHESIVE FILMS	100293.0	MMS3
99	531-71	RM016255	2022-03-03	FM 300 AND FM 300-1 ADHESIVE FILMS	100293.0	MMS3

100 rows × 9 columns

Scenarios II[I]:

In [17]:
```
s3_cas = cas_total[['Unnamed: 0', 'ID', 'FROM_LOCATION', 'TO_LOCATION',
'Part',                  'Kit_Quantity', 'Unit', 'KITS', 'UPDATED_BY',
'TRADE_NAME', 'KI                    'SPEC_MATERIAL_TYPE',
'SPEC_MATERIAL_CLASS', 'DOCUMENT_SPEC', '
                  'Specific_gravity', 'Usage_gallons', 'Part_specific']]

s3_cas = s3_cas[['TO_LOCATION', 'Part', 'TRANSACTION_DATE', 'TRADE_NAME',
'SDS_N                                    'DOCUMENT_SPEC', 'Year',
'Specific_gravity',

s3_cas.dropna()
```

Out[17]:
```
s3_cas.shape
```

```
print(s1_trans_concat.isnull().sum())
print('-------------------------------------------------')
print(s2_trans_concat.isnull().sum())
print('-------------------------------------------------')
print(s3_cas.isnull().sum())
print('-------------------------------------------------')
```

```
(1173758, 9)
```

In [18]:

```
TO_LOCATION         0
Part                0
TRANSACTION_DATE    0
TRADE_NAME          0
SDS_Number          0
DOCUMENT_SPEC       0
Year                0
Specific_gravity
0 Usage_gallons
0 dtype: int64
-------------------------------------------------
TO_LOCATION         0
Part                0
TRANSACTION_DATE    0
TRADE_NAME          0
```

```
SDS_Number          0

DOCUMENT_SPEC       0

Year                0

Specific_gravity
0 Usage_gallons
0 dtype: int64

------------------------------------------------

TO_LOCATION             0

Part                    0

TRANSACTION_DATE        0

TRADE_NAME              0

SDS_Number              0

DOCUMENT_SPEC       28385

Year                    0

Specific_gravity
5645 Usage_gallons
390 dtype: int64

------------------------------------------------
```

In [ ]:

In [19]:
```
print(s1_trans_concat.head())
 print(s1_trans_concat.tail())
```

```
  TO_LOCATION      Part TRANSACTION_DATE          TRADE_NAME
SDS_Number  \

  0      531-71  RM016244       2022-04-28  HEXPLY M73 PREPREG
129805.0    1      531-71  RM016244       2022-04-28  HEXPLY M73
 PREPREG    129805.0    2      531-71  RM016244       2022-04-28
   HEXPLY M73 PREPREG    129805.0    3      531-71  RM016244
           2022-04-28  HEXPLY M73 PREPREG    129805.0
4      531-71  RM016244       2022-04-28  HEXPLY M73 PREPREG
129805.0

  DOCUMENT_SPEC    Year  Specific_gravity  Usage_gallons

0       MMS5025  2022.0               1.4
10.495889   1       MMS5025  2022.0
1.4      10.495889   2       MMS5025  2022.0
1.4      10.495889   3       MMS5025  2022.0
1.4      10.495889   4       MMS5025  2022.0
1.4      10.495889
```

```
      TO_LOCATION      Part TRANSACTION_DATE             TRADE_NAME
\

24695      520-62  RM021279       2016-06-07  TIOLUBE 75/75 LOW
VOC    24707      560-31  RM021372       2016-12-20      MOBIL
JET OIL 254    24708      560-31  RM021372       2016-10-20
MOBIL JET OIL 254    24709      560-31  RM021372       2016-08-
31      MOBIL JET OIL 254    24710      560-31  RM021372
2016-08-31      MOBIL JET OIL 254

       SDS_Number    DOCUMENT_SPEC    Year  Specific_gravity
Usage_gallons

24695    107024.0       MIL-L-46147  2016.0               1.34
0.25   24707    36553.0  HMS20-1267/2441  2016.0            0.99
10.50   24708    36553.0  HMS20-1267/2441  2016.0            0.99
12.00   24709    36553.0  HMS20-1267/2441  2016.0            0.99
2.25
24710     36553.0  HMS20-1267/2441  2016.0              0.99
3.75
```

```
In [20]: pd.options.display.max_rows =
          999
          pd.set_option("display.max_colu
          mns", None)

print(s2_trans_concat.head(10))
print(s2_trans_concat.tail(10))
```

```
  TO_LOCATION      Part TRANSACTION_DATE          TRADE_NAME
SDS_Number  \

  0      531-71  RM016244       2022-04-28  HEXPLY M73 PREPREG
129805.0    1      531-71  RM016244       2022-04-28  HEXPLY M73
PREPREG    129805.0    2      531-71  RM016244       2022-04-28
   HEXPLY M73 PREPREG    129805.0    3      531-71  RM016244
  2022-04-28  HEXPLY M73 PREPREG    129805.0    4      531-71
 RM016244       2022-04-28  HEXPLY M73 PREPREG    129805.0    5
     531-71  RM016244       2022-04-28  HEXPLY M73 PREPREG
129805.0    6      531-71  RM016244       2022-04-28  HEXPLY M73
PREPREG    129805.0    7      531-71  RM016244       2022-04-28
   HEXPLY M73 PREPREG    129805.0    8      531-71  RM016244
          2022-04-28  HEXPLY M73 PREPREG    129805.0
9      531-71  RM016244       2022-04-28  HEXPLY M73 PREPREG
129805.0

  DOCUMENT_SPEC    Year  Specific_gravity  Usage_gallons

0       MMS5025  2022.0               1.4
10.495889   1       MMS5025  2022.0
1.4      10.495889   2       MMS5025  2022.0
1.4      10.495889   3       MMS5025  2022.0
1.4      10.495889   4       MMS5025  2022.0
1.4      10.495889   5       MMS5025  2022.0
1.4      10.495889   6       MMS5025  2022.0
```

```
1.4      10.495889   7       MMS5025  2022.0
1.4      10.495889   8       MMS5025  2022.0
1.4      10.495889   9       MMS5025  2022.0
1.4      10.495889

      TO_LOCATION      Part TRANSACTION_DATE
TRADE_NAME  \

24662      531-107   RM021205         2016-05-27   44GN011CAT
CURING SOLUTION    24664       520-15  RM021207        2016-
01-19    1544  ROSIN  SOLDERING  FLUX      24677       543-ALL
RM021258       2016-08-31                    6050-11    24693
520-10  RM021276       2016-04-13       EVERLUBE 620C DILUTED
24694       520-62  RM021279        2016-08-15        TIOLUBE
75/75 LOW VOC    24695       520-62  RM021279        2016-06-
07       TIOLUBE 75/75 LOW VOC    24707      560-31  RM021372
2016-12-20            MOBIL JET OIL 254    24708       560-31
RM021372       2016-10-20           MOBIL JET OIL 254    24709
560-31  RM021372       2016-08-31           MOBIL JET OIL 254
24710      560-31  RM021372       2016-08-31           MOBIL
JET OIL 254

       SDS_Number    DOCUMENT_SPEC    Year  Specific_gravity
Usage_gallons

24662    149653.0          BMS10-11  2016.0            1.88000
0.019531    24664         4214.0               NO_SPEC   2016.0
0.93000        0.015625    24677      51577.0            STM0905
2016.0             1.30000        0.250000    24693      46803.0
MIL-L-46010  2016.0             0.90071        1.000000    24694
107024.0      MIL-L-46147  2016.0          1.34000       0.250000
24695    107024.0       MIL-L-46147  2016.0             1.34000
0.250000      24707         36553.0    HMS20-1267/2441    2016.0
0.99000        10.500000    24708       36553.0   HMS20-1267/2441
2016.0             0.99000       12.000000    24709      36553.0
HMS20-1267/2441  2016.0           0.99000       2.250000
24710     36553.0  HMS20-1267/2441  2016.0           0.99000
3.750000
```

In [21]:

```
pd.options.display.max_rows = 999
pd.set_option("display.max_columns", None)

print(s3_cas.head(10))
print(s3_cas.tail(10))
```

```
  TO_LOCATION      Part TRANSACTION_DATE          TRADE_NAME
SDS_Number  \

  0      531-71  RM016244       2022-04-28  HEXPLY M73 PREPREG
129805.0    1      531-71  RM016244       2022-04-28  HEXPLY M73
 PREPREG    129805.0    2      531-71  RM016244       2022-04-28
   HEXPLY M73 PREPREG    129805.0    3      531-71  RM016244
  2022-04-28  HEXPLY M73 PREPREG    129805.0    4      531-71
 RM016244       2022-04-28  HEXPLY M73 PREPREG    129805.0    5
     531-71  RM016244       2022-04-28  HEXPLY M73 PREPREG
```

```
129805.0    6      531-71  RM016244       2022-04-28  HEXPLY M73
 PREPREG    129805.0    7      531-71  RM016244       2022-04-28
   HEXPLY M73 PREPREG    129805.0    8      531-71  RM016244
           2022-04-28  HEXPLY M73 PREPREG    129805.0
9      531-71  RM016244       2022-04-28  HEXPLY M73 PREPREG
129805.0
```

```
  DOCUMENT_SPEC    Year  Specific_gravity  Usage_gallons
```

```
0       MMS5025  2022.0              1.4
10.495889   1       MMS5025  2022.0
1.4      10.495889   2       MMS5025  2022.0
1.4      10.495889   3       MMS5025  2022.0
1.4      10.495889   4       MMS5025  2022.0
1.4      10.495889   5       MMS5025  2022.0
1.4      10.495889   6       MMS5025  2022.0
1.4      10.495889   7       MMS5025  2022.0
1.4      10.495889   8       MMS5025  2022.0
1.4      10.495889   9       MMS5025  2022.0
1.4      10.495889
```

```
      TO_LOCATION          Part TRANSACTION_DATE
TRADE_NAME  \
```

```
24758  531-MPSLAB  EPIKURE 3274      2016-11-30  EPIKURE CURING
AGENT 3274    0         520-192        NP1578       2016-12-21
DUSTING GAS/FREEZE SPRAY    1          520-192         NP1578
2016-12-12    DUSTING GAS/FREEZE SPRAY    2           520-192
NP1578      2016-09-19   DUSTING GAS/FREEZE SPRAY    3     531-
MPSLAB       NP1578      2016-08-30   DUSTING GAS/FREEZE SPRAY
4           531-65         NP1578        2016-07-05    DUSTING
GAS/FREEZE SPRAY    5         520-192        NP1578       2016-
06-24   DUSTING GAS/FREEZE SPRAY    6      560-TC500       NP1578
2016-06-23    DUSTING GAS/FREEZE SPRAY    7         560-TC500
NP1578       2016-06-22   DUSTING GAS/FREEZE SPRAY
8       560-TC500        NP1578       2016-04-22   DUSTING
GAS/FREEZE SPRAY
```

```
       SDS_Number DOCUMENT_SPEC    Year  Specific_gravity
Usage_gallons
```

```
24758    136221.0          None  2016.0               1.0
0.015625    0            83377.0             None   2016.0
1.0       2.940000   1         83377.0          None  2016.0
1.0       2.940000   2         83377.0          None  2016.0
1.0       3.920000   3         83377.0          None  2016.0
1.0      11.760000   4         83377.0          None  2016.0
1.0       4.900000   5         83377.0          None  2016.0
1.0       1.960000   6         83377.0          None  2016.0
1.0       2.940000   7         83377.0          None  2016.0
1.0       1.960000
8         83377.0          None  2016.0              1.0
4.900000
```

In [ ]:

```
In [22]: csv_s1_trans_concat =
s1_trans_concat.to_csv('C:/Users/sohra/Desktop/Untitled Fo
csv_s2_trans_concat =
s2_trans_concat.to_csv('C:/Users/sohra/Desktop/Untitled Fo
csv_s3_trans_concat = s3_cas.to_csv('C:/Users/sohra/Desktop/Untitled
Folder/csv_
```

In [ ]:

Read excel files

```
In [23]: org_level = read_excel('C:/Users/sohra/Desktop/Untitled
```

```
accounting_departments = read_excel('C:/Users/sohra/Desktop/Untitled
Folder/Copy new_column_names = ['Row', '', 'Empl Stat Cd', 'Acctg Bus Unit
Cd', 'Acctg Locat
'Business Unit Nm', 'Program']

accounting_departments = accounting_departments.iloc[3:]
accounting_departments.columns = new_column_names
accounting_departments
```

```
Folder/MasterOrgFile_Org In [24]:
```

```
In [25]:

Mesa_Square_Footage = read_excel('C:/Users/sohra/Desktop/Untitled
Folder/2021.5
T
Mesa_Square_Footage.drop(index=Mesa_Square_Footage.index[0:3], axis=0,
inplace=1
new_column_names = ['Region', 'Site', 'Property', 'Property Name', 'Bldg #',d
'B 'Bldg Address 1', 'Bldg Address 2', 'City',e
'State','Zip Co 'County', 'Occupancy Pool', 'Occupancyc
Pool Name', 'Floor',o
'Floor Area', 'Floor Area Name', 'Floor Area Type', 'LongO
T 'Subloc/Room (a)', 'Subloc/Room Name', 'Space Type',i
'Space 'Class (b)', 'Class Name', 'Class Type', 'Occ/Vacr
(c)', 'A 'Acctg Org Name', 'Subloc / Room SqFt', 'Seci
Circ SqFt', 'T 'Subloc/Room Attributes','BEMSID-Name',
'BEMSID', 'Name', ' 'OvHd Alloc Name', 'Pool Accounting',
'Division Code', 'Div 'Subdivision Code', 'Subdivision
Name', 'Resource Subcatego 'Labor Code', 'Pool-OBA',
'Finance Site Code', 'Finance Sub 'Work Order','Loan
Pool','SubPool', 'Business Unit', 'Locat
```

```
be_SDS_Chemicals = read_excel('C:/Users/sohra/Desktop/Untitled
Folder/be_SDS_Che
```

```
 Mesa_Square_Footage.columns = new_column_names
 Mesa_Square_Footage
```

In [26]:

```
Mesa_POU_Locations = read_excel('C:/Users/sohra/Desktop/Untitled
Folder/2022.7.2
```

In [27]:

```
Manager_Mesa_Site_exportedResults = read_csv('C:/Users/sohra/Desktop/Untitled
Fo
```

In [28]:

In [ ]:

Basic EDA

In [2]:

```
import numpy as np
import pandas as pd
import datetime as dt
import matplotlib.pyplot as plt



from scipy import stats from matplotlib import
rcParams from sklearn.preprocessing import
scale from statsmodels.tsa.arima.model import
ARIMA from sklearn.linear_model import
LinearRegression
```

In [ ]:

```
plt.rcParams["figure.figsize"] = 10, 6      # Set figure size
```

In [3]:
```
# read data and assign variable names

df = pd.read_csv('df.csv')

df = df[['TO_LOCATION', 'Part', 'TRANSACTION_DATE', 'TRADE_NAME',
'SDS_Number',
        'DOCUMENT_SPEC', 'Year', 'Specific_gravity', 'Usage_gallons']]

print(f'Data frame shape: {df.shape}')

df.head()
```

```
Data frame shape: (1177217, 9)
```

Out[3]:

	TO_LOCATION	Part	TRANSACTION_DATE	TRADE_NAME	SDS_Number DOCUMENT_SPE
0	531-71	RM016244	2022-04-28	HEXPLY M73	129805.0 MMS502 PREPREG
1	531-71	RM016244	2022-04-28	HEXPLY M73	129805.0 MMS502 PREPREG
2	531-71	RM016244	2022-04-28	HEXPLY M73	129805.0 MMS502 PREPREG
3	531-71	RM016244	2022-04-28	HEXPLY M73	129805.0 MMS502 PREPREG
4	531-71	RM016244	2022-04-28	HEXPLY M73	129805.0 MMS502 PREPREG

```
df['TRANSACTION_DATE'] =
pd.to_datetime(df['TRANSACTION_DATE']) df['year'] =
df['TRANSACTION_DATE'].dt.year df['month'] =
df['TRANSACTION_DATE'].dt.month df['day'] =
df['TRANSACTION_DATE'].dt.day df.head()
```

Basic EDA on raw data

In [16]:

Out[16]:

	TO_LOCATION	Part	TRANSACTION_DATE	TRADE_NAME	SDS_Number DOCUMENT_SPE
0	531-71	RM016244	2022-04-28	HEXPLY M73 PREPREG	129805.0 MMS502

TO_LOCATION Part TRANSACTION_DATE TRADE_NAME SDS_Number DOCUMENT_SPE

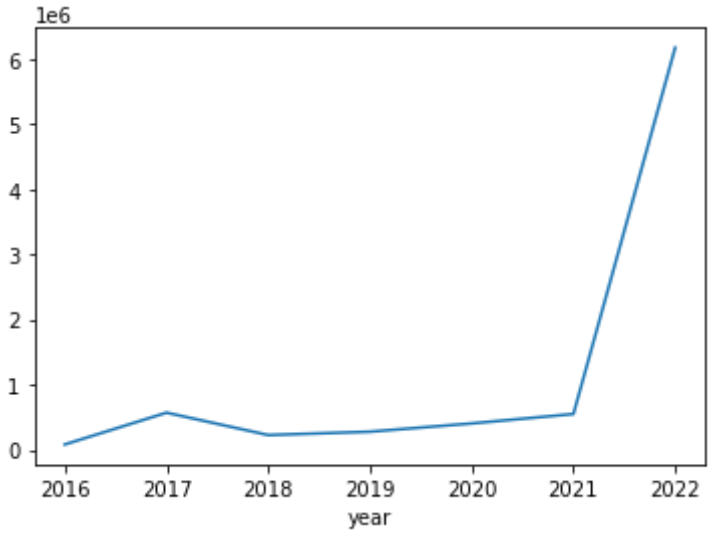
1	531-71	RM016244	2022-04-28	HEXPLY M73 PREPREG	129805.0 MMS502
2	531-71	RM016244	2022-04-28	HEXPLY M73 PREPREG	129805.0 MMS502
3	531-71	RM016244	2022-04-28	HEXPLY M73 PREPREG	129805.0 MMS502

531-71 RM016244 2022-04-28 HEXPLY M73 PREPREG 129805.0 MMS502

In [ ]:
```
df.to_csv('arima.csv')
```

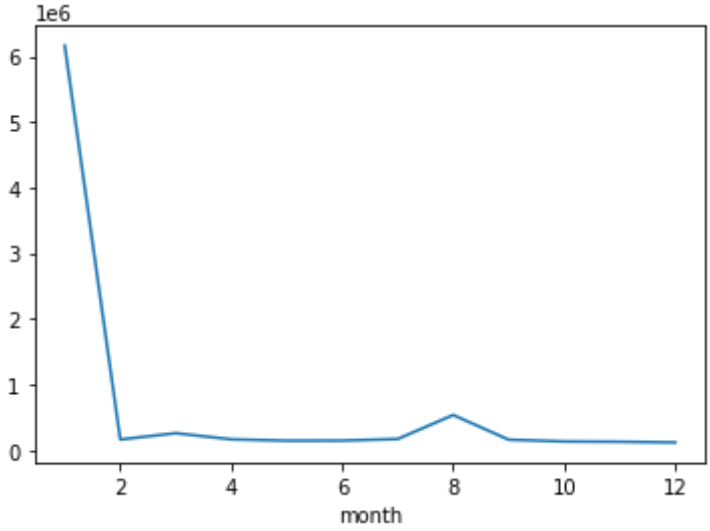
In [22]:
```
df_use_year = df.groupby(['year'])['Usage_gallons'].sum()
df_use_year.plot()
```

Out[22]:
```
<AxesSubplot:xlabel='year'>
```



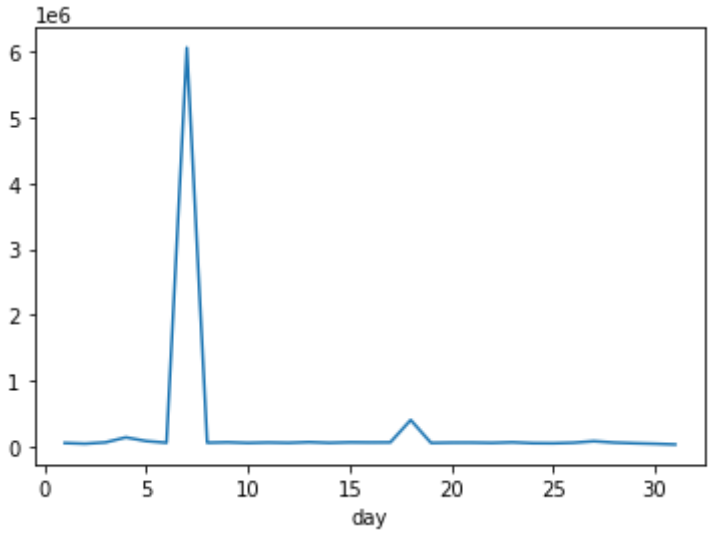

```
In [23]: df_use_month = df.groupby(['month'])['Usage_gallons'].sum()
         df_use_month.plot()
```

```
Out[23]: <AxesSubplot:xlabel='month'>
```



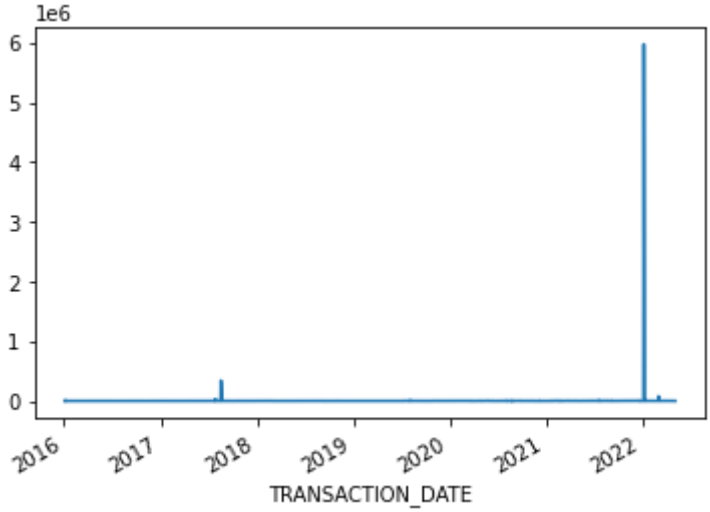

```
In [24]: df_use_month = df.groupby(['day'])['Usage_gallons'].sum()
         df_use_month.plot()
```

```
Out[24]: <AxesSubplot:xlabel='day'>
```




```
In [27]: df_use_date =
          df.groupby(['TRANSACTION_DATE'])['Usage_gallons'].sum()
          df_use_date.plot()
```

Out[27]: <AxesSubplot:xlabel='TRANSACTION_DATE'>




```
In [28]: SDS_use_total = df.groupby(['SDS_Number']).sum()
         SDS_use_total
```

Out[28]:

SDS_Number	Year	Specific_gravity	Usage_gallons	year	month	day
169.0	217840.0	94.1004	2.000000	217840	522	2368
305.0	54567.0	31.3200	58.264218	54567	135	624

490.0	5125801.0	2208.9300	427.500000	5125801	16842	43548
525.0	3954157.0	1918.7000	164.265625	3954157	11752	29171
587.0	700756.0	329.6500	1865.000000	700756	2418	6478
...	...	...	...	...	...	...

306766.0 12126.0 6.0000 0.140625 12126 18 75 306947.0 691304.0 342.0000

22.266388 691304 2048 5347 306951.0 84840.0 42.0000 2.625000 84840 336 686

307083.0 303234.0 465.0000 3.843750 303234 624 2904 307192.0 816628.0 492.8800 9.595886

816628 2420 6315

739 rows × 6 columns

```
In [32]: SDS_use_total = df.groupby(['SDS_Number'])['Usage_gallons']
          SDS_use_total.head()
```

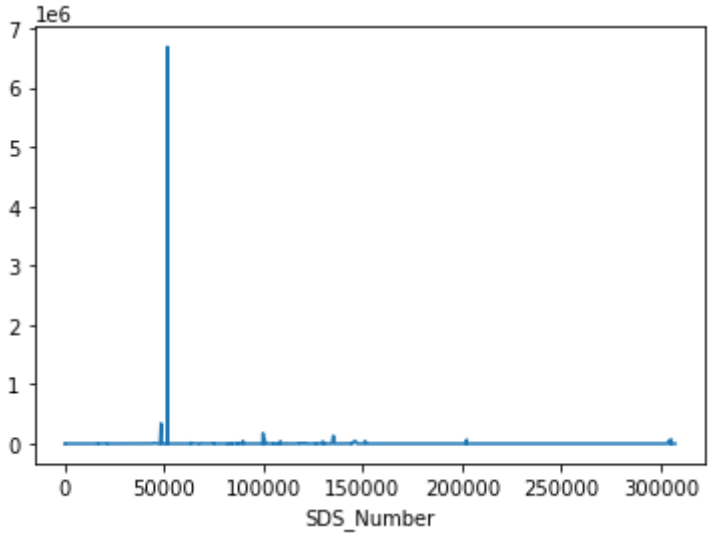
```
Out[32]: 0
         10.495889 1
         10.495889 2
         10.495889 3
         10.495889 4
         10.495889
         ...
         1171306
         1.000000 1175649
         0.039062 1175650
         0.039062 1176727
         0.125000 1176728
         0.125000
         Name: Usage_gallons, Length: 3645, dtype: float64
```

```
In [38]: SDS_use_total =
          df.groupby(['SDS_Number'])['Usage_gallons'].sum()
          print('Most used SDS_Number: ',
          SDS_use_total.idxmax(), '\n')
          print(SDS_use_total.head()) SDS_use_total.plot()
```

```
         Most used SDS_Number:  51787.0

         SDS_Number
         169.0
         2.000000
         305.0
         58.264218
         490.0
         427.50000
         0   525.0
         164.26562
         5   587.0
         1865.0000
         00
```

```
Name: Usage_gallons, dtype: float64
```

Out[38]: <AxesSubplot:xlabel='SDS_Number'>




```
In [40]: Name_use_total = df.groupby(['TRADE_NAME'])['Usage_gallons']
          Name_use_total.head()
```

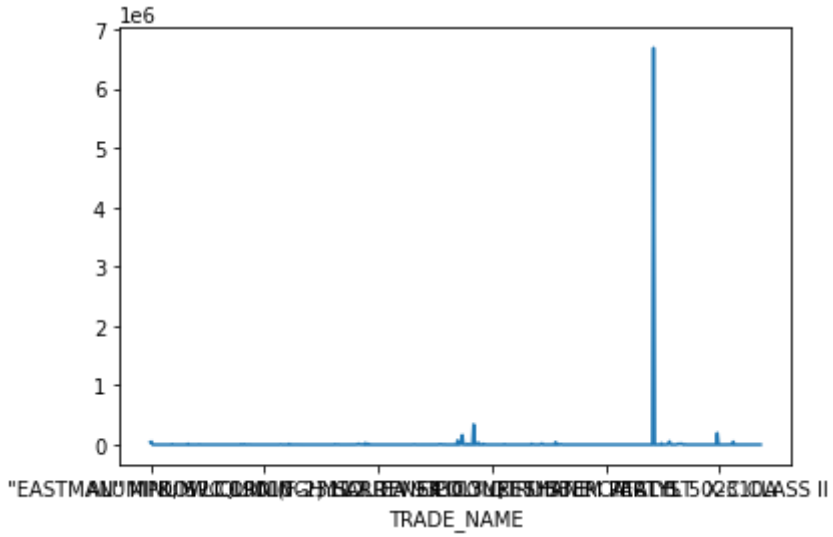
```
Out[40]: 0
         10.495889 1
         10.495889 2
         10.495889 3
         10.495889 4
         10.495889
         ...
         1176728
         0.125000
         1177142
         0.062500
         1177168
         0.000528
         1177169
         0.001057
         1177209
         0.250000
         Name: Usage_gallons, Length: 2655, dtype: float64
```

```
In [41]: Name_use_total =
          df.groupby(['TRADE_NAME'])['Usage_gallons'].sum()
          print('Most used chemical: ',
          Name_use_total.idxmax(), '\n')
          print(Name_use_total.head()) Name_use_total.plot()
```

```
Most used chemical:  R 913 PREPREG

TRADE_NAME
```

```
"EASTMAN"        MPK;        SPC        19010
41717.562500 "SUVA" 134A ; HFC 134A ; VT1505
; SUVA        49435.704417 #86 SCOTCH BRAND
POLYURETHANE  PROTECTIVE           15.875000
020X456                              THINNER
1671.000000   02Y040ACAT   CURING   SOLUTION
275.992188   Name:   Usage_gallons,   dtype:
float64
```

Out[41]: <AxesSubplot:xlabel='TRADE_NAME'>



In [42]:
```
Part_use_total = df.groupby(['Part'])['Usage_gallons']
Part_use_total.head()
```

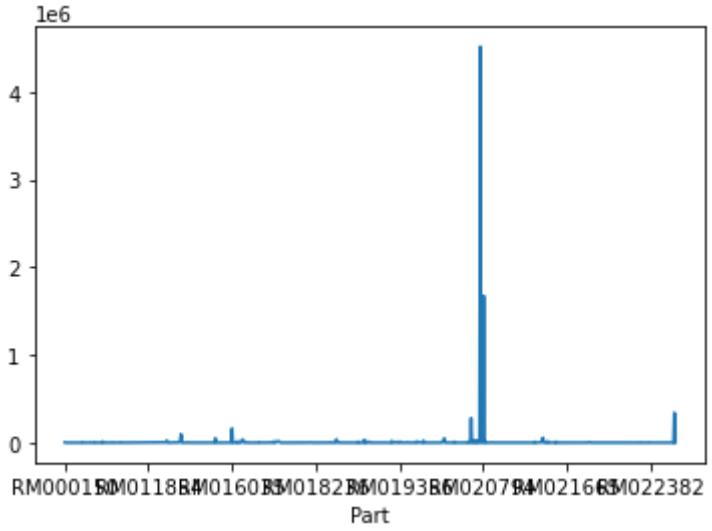
Out[42]:
```
0
10.495889 1
10.495889 2
10.495889 3
10.495889 4
10.495889
...
1177142
0.062500
1177168
0.000528
1177169
0.001057
1177170
0.062500
1177209
0.250000
Name: Usage_gallons, Length: 3602, dtype: float64
```

In [43]:
```
Part_use_total =
 df.groupby(['Part'])['Usage_gallons'].sum()
 print('Most used part: ',
```

```
Part_use_total.idxmax(), '\n')
print(Part_use_total.head())
Part_use_total.plot()
```

```
Most used part:  RM020789

Part
RM000150    3608.187500
RM000422       3.000000
RM000568      11.953125
RM000688      24.179688
RM001150      86.140625
Name: Usage_gallons, dtype: float64
```

Out[43]: <AxesSubplot:xlabel='Part'>



In [44]:
```
Location_use_total =
 df.groupby(['TO_LOCATION'])['Usage_gallons']
 Location_use_total.head()
```

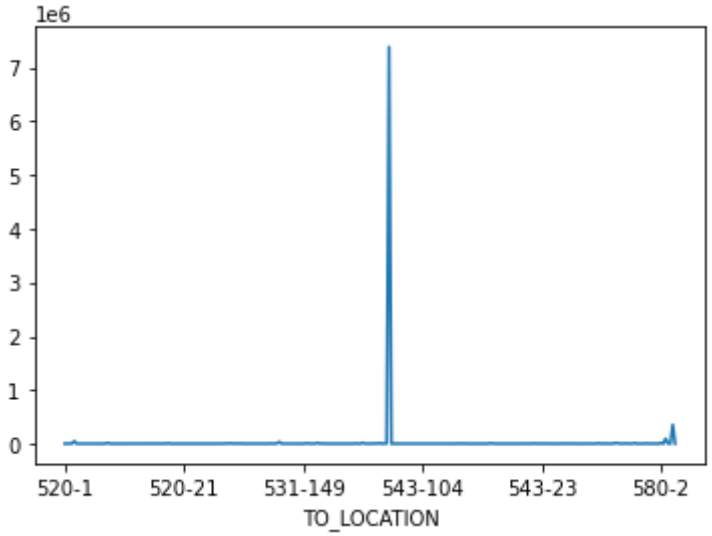
Out[44]:
```
0
10.495889 1
10.495889 2
10.495889 3
10.495889 4
10.495889
...
1076452
0.125000
1086950
1.000000
1162708
30.000000
1162717
25.000000
1177209
0.250000
```

```
Name: Usage_gallons, Length: 1275, dtype: float64
```

```
In [45]: Location_use_total =
          df.groupby(['TO_LOCATION'])['Usage_gallons'].sum(
          ) print('Most used part: ',
          Location_use_total.idxmax(), '\n')
print(Location_use_total.head()) Location_use_total.plot()
```

```
Most used part:  531-71

TO_LOCATION
520-1
298.561784 520-10
1225.764760 520-
101
1626.241163 520-
104
1954.341773 520-
105
47461.718019
Name: Usage_gallons, dtype: float64
```

Out[45]: <AxesSubplot:xlabel='TO_LOCATION'>



ARIMA

In [1]:
```
import pandas as pd
import datetime as dt
import matplotlib.pyplot as plt

from matplotlib import rcParams
from pandas.tseries.offsets import DateOffset from
statsmodels.tsa.arima.model import ARIMA from
statsmodels.tsa.stattools import adfuller from
statsmodels.tsa.statespace.sarimax import SARIMAX from
statsmodels.graphics.tsaplots import plot_acf, plot_pacf
```

In [2]:
```
import statsmodels.tsa.arima.model as  dd
```

In [3]:
```
plt.rcParams["figure.figsize"] = 10, 6
```

In [4]:
```
df = pd.read_csv('arima.csv')     # read the final file df =
 df[['TRANSACTION_DATE', 'Usage_gallons']]     # Choose neede coumns
 for pred print(f'Data frame shape: {df.shape}') print(df.head())
```

```
Data frame shape: (1177158, 2)

  TRANSACTION_DATE  Usage_gallons

0       2022-04-28
10.495889 1       2022-
04-28      10.495889 2
2022-04-28
10.495889 3       2022-
04-28      10.495889 4
2022-04-28
10.495889
```

In [5]:
```
df['TRANSACTION_DATE'] = pd.to_datetime(df['TRANSACTION_DATE'])
 # set tran df = df[df['TRANSACTION_DATE'] < '2022-05-01']        #
 filter last month  print(f'Data frame shape: {df.shape}')
 print(df.head()) print(df.tail())
```

```
Data frame shape: (1174806, 2)

  TRANSACTION_DATE  Usage_gallons

0       2022-04-28
10.495889 1       2022-
04-28      10.495889 2
2022-04-28
10.495889 3       2022-
04-28      10.495889 4
2022-04-28
10.495889

        TRANSACTION_DATE  Usage_gallons

1177153           2016-06-07
0.25 1177154        2016-12-
```

```
20              10.50  1177155
2016-10-20              12.00
1177156           2016-08-31
2.25 1177157        2016-08-
31           3.75
```

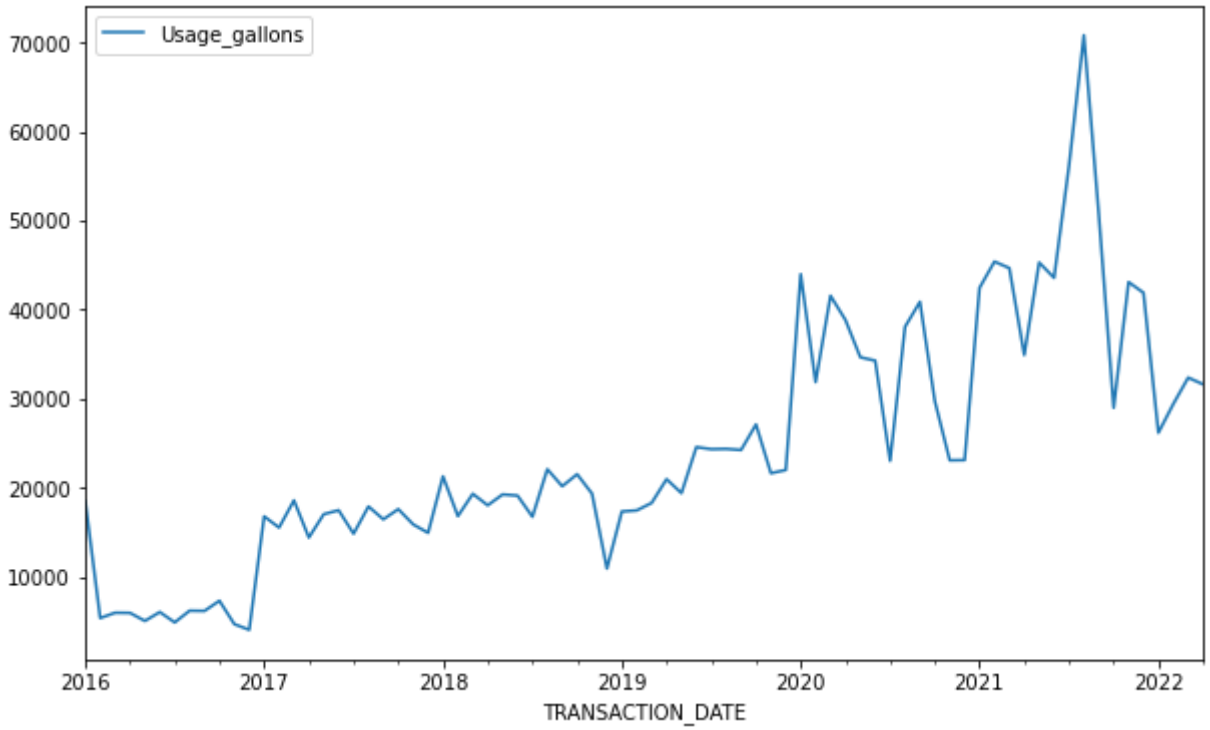
```
In [6]: # group by month and calculate total cunsumptions per month arimaz =
        df.groupby(pd.Grouper(key='TRANSACTION_DATE', axis=0,
        freq='M')).sum() print(f'Data frame shape: {arimaz.shape}')
        print(arimaz.head())
```

```
Data frame shape: (76, 1)

                  Usage_gallons

TRANSACTION_DATE

2016-01-31
18467.922976   2016-
02-29
5349.208348  2016-03-
31
5948.213152  2016-04-
30
5930.484263
2016-05-31          5028.455578
```

```
In [7]: print(arimaz.head())
        print(arimaz.tail())
        arimaz.shape
```

```
                  Usage_gallons

TRANSACTION_DATE

2016-01-31
18467.922976   2016-
02-29
5349.208348  2016-03-
31
5948.213152  2016-04-
30
5930.484263  2016-05-
31
5028.455578
Usage_gallons
TRANSACTION_DATE

2021-12-31
41890.167389 2022-01-31
26171.658883 2022-02-28
29400.786396 2022-03-31
32361.026412

2022-04-30
31609.526980 Out[7]: (76, 1)
```

```
In [8]: type(arimaz)

Out[8]: pandas.core.frame.DataFrame

In [9]: arimaz.plot()

Out[9]: <AxesSubplot:xlabel='TRANSACTION_DATE'>
```




```
In [10]: result=adfuller(arimaz['Usage_gallons']) #to help you, we added the
          names of every value dict(zip(['adf', 'pvalue', 'usedlag', 'nobs',
          'critical' 'values', 'icbest'],res

Out[10]: {'adf': -1.6736608458992983,
          'pvalue': 0.4447317279527293,
          'usedlag': 2,
          'nobs': 73,
          'criticalvalues': {'1%': -
         3.5232835753964475,   '5%': -
         2.902030597326081,
           '10%': -2.5883710883843123},
          'icbest': 1303.8129598480937}
```

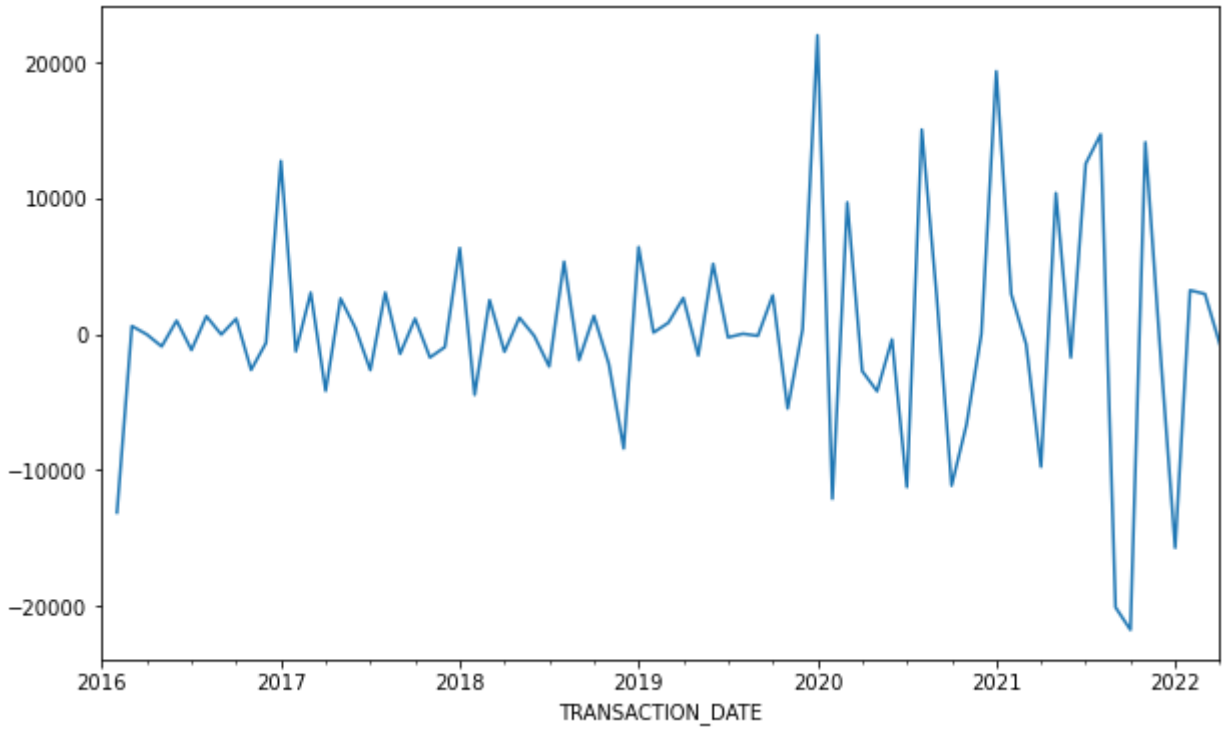
Since p-value is greater than 0.05, As expected we failed to reject the Null Hypothesis and the series has a unit root thus is not stationary.

Transform Non-Stationary to Stationary using Differencing (the d and D parameters) The next step is to transform our data to Stationary so we will have an estimate for d and D parameters we will use in the model. This can be done using Differencing and it's performed by subtracting the previous observation from the current observation.

$difference(T) = observation(T) - observation(T-1)$ Then, we will test it again for stationarity using the Augmented Dickey-Fuller test and if it's stationary we will proceed to our next step. If not we will apply differencing again till we have a stationary series. Differencing can be done very easily with pandas using the shift function.

In [11]:
```
arimaz['1difference']=arimaz['Usage_gallons']-
arimaz['Usage_gallons'].shift(1) arimaz['1difference'].plot()
```

Out[11]: <AxesSubplot:xlabel='TRANSACTION_DATE'>



In [12]:
```
result=adfuller((arimaz['1difference']).dropna())
#to help you, we added the names of every value
dict(zip(['adf', 'pvalue', 'usedlag', 'nobs', 'critical' 'values',
'icbest'],res
```

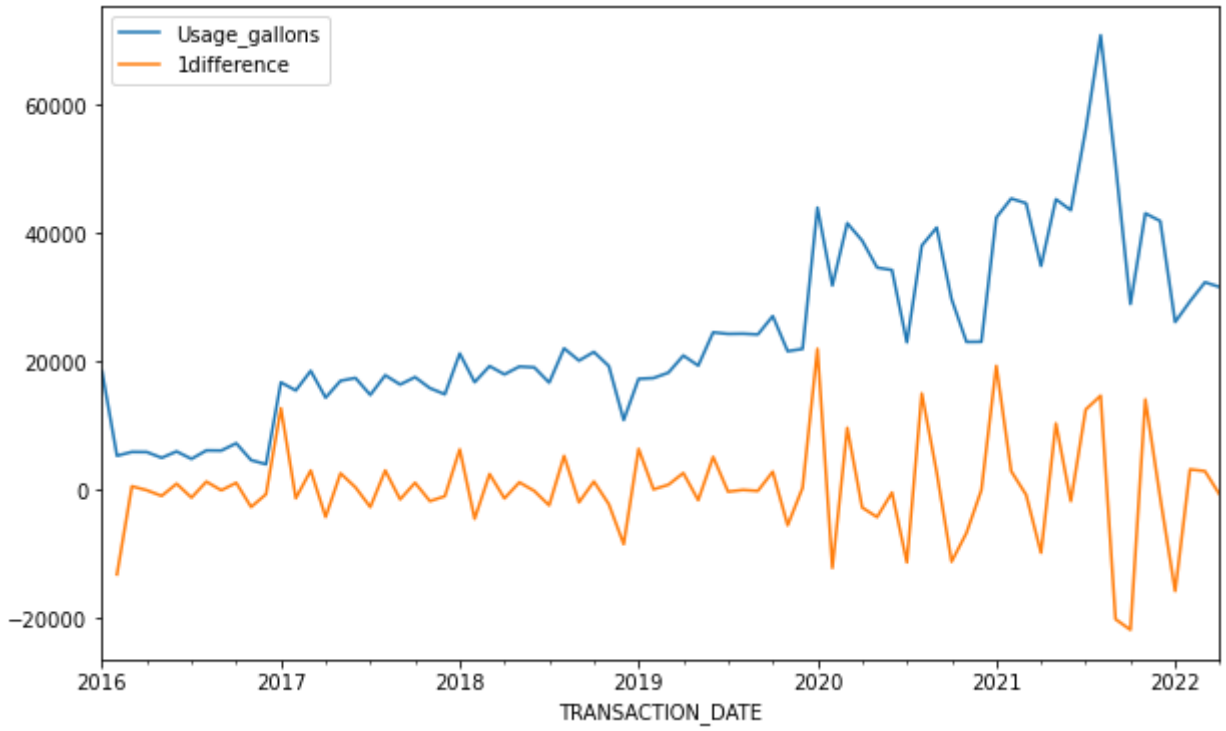
Out[12]:

```
{'adf': -4.193589166184938,

 'pvalue': 0.0006755592722814878,

 'usedlag': 12,

 'nobs': 62,

 'criticalvalues': {'1%': -3.540522678829176,

  '5%': -2.9094272025108254,

  '10%': -2.5923136524453696},

 'icbest':
```

```
arimaz.plot()
```

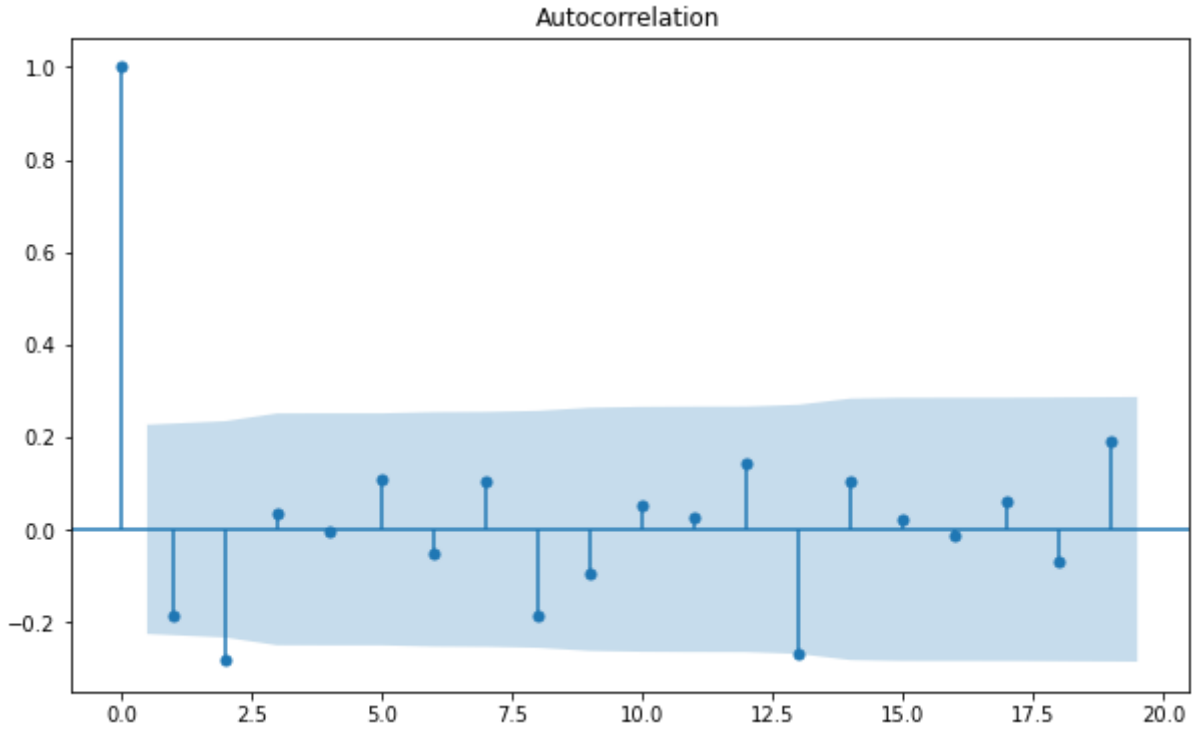
```
1281.6118533940491} In [13]:
```

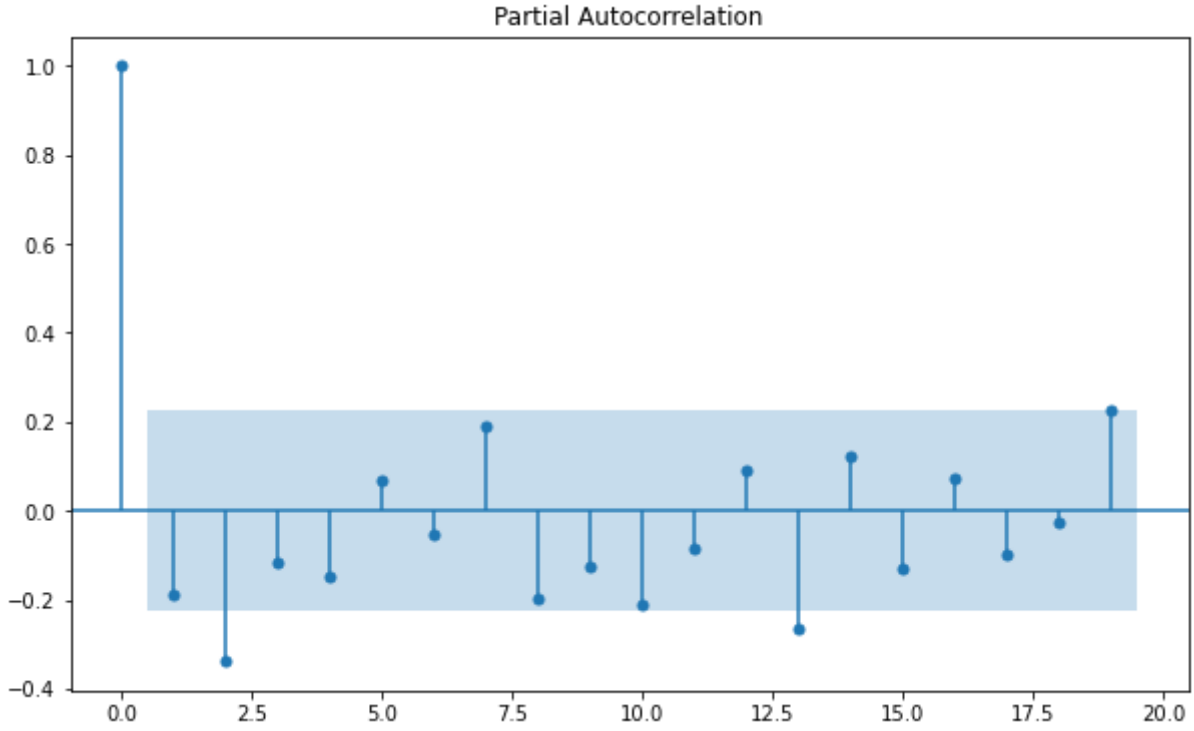
Out[13]: <AxesSubplot:xlabel='TRANSACTION_DATE'>



The p-value is less than 0.05 so we can reject the null hypothesis. That means the first difference is stationary and that suggests that a good estimate for the value d is 1.

In [14]:
```
fig1=plot_acf(arimaz['1difference'].dropna())
fig2=plot_pacf(arimaz['1difference'].dropna())
```

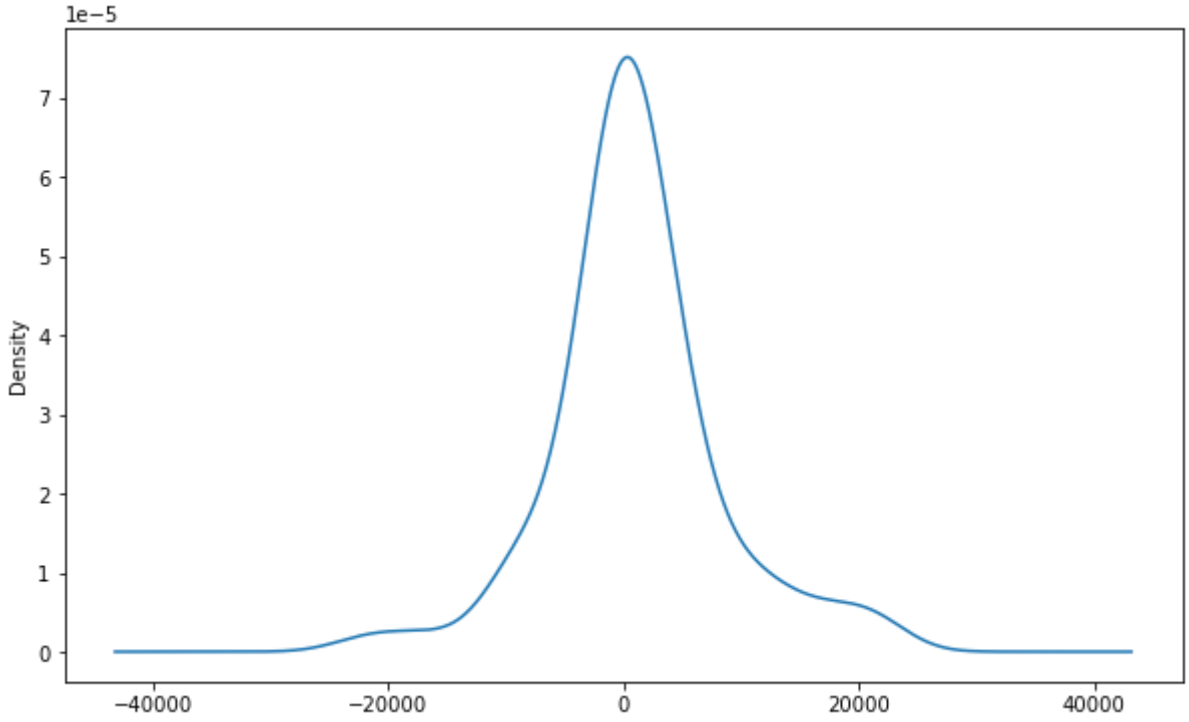




The ARIMA model with calculated values for p, d, q

```
In [15]: model=ARIMA(arimaz['Usage_gallons'],order=(1,1,1))
         result=model.fit()

In [16]: result.resid.plot(kind='kde')
```

Out[16]: <AxesSubplot:ylabel='Density'>




```
In [17]: new_dates=[arimaz.index[-2]+DateOffset(months=x) for x in
             range(1,48)] df_pred=pd.DataFrame(index=new_dates,columns
             =arimaz.columns)
#print(df_pred.head())
#print(df_pred.tail())
```

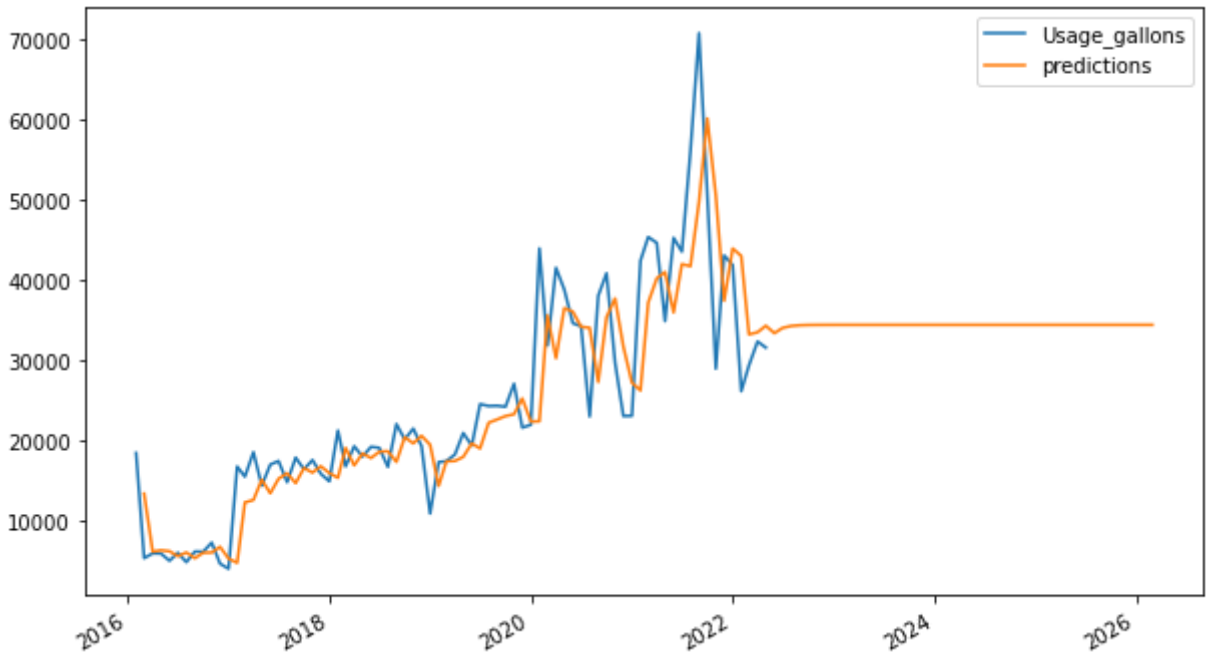
```
In [18]:     df2=pd.concat([arimaz,df_pred])
         df2['predictions']=result.predict(star
         t=1,end=123)
         df2[['Usage_gallons','predictions']].p
         lot() df2_summ = model.fit().summary()
         print(df2_summ)
```

SARIMAX Results

==========

Dep. Variable: Usage_gallons No. Observations:
76

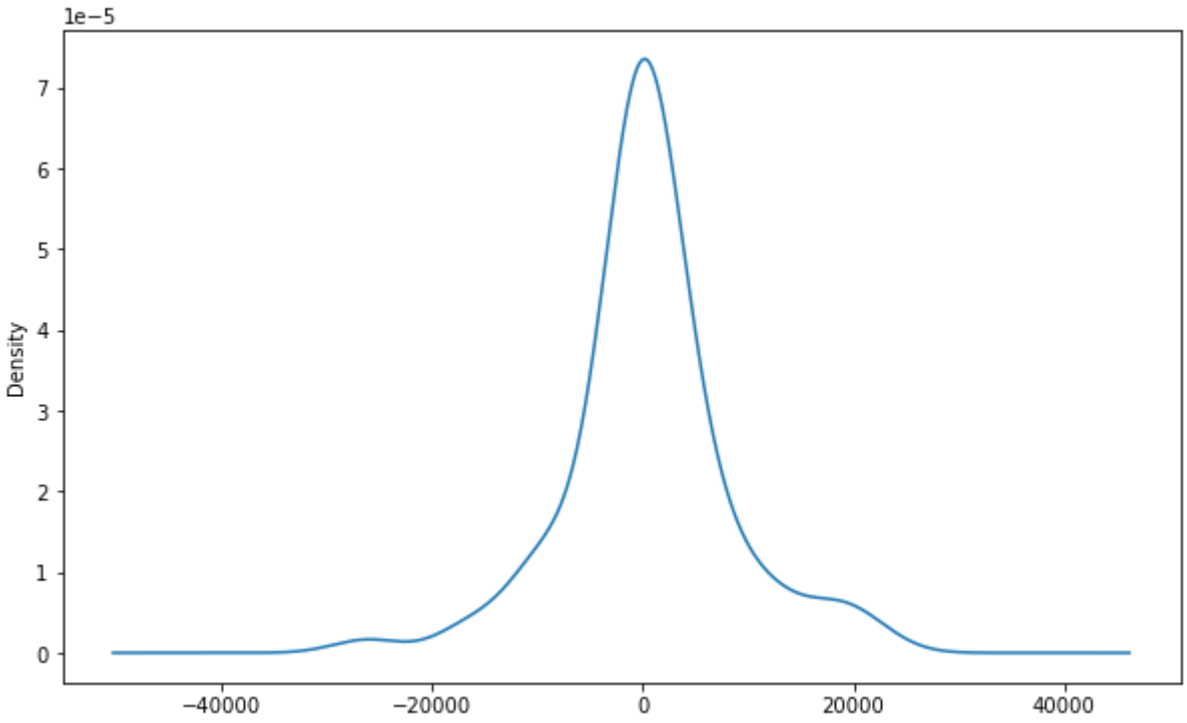
```
Model:                  ARIMA(1, 1, 1)   Log Likelihood
-769.831

Date:                Tue, 23 Aug 2022   AIC
1545.661

Time:                        21:30:02   BIC
1552.614

Sample:                    01-31-2016   HQIC
1548.438

                         - 04-30-2022
Covariance Type:                  opg
==============================================================================
==========                    coef    std err          z      P>|z|
[0.025      0.975] ------------------------------------------------
------------------------------ ar.L1          0.3666      0.159
2.305      0.021       0.055       0.678 ma.L1         -0.7618
0.099     -7.668      0.000      -0.956      -0.567 sigma2
5.194e+07   1.65e-09   3.14e+16      0.000    5.19e+07    5.19e+07
==============================================================================
============ ===

Ljung-Box (L1) (Q):                   0.14   Jarque-Bera (JB):
1

8.62

Prob(Q):                              0.71   Prob(JB):

0.00

Heteroskedasticity (H):               7.32   Skew:

0.18

Prob(H) (two-sided):                  0.00   Kurtosis:

5.41

==============================================================================
============ ===

Warnings:

[1] Covariance matrix calculated using the outer product of
gradients (complex-s tep). [2] Covariance matrix is singular or
near-singular, with condition number 5.69e+ 31. Standard errors may
be unstable.
```




```
In [19]: model=ARIMA(arimaz['Usage_gallons'],order=(1,1,0))
         result=model.fit()
```

```
In [20]: result.resid.plot(kind='kde')
```

```
Out[20]: <AxesSubplot:ylabel='Density'>
```

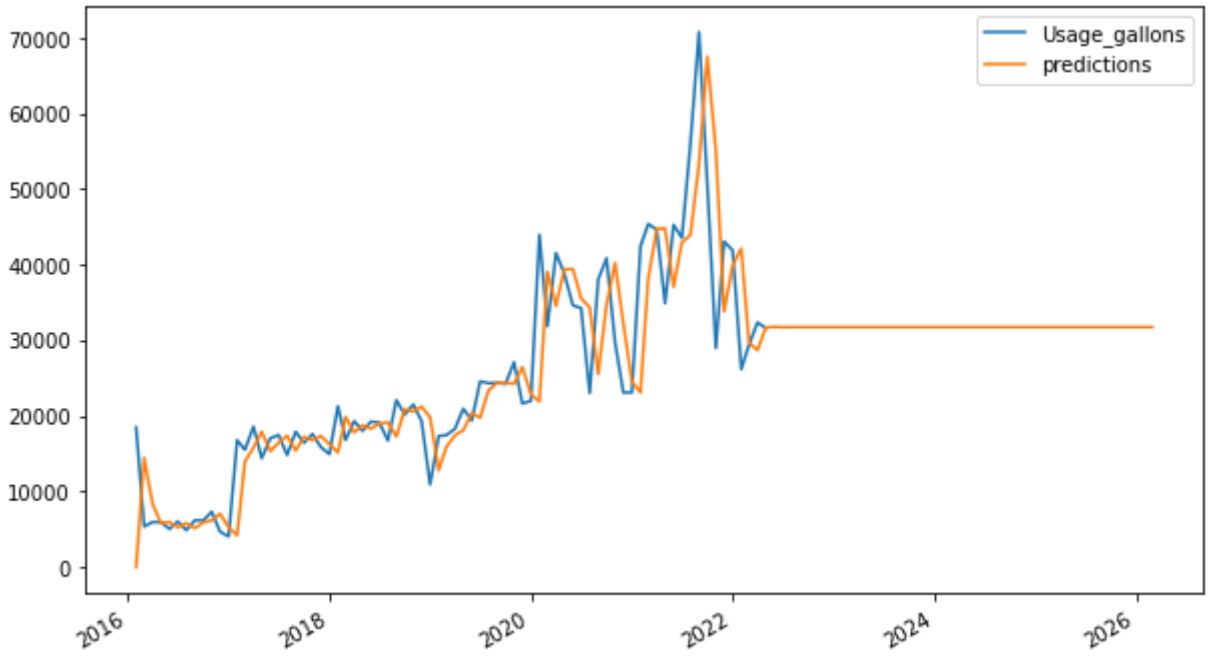


In [21]:

```
df3=pd.concat([arimaz,df_pred])
df3['predictions']=result.predict(start=0,end=123)
df3[['Usage_gallons','predictions']].plot()
df3_summ = model.fit().summary()
print(df3_summ)
```

```
                               SARIMAX Results
==============================================================================
==========

Dep. Variable:          Usage_gallons   No. Observations:
76

Model:                 ARIMA(1, 1, 0)   Log Likelihood
-774.237

Date:                Tue, 23 Aug 2022   AIC
1552.475

Time:                        21:30:03   BIC
1557.110

Sample:                    01-31-2016   HQIC
1554.325

                         - 04-30-2022
Covariance Type:                  opg
==============================================================================
==========                    coef    std err          z      P>|z|
[0.025      0.975] ----------------------------------------------
---------------------------- ar.L1         -0.2227      0.072     -
3.102      0.002      -0.363      -0.082 sigma2      5.375e+07
3.71e-10   1.45e+17      0.000    5.37e+07    5.37e+07
==============================================================================
============ ===

Ljung-Box (L1) (Q):                   0.12   Jarque-Bera (JB):
2

2.83

Prob(Q):                              0.73   Prob(JB):

0.00

Heteroskedasticity (H):               8.36   Skew:
-

0.15

Prob(H) (two-sided):                  0.00   Kurtosis:

5.69
```

```
==========================================================================
============

===

Warnings:

[1] Covariance matrix calculated using the outer product of
gradients (complex-s tep). [2] Covariance matrix is singular or
near-singular, with condition number 1.48e+ 33. Standard errors may
be unstable.
```

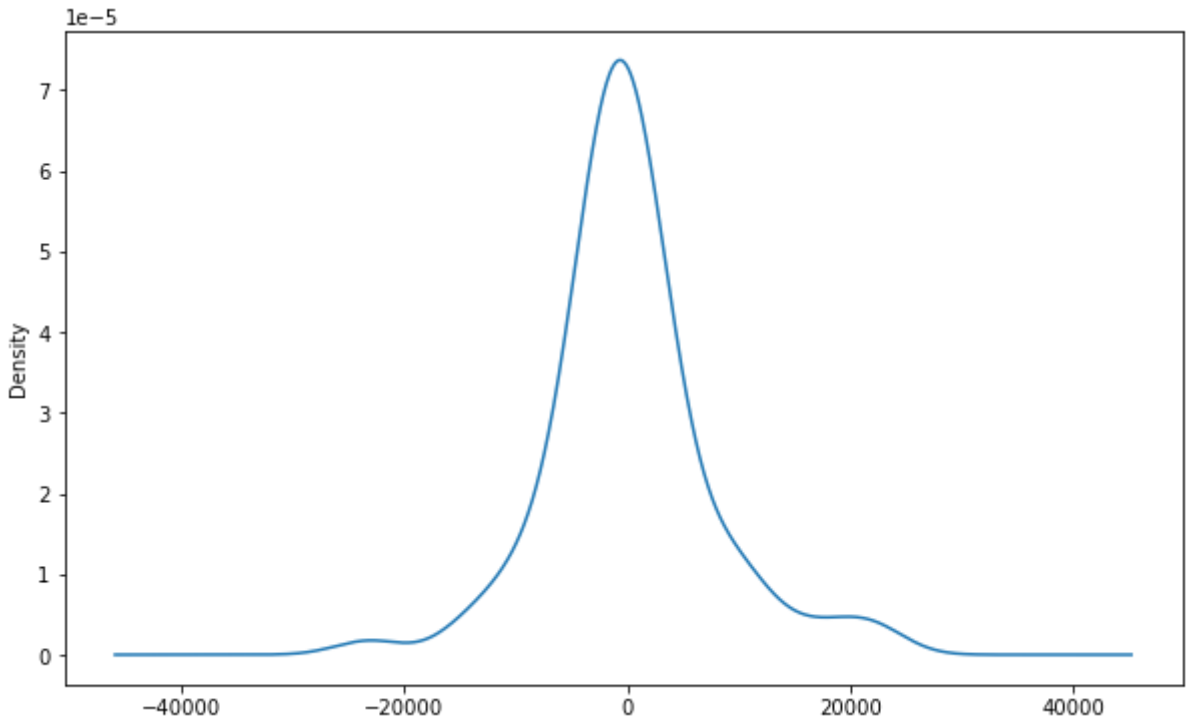


In [22]:
```
model=ARIMA(arimaz['Usage_gallons'],order=(1,0,1))
#model=SARIMAX(arimaz['Usage_gallons'],order=(1,0,0),seasonal_order=(1, 1, 0,
12 result=model.fit()
```

```
result.resid.plot(kind='kde')
```

In [23]:

Out[23]: <AxesSubplot:ylabel='Density'>



In [24]:

```
df4=pd.concat([arimaz,df_pred])
df4['predictions']=result.predict(start=0,end=123)
df4[['Usage_gallons','predictions']].plot()
df4_summ = model.fit().summary() print(df4_summ)
```

SARIMAX Results

==========

======== Dep. Variable: Usage_gallons No. Observations:
76

Model: ARIMA(1, 0, 1) Log Likelihood
-782.297

Date: Tue, 23 Aug 2022 AIC
1572.594

Time: 21:30:04 BIC
1581.916

Sample: 01-31-2016 HQIC
1576.319

- 04-30-2022

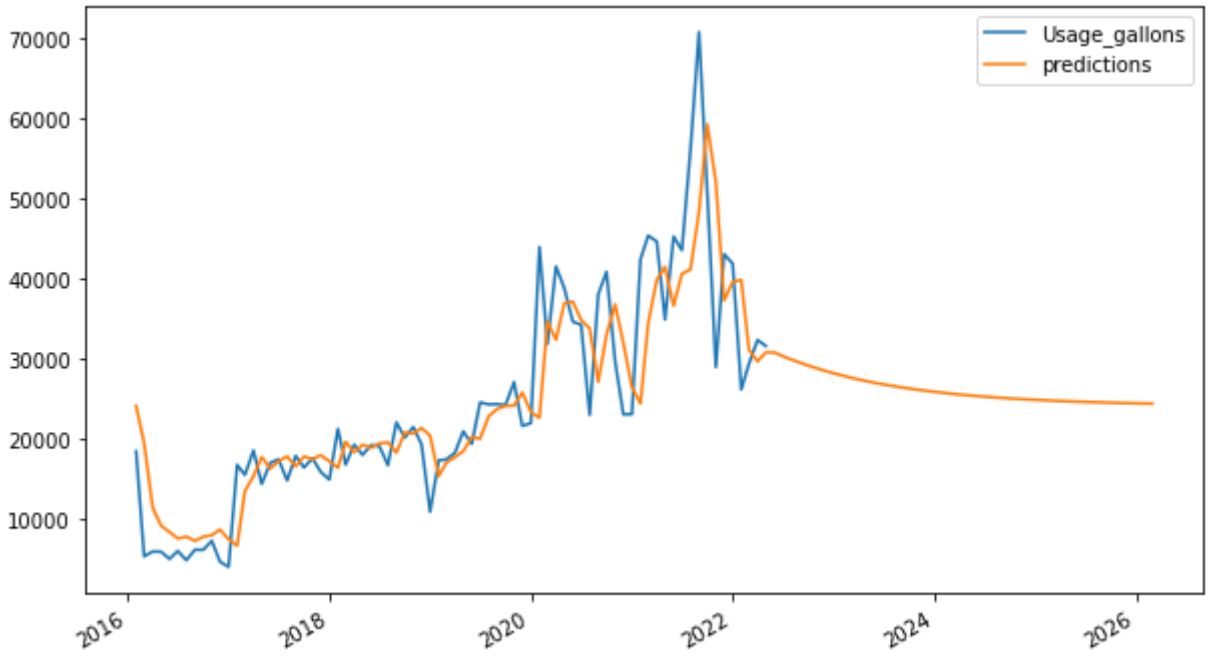
Covariance Type: opg

==========

========== coef std err z P>|z|
[0.025 0.975] ----------

---------- const 2.413e+04 8839.908
2.729 0.006 6802.252 4.15e+04 ar.L1 0.9324
0.042 21.952 0.000 0.849 1.016 ma.L1 -

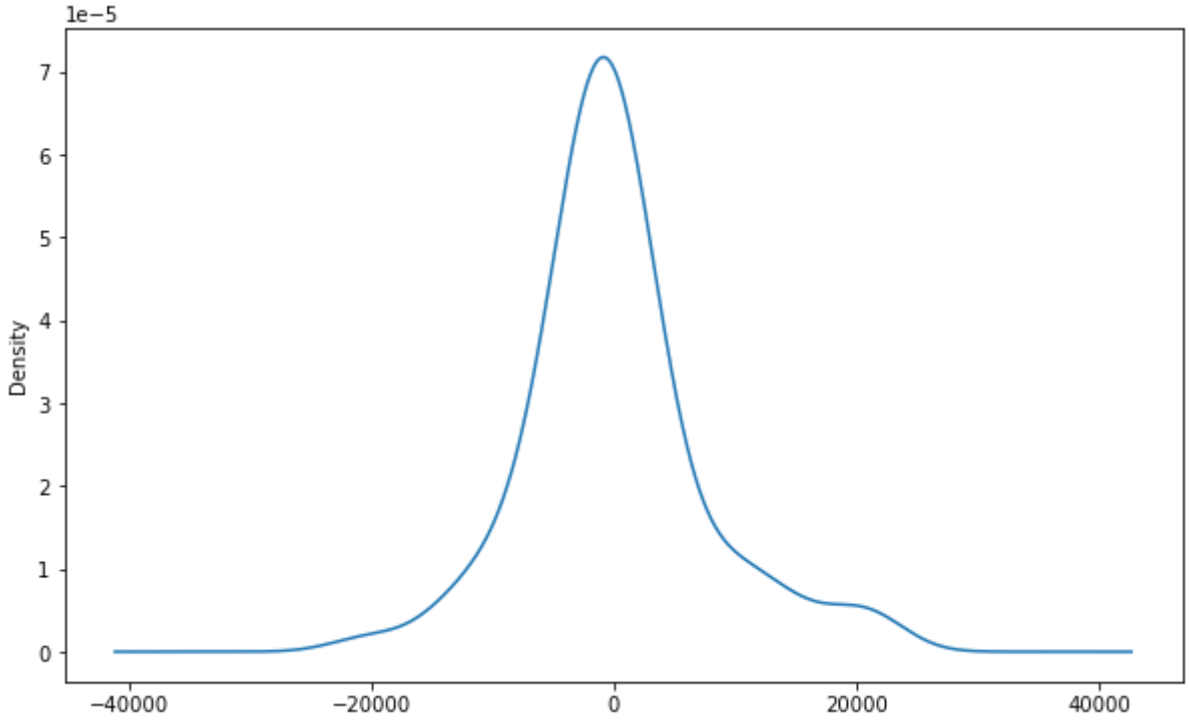
```
0.3735       0.117     -3.204       0.001      -0.602      -0.145
sigma2       5.078e+07       5.099   9.96e+06       0.000    5.08e+07
5.08e+07
==============================================================================
============ ===

Ljung-Box (L1) (Q):                   0.85   Jarque-Bera (JB):
2

2.89

Prob(Q):                              0.36   Prob(JB):

0.00

Heteroskedasticity (H):               5.68   Skew:

0.43

Prob(H) (two-sided):                  0.00   Kurtosis:

5.55

==============================================================================
============ ===

Warnings:

[1] Covariance matrix calculated using the outer product of
gradients (complex-s tep). [2] Covariance matrix is singular or
near-singular, with condition number 2.51e+ 22. Standard errors may
be unstable.
```



```
In [25]: model=ARIMA(arimaz['Usage_gallons'],order=(2,0,0))
          result=model.fit()
```

```
In [26]: result.resid.plot(kind='kde')
```

```
Out[26]: <AxesSubplot:ylabel='Density'>
```

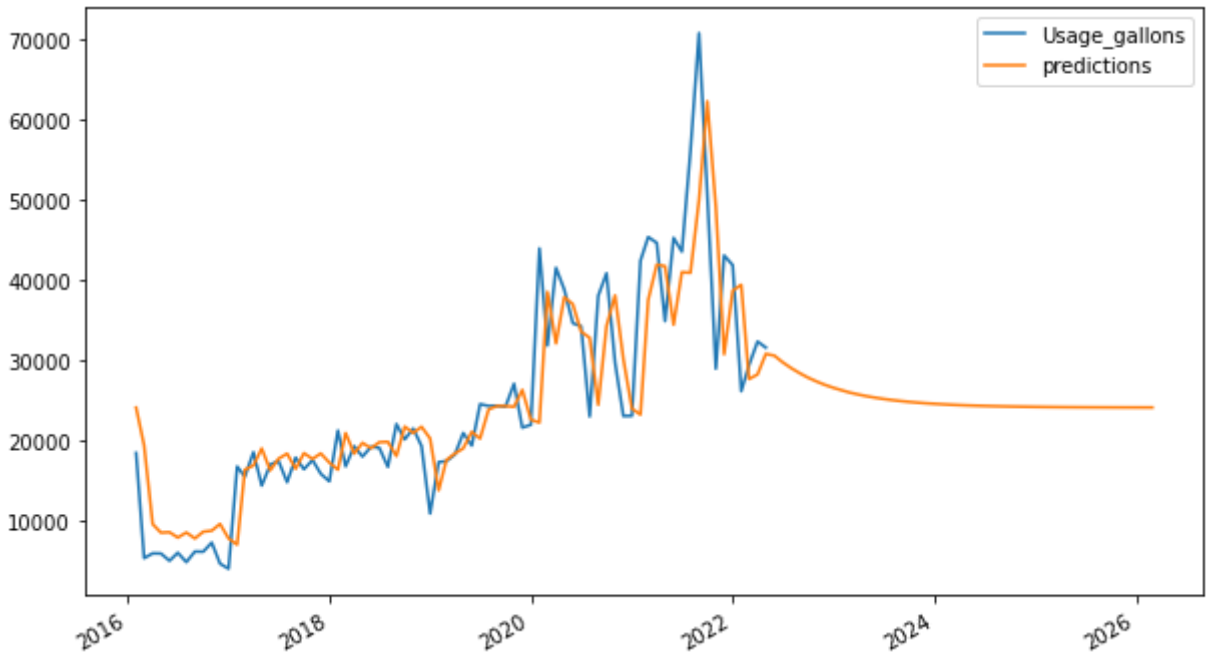



```
In [27]:
```

```
df6=pd.concat([arimaz,df_pred])
df6['predictions']=result.predict(start=0,end=123)
df6[['Usage_gallons','predictions']].plot()
df6_summ = model.fit().summary()
print(df6_summ)
```

```
                               SARIMAX Results
==============================================================================
 ======== Dep. Variable:          Usage_gallons   No. Observations:
76
Model:                 ARIMA(2, 0, 0)   Log Likelihood
-783.309

Date:                Tue, 23 Aug 2022   AIC
1574.619

Time:                        21:30:04   BIC
1583.942
```

```
Sample:                     01-31-2016   HQIC
1578.345

                          - 04-30-2022

Covariance Type:                  opg
==============================================================================
==========                   coef    std err          z      P>|z|
[0.025      0.975] ---------------------------------------------
---------------------------- const       2.413e+04   6580.414
3.667      0.000    1.12e+04     3.7e+04 ar.L1          0.7391
0.084      8.819      0.000       0.575       0.903 ar.L2
0.1151      0.077      1.497      0.134      -0.036       0.266
sigma2      5.023e+07      3.612   1.39e+07      0.000    5.02e+07
5.02e+07
==============================================================================
============ ===

Ljung-Box (L1) (Q):                   0.02   Jarque-Bera (JB):
1

5.29

Prob(Q):                              0.88   Prob(JB):

0.00

Heteroskedasticity (H):               5.48   Skew:

0.64

Prob(H) (two-sided):                  0.00   Kurtosis:

4.79

==============================================================================
============

===

Warnings:

[1] Covariance matrix calculated using the outer product of
gradients (complex-s tep). [2] Covariance matrix is singular or
near-singular, with condition number 2.99e+ 22. Standard errors may
be unstable.
```



Predictions for main model

In [28]:
```
print(df2.tail(46))
```

```
            Usage_gallons  1difference   predictions

  2022-05-31            NaN          NaN
  33404.634472 2022-06-30            NaN
NaN  34062.725034 2022-07-31            NaN
NaN  34303.982558 2022-08-31            NaN
NaN  34392.428125 2022-09-30            NaN
NaN  34424.852475 2022-10-31            NaN
NaN  34436.739317 2022-11-30            NaN
NaN  34441.097060 2022-12-31            NaN
NaN  34442.694619 2023-01-31            NaN
             NaN  34443.280288
   2023-02-28            NaN          NaN
   34443.494995 2023-03-31            NaN
NaN  34443.573708 2023-04-30            NaN
NaN  34443.602564 2023-05-31            NaN
NaN  34443.613143 2023-06-30            NaN
NaN  34443.617021 2023-07-31            NaN
NaN  34443.618442 2023-08-31            NaN
NaN  34443.618964 2023-09-30            NaN
NaN  34443.619155 2023-10-31            NaN
NaN  34443.619225 2023-11-30            NaN
NaN  34443.619250 2023-12-31            NaN
NaN  34443.619260 2024-01-31            NaN
NaN  34443.619263 2024-02-29            NaN
NaN  34443.619265 2024-03-31            NaN
NaN  34443.619265 2024-04-30            NaN
NaN  34443.619265 2024-05-31            NaN
NaN  34443.619265 2024-06-30            NaN
```

```
NaN  34443.619265 2024-07-31            NaN
NaN  34443.619265 2024-08-31            NaN
NaN  34443.619265 2024-09-30            NaN
NaN  34443.619265 2024-10-31            NaN
NaN  34443.619265 2024-11-30            NaN
NaN  34443.619265 2024-12-31            NaN
NaN  34443.619265 2025-01-31            NaN
NaN  34443.619265 2025-02-28            NaN
NaN  34443.619265 2025-03-31            NaN
NaN  34443.619265 2025-04-30            NaN
NaN  34443.619265 2025-05-31            NaN
NaN  34443.619265 2025-06-30            NaN
NaN  34443.619265 2025-07-31            NaN
NaN  34443.619265 2025-08-31            NaN
NaN  34443.619265 2025-09-30            NaN
NaN  34443.619265 2025-10-31            NaN
NaN  34443.619265 2025-11-30            NaN
NaN  34443.619265 2025-12-31            NaN
NaN  34443.619265 2026-01-31            NaN
             NaN  34443.619265
2026-02-28            NaN          NaN  34443.619265
```

Predictions for main model

In [29]:
```
print(df3.tail(46))
print(df4.tail(46))
print(df6.tail(46))
```

```
            Usage_gallons  1difference   predictions

2022-05-31            NaN          NaN  31776.915450

  2022-06-30            NaN          NaN
  31739.631456 2022-07-31            NaN
NaN  31747.936068 2022-08-31            NaN
NaN  31746.086304 2022-09-30            NaN
NaN  31746.498319 2022-10-31            NaN
NaN  31746.406547 2022-11-30            NaN
NaN  31746.426989 2022-12-31            NaN
NaN  31746.422435 2023-01-31            NaN
NaN  31746.423450 2023-02-28            NaN
NaN  31746.423224 2023-03-31            NaN
NaN  31746.423274 2023-04-30            NaN
NaN  31746.423263 2023-05-31            NaN
NaN  31746.423265 2023-06-30            NaN
NaN  31746.423265 2023-07-31            NaN
NaN  31746.423265 2023-08-31            NaN
NaN  31746.423265 2023-09-30            NaN
NaN  31746.423265 2023-10-31            NaN
NaN  31746.423265 2023-11-30            NaN
NaN  31746.423265 2023-12-31            NaN
             NaN  31746.423265
2024-01-31            NaN          NaN
31746.423265 2024-02-29            NaN
NaN  31746.423265 2024-03-31            NaN
NaN  31746.423265 2024-04-30            NaN
NaN  31746.423265 2024-05-31            NaN
```

```
NaN  31746.423265 2024-06-30          NaN
NaN  31746.423265 2024-07-31          NaN
NaN  31746.423265 2024-08-31          NaN
NaN  31746.423265 2024-09-30          NaN
NaN  31746.423265 2024-10-31          NaN
NaN  31746.423265 2024-11-30          NaN
NaN  31746.423265 2024-12-31          NaN
NaN  31746.423265 2025-01-31          NaN
NaN  31746.423265 2025-02-28          NaN
NaN  31746.423265 2025-03-31          NaN
NaN  31746.423265 2025-04-30          NaN
NaN  31746.423265 2025-05-31          NaN
NaN  31746.423265 2025-06-30          NaN
NaN  31746.423265 2025-07-31          NaN
NaN  31746.423265 2025-08-31          NaN
NaN  31746.423265 2025-09-30          NaN
NaN  31746.423265 2025-10-31          NaN
NaN  31746.423265 2025-11-30          NaN
NaN  31746.423265 2025-12-31          NaN
NaN  31746.423265 2026-01-31          NaN
NaN  31746.423265 2026-02-28          NaN
NaN  31746.423265           Usage_gallons
1difference   predictions
2022-05-31            NaN          NaN
30805.175102 2022-06-30            NaN          NaN
30354.065721 2022-07-31            NaN          NaN
29933.433954 2022-08-31            NaN          NaN
29541.220687 2022-09-30            NaN          NaN
29175.505924 2022-10-31            NaN          NaN
28834.499389 2022-11-30            NaN          NaN
28516.531756 2022-12-31            NaN          NaN
28220.046484 2023-01-31            NaN          NaN
27943.592193 2023-02-28            NaN          NaN
27685.815560 2023-03-31            NaN          NaN
27445.454696 2023-04-30            NaN          NaN
27221.332964 2023-05-31            NaN          NaN
27012.353226 2023-06-30            NaN          NaN
26817.492467 2023-07-31            NaN          NaN
26635.796786 2023-08-31            NaN          NaN
26466.376733 2023-09-30            NaN          NaN
26308.402948 2023-10-31            NaN          NaN
26161.102104

2023-11-30            NaN          NaN  26023.753121

2023-12-31            NaN          NaN
25895.683638 2024-01-31            NaN
NaN  25776.266717 2024-02-29           NaN
NaN  25664.917779 2024-03-31           NaN
NaN  25561.091739 2024-04-30           NaN
NaN  25464.280338 2024-05-31           NaN
NaN  25374.009659 2024-06-30           NaN
NaN  25289.837798 2024-07-31           NaN
NaN  25211.352713 2024-08-31           NaN
NaN  25138.170195 2024-09-30           NaN
```

```
NaN  25069.931996 2024-10-31            NaN
NaN  25006.304070 2024-11-30            NaN
NaN  24946.974940 2024-12-31            NaN
NaN  24891.654173 2025-01-31            NaN
NaN  24840.070959 2025-02-28            NaN
NaN  24791.972782 2025-03-31            NaN
NaN  24747.124189 2025-04-30            NaN
NaN  24705.305632 2025-05-31            NaN
NaN  24666.312399 2025-06-30            NaN
NaN  24629.953606

2025-07-31            NaN          NaN
24596.051267 2025-08-31            NaN          NaN
24564.439419 2025-09-30            NaN          NaN
24534.963314 2025-10-31            NaN          NaN
24507.478659 2025-11-30            NaN          NaN
24481.850907 2025-12-31            NaN          NaN
24457.954603 2026-01-31            NaN          NaN
24435.672769 2026-02-28            NaN          NaN
24414.896329            Usage_gallons  1difference
predictions

2022-05-31            NaN          NaN
30604.816170 2022-06-30            NaN          NaN
29775.790755 2022-07-31            NaN          NaN
29047.463652 2022-08-31            NaN          NaN
28413.777796 2022-09-30            NaN          NaN
27861.626323 2022-10-31            NaN          NaN
27380.625456 2022-11-30            NaN          NaN
26961.592548 2022-12-31            NaN          NaN
26596.546054 2023-01-31            NaN          NaN
26278.530348 2023-02-28            NaN          NaN
26001.486248 2023-03-31            NaN          NaN
25760.135166 2023-04-30            NaN          NaN
25549.878588 2023-05-31            NaN          NaN
25366.710448 2023-06-30            NaN          NaN
25207.140803 2023-07-31            NaN          NaN
25068.129337 2023-08-31            NaN          NaN
24947.027435 2023-09-30            NaN          NaN
24841.527714 2023-10-31            NaN          NaN
24749.620066 2023-11-30            NaN          NaN
24669.553354 2023-12-31            NaN          NaN
24599.802049 2024-01-31            NaN          NaN
24539.037164 2024-02-29            NaN          NaN
24486.100932 2024-03-31            NaN          NaN
24439.984748 2024-04-30            NaN          NaN
24399.809949 2024-05-31            NaN          NaN
24364.811077 2024-06-30            NaN          NaN
24334.321289 2024-07-31            NaN          NaN
24307.759658 2024-08-31            NaN          NaN
24284.620099 2024-09-30            NaN          NaN
24264.461728 2024-10-31            NaN          NaN
24246.900464 2024-11-30            NaN          NaN
```

```
24231.601709 2024-12-31            NaN          NaN
24218.273970 2025-01-31            NaN          NaN
24206.663313 2025-02-28            NaN          NaN
24196.548517 2025-03-31            NaN          NaN
24187.736862 2025-04-30            NaN          NaN
24180.060458

2025-05-31            NaN          NaN  24173.373044

2025-06-30            NaN          NaN
24167.547205 2025-07-31            NaN
NaN  24162.471938 2025-08-31            NaN
NaN  24158.050544 2025-09-30            NaN
NaN  24154.198782 2025-10-31            NaN
NaN  24150.843261 2025-11-30            NaN
NaN  24147.920050 2025-12-31            NaN
NaN  24145.373451 2026-01-31            NaN
NaN  24143.154943 2026-02-28            NaN
NaN  24141.222257
```

SARIMA

In [1]:
```
import pandas as pd
import datetime as dt
import matplotlib.pyplot as plt

from matplotlib import rcParams
from pandas.tseries.offsets import DateOffset from
statsmodels.tsa.arima.model import ARIMA from
statsmodels.tsa.stattools import adfuller from
statsmodels.tsa.statespace.sarimax import SARIMAX from
statsmodels.graphics.tsaplots import plot_acf, plot_pacf
```

In [2]:
```
plt.rcParams["figure.figsize"] = 14, 6
```

In [3]:
```
df = pd.read_csv('arima.csv')     # read the final file df =
 df[['TRANSACTION_DATE', 'Usage_gallons']]    # Choose needed columns
 for pr print(f'Data frame shape: {df.shape}') print(df.head())
 print(df.tail())
```

```
Data frame shape: (1177158, 2)
  TRANSACTION_DATE  Usage_gallons
0       2022-04-28
10.495889 1       2022-
04-28      10.495889 2
2022-04-28
10.495889 3       2022-
04-28      10.495889 4
2022-04-28
10.495889
```

```
        TRANSACTION_DATE  Usage_gallons
1177153           2016-06-07
0.25 1177154        2016-12-
20             10.50 1177155
2016-10-20             12.00
1177156           2016-08-31
2.25 1177157        2016-08-
31           3.75
```

```
In [4]: df['TRANSACTION_DATE'] = pd.to_datetime(df['TRANSACTION_DATE'])
        # set tran df = df[df['TRANSACTION_DATE'] < '2022-05-01']        #
        filter last month  print(f'Data frame shape: {df.shape}')
        print(df.head()) print(df.tail())
```

```
Data frame shape: (1174806, 2)
  TRANSACTION_DATE  Usage_gallons
0       2022-04-28
10.495889 1       2022-
04-28      10.495889 2
2022-04-28
10.495889 3       2022-
04-28      10.495889 4
2022-04-28
10.495889
        TRANSACTION_DATE  Usage_gallons
1177153           2016-06-07
0.25 1177154        2016-12-
20             10.50 1177155
2016-10-20             12.00
1177156           2016-08-31
2.25 1177157        2016-08-
31           3.75
```

```
In [5]: # group by month and calculate total consumptions per month
        arimaz = df.groupby(pd.Grouper(key='TRANSACTION_DATE', axis=0,
        freq='M')).sum() print(f'Data frame shape: {arimaz.shape}')
        print(arimaz.head()) print(arimaz.tail())
```

```
Data frame shape: (76, 1)
                   Usage_gallons
TRANSACTION_DATE
2016-01-31
18467.922976   2016-
02-29
5349.208348 2016-03-
31
5948.213152 2016-04-
30
5930.484263 2016-05-
31
5028.455578
Usage_gallons
TRANSACTION_DATE
```

```
2021-12-31          41890.167389
2022-01-31
26171.658883 2022-02-28
29400.786396 2022-03-31
32361.026412 2022-04-30
31609.526980
```

Since data for the last month is not completly available, the may of 2022 data should be deleted

In [6]:
```
print(arimaz.head())
print(arimaz.tail())
arimaz.shape
```

```
                  Usage_gallons
TRANSACTION_DATE
2016-01-31
18467.922976   2016-
02-29
5349.208348  2016-03-
31
5948.213152  2016-04-
30
5930.484263  2016-05-
31
5028.455578
                  Usage_gallons
TRANSACTION_DATE
2021-12-31          41890.167389
2022-01-31
26171.658883 2022-02-28
29400.786396 2022-03-31
32361.026412
2022-04-30
```

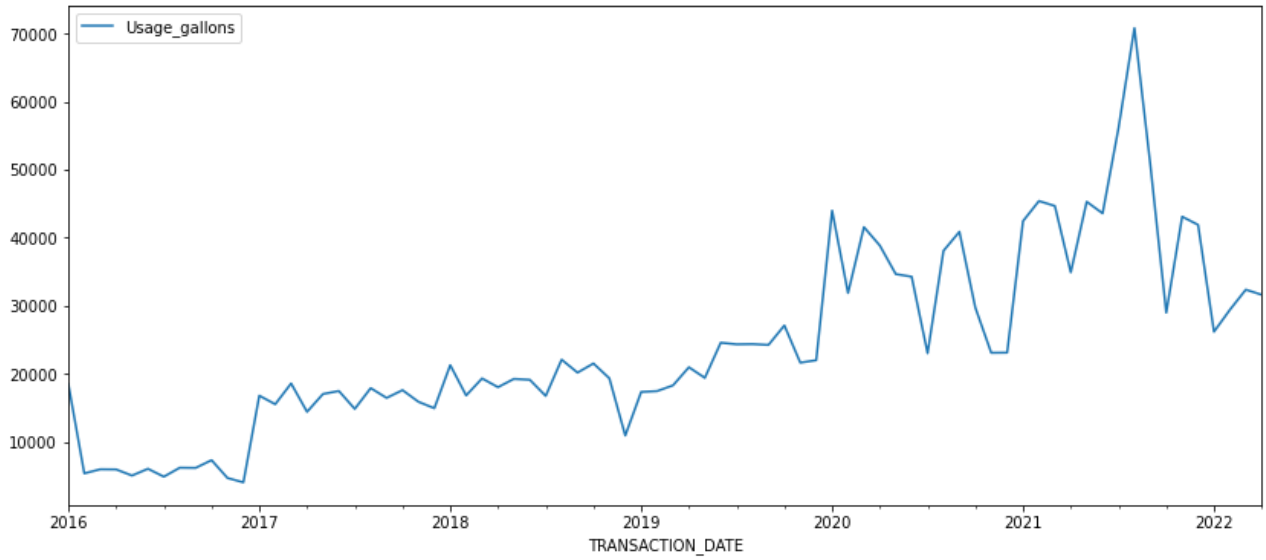
31609.526980 Out[6]: (76, 1)

In [7]:
```
type(arimaz)
```

Out[7]: pandas.core.frame.DataFrame

In [8]:
```
arimaz.plot()
```

Out[8]: <AxesSubplot:xlabel='TRANSACTION_DATE'>



In [9]:
```
result=adfuller(arimaz['Usage_gallons']) #to
help you, we added the names of every value
dict(zip(['adf', 'pvalue', 'usedlag', 'nobs', 'critical' 'values',
'icbest'],res
```

Out[9]:
```
{'adf': -1.6736608458992983,
 'pvalue': 0.4447317279527293,
 'usedlag': 2,
 'nobs': 73,
 'criticalvalues': {'1%': -
3.5232835753964475,   '5%': -
2.902030597326081,
  '10%': -2.5883710883843123},
 'icbest': 1303.8129598480937}
```

Since p-value is greater than 0.05, as expected we failed to reject the Null Hypothesis and the series has a unit root thus is not stationary.

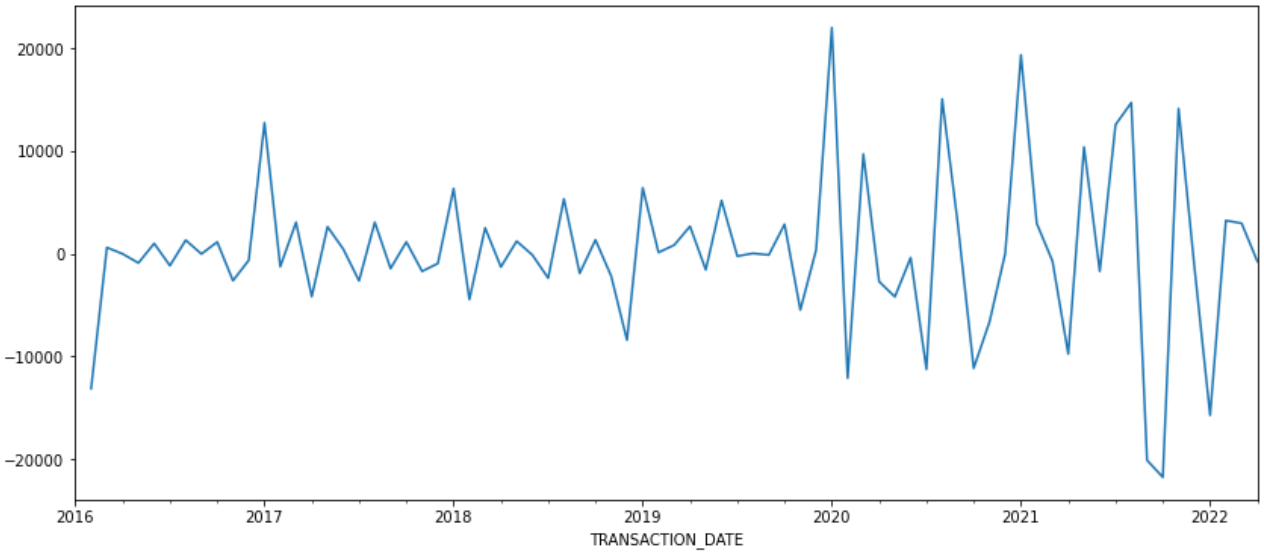
Transform Non-Stationary to Stationary using Differencing (the d and D parameters) The next step is to transform our data to Stationary so we will have an estimate for d and D parameters we will use in the model. This can be done using Differencing and it's performed by subtracting the previous observation from the current observation.

difference(T) = observation(T) — observation(T −1)

Then, we will test it again for stationarity using the Augmented Dickey-Fuller test and if it's stationary we will proceed to our next step. If not we will apply differencing again till we have a stationary series. Differencing can be done very easily with pandas using the shift function.

In [10]:
```
arimaz['1difference']=arimaz['Usage_gallons']-
arimaz['Usage_gallons'].shift(1) arimaz['1difference'].plot()
```

Out[10]: <AxesSubplot:xlabel='TRANSACTION_DATE'>



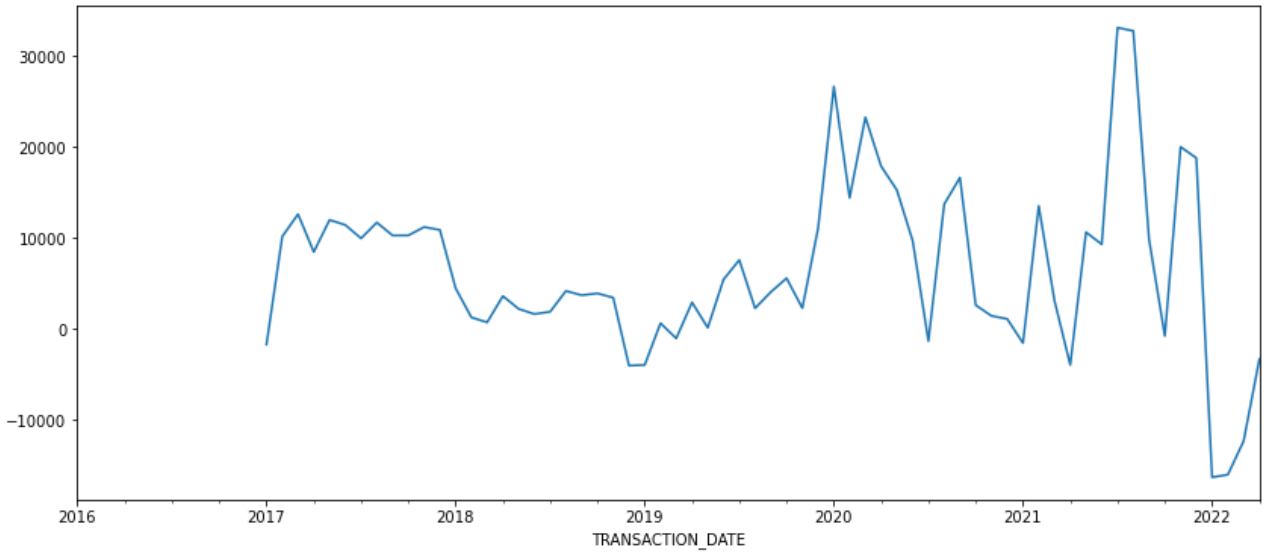
In [11]:
```
result=adfuller((arimaz['1difference']).dropna())
#to help you, we added the names of every value
dict(zip(['adf', 'pvalue', 'usedlag', 'nobs', 'critical' 'values',
'icbest'],res
```

Out[11]:
```
{'adf': -4.193589166184938,
 'pvalue': 0.0006755592722814878,
 'usedlag': 12,
 'nobs': 62,
 'criticalvalues': {'1%': -3.540522678829176,
  '5%': -2.9094272025108254,
  '10%': -2.5923136524453696},
 'icbest': 1281.6118533940491}
```

The p-value is less than 0.05 so we can reject the null hypothesis. That means the first difference is stationary and that suggests that a good estimate for the value d is 1.

Our data are seasonal so we need to estimate also the D value which is the same as the d value but for Seasonal Difference. The seasonal difference can be computed by shifting the data by the number of rows per season (in our example 12 month per year) and subtracting them from the previous season. This is not the first seasonal difference. If we get that the seasonal difference is stationary then the D value will be 0. If not then we will compute the seasonal first difference.

In [12]:
```
arimaz['Seasonal_Difference']=arimaz['Usage_gallons']-
arimaz['Usage_gallons'].sh ax=arimaz['Seasonal_Difference'].plot()
```

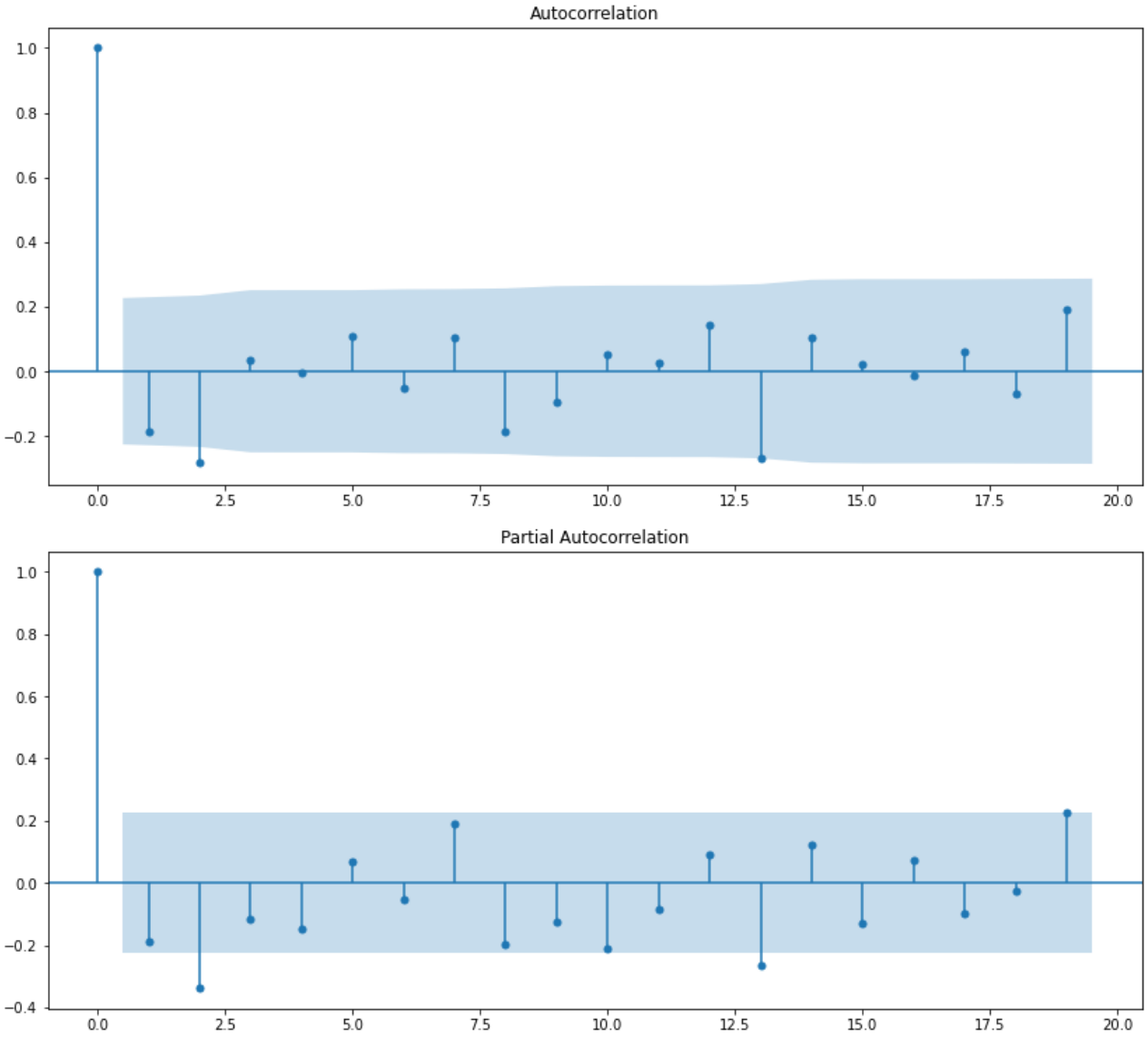


In [13]:
```
result=adfuller((arimaz['Seasonal_Difference']).dropna())
#to help you, we added the names of every value
dict(zip(['adf', 'pvalue', 'usedlag', 'nobs', 'critical' 'values',
'icbest'],res
```

Out[13]:
```
{'adf': -3.4221391202075835,
 'pvalue': 0.010226909181380008,
 'usedlag': 6,
 'nobs': 57,
 'criticalvalues': {'1%': -3.5506699942762414,
  '5%': -2.913766394626147,
  '10%': -2.5946240473991997},
 'icbest': 1091.4797304043595}
```

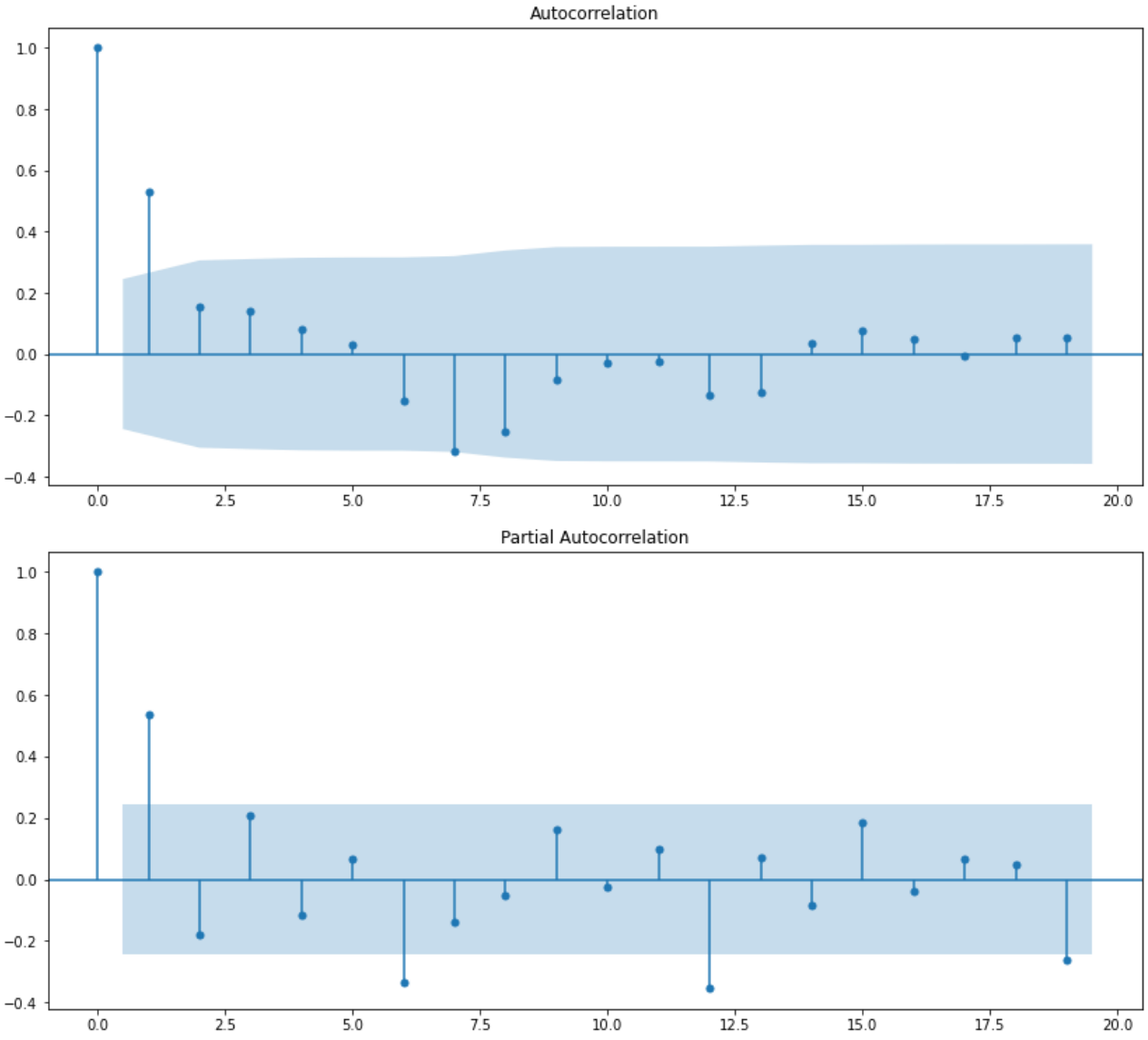
The p-value is less than 0.05 thus it's stationary and we don't have to use differencing. That suggests using 1 for the D value.

In [14]:
```
fig1=plot_acf(arimaz['1difference'].dropna())
fig2=plot_pacf(arimaz['1difference'].dropna())
```



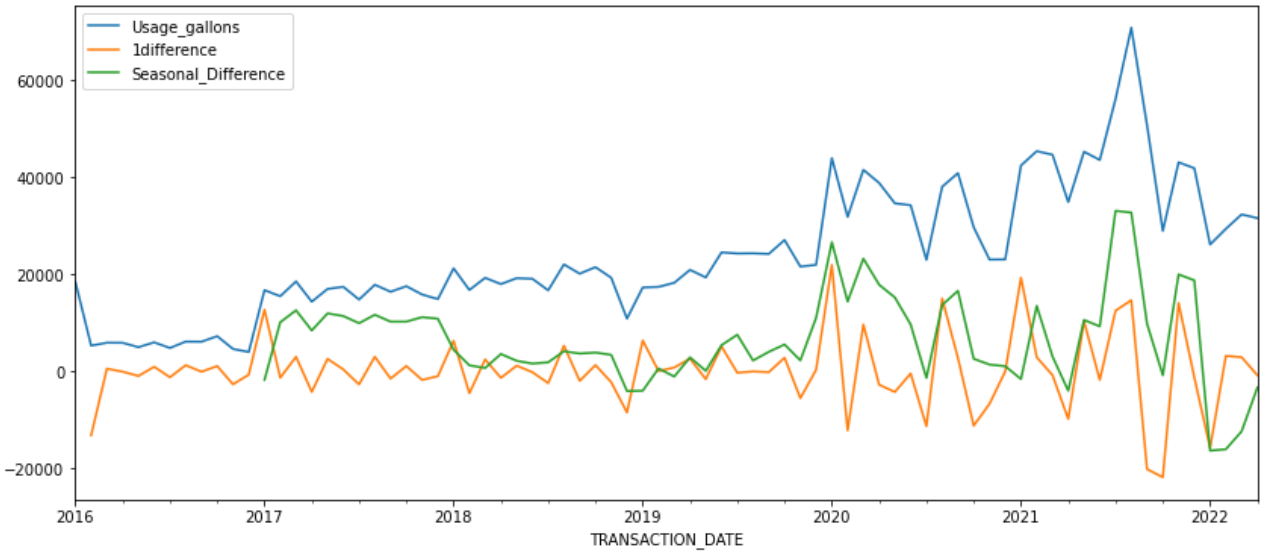
We have a gradual decrease in the Autocorrelation plot and a sharp cut-off in the Partial Autocorrelation plot at lag-2. This suggests using AR and MA in orders in other words p = 1, q = 1.

In [15]:
```
fig1=plot_acf(arimaz['Seasonal_Difference'].dropna())
fig2=plot_pacf(arimaz['Seasonal_Difference'].dropna())
```



In [16]:
```
arimaz.plot()
```

Out[16]: <AxesSubplot:xlabel='TRANSACTION_DATE'>



We have a gradual decrease in the Autocorrelation plot and a sharp cut-off in the Partial Autocorrelation plot. This suggests using AR and not over the value of 1 for the seasonal part of the ARIMA.

The values we chose may not be optimum. You can play around with these parameters to fine-tune the model having as a guide the rules we mentioned above.

The SARIMAX model with calculated values for p, d, q,

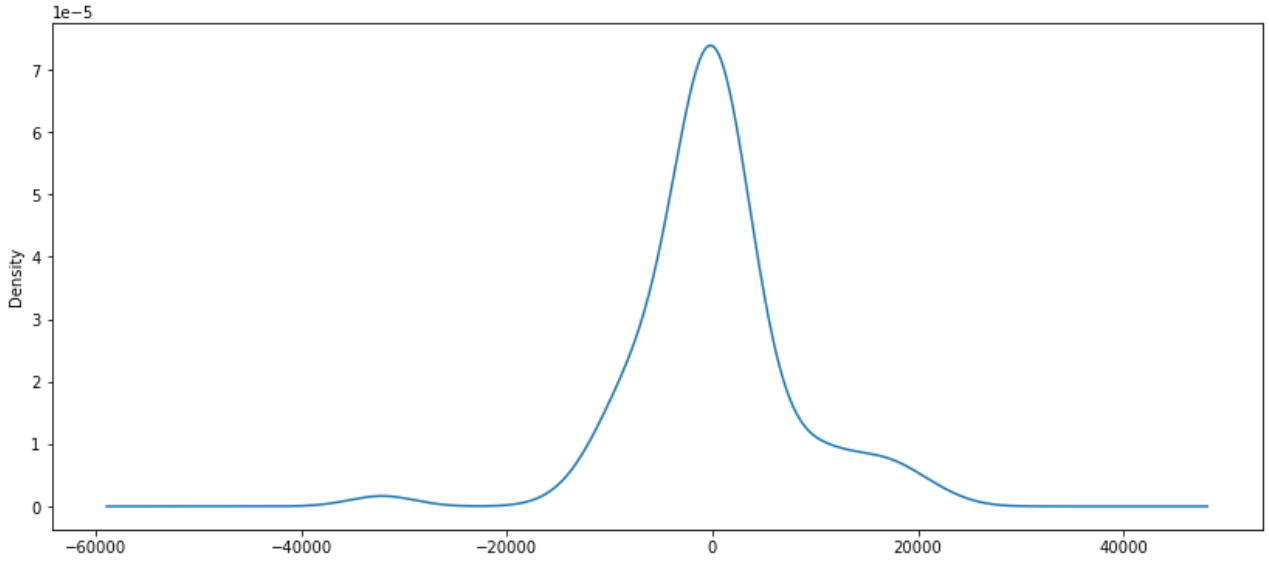
```
model=SARIMAX(arimaz['Usage_gallons'],order=(1,1,1),seasonal_order=(1, 1, 0,
12) result=model.fit()
```

P, D, Q, m

In [17]:

In [18]:
```
result.resid.plot(kind='kde')
```

Out[18]: <AxesSubplot:ylabel='Density'>

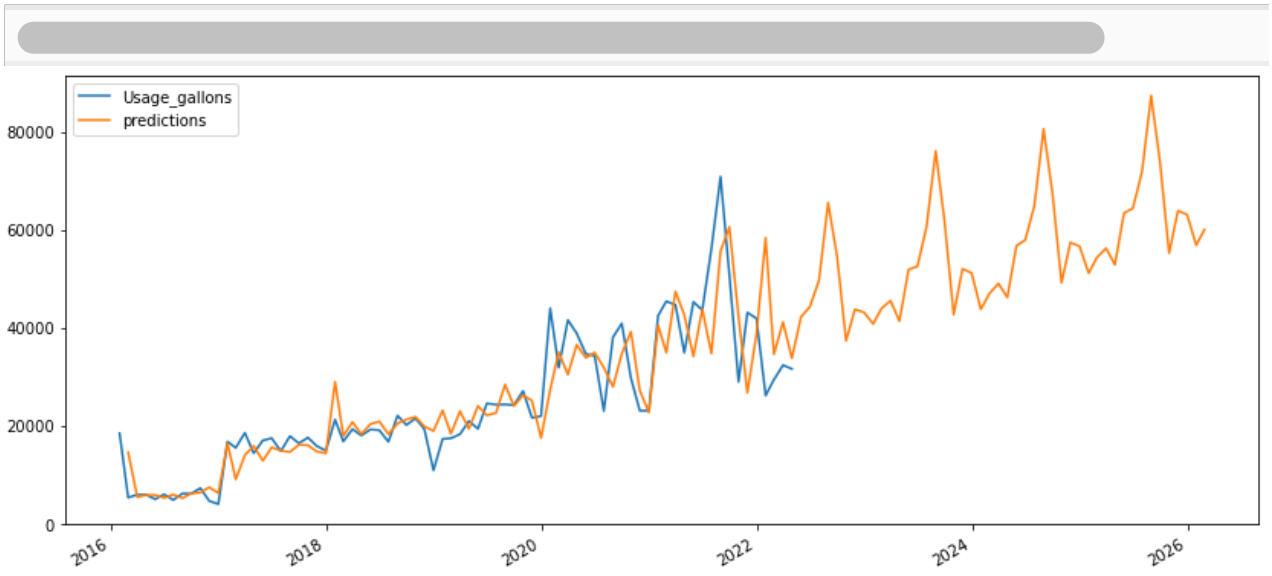



```
In [19]: new_dates=[arimaz.index[-2]+DateOffset(months=x) for x in
          range(1,48)] df_pred=pd.DataFrame(index=new_dates,columns
          =arimaz.columns)
#print(df_pred.head())
#print(df_pred.tail())
```

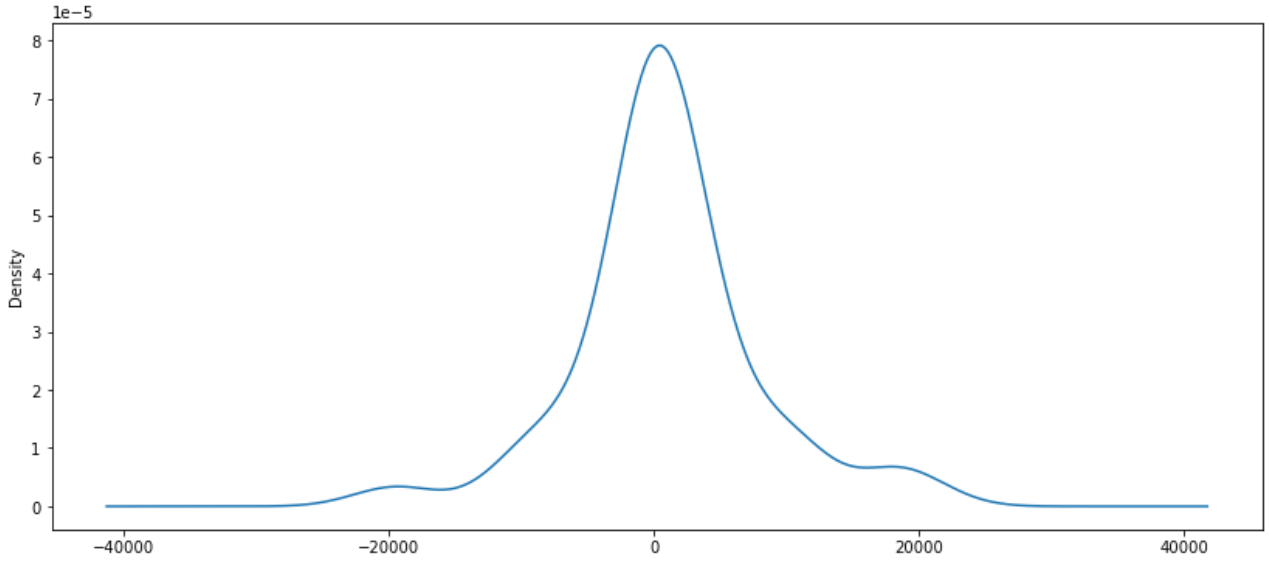
```
In [20]: df2=pd.concat([arimaz,df_pred])
df2['predictions']=result.predict(star
t=1,end=122)
df2[['Usage_gallons','predictions']].p
lot() df2_summ = model.fit().summary()
print(df2_summ)
```

```
                                     SARIMAX Results
==========================================================================
============ =========== Dep. Variable:
Usage_gallons   No. Observations:
76
Model:             SARIMAX(1, 1, 1)x(1, 1, [], 12)   Log Likelihood
-654.193
Date:                             Tue, 23 Aug 2022   AIC
1316.386
Time:                                    21:32:50   BIC
1324.959
Sample:                                01-31-2016   HQIC
1319.758
                                     - 04-30-2022
Covariance Type:                              opg
==========================================================================
==========                  coef    std err          z      P>|z|
[0.025      0.975] ---------------------------------------------
---------------------------- ar.L1          0.5681      0.099
5.737      0.000       0.374       0.762 ma.L1         -1.0000
```

```
0.112     -8.960       0.000       -1.219       -0.781 ar.S.L12      -
0.3797      0.159     -2.384        0.017       -0.692       -0.068
sigma2       5.93e+07   1.88e-09   3.15e+16       0.000     5.93e+07
5.93e+07
==============================================================================
============ ===
Ljung-Box (L1) (Q):                   0.20   Jarque-Bera (JB):
4
6.43
Prob(Q):                              0.65   Prob(JB):
0.00
Heteroskedasticity (H):              17.91   Skew:
-
0.47
Prob(H) (two-sided):                  0.00   Kurtosis:
7.10
==============================================================================
============
===

Warnings:
[1] Covariance matrix calculated using the outer product of
gradients (complex-s tep). [2] Covariance matrix is singular or
near-singular, with condition number 6.12e+ 31. Standard errors may
be unstable.
```



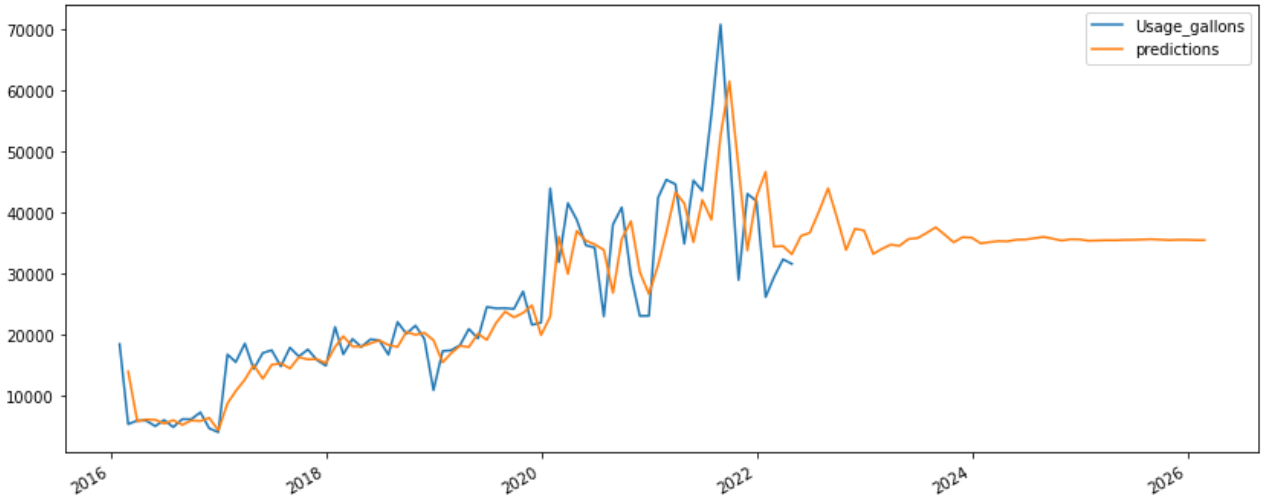

```
In [21]: model=SARIMAX(arimaz['Usage_gallons'].dropna(),
          order=(1,1,1),seasonal_order=(1, result=model.fit()

In [22]: result.resid.plot(kind='kde')

Out[22]: <AxesSubplot:ylabel='Density'>
```




```
In [23]:  df3=pd.concat([arimaz,df_pred])
          df3['predictions']=result.predict(star
          t=1,end=122)
          df3[['Usage_gallons','predictions']].p
          lot()  df3_summ = model.fit().summary()
          print(df3_summ)
```

```
                                      SARIMAX Results
==========================================================================
============ =========== Dep. Variable:
Usage_gallons   No. Observations:
76
Model:             SARIMAX(1, 1, 1)x(1, 0, [], 12)   Log Likelihood
-768.368
Date:                               Tue, 23 Aug 2022   AIC
1544.736
Time:                                     21:32:51   BIC
1554.006
Sample:                                 01-31-2016   HQIC
1548.437
                                      - 04-30-2022
Covariance Type:                                opg
==========================================================================
==========                  coef    std err          z      P>|z|
[0.025      0.975] ---------------------------------------------
---------------------------- ar.L1          0.4572      0.148
3.092       0.002       0.167       0.747 ma.L1         -0.8233
0.093      -8.898       0.000      -1.005      -0.642 ar.S.L12
0.2444       0.094       2.589       0.010       0.059       0.429
sigma2       5.194e+07   1.09e-09   4.78e+16       0.000    5.19e+07
5.19e+07
==========================================================================
============ ===
```

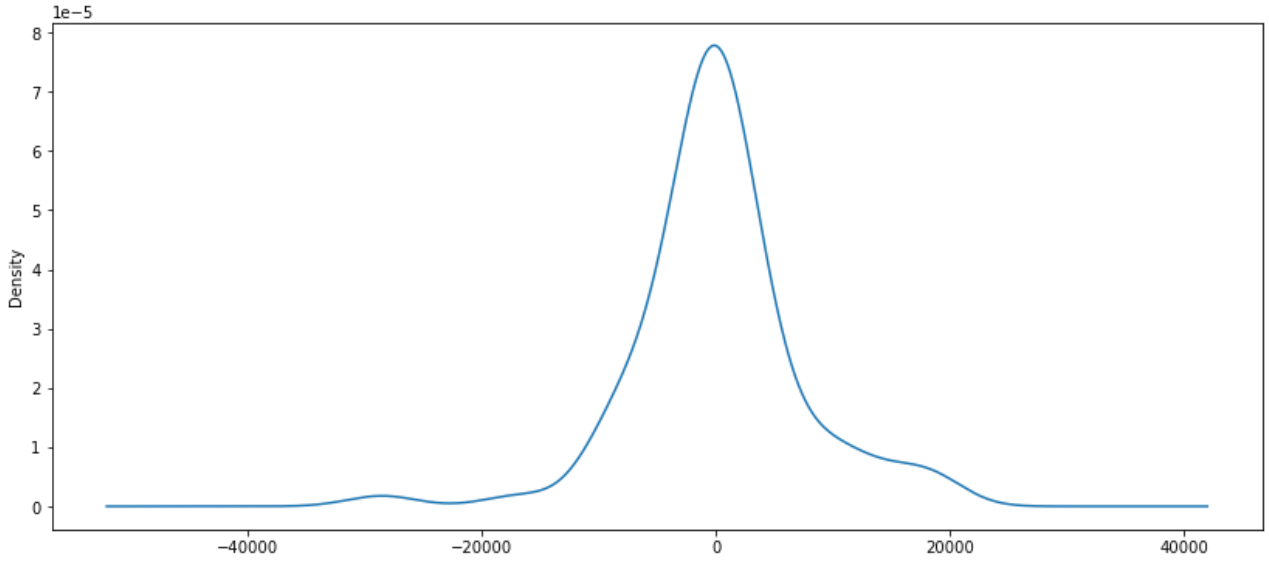
```
Ljung-Box (L1) (Q):                   0.24   Jarque-Bera (JB):
1
6.96
Prob(Q):                              0.62   Prob(JB):
0.00
Heteroskedasticity (H):               9.52   Skew:
0.03
Prob(H) (two-sided):                  0.00   Kurtosis:
5.33
===============================================================================
============
===

Warnings:
[1] Covariance matrix calculated using the outer product of
gradients (complex-s tep). [2] Covariance matrix is singular or
near-singular, with condition number 1.57e+ 31. Standard errors may
be unstable.
```



In [24]:
```
model=SARIMAX(arimaz['Usage_gallons'],order=(1,1,1),seasonal_order=(1, 1, 1,
12) result=model.fit()
```

In [25]:
```
result.resid.plot(kind='kde')
```

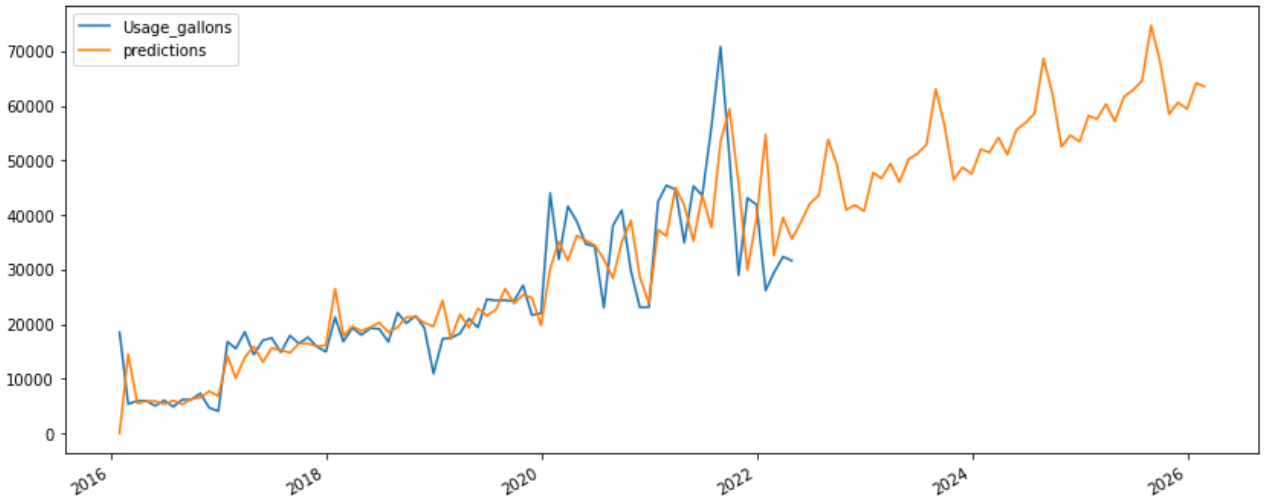
Out[25]:
```
<AxesSubplot:ylabel='Density'>
```



In [26]:
```
df4=pd.concat([arimaz,df_pred])
df4['predictions']=result.predict(start=0,end=122)
df4[['Usage_gallons','predictions']].plot()
df4_summ = model.fit().summary() print(df4_summ)
```

```
                                     SARIMAX Results
==========================================================================
============ ==========
Dep. Variable:                         Usage_gallons   No.
Observations:
76
Model:             SARIMAX(1, 1, 1)x(1, 1, 1, 12)   Log Likelihood
-653.175
Date:                              Tue, 23 Aug 2022   AIC
1316.351
Time:                                     21:32:52   BIC
1327.067
Sample:                                 01-31-2016   HQIC
1320.565
                                      - 04-30-2022
Covariance Type:                                opg
==========================================================================
==========                    coef    std err          z      P>|z|
[0.025      0.975] ----------------------------------------------
---------------------------- ar.L1          0.5579      0.219
2.546      0.011       0.128       0.987 ma.L1         -0.9646
0.128     -7.550      0.000      -1.215      -0.714 ar.S.L12      -
0.1102      0.594     -0.186       0.853      -1.274       1.053
ma.S.L12      -0.6084      0.494     -1.230      0.219      -1.577
0.361 sigma2      8.079e+07   1.63e-08   4.94e+15      0.000
8.08e+07    8.08e+07
==========================================================================
============ ===
```

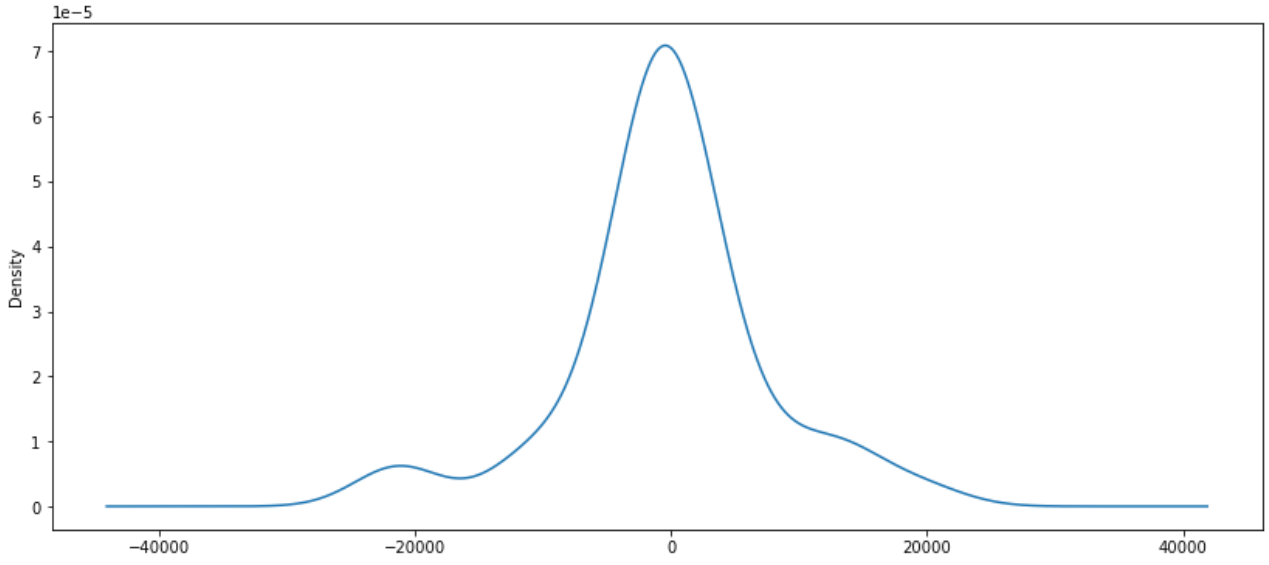
```
Ljung-Box (L1) (Q):                   0.18   Jarque-Bera (JB):
3
5.63
Prob(Q):                              0.67   Prob(JB):
0.00
Heteroskedasticity (H):              23.55   Skew:
-
0.53
Prob(H) (two-sided):                  0.00   Kurtosis:
6.53
===============================================================================
============
===

Warnings:
[1] Covariance matrix calculated using the outer product of
gradients (complex-s tep). [2] Covariance matrix is singular or
near-singular, with condition number 2.24e+ 31. Standard errors may
be unstable.
```



In [27]:
```
model=SARIMAX(arimaz['Usage_gallons'],order=(1,2,1),seasonal_order=(
1, 0, 0, 12) result=model.fit()
```

In [28]:
```
result.resid.plot(kind='kde')
```

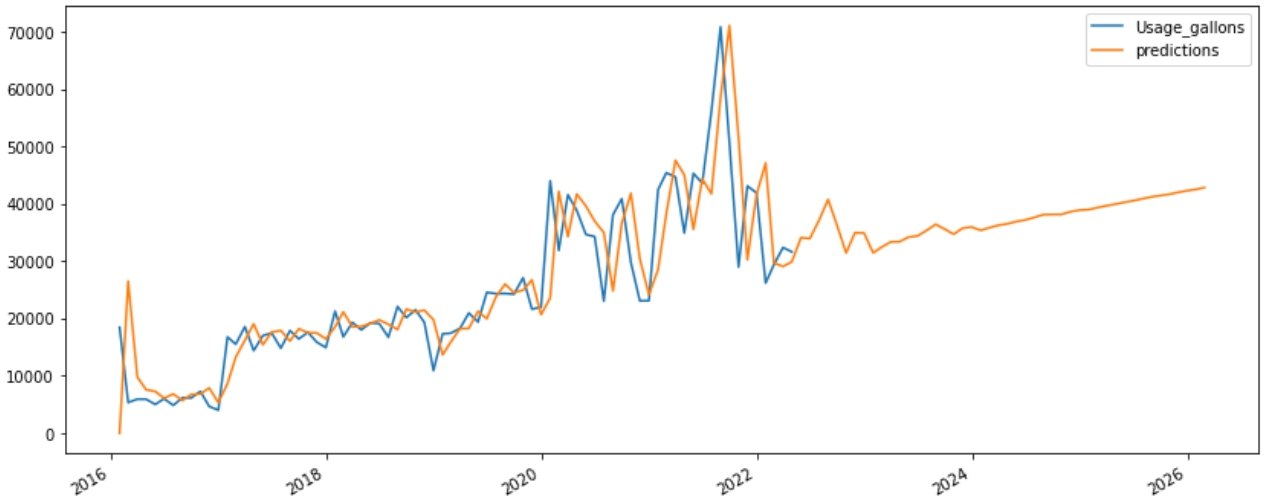
Out[28]:
```
<AxesSubplot:ylabel='Density'>
```




```
In [29]: df6=pd.concat([arimaz,df_pred])
         df6['predictions']=result.predict(star
         t=0,end=122)
         df6[['Usage_gallons','predictions']].p
         lot() df6_summ = model.fit().summary()
         print(df6_summ)
```

```
                                      SARIMAX Results
==========================================================================
============ =========== Dep. Variable:
Usage_gallons   No. Observations:
76
Model:             SARIMAX(1, 2, 1)x(1, 0, [], 12)   Log Likelihood
-766.497
Date:                             Tue, 23 Aug 2022   AIC
1540.994
Time:                                    21:32:53   BIC
1550.210
Sample:                                01-31-2016   HQIC
1544.671
                                     - 04-30-2022
Covariance Type:                              opg
==========================================================================
==========                  coef    std err          z      P>|z|
[0.025      0.975] ---------------------------------------------
---------------------------- ar.L1         -0.1310      0.126     -
1.042      0.298      -0.377       0.115 ma.L1         -0.9863
0.134     -7.364      0.000      -1.249      -0.724 ar.S.L12
0.2337      0.133      1.763       0.078      -0.026       0.493
sigma2      7.877e+07   1.36e-09   5.79e+16      0.000    7.88e+07
7.88e+07
==========================================================================
============ ===
```

```
Ljung-Box (L1) (Q):                   0.40   Jarque-Bera (JB):
1
5.76
Prob(Q):                              0.53   Prob(JB):
0.00
Heteroskedasticity (H):              16.61   Skew:
-
0.34
Prob(H) (two-sided):                  0.00   Kurtosis:
5.15
===============================================================================
============
===

Warnings:
[1] Covariance matrix calculated using the outer product of
gradients (complex-s tep). [2] Covariance matrix is singular or
near-singular, with condition number 1.5e+3 2. Standard errors may
be unstable.
```



Predictions for main model

In [30]:
```
print(df2.tail(46))
```

```
            Usage_gallons  1difference  Seasonal_Difference
predictions
  2022-05-31            NaN          NaN                  NaN
      42201.437028 2022-06-30           NaN          NaN
   NaN  44256.664973 2022-07-31           NaN          NaN
   NaN  49620.948996 2022-08-31           NaN          NaN
   NaN  65510.189271 2022-09-30           NaN          NaN
   NaN  54707.853262 2022-10-31           NaN          NaN
   NaN  37324.181535 2022-11-30           NaN          NaN
   NaN  43749.996096 2022-12-31           NaN          NaN
   NaN  43121.715036 2023-01-31           NaN          NaN
   NaN  40780.832990 2023-02-28           NaN          NaN
```

```
  NaN  43938.876869 2023-03-31            NaN           NaN
  NaN  45518.255471 2023-04-30            NaN           NaN
  NaN  41357.947372 2023-05-31            NaN           NaN
  NaN  51876.654943 2023-06-30            NaN           NaN
  NaN  52504.176563 2023-07-31            NaN           NaN
  NaN  60598.065962 2023-08-31            NaN           NaN
                    NaN  76037.476585
2023-09-30           NaN          NaN                   NaN
    61706.203863 2023-10-31            NaN          NaN
  NaN  42663.727645 2023-11-30            NaN           NaN
  NaN  52011.743749 2023-12-31            NaN           NaN
  NaN  51165.275779 2024-01-31            NaN           NaN
  NaN  43744.317761 2024-02-29            NaN           NaN
  NaN  46929.395667 2024-03-31            NaN           NaN
  NaN  49033.178408 2024-04-30            NaN           NaN
  NaN  46167.376501 2024-05-31            NaN           NaN
  NaN  56713.890018 2024-06-30            NaN           NaN
  NaN  57883.586523 2024-07-31            NaN           NaN
  NaN  64940.912114 2024-08-31            NaN           NaN
  NaN  80551.146653 2024-09-30            NaN           NaN
  NaN  67559.986521 2024-10-31            NaN           NaN
  NaN  49147.440854 2024-11-30            NaN           NaN
  NaN  57385.752735 2024-12-31            NaN           NaN
  NaN  56622.141268 2025-01-31            NaN           NaN
  NaN  51130.339457 2025-02-28            NaN           NaN
  NaN  54305.151251 2025-03-31            NaN           NaN
  NaN  56209.791827 2025-04-30            NaN           NaN
  NaN  52852.401902 2025-05-31            NaN           NaN
  NaN  63388.356134 2025-06-30            NaN           NaN
  NaN  64352.162022 2025-07-31            NaN           NaN
  NaN  71803.122166 2025-08-31            NaN           NaN
  NaN  87348.486416 2025-09-30            NaN           NaN
  NaN  73848.419247 2025-10-31            NaN           NaN
  NaN  55196.657792 2025-11-30            NaN           NaN
  NaN  63856.379257 2025-12-31            NaN           NaN
  NaN  63061.303078 2026-01-31            NaN           NaN
                    NaN  56836.905231
2026-02-28           NaN          NaN                   NaN
60015.615576
```

Predictions for other built models

In [31]:
```
print(df3.tail(46))
print(df4.tail(46))
print(df6.tail(46))
```

```
             Usage_gallons  1difference  Seasonal_Difference
                                                 predictions
2022-05-31             NaN          NaN                   NaN
    36192.387451 2022-06-30            NaN          NaN
  NaN  36708.838891 2022-07-31            NaN           NaN
  NaN  40202.424166 2022-08-31            NaN           NaN
  NaN  43990.475935 2022-09-30            NaN           NaN
  NaN  39168.850381 2022-10-31            NaN           NaN
  NaN  33894.060166 2022-11-30            NaN           NaN
  NaN  37363.263639 2022-12-31            NaN           NaN
  NaN  37077.792108 2023-01-31            NaN           NaN
```

```
   NaN  33240.630161 2023-02-28            NaN           NaN
   NaN  34031.503626 2023-03-31            NaN           NaN
   NaN  34755.701220 2023-04-30            NaN           NaN
   NaN  34572.433531 2023-05-31            NaN           NaN
   NaN  35692.498612 2023-06-30            NaN           NaN
   NaN  35818.779679 2023-07-31            NaN           NaN
   NaN  36672.528202 2023-08-31            NaN           NaN
   NaN  37598.214922 2023-09-30            NaN           NaN
   NaN  36419.981835 2023-10-31            NaN           NaN
   NaN  35131.006744 2023-11-30            NaN           NaN
   NaN  35978.763093 2023-12-31            NaN           NaN
   NaN  35909.004317 2024-01-31            NaN           NaN
   NaN  34971.333372 2024-02-29            NaN           NaN
   NaN  35164.595973 2024-03-31            NaN           NaN
   NaN  35341.565188 2024-04-30            NaN           NaN
   NaN  35296.780858 2024-05-31            NaN           NaN
   NaN  35570.486508 2024-06-30            NaN           NaN
   NaN  35601.345292 2024-07-31            NaN           NaN
                     NaN  35809.972263
2024-08-31           NaN           NaN                    NaN
36036.178470  2024-09-30                 NaN               NaN
NaN   35748.258562 2024-10-31              NaN              NaN
NaN   35433.277092 2024-11-30              NaN              NaN
NaN   35640.439777 2024-12-31              NaN              NaN
NaN   35623.393116 2025-01-31              NaN              NaN
NaN   35394.258376 2025-02-28              NaN              NaN
NaN   35441.485152 2025-03-31              NaN              NaN
NaN   35484.730382 2025-04-30              NaN              NaN
NaN   35473.786621 2025-05-31              NaN              NaN
NaN   35540.670930 2025-06-30              NaN              NaN
NaN   35548.211763 2025-07-31              NaN              NaN
NaN   35599.193066 2025-08-31              NaN              NaN
NaN   35654.470134 2025-09-30              NaN              NaN
NaN   35584.112346 2025-10-31              NaN              NaN
NaN   35507.141637 2025-11-30              NaN              NaN
NaN   35557.765119 2025-12-31              NaN              NaN
NaN   35553.599498 2026-01-31              NaN              NaN
NaN   35497.606796 2026-02-28              NaN              NaN
NaN   35509.147407               Usage_gallons   1difference
Seasonal_Difference   predictions
2022-05-31           NaN          NaN                   NaN
38735.867802 2022-06-30           NaN          NaN
NaN  42110.360578 2022-07-31            NaN          NaN
NaN  43614.141895 2022-08-31            NaN          NaN
NaN  53829.158128 2022-09-30            NaN          NaN
NaN  49147.224313 2022-10-31            NaN          NaN
NaN  40933.050968 2022-11-30            NaN          NaN
NaN  41809.939085 2022-12-31            NaN          NaN
NaN  40691.499456 2023-01-31            NaN          NaN
NaN  47722.835049 2023-02-28            NaN          NaN
NaN  46690.023598 2023-03-31            NaN          NaN
NaN  49403.079521 2023-04-30            NaN          NaN
NaN  46002.350703 2023-05-31            NaN          NaN
NaN  50221.952582 2023-06-30            NaN          NaN
NaN  51213.539914 2023-07-31            NaN          NaN
```

```
NaN  52918.090894 2023-08-31            NaN          NaN
NaN  63060.529052 2023-09-30            NaN          NaN
NaN  56362.926122 2023-10-31            NaN          NaN
NaN  46479.799932 2023-11-30            NaN          NaN
NaN  48718.058349 2023-12-31            NaN          NaN
NaN  47535.325684 2024-01-31            NaN          NaN
NaN  52028.210706 2024-02-29            NaN          NaN
NaN  51448.100861 2024-03-31            NaN          NaN
NaN  54178.860830 2024-04-30            NaN          NaN
NaN  51064.840559 2024-05-31            NaN          NaN
NaN  55601.891060 2024-06-30            NaN          NaN
NaN  56854.496441 2024-07-31            NaN          NaN
NaN  58535.991159 2024-08-31            NaN          NaN
NaN  68685.913893 2024-09-30            NaN          NaN
NaN  62210.216973 2024-10-31            NaN          NaN
NaN  52510.904368 2024-11-30            NaN          NaN
NaN  54599.004832 2024-12-31            NaN          NaN
NaN  53423.309408 2025-01-31            NaN          NaN
NaN  58195.988335 2025-02-28            NaN          NaN
NaN  57565.960134 2025-03-31            NaN          NaN
NaN  60294.759845 2025-04-30            NaN          NaN
NaN  57149.129899 2025-05-31            NaN          NaN
NaN  61651.184287 2025-06-30            NaN          NaN
NaN  62875.015277 2025-07-31            NaN          NaN
NaN  64559.050722 2025-08-31            NaN          NaN
NaN  74708.147937 2025-09-30            NaN          NaN
NaN  68207.989322 2025-10-31            NaN          NaN
NaN  58488.414192 2025-11-30            NaN          NaN
NaN  60593.066982 2025-12-31            NaN          NaN
NaN  59416.595775 2026-01-31            NaN          NaN
NaN  64158.432069
2026-02-28            NaN          NaN                  NaN
63533.906519            Usage_gallons  1difference
Seasonal_Difference   predictions
 2022-05-31            NaN          NaN                  NaN
     34089.291570 2022-06-30            NaN          NaN
   NaN  33935.604502 2022-07-31            NaN          NaN
   NaN  37089.095938 2022-08-31            NaN          NaN
   NaN  40749.458843 2022-09-30            NaN          NaN
   NaN  36277.617482 2022-10-31            NaN          NaN
   NaN  31418.874980 2022-11-30            NaN          NaN
   NaN  34943.027308 2022-12-31            NaN          NaN
   NaN  34886.362915 2023-01-31            NaN          NaN
   NaN  31437.726826 2023-02-28            NaN          NaN
   NaN  32416.807717 2023-03-31            NaN          NaN
   NaN  33333.054501 2023-04-30            NaN          NaN
   NaN  33381.935473 2023-05-31            NaN          NaN
   NaN  34185.903914 2023-06-30            NaN          NaN
   NaN  34374.482732 2023-07-31            NaN          NaN
   NaN  35335.888849 2023-08-31            NaN          NaN
   NaN  36415.741583 2023-09-30            NaN          NaN
   NaN  35595.246499 2023-10-31            NaN          NaN
   NaN  34684.339675 2023-11-30            NaN          NaN
   NaN  35732.362482 2023-12-31            NaN          NaN
   NaN  35943.613730 2024-01-31            NaN          NaN
```

```
NaN  35362.223057 2024-02-29           NaN          NaN
NaN  35815.509087 2024-03-31           NaN          NaN
NaN  36254.111933 2024-04-30           NaN          NaN
NaN  36490.027207 2024-05-31           NaN          NaN
NaN  36902.392652 2024-06-30           NaN          NaN
NaN  37170.952765 2024-07-31           NaN          NaN
NaN  37620.108523 2024-08-31           NaN          NaN
NaN  38096.943095 2024-09-30           NaN          NaN
NaN  38129.701041 2024-10-31           NaN          NaN
NaN  38141.331413 2024-11-30           NaN          NaN
NaN  38610.727912 2024-12-31           NaN          NaN
NaN  38884.586158 2025-01-31           NaN          NaN
NaN  38973.218445 2025-02-28           NaN          NaN
NaN  39303.635806 2025-03-31           NaN          NaN
NaN  39630.621974 2025-04-30           NaN          NaN
NaN  39910.243754 2025-05-31           NaN          NaN
NaN  40231.098720 2025-06-30           NaN          NaN
NaN  40518.349004 2025-07-31           NaN          NaN
NaN  40847.801195 2025-08-31           NaN          NaN
NaN  41183.721421 2025-09-30           NaN          NaN
NaN  41415.869039 2025-10-31           NaN          NaN
NaN  41643.079527 2025-11-30           NaN          NaN
NaN  41977.261611 2025-12-31           NaN          NaN
NaN  42265.749971 2026-01-31           NaN          NaN
                     NaN  42510.954403
2026-02-28             NaN          NaN                  NaN
                                                 42812.659597
```

In [ ]:

In [ ]: