Supply_Chain_Analytics (1)

May 10, 2025

[ ]:
```
# Load all the modules we need
# For plotting
import matplotlib.pyplot as plt
import matplotlib
import seaborn as sns
import datetime

# For ML
import sklearn

# For data manipulation
import numpy as np
import pandas as pd

# This makes all the plots to be shown within jupyter
%matplotlib inline
# Setting the default plot size
matplotlib.rcParams['figure.figsize'] = (20.0, 10.0)
```

First load the Excel sheet using pandas and then load each page into a new data frame

[ ]:
```
xlsx = pd.ExcelFile('Project.xlsx')
```

[ ]:
```
customer_order = xlsx.parse('customer order')
material_master = xlsx.parse('Material Master')
sales_past_demand = xlsx.parse('sales past demand')
customer_master = xlsx.parse('Customer Master')
invoice = xlsx.parse('Invoice')
stock_master = xlsx.parse('stock master')
delivery_data = xlsx.parse('Delivery data')
booking_data = xlsx.parse('Booking details')
```

1 Pre-processing of data

1.1 customer order

[ ]:
```
pd.isnull(customer_order).describe()
```

[ ]:
```
          SONO   ITEM   PTNO   DESC   DATE ORD_QTY   CUST   PLNT  Price  \
count   39398  39398  39398  39398  39398   39398  39398  39398  39398
unique      1      1      1      1      1       1      1      2      1
top     False  False  False  False  False   False  False  False  False
freq    39398  39398  39398  39398  39398   39398  39398  39388  39398

       customer PO ref PO date
count            39398   39398
unique               1       1
top              False   False
freq             39398   39398
```

There is some missing data in PLNT. Lets see what they are.

Since there are only 10 values missing, lets fill it with the most common PLNT values.

[ ]:
```
customer_order.PLNT.describe()
```

[ ]:
```
count    39388.000000
mean       861.837869
std        196.784207
min        130.000000
25%        930.000000
50%        930.000000
75%        930.000000
max        930.000000
Name: PLNT, dtype: float64
```

[ ]:
```
customer_order.PLNT.fillna(930, inplace=True)
```

[ ]:
```
pd.isnull(customer_order).describe()
```

[ ]:
```
          SONO   ITEM   PTNO   DESC   DATE ORD_QTY   CUST   PLNT  Price  \
count   39398  39398  39398  39398  39398   39398  39398  39398  39398
unique      1      1      1      1      1       1      1      1      1
top     False  False  False  False  False   False  False  False  False
freq    39398  39398  39398  39398  39398   39398  39398  39398  39398

       customer PO ref PO date
count            39398   39398
unique               1       1
top              False   False
freq             39398   39398
```

pd.to_datetime(customer_order['PO date']) gives an error due to some strings which are set to 00:00:00. So we need to remove these first, we do this by setting PO date to the DATE

[ ]:
```
customer_order[customer_order['PO date'] == datetime.time(0)].count()
```

[ ]:
```
SONO              397
ITEM              397
PTNO              397
DESC              397
DATE              397
ORD_QTY           397
CUST              397
PLNT              397
Price             397
customer PO ref   397
PO date           397
dtype: int64
```

1.1.1 Check date format

[ ]:
```
customer_order['DATE'][:2]
```

[ ]:
```
0    23-08-2016
1    23-08-2016
Name: DATE, dtype: object
```

[ ]:
```
customer_order['PO date'][:2]
```

[ ]:
```
0    2016-08-23 00:00:00
1    2206-08-19 00:00:00
Name: PO date, dtype: object
```

Fill in the missing 'PO date' with the corresponding values from 'DATE'. To do this, first get the month and day from each corresponding 'DATE' value. and using this and the year (2016) create a datetime.date and assign it to 'PO DATE'

[ ]:
```
tmp = customer_order[customer_order['PO date'] == datetime.time(0)]
day, month = [], []
for i in tmp['DATE'].str.split('-').values:
    day.append(int(i[0]))
    month.append(int(i[1]))
customer_order.loc[customer_order['PO date'] == datetime.time(0), 'PO date'] =␣
 ↪[datetime.date(2016, m, d) for m, d in zip(month, day)]
```

[ ]:
```
customer_order['PO date'] = pd.to_datetime(customer_order['PO date'],␣
 ↪format='%Y/%m/%d')
```

```
customer_order['DATE'] = pd.to_datetime(customer_order['DATE'],␣
 ↪format='%d-%m-%Y')
```

[ ]:
```
x = (customer_order['DATE'] - customer_order['PO date'])
customer_order[x.dt.days < 0]
```

[ ]:

	SONO	ITEM	PTNO	\
1	101195540	10	6114-80-7101I.	
3091	101196033	10	21M-939-2261I.	
3092	101196033	20	209-939-7120I.	
3093	101196033	30	209-939-7110I.	
3094	101196033	40	07155-01125I.	
3095	101196033	50	195-30-18271I.	
3096	101196033	60	04064-08530I.	
3097	101196033	70	707-99-56220I.	
3098	101196033	80	707-99-77160I.	
3099	101196033	90	707-99-67830I.	
3100	101196033	100	707-99-69700I.	
9547	103122797	20	202-63-N1140.	
9549	103122801	30	07000-13025I.	
9550	103122801	40	07000-13032.	
10288	103124522	20	WL10430C4381.	
10523	103124893	10	C0010010.	
10524	103124893	20	S290AFA27005.	
10525	103124893	30	S290AFA27055.	
10526	103124893	40	WL10675C2267.	
10987	103125462	10	V30641148.	
10988	103125462	20	207-60-71183I.	
10989	103125462	30	6736-51-5142I.	
12827	101190460	10	207-60-71183I.	
12828	101190460	20	600-185-4100I.	
12829	101190460	30	600-181-6740I.	
12830	101190460	40	04120-21746I.	
12831	101190461	10	20Y-26-22270I.	
12832	101190461	20	6217-71-6640I.	
12833	101190461	30	6217-71-6650I.	
12834	101190461	40	209-38-12460I.	
…	…	…	…	
31380	101193919	140	M30441178.	
31381	101193920	10	21W-06-22120I.	
31382	101193920	30	M30441178.	
31383	101193920	40	600-319-5611I.	
31384	101193920	50	6736-51-5142I.	
31385	101193920	60	707-98-45280I.	
31386	101193920	70	600-319-5611I.	
31387	101193920	80	600-319-3610I.	
31388	101193920	90	600-181-6740I.	

31389	101193920	110	707-52-15430I.
31390	101193920	120	07000-13032.
31391	101193920	130	20Y-27-11561I.
31392	101193920	140	09244-02496.
31393	101193920	150	207-60-71183I.
31394	101193920	160	04121-21744I.
31395	101193920	170	600-311-9733I.
31396	101193920	180	6732-81-3380I.
31397	101193920	190	600-185-4100I.
31398	101193920	200	D205-70-19570RCI.
31399	101193920	210	04121-21744I.
31400	101193920	220	7834-27-3003I.
31401	101193920	230	H30441174.
31402	101193920	240	K30441176.
31403	101193920	250	6736-51-5142I.
31404	101193920	260	M30741167.
31405	101193920	270	707-99-25870I.
31406	101193920	280	V30541167.
31407	101193920	290	600-319-3610I.
31408	101193920	300	206-06-61130I.
34644	101194455	10	7834-41-2003I.

	DESC	DATE	ORD_QTY	CUST
1	ELEMENT ASS'Y (GD623) ^$	2016-08-23	3	LUA1053
3091	ADAPTER (PC600LC-8R)	2016-09-07	3	A1284
3092	SHIM (PC1250-7)	2016-09-07	1	A1284
3093	SHIM (PC1250-7)	2016-09-07	1	A1284
3094	WEAR RING	2016-09-07	2	A1284
3095	PACKING (D355A-1) 195-30-14210I	2016-09-07	2	A1284
3096	SNAP RING (D275)	2016-09-07	2	A1284
3097	SERVICE KIT (PC600LC-6)	2016-09-07	2	A1284
3098	SERVICE KIT (PC600-6)	2016-09-07	2	A1284
3099	SERVICE KIT (PC600LC-6)	2016-09-07	2	A1284
3100	SERVICE KIT (PC600)	2016-09-07	2	A1284
9547	TUBE (PC130-7)	2016-05-26	1	BHP0025
9549	O RING	2016-05-26	5	BHM0072
9550	O RING (PC71)	2016-05-26	5	BHM0072
10288	HEX.HEAD BOLT M24X365-CL10.9-ZNC	2016-07-15	1	AHS0170
10523	CONNECTION BELLOW WL2010	2016-07-28	1	CGN0054
10524	O-RING FACE SEAL -12	2016-07-28	1	CGN0054
10525	O-RING BOSS END -12	2016-07-28	1	CGN0054
10526	HOSE	2016-07-28	1	CGN0054
10987	1000 Hrs FILTER KIT (PC210-8MO) (KGO)	2016-08-26	2	DES0418
10988	ELEMENT (PC210-8MO) ^$	2016-08-26	2	DES0418
10989	ENGINE OIL FILTER (PC210-8) ^$	2016-08-26	5	DES0418
12827	ELEMENT (PC210-8MO) ^$	2016-04-05	4	LUA1051
12828	FILTER ASSY (PC210LC-8) ^$	2016-04-05	2	LUA1051

12829	AIR CLEANER ELEMENT ASSY (PC200-6) ^$	2016-04-05	4	LUA1051	
12830	V-BELT (PC210-8)	2016-04-05	2	LUA1051	
12831	RING (PC200-6)	2016-04-05	2	LUA1053	
12832	CLAMP (PC600)	2016-04-05	2	LUA1053	
12833	CLAMP (PC600)	2016-04-05	2	LUA1053	
12834	FILTER (PC600) (209-38-12550I)	2016-04-05	1	LUA1053	
…	…	…	…	…	
31380	KOMATSU Powtr TO 30 (20 LTR)	2016-07-07	8	DEI0089	
31381	LAMP ASSY (PC130-7)	2016-07-07	1	DEI0067	
31382	KOMATSU Powtr TO 30 (20 LTR)	2016-07-07	6	DEI0067	
31383	CARTRIDGE (PC210-8MO) ^$	2016-07-07	6	DEI0067	
31384	ENGINE OIL FILTER (PC210-8) ^$	2016-07-07	8	DEI0067	
31385	BUCKET CYLINDER SEAL KIT (PC200-6)	2016-07-07	2	DEI0067	
31386	CARTRIDGE (PC210-8MO) ^$	2016-07-07	8	DEI0067	
31387	FUEL PRE FILTER (PC210LC-8) ^$	2016-07-07	8	DEI0067	
31388	AIR CLEANER ELEMENT ASSY (PC200-6) ^$	2016-07-07	4	DEI0067	
31389	BUSHING (07177-07030I)	2016-07-07	4	DEI0067	
31390	O RING (PC71)	2016-07-07	20	DEI0067	
31391	BOLT (PC200-6)	2016-07-07	20	DEI0067	
31392	PIN ASSY (PC200-6)	2016-07-07	30	DEI0067	
31393	ELEMENT (PC210-8MO) ^$	2016-07-07	2	DEI0067	
31394	V-BELT (PC200-6)	2016-07-07	1	DEI0067	
31395	SEPARATOR ASSY (PC200-6) ^$	2016-07-07	1	DEI0067	
31396	V-BELT (PC200-6)	2016-07-07	2	DEI0067	
31397	FILTER ASSY (PC210LC-8) ^$	2016-07-07	2	DEI0067	
31398	TOOTH DURA RC (PC200-6) ^`	2016-07-07	10	DEI0067	
31399	V-BELT (PC200-6)	2016-07-07	2	DEI0067	
31400	PUMP CONTROLLER (LTK PC200-6)(PC300-6)	2016-07-07	1	DEI0067	
31401	KOMATSU DEO 15W40 DH (20 LTR)	2016-07-07	8	DEI0067	
31402	KOMATSU HO46-HM (20 LTR)	2016-07-07	2	DEI0067	
31403	ENGINE OIL FILTER (PC210-8) ^$	2016-07-07	5	DEI0067	
31404	1000 Hrs FILTER KIT (PC200-6)	2016-07-07	2	DEI0067	
31405	BUCKET CYLINDER SEAL KIT (PC130-7)	2016-07-07	1	DEI0067	
31406	TOOTH POINT SET 25 NOS (PC130,200, 210)	2016-07-07	3	DEI0067	
31407	FUEL PRE FILTER (PC210LC-8) ^$	2016-07-07	5	DEI0067	
31408	SWITCH (PC300SE-6)	2016-07-07	1	DEI0067	
34644	GOVERNOR MOTER ASSY (PC200-6, PC130-7)	2016-07-22	1	CAS0500	

	PLNT	Price	customer PO ref	PO date
1	930.0	6024	SPR 130_ Ms MONTE CA	2206-08-19
3091	930.0	36422	1200617997	2016-09-10
3092	930.0	366	1200617997	2016-09-10
3093	930.0	303	1200617997	2016-09-10
3094	930.0	1848	1200617997	2016-09-10
3095	930.0	1763	1200617997	2016-09-10
3096	930.0	375	1200617997	2016-09-10
3097	930.0	45217	1200617997	2016-09-10

3098	930.0	82255		1200617997	2016-09-10
3099	930.0	46875		1200617997	2016-09-10
3100	930.0	67645		1200617997	2016-09-10
9547	130.0	5646		ANKIT	2016-06-01
9549	130.0	542		Madanram	2016-06-01
9550	130.0	21		Madanram	2016-06-01
10288	930.0	764		Counter bolt	2016-07-16
10523	930.0	4481		3000602559	2016-08-22
10524	930.0	6		3000602559	2016-08-22
10525	930.0	6		3000602559	2016-08-22
10526	930.0	510		3000602559	2016-08-22
10987	930.0	0		FOC NS	2016-08-28
10988	930.0	0		FOC NS	2016-08-28
10989	930.0	0		FOC NS	2016-08-28
12827	930.0	6899	SPR_2	PNC Infratech	2016-05-04
12828	930.0	10023	SPR_2	PNC Infratech	2016-05-04
12829	930.0	7217	SPR_2	PNC Infratech	2016-05-04
12830	930.0	1610	SPR_2	PNC Infratech	2016-05-04
12831	930.0	276	SPR_1	JPA Amelia	2016-05-04
12832	930.0	808	SPR_1	JPA Amelia	2016-05-04
12833	930.0	744	SPR_1	JPA Amelia	2016-05-04
12834	930.0	11275	SPR_1	JPA Amelia	2016-05-04
…	…	…		…	…
31380	930.0	2908	July 1st Order Thru		2016-07-29
31381	930.0	9419	July 1st Order Thru		2016-07-29
31382	930.0	2908	July 1st Order Thru		2016-07-29
31383	930.0	3523	July 1st Order Thru		2016-07-29
31384	930.0	1754	July 1st Order Thru		2016-07-29
31385	930.0	7365	July 1st Order Thru		2016-07-29
31386	930.0	3523	July 1st Order Thru		2016-07-29
31387	930.0	3178	July 1st Order Thru		2016-07-29
31388	930.0	7465	July 1st Order Thru		2016-07-29
31389	930.0	1812	July 1st Order Thru		2016-07-29
31390	930.0	18	July 1st Order Thru		2016-07-29
31391	930.0	314	July 1st Order Thru		2016-07-29
31392	930.0	197	July 1st Order Thru		2016-07-29
31393	930.0	7136	July 1st Order Thru		2016-07-29
31394	930.0	4259	July 1st Order Thru		2016-07-29
31395	930.0	11869	July 1st Order Thru		2016-07-29
31396	930.0	4263	July 1st Order Thru		2016-07-29
31397	930.0	10367	July 1st Order Thru		2016-07-29
31398	930.0	887	July 1st Order Thru		2016-07-29
31399	930.0	4259	July 1st Order Thru		2016-07-29
31400	930.0	29277	July 1st Order Thru		2016-07-29
31401	930.0	3137	July 1st Order Thru		2016-07-29
31402	930.0	2658	July 1st Order Thru		2016-07-29
31403	930.0	1754	July 1st Order Thru		2016-07-29

```
31404  930.0  23786   July 1st Order Thru 2016-07-29
31405  930.0   5463   July 1st Order Thru 2016-07-29
31406  930.0  16292   July 1st Order Thru 2016-07-29
31407  930.0   3178   July 1st Order Thru 2016-07-29
31408  930.0   4017   July 1st Order Thru 2016-07-29
34644  930.0  59696  SRL/16-17/053 (Anand 2016-07-26

[83 rows x 11 columns]
```

As you can see above there are still ~80 rows which have date > po date, this could perhaps be deleted.

[ ]:
```
customer_order.drop(customer_order[x.dt.days < 0].index, inplace=True)
```

1.2 material_master

[ ]:
```
material_master.head()
```

[ ]:
```
  Material code`            Material Description Type Unit  Model  \
0  01010-61435I.             BOLT (01010-51435I)  ROH   EA  PC450
1  01010-61455I.  BOLT (D65E-12) (01010-31455I.)  ROH   EA    D65
2  01010-61635I.                     BOLT (GD511)  ROH   EA  GD511
3  01010-61645I.            BOLT (01010-81645I.)  ROH   EA   D475
4  01010-61650I.    BOLT (HD465) (01010-81650I.)  ROH   EA  HD465

   safety stock  Demand
0             8      30
1             1       2
2             1      12
3             8      32
4             5      18
```

[ ]:
```
pd.isnull(material_master).describe()
```

[ ]:
```
       Material code` Material Description   Type   Unit  Model safety stock  \
count            6022                 6022   6022   6022   6022         6022
unique              1                    1      1      1      1            1
top             False                False  False  False  False        False
freq             6022                 6022   6022   6022   6022         6022

       Demand
count    6022
unique      1
top     False
freq     6022
```

No null values here, lets just remove unwated columns: 1. Type 2. Unit

[ ]:
```
material_master.drop(['Type', 'Unit'], axis=1, inplace=True)
```

[ ]:
```
material_master.columns
```

[ ]:
```
Index([u'Material code`', u'Material Description', u'Model', u'safety stock',
       u'Demand'],
      dtype='object')
```

[ ]:
```
material_master.rename(columns={'Material code`': 'Material code'},␣
↪inplace=True)
```

1.3 sales past demand

[ ]:
```
sales_past_demand.head()
```

[ ]:

	Material code`	DEM36	DEM35	DEM34	DEM33	DEM32	DEM31	DEM30	DEM29	\
0	01010-61435I.	6	0	0	0	0	0	5	0	
1	01010-61455I.	0	0	0	0	0	0	0	0	
2	01010-61635I.	0	0	0	0	0	0	0	0	
3	01010-61645I.	0	0	0	0	0	0	0	0	
4	01010-61650I.	0	0	0	0	0	0	0	0	

	DEM28	…	DEM10	DEM9	DEM8	DEM7	DEM6	DEM5	DEM4	DEM3	DEM2	DEM1
0	0	…	0	4	0	0	16	12	2	0	0	0
1	0	…	0	0	0	0	0	0	0	2	0	0
2	0	…	0	0	0	0	0	0	0	8	1	0
3	0	…	0	0	0	0	0	0	0	0	2	0
4	0	…	0	0	0	0	0	2	0	0	0	0

[5 rows x 37 columns]

This holds the sales demands for given materials for the last 36months, this could be used to make predicitons for future demands. No pre-processing required here

[ ]:
```
sales_past_demand.columns
```

[ ]:
```
Index([u'Material code`', u'DEM36', u'DEM35', u'DEM34', u'DEM33', u'DEM32',
       u'DEM31', u'DEM30', u'DEM29', u'DEM28', u'DEM27', u'DEM26', u'DEM25',
       u'DEM24', u'DEM23', u'DEM22', u'DEM21', u'DEM20', u'DEM19', u'DEM18',
       u'DEM17', u'DEM16', u'DEM15', u'DEM14', u'DEM13', u'DEM12', u'DEM11',
       u'DEM10', u'DEM9', u'DEM8', u'DEM7', u'DEM6', u'DEM5', u'DEM4', u'DEM3',
       u'DEM2', u'DEM1'],
      dtype='object')
```

[ ]:
```
sales_past_demand.rename(columns={'Material code`': 'Material code'},␣
↪inplace=True)
```

1.4 customer_master

[ ]:
```
customer_master.head()
```

[ ]:
```
  customer code                     Name                                Street  \
0         A1283                ACC LIMITED                   P O CEMENT NAGAR
1         A1284  ASSOCIATED CEMENT CO LTD                         P.O. KYMORE
2         A1285                ACC LIMITED          1 P.O.JAMUL CEMENT WORKS,
3         A1438     AMBUJA CEMENT LIMITED              (Unit : Ambujanagar),
4         A1896       AMBUJA CEMENTS LTD.  P.O. Rawan, Tehsil: Baloda Bazar

                      City  PostalCode          Region Industry
0               CHANDRAPUR      442502     Maharashtra   Mining
1                    KATNI      483880  Madhya Pradesh   Cement
2       DIST : DURG (C.G.)      490024     Chattisgarh   Mining
3  Taluka Kodinar,Junagadh      362715         Gujarat   Cement
4                   RAIPUR      493331     Chattisgarh   Mining
```

[ ]:
```
pd.isnull(customer_master).describe()
```

[ ]:
```
        customer code  Name  Street   City PostalCode Region Industry
count             976   976    976    976        976    976      976
unique              1     1      1      1          1      1        1
top             False  False  False  False      False  False    False
freq              976   976    976    976        976    976      976
```

No null values here, lets remove the unwated columns: 1. Street, 2. City

[ ]:
```
customer_master.drop(['Street', 'City'], axis=1, inplace=True)
```

[ ]:
```
customer_master.columns
```

[ ]:
```
Index([u'customer code', u'Name ', u'PostalCode', u'Region', u'Industry'],
dtype='object')
```

[ ]:
```
customer_master.rename(columns={'Name ': 'Name'}, inplace=True)
```

1.5 invoice

[ ]:
```
invoice.head()
```

[ ]:
```
   Bill.Doc.  Item   Material code                  Description  \
0   90025694    10  7861-93-2330I.                SENSOR (PC300)
1   90025696    60  207-32-61260I.  SEAL (PC300) (150-32-11260I)
2   90025696   120  207-70-34230I.           BUSHING (PC300SE-7)
3   90025697    10  23B-27-11132I.                   HUB (GD623)
4   90025698    20  6732-51-8330I.              O-RING (PC200-6)
```

```
   Required quantity  Billed Quantity  Value  delivery doc  RefItm  Sales ord  \
0                  2                2  11759      81223222      10  101192139
1                 12                9   1599      81224013      60  101192000
2                  2                1   6267      81224013     120  101192000
3                  1                1  31718      81223587      10  101190426
4                  2                1     60      81223590      20  101191683

   so Item  Plnt  Bill date
0       10   930 2016-05-26
1       60   930 2016-05-26
2      120   930 2016-05-26
3       10   930 2016-05-26
4       20   930 2016-05-26
```

[ ]:
```
pd.isnull(invoice).describe()
```

[ ]:
```
        Bill.Doc.   Item Material code Description Required quantity  \
count      39343  39343         39343       39343             39343
unique         1      1             1           1                 1
top        False  False         False       False             False
freq       39343  39343         39343       39343             39343

        Billed Quantity  Value delivery doc RefItm Sales ord so Item   Plnt  \
count             39343  39343        39343  39343     39343   39343  39343
unique                1      1            1      1         1       1      1
top               False  False        False  False     False   False  False
freq              39343  39343        39343  39343     39343   39343  39343

        Bill date
count       39343
unique          1
top         False
freq        39343
```

No null values here too. Lets remove unwanted columns: 1. Refitm 2. so item

[ ]:
```
invoice.drop(['so Item', 'RefItm'], axis=1, inplace=True)
```

Convert the date to pd_datetime

[ ]:
```
invoice['Bill date'] = pd.to_datetime(invoice['Bill date'], format='%Y/%m/%d')
```

[ ]:
```
invoice.columns
```

[ ]:
```
Index([u'Bill.Doc.', u'Item', u'Material code', u'Description',
       u'Required quantity', u'Billed Quantity', u'Value', u'delivery doc',
       u'Sales ord', u'Plnt', u'Bill date'],
      dtype='object')
```

1.6 stock_master

[ ]:
```
stock_master.head()
```

[ ]:
```
        Material  ValA  DocumentNo  Year  Itm    D/C  Amount     Quantity BUn  \
0  01010-61435I.   930   920006718  2016    2  Recpt  167.56            4  EA
1  01010-61435I.   930   920006757  2016   21   Issu  167.56            4  EA
2  01010-61435I.   930   920009004  2016    2  Recpt  502.68           12  EA
3  01010-61435I.   930   920009049  2016    1   Issu  502.68           12  EA
4  01010-61435I.   930   920011287  2016   13  Recpt  708.00           15  EA

  Pstng Date
0 2016-04-22
1 2016-04-22
2 2016-04-29
3 2016-04-29
4 2016-05-04
```

[ ]:
```
pd.isnull(stock_master).describe()
```

[ ]:
```
       Material   ValA DocumentNo   Year    Itm    D/C Amount    Quantity  \
count     68555  68555      68555  68555  68555  68555  68555       68555
unique        1      1          1      1      1      1      1           1
top       False  False      False  False  False  False  False       False
freq      68555  68555      68555  68555  68555  68555  68555       68555

          BUn Pstng Date
count   68555      68555
unique      1          1
top     False      False
freq    68555      68555
```

No null values here. Lets remove unwanted columns: 1. Year 2. ValA

[ ]:
```
stock_master.drop(['ValA', 'Year'], axis=1, inplace=True)
```

Convert date to pd_datetime

[ ]:
```
stock_master['Pstng Date'] = pd.to_datetime(stock_master['Pstng Date'],␣
 ↪format='%Y/%m/%d')
```

[ ]:
```
stock_master.columns
```

[ ]:
```
Index([u'Material', u'DocumentNo', u'Itm', u'D/C', u'Amount', u'   Quantity',
       u'BUn', u'Pstng Date'],
      dtype='object')
```

[ ]:
```
stock_master.rename(columns={'   Quantity': 'Quantity'}, inplace=True)
```

1.7 delivery data

[ ]:
```
delivery_data.head()
```

[ ]:
```
   Delivery no  delivery Item        Material  Plnt  Delivery quantity Unit  \
0     81211954             10  600-181-6740I.   299                   1   EA
1     81211955             10  600-181-6740I.   299                   1   EA
2     81211957             10  600-411-1151I.   299                   1   EA
3     81211967             10  600-411-1151I.   299                   1   EA
4     81212008             10   WL10670A2582.   930                   1   EA

        date                              Description  sales ord   \
0 2016-04-04   AIR CLEANER ELEMENT ASSY (PC200-6) ^$   103121287
1 2016-04-04   AIR CLEANER ELEMENT ASSY (PC200-6) ^$   103121288
2 2016-04-04  CORROSION RESISTOR FILTER (PC200-6) ^$   103121290
3 2016-04-04  CORROSION RESISTOR FILTER (PC200-6) ^$   103121289
4 2016-04-05                    TUBE-LIFT BORE SPOOL   103121296

   sale ord item
0             10
1             10
2             10
3             10
4             10
```

Dropping plant and unit, as its not needed

[ ]:
```
delivery_data.drop(['Plnt', 'Unit'], axis=1, inplace=True)
```

[ ]:
```
pd.isnull(delivery_data).describe()
```

[ ]:
```
        Delivery no delivery Item Material Delivery quantity   date  \
count         40356         40356    40356             40356  40356
unique            1             1        1                 1      1
top           False         False    False             False  False
freq          40356         40356    40356             40356  40356

        Description sales ord  sale ord item
count         40356      40356         40356
unique            1          1             1
top           False      False         False
freq          40356      40356         40356
```

No missing data here

[ ]:
```
delivery_data.columns
```

```
[ ]: Index([u'Delivery no', u'delivery Item', u'Material', u'Delivery quantity',
            u'date', u'Description', u'sales ord ', u'sale ord item'],
           dtype='object')
```

```
[ ]: delivery_data.rename(columns={'sales ord ': 'sales ord'}, inplace=True)
```

1.8 booking_data

```
[ ]: booking_data.head()
```

```
[ ]:    Delivery no Delivery date  ShPt SOrg.          Consignment details  \
    0     81211954    2016-04-04   299    50  LOCAL COLLECTION - PRASHANT
    1     81211955    2016-04-04   299    50  LOCAL COLLECTION - PRASHANT
    2     81211957    2016-04-04   299    50  LOCAL COLLECTION - PRASHANT
    3     81211967    2016-04-04   299    50  LOCAL COLLECTION - PRASHANT
    4     81212008    2016-04-05   930    50     BYHAND RAKESH 05-04-2016

        GC date Recpt date
    0 2016-04-11 2016-05-05
    1 2016-04-11 2016-05-05
    2 2016-04-11 2016-05-05
    3 2016-04-11 2016-05-05
    4 2016-04-05 2016-04-06
```

```
[ ]: pd.isnull(booking_data).describe()
```

```
[ ]:        Delivery no Delivery date   ShPt  SOrg. Consignment details GC date  \
    count          8445          8445   8445   8445                8445    8445
    unique            1             1      1      1                   2       2
    top           False         False  False  False               False   False
    freq           8445          8445   8445   8445                7559    8441

           Recpt date
    count        8445
    unique          2
    top         False
    freq         8443
```

There are null values in Consigment details, GC date and Recpt date, lets have a look

```
[ ]: booking_data.loc[pd.isnull(booking_data['Consignment details']), 'Consignment␣
     ↪details'] = "No details"
```

For GC date and Recpt date, we can only 3~4 rows are missing data, lets drop these rows

```
[ ]: booking_data.drop(booking_data[pd.isnull(booking_data['GC date'])].index,␣
     ↪inplace=True)
```

[ ]:
```
booking_data[pd.isnull(booking_data['Recpt date'])]
```

[ ]:
```
Empty DataFrame
Columns: [Delivery no, Delivery date, ShPt, SOrg., Consignment details, GC date,
Recpt date]
Index: []
```

Luckily the fields which were missing the GC date and Recpt date, were overlapping. Lets drop unwanted rows: 1. ShPt 2. SOrg.

[ ]:
```
booking_data.drop(['ShPt', 'SOrg.'], axis=1, inplace=True)
```

2 Merging the tables

Since there are multiple tables and there is a strong relation amongst these tables, we could merge the tables for easier access and manipulation

2.0.1 Customer Order and Master

[ ]:
```
customer_order.head(2)
```

[ ]:
```
        SONO  ITEM            PTNO                              DESC  \
0  101195538    10  707-99-67090I.  ARM CYLINDER SEAL KIT (PC300-7)
2  101195542    10  723-40-50601I.           RELIEF VALVE (PC200-6)

        DATE  ORD_QTY     CUST   PLNT  Price customer PO ref    PO date
0 2016-08-23        1  CNG0101  930.0  14253      0100048101 2016-08-23
2 2016-08-23        2  BAC0093  930.0   9113      0100048104 2016-08-23
```

[ ]:
```
customer_master.head(2)
```

[ ]:
```
  customer code                     Name  PostalCode          Region Industry
0         A1283                ACC LIMITED      442502     Maharashtra   Mining
1         A1284  ASSOCIATED CEMENT CO LTD      483880  Madhya Pradesh   Cement
```

[ ]:
```
customer = pd.merge(customer_master, customer_order, left_on=['customer code'],␣
↪right_on=['CUST'])
```

[ ]:
```
customer.drop(['CUST'], axis=1, inplace=True)
```

[ ]:
```
print('Customer order:', len(customer_order))
print('Customer master:', len(customer_master))
print('Customer:', len(customer))
```

```
('Customer order:', 39315)
('Customer master:', 976)
('Customer:', 39315)
```

No data loss here

2.1 Bill

[ ]:
```
print(len(invoice))
invoice.head(2)
```

39343

[ ]:
```
   Bill.Doc.  Item   Material code                  Description  \
0   90025694    10  7861-93-2330I.                SENSOR (PC300)
1   90025696    60  207-32-61260I.  SEAL (PC300) (150-32-11260I)

   Required quantity  Billed Quantity  Value  delivery doc  Sales ord  Plnt  \
0                  2                2  11759      81223222  101192139   930
1                 12                9   1599      81224013  101192000   930

   Bill date
0 2016-05-26
1 2016-05-26
```

[ ]:
```
print(len(delivery_data))
delivery_data.head(2)
```

40356

[ ]:
```
   Delivery no  delivery Item        Material  Delivery quantity       date  \
0     81211954             10  600-181-6740I.                  1 2016-04-04
1     81211955             10  600-181-6740I.                  1 2016-04-04

                            Description  sales ord  sale ord item
0  AIR CLEANER ELEMENT ASSY (PC200-6) ^$  103121287             10
1  AIR CLEANER ELEMENT ASSY (PC200-6) ^$  103121288             10
```

We can see that there is no unique key, lets try and find a combination of keys to get a unique key

[ ]:
```
delivery_data[(delivery_data['sales ord'] == 101190353) & (delivery_data['sale␣
 ↪ord item'] == 50)]
```

[ ]:
```
       Delivery no  delivery Item       Material  Delivery quantity  \
809       81213104             50  07000-12115I.                  3
15513     81229842             50  07000-12115I.                  7

            date    Description  sales ord  sale ord item
809   2016-04-11  O-RING (D275)  101190353             50
15513 2016-06-16  O-RING (D275)  101190353             50
```

[ ]:
```
invoice[(invoice['Sales ord'] == 101190353) & (invoice['Item'] == 50)]
```

[ ]:
```
        Bill.Doc.  Item  Material code    Description  Required quantity  \
2554    167206633    50  07000-12115I.  O-RING (D275)                 10
14231   167210377    50  07000-12115I.  O-RING (D275)                 10

        Billed Quantity  Value  delivery doc  Sales ord  Plnt  Bill date
2554                  3    698      81213104  101190353   930 2016-04-12
14231                 7   1628      81229842  101190353   930 2016-06-17
```

Using [‘Sales ord’, ‘Item’, ‘Billed Quantity’] from invoice and [‘sales ord’, ‘sale ord item’, ‘Delivery quantity’] from delivery data, we can get the unique row for merging

[ ]:
```
bill = pd.merge(invoice, delivery_data, left_on=['Sales ord', 'Item', 'Billed␣
↪Quantity'], right_on=['sales ord', 'sale ord item', 'Delivery quantity'])
```

[ ]:
```
bill.head(2)
```

[ ]:
```
   Bill.Doc.  Item   Material code   Description_x  Required quantity  \
0   90025694    10  7861-93-2330I.  SENSOR (PC300)                  2
1  167208916    10  7861-93-2330I.  SENSOR (PC300)                  2

   Billed Quantity  Value  delivery doc  Sales ord  Plnt  Bill date  \
0                2  11759      81223222  101192139   930 2016-05-26
1                2  11759      81223222  101192139   930 2016-05-24

   Delivery no  delivery Item        Material  Delivery quantity       date  \
0     81226681             10  7861-93-2330I.                  2 2016-06-06
1     81226681             10  7861-93-2330I.                  2 2016-06-06

    Description_y  sales ord  sale ord item
0  SENSOR (PC300)  101192139             10
1  SENSOR (PC300)  101192139             10
```

[ ]:
```
bill.drop(['sales ord', 'sale ord item', 'Delivery quantity'], axis=1,␣
↪inplace=True)
```

[ ]:
```
bill.drop(['Description_x'], axis = 1, inplace=True)
bill.rename(columns={'Description_y':'Description'}, inplace=True)
```

[ ]:
```
bill.rename(columns={'date':'delivery date'}, inplace=True)
```

[ ]:
```
(bill['delivery Item'] == bill['Item']).value_counts()
```

[ ]:
```
True    40307
dtype: int64
```

[ ]:
```
bill.drop(['Item'], axis=1, inplace=True)
```

[ ]:
```
len(bill.groupby(['Delivery no']).count())
```

[ ]: 7844

[ ]:
```
bill.head(2)
```

[ ]:
```
   Bill.Doc.   Material code  Required quantity  Billed Quantity  Value  \
0   90025694  7861-93-2330I.                  2                2  11759
1  167208916  7861-93-2330I.                  2                2  11759

   delivery doc  Sales ord  Plnt  Bill date  Delivery no  delivery Item  \
0      81223222  101192139   930 2016-05-26     81226681             10
1      81223222  101192139   930 2016-05-24     81226681             10

         Material delivery date     Description
0  7861-93-2330I.    2016-06-06  SENSOR (PC300)
1  7861-93-2330I.    2016-06-06  SENSOR (PC300)
```

[ ]:
```
print(len(booking_data))
booking_data.head(2)
```

```
8441
```

[ ]:
```
   Delivery no Delivery date          Consignment details    GC date  \
0     81211954    2016-04-04  LOCAL COLLECTION - PRASHANT 2016-04-11
1     81211955    2016-04-04  LOCAL COLLECTION - PRASHANT 2016-04-11

  Recpt date
0 2016-05-05
1 2016-05-05
```

Now we can merge the booking data onto the bill

[ ]:
```
bill = pd.merge(bill, booking_data, on='Delivery no', how='left')
```

[ ]:
```
bill.head(2)
```

[ ]:
```
   Bill.Doc.   Material code  Required quantity  Billed Quantity  Value  \
0   90025694  7861-93-2330I.                  2                2  11759
1  167208916  7861-93-2330I.                  2                2  11759

   delivery doc  Sales ord  Plnt  Bill date  Delivery no  delivery Item  \
0      81223222  101192139   930 2016-05-26     81226681             10
1      81223222  101192139   930 2016-05-24     81226681             10

         Material delivery date     Description Delivery date  \
0  7861-93-2330I.    2016-06-06  SENSOR (PC300)    2016-06-06
1  7861-93-2330I.    2016-06-06  SENSOR (PC300)    2016-06-06
```

```
           Consignment details    GC date Recpt date
0  806869210 TCI 09-06-2016 GD 2016-06-06 2016-06-17
1  806869210 TCI 09-06-2016 GD 2016-06-06 2016-06-17
```

3 Analysis of Duplicates

3.1 Bill

[ ]:
```
bill[bill['Sales ord'] == 101192139]
```

[ ]:
```
   Bill.Doc.   Material code  Required quantity  Billed Quantity  Value  \
0   90025694  7861-93-2330I.                  2                2  11759
1  167208916  7861-93-2330I.                  2                2  11759
2  167209909  7861-93-2330I.                  2                2  11759

   delivery doc  Sales ord  Plnt  Bill date  Delivery no  delivery Item  \
0      81223222  101192139   930 2016-05-26     81226681             10
1      81223222  101192139   930 2016-05-24     81226681             10
2      81226681  101192139   930 2016-06-09     81226681             10

         Material delivery date     Description Delivery date  \
0  7861-93-2330I.    2016-06-06  SENSOR (PC300)    2016-06-06
1  7861-93-2330I.    2016-06-06  SENSOR (PC300)    2016-06-06
2  7861-93-2330I.    2016-06-06  SENSOR (PC300)    2016-06-06

           Consignment details    GC date Recpt date
0  806869210 TCI 09-06-2016 GD 2016-06-06 2016-06-17
1  806869210 TCI 09-06-2016 GD 2016-06-06 2016-06-17
2  806869210 TCI 09-06-2016 GD 2016-06-06 2016-06-17
```

[ ]:
```
bill[bill.duplicated(['Sales ord', 'Delivery no', 'Description', 'Consignment␣
↪details', 'Value'])]
```

[ ]:
```
       Bill.Doc.     Material code  Required quantity  Billed Quantity  Value  \
1      167208916    7861-93-2330I.                  2                2  11759
2      167209909    7861-93-2330I.                  2                2  11759
4      167209055    207-32-61260I.                 12                9   1599
5      167209887    207-32-61260I.                 12                9   1599
10     167209055    207-70-34230I.                  2                1   6267
11     167209055    207-70-34230I.                  2                1   6267
12     167209887    207-70-34230I.                  2                1   6267
13     167209887    207-70-34230I.                  2                1   6267
15     167209034    23B-27-11132I.                  1                1  31718
16     167209884    23B-27-11132I.                  1                1  31718
21     167209028    6732-51-8330I.                  2                1     60
22     167209028    6732-51-8330I.                  2                1     60
```

23	167209895	6732-51-8330I.	2		1	60
24	167209895	6732-51-8330I.	2		1	60
26	167209028	707-98-12690I.	1		1	4334
27	167209895	707-98-12690I.	1		1	4334
29	167209028	7700-85-8610I.	1		1	1892
30	167209895	7700-85-8610I.	1		1	1892
32	167209028	23B-27-31690I.	1		1	20980
33	167209895	23B-27-31690I.	1		1	20980
35	167209027	KD0-23750-0050I.	3		3	5608
36	167209894	KD0-23750-0050I.	3		3	5608
38	167209027	ND028530-1100I.	3		3	524
39	167209894	ND028530-1100I.	3		3	524
42	167209027	ND053660-0443I.	2		1	583
43	167209027	ND053660-0443I.	2		1	583
44	167209894	ND053660-0443I.	2		1	583
45	167209894	ND053660-0443I.	2		1	583
46	167210178	ND053660-0443I.	2		1	583
47	167210178	ND053660-0443I.	2		1	583
…	…	…	…	…	…	
39331	335007144	D01139814.	2		1	30377
39332	335007144	D01139814.	2		1	30377
39389	335007262	L11430941.	6		3	21070
39390	335007262	L11430941.	6		3	21070
39484	335007170	A03731957.	3		3	4160
39485	335007170	Q03231997.	3		3	6032
39492	335007286	U20131803.	2		1	7698
39493	335007286	U20131803.	2		1	7698
39521	335007273	D02649695.	6		3	5379
39522	335007273	D02649695.	6		3	5379
39567	335007198	G30249675.	2		1	50542
39568	335007198	G30249675.	2		1	50542
39626	335007277	T20854646.	2		1	5409
39627	335007277	T20854646.	2		1	5409
39687	335007285	R00030444.	30		15	354
39688	335007285	R00030444.	30		15	354
39698	335007278	J10234548.	2		1	2665
39699	335007278	J10234548.	2		1	2665
39702	335007391	F1249664.	2		1	22734
39703	335007391	F1249664.	2		1	22734
39781	335007286	P21451055.	1		1	3998
39857	335007319	600-319-3841I.	2		1	5787
39858	335007319	600-319-3841I.	2		1	5787
39899	335007394	C20141884.	8		4	5622
39900	335007394	C20141884.	8		4	5622
39974	335007397	W20251014.	2		1	2355
39975	335007397	W20251014.	2		1	2355
40056	369003442	20Y-60-21470I.	2		1	353

```
40057  369003442    20Y-60-21470I.                  2                1    353
40132  369003463        K20036942.                 20               20    413

       delivery doc  Sales ord  Plnt  Bill date  Delivery no  delivery Item  \
1          81223222  101192139   930 2016-05-24     81226681             10
2          81226681  101192139   930 2016-06-09     81226681             10
4          81224013  101192000   930 2016-05-26     81227687             60
5          81227687  101192000   930 2016-06-09     81227687             60
10         81224013  101192000   930 2016-05-26     81222304            120
11         81224013  101192000   930 2016-05-26     81227687            120
12         81227687  101192000   930 2016-06-09     81222304            120
13         81227687  101192000   930 2016-06-09     81227687            120
15         81223587  101190426   930 2016-05-26     81227680             10
16         81227680  101190426   930 2016-06-09     81227680             10
21         81223590  101191683   930 2016-05-26     81219949             20
22         81223590  101191683   930 2016-05-26     81227685             20
23         81227685  101191683   930 2016-06-09     81219949             20
24         81227685  101191683   930 2016-06-09     81227685             20
26         81223590  101191683   930 2016-05-26     81227685            120
27         81227685  101191683   930 2016-06-09     81227685            120
29         81223590  101191683   930 2016-05-26     81227685            130
30         81227685  101191683   930 2016-06-09     81227685            130
32         81223590  101191683   930 2016-05-26     81227685            160
33         81227685  101191683   930 2016-06-09     81227685            160
35         81223588  101191636   930 2016-05-26     81227683             10
36         81227683  101191636   930 2016-06-09     81227683             10
38         81223588  101191636   930 2016-05-26     81227683             20
39         81227683  101191636   930 2016-06-09     81227683             20
42         81223588  101191636   930 2016-05-26     81227683             30
43         81223588  101191636   930 2016-05-26     81228956             30
44         81227683  101191636   930 2016-06-09     81227683             30
45         81227683  101191636   930 2016-06-09     81228956             30
46         81228956  101191636   930 2016-06-14     81227683             30
47         81228956  101191636   930 2016-06-14     81228956             30
…             …         …   …        …            …            …
39331      81226985  101191952   333 2016-06-06     81226984             50
39332      81226985  101191952   333 2016-06-06     81226985             50
39389      81243900  101193170   333 2016-08-17     81232159            160
39390      81243900  101193170   333 2016-08-17     81243900            160
39484      81233560  101192462   333 2016-06-30     81233560            530
39485      81233560  101192462   333 2016-06-30     81233560            540
39492      81246931  101192462   333 2016-08-30     81233560            600
39493      81246931  101192462   333 2016-08-30     81246931            600
39521      81246934  101193698   333 2016-08-30     81233721             30
39522      81246934  101193698   333 2016-08-30     81246934             30
39567      81239894  101194235   333 2016-07-28     81239893            170
39568      81239894  101194235   333 2016-07-28     81239894            170
```

39626	81246944	101194399	333	2016-08-30	81240539	80
39627	81246944	101194399	333	2016-08-30	81246944	80
39687	81246943	101194252	333	2016-08-30	81240550	390
39688	81246943	101194252	333	2016-08-30	81246943	390
39698	81246948	101194955	333	2016-08-30	81241441	20
39699	81246948	101194955	333	2016-08-30	81246948	20
39702	81260355	101194955	333	2016-10-28	81241441	30
39703	81260355	101194955	333	2016-10-28	81260355	30
39781	81246931	101192462	333	2016-08-30	81246931	420
39857	81249435	101196105	333	2016-09-12	81249434	10
39858	81249435	101196105	333	2016-09-12	81249435	10
39899	81260358	101196756	333	2016-10-28	81253602	40
39900	81260358	101196756	333	2016-10-28	81260358	40
39974	81260365	101197116	333	2016-10-28	81254159	230
39975	81260365	101197116	333	2016-10-28	81260365	230
40056	81226010	101191109	320	2016-05-31	81218184	40
40057	81226010	101191109	320	2016-05-31	81226010	40
40132	81240605	101194794	320	2016-07-30	81240605	40

	Material	delivery date	\
1	7861-93-2330I.	2016-06-06	
2	7861-93-2330I.	2016-06-06	
4	207-32-61260I.	2016-06-08	
5	207-32-61260I.	2016-06-08	
10	207-70-34230I.	2016-05-19	
11	207-70-34230I.	2016-06-08	
12	207-70-34230I.	2016-05-19	
13	207-70-34230I.	2016-06-08	
15	23B-27-11132I.	2016-06-08	
16	23B-27-11132I.	2016-06-08	
21	6732-51-8330I.	2016-05-09	
22	6732-51-8330I.	2016-06-08	
23	6732-51-8330I.	2016-05-09	
24	6732-51-8330I.	2016-06-08	
26	707-98-12690I.	2016-06-08	
27	707-98-12690I.	2016-06-08	
29	7700-85-8610I.	2016-06-08	
30	7700-85-8610I.	2016-06-08	
32	23B-27-31690I.	2016-06-08	
33	23B-27-31690I.	2016-06-08	
35	KD0-23750-0050I.	2016-06-08	
36	KD0-23750-0050I.	2016-06-08	
38	ND028530-1100I.	2016-06-08	
39	ND028530-1100I.	2016-06-08	
42	ND053660-0443I.	2016-06-08	
43	ND053660-0443I.	2016-06-13	
44	ND053660-0443I.	2016-06-08	

```
45      ND053660-0443I.    2016-06-13
46      ND053660-0443I.    2016-06-08
47      ND053660-0443I.    2016-06-13
…                  …           …
39331        D01139814.    2016-06-06
39332        D01139814.    2016-06-06
39389        L11430941.    2016-06-27
39390        L11430941.    2016-08-17
39484        A03731957.    2016-06-30
39485        Q03231997.    2016-06-30
39492        U20131803.    2016-06-30
39493        U20131803.    2016-08-30
39521        D02649695.    2016-06-30
39522        D02649695.    2016-08-30
39567        G30249675.    2016-07-28
39568        G30249675.    2016-07-28
39626        T20854646.    2016-07-30
39627        T20854646.    2016-08-30
39687        R00030444.    2016-07-30
39688        R00030444.    2016-08-30
39698        J10234548.    2016-08-04
39699        J10234548.    2016-08-30
39702         F1249664.    2016-08-04
39703         F1249664.    2016-10-28
39781        P21451055.    2016-08-30
39857    600-319-3841I.    2016-09-12
39858    600-319-3841I.    2016-09-12
39899        C20141884.    2016-09-29
39900        C20141884.    2016-10-28
39974        W20251014.    2016-09-30
39975        W20251014.    2016-10-28
40056    20Y-60-21470I.    2016-04-30
40057    20Y-60-21470I.    2016-05-31
40132        K20036942.    2016-07-30

                                    Description Delivery date  \
1                                  SENSOR (PC300)    2016-06-06
2                                  SENSOR (PC300)    2016-06-06
4                 SEAL (PC300) (150-32-11260I)    2016-06-08
5                 SEAL (PC300) (150-32-11260I)    2016-06-08
10                         BUSHING (PC300SE-7)    2016-05-19
11                         BUSHING (PC300SE-7)    2016-06-08
12                         BUSHING (PC300SE-7)    2016-05-19
13                         BUSHING (PC300SE-7)    2016-06-08
15                                 HUB (GD623)    2016-06-08
16                                 HUB (GD623)    2016-06-08
21                            O-RING (PC200-6)    2016-05-09
```

22	O-RING (PC200-6)	2016-06-08
23	O-RING (PC200-6)	2016-05-09
24	O-RING (PC200-6)	2016-06-08
26	SERVICE KIT (GD555)	2016-06-08
27	SERVICE KIT (GD555)	2016-06-08
29	SENSOR GD555	2016-06-08
30	SENSOR GD555	2016-06-08
32	KIT (GD555)	2016-06-08
33	KIT (GD555)	2016-06-08
35	BRUSH HOLDER	2016-06-08
36	BRUSH HOLDER	2016-06-08
38	BRUSH ASS'Y	2016-06-08
39	BRUSH ASS'Y	2016-06-08
42	PLUNGER (GD555)	2016-06-08
43	PLUNGER (GD555)	2016-06-13
44	PLUNGER (GD555)	2016-06-08
45	PLUNGER (GD555)	2016-06-13
46	PLUNGER (GD555)	2016-06-08
47	PLUNGER (GD555)	2016-06-13
…	…	…
39331	BEARING (D1139814.)	2016-06-06
39332	BEARING (D1139814.)	2016-06-06
39389	38/DT PIN , DIA 75 (300CKE-2)	2016-06-27
39390	38/DT PIN , DIA 75 (300CKE-2)	2016-08-17
39484	HOSE L500 (90CKD-3)	2016-06-30
39485	HOSE B 15.5 L630 MMC NUT (90CKD-3)	2016-06-30
39492	HOSE (UOM - METERS - PUR MOQ - 30 METER	2016-06-30
39493	HOSE (UOM - METERS - PUR MOQ - 30 METER	2016-08-30
39521	SEAL KIT	2016-06-30
39522	SEAL KIT	2016-08-30
39567	SEAL KIT (300CK B/A STICK P305/27)	2016-07-28
39568	SEAL KIT (300CK B/A STICK P305/27)	2016-07-28
39626	PILOT HOSE KIT - CABIN FLOOR HYD CKT 300	2016-07-30
39627	PILOT HOSE KIT - CABIN FLOOR HYD CKT 300	2016-08-30
39687	O RING (300CK)	2016-07-30
39688	O RING (300CK)	2016-08-30
39698	STOP NUT M 140X (300CKD)	2016-08-04
39699	STOP NUT M 140X (300CKD)	2016-08-30
39702	SEAL KIT (180CKD)	2016-08-04
39703	SEAL KIT (180CKD)	2016-10-28
39781	TUBE	2016-08-30
39857	CARTRIDGE (PC450-7) ^$	2016-09-12
39858	CARTRIDGE (PC450-7) ^$	2016-09-12
39899	H.T. BRUSH (300CKE-2)	2016-09-29
39900	H.T. BRUSH (300CKE-2)	2016-10-28
39974	TUBE DIA 19X27 (300CKD)	2016-09-30
39975	TUBE DIA 19X27 (300CKD)	2016-10-28

```
40056     BREATHER ELEMENT (PC200-6/PC210-8) ^$    2016-04-30
40057     BREATHER ELEMENT (PC200-6/PC210-8) ^$    2016-05-31
40132                   WORM HOSE CLIP (300CKD)    2016-07-30

               Consignment details    GC date Recpt date
1      806869210 TCI 09-06-2016 GD 2016-06-06 2016-06-17
2      806869210 TCI 09-06-2016 GD 2016-06-06 2016-06-17
4        442177440 GATI 10-06-2016 2016-06-08 2016-06-15
5        442177440 GATI 10-06-2016 2016-06-08 2016-06-15
10        725260093 XPS 19-05-2016 2016-05-19 2016-05-21
11       442177440 GATI 10-06-2016 2016-06-08 2016-06-15
12        725260093 XPS 19-05-2016 2016-05-19 2016-05-21
13       442177440 GATI 10-06-2016 2016-06-08 2016-06-15
15       442177426 GATI 09-06-2016 2016-06-08 2016-06-15
16       442177426 GATI 09-06-2016 2016-06-08 2016-06-15
21       437704293 GATI 10-05-2016 2016-05-09 2016-05-14
22       442177464 GATI 09-06-2016 2016-06-08 2016-06-18
23       437704293 GATI 10-05-2016 2016-05-09 2016-05-14
24       442177464 GATI 09-06-2016 2016-06-08 2016-06-18
26       442177464 GATI 09-06-2016 2016-06-08 2016-06-18
27       442177464 GATI 09-06-2016 2016-06-08 2016-06-18
29       442177464 GATI 09-06-2016 2016-06-08 2016-06-18
30       442177464 GATI 09-06-2016 2016-06-08 2016-06-18
32       442177464 GATI 09-06-2016 2016-06-08 2016-06-18
33       442177464 GATI 09-06-2016 2016-06-08 2016-06-18
35       442177457 GATI 09-06-2016 2016-06-08 2016-06-22
36       442177457 GATI 09-06-2016 2016-06-08 2016-06-22
38       442177457 GATI 09-06-2016 2016-06-08 2016-06-22
39       442177457 GATI 09-06-2016 2016-06-08 2016-06-22
42       442177457 GATI 09-06-2016 2016-06-08 2016-06-22
43       442177310 GATI 14-06-2016 2016-06-13 2016-06-22
44       442177457 GATI 09-06-2016 2016-06-08 2016-06-22
45       442177310 GATI 14-06-2016 2016-06-13 2016-06-22
46       442177457 GATI 09-06-2016 2016-06-08 2016-06-22
47       442177310 GATI 14-06-2016 2016-06-13 2016-06-22
…                           …          …          …
39331                  No details 2016-06-06 2016-06-06
39332                  No details 2016-06-06 2016-06-06
39389                  No details 2016-06-27 2016-06-27
39390                  No details 2016-08-17 2016-08-17
39484                  No details 2016-06-30 2016-06-30
39485                  No details 2016-06-30 2016-06-30
39492                  No details 2016-06-30 2016-06-30
39493                  No details 2016-08-30 2016-08-30
39521                  No details 2016-06-30 2016-06-30
39522                  No details 2016-08-30 2016-08-30
39567                  No details 2016-07-28 2016-07-28
```

```
39568                   No details 2016-07-28 2016-07-28
39626                   No details 2016-07-30 2016-07-30
39627                   No details 2016-08-30 2016-08-30
39687                   No details 2016-07-30 2016-07-30
39688                   No details 2016-08-30 2016-08-30
39698                   No details 2016-08-04 2016-08-04
39699                   No details 2016-08-30 2016-08-30
39702                   No details 2016-08-04 2016-08-04
39703                   No details 2016-10-28 2016-10-28
39781                   No details 2016-08-30 2016-08-30
39857                   No details 2016-09-12 2016-09-12
39858                   No details 2016-09-12 2016-09-12
39899                   No details 2016-09-29 2016-09-29
39900                   No details 2016-10-28 2016-10-28
39974                   No details 2016-09-30 2016-09-30
39975                   No details 2016-10-28 2016-10-28
40056                   No details 2016-04-30 2016-04-30
40057                   No details 2016-05-31 2016-05-31
40132                   No details 2016-07-30 2016-07-30

[1943 rows x 18 columns]
```

lets drop these duplicates

[ ]:
```
bill.drop(bill[bill.duplicated(['Sales ord', 'Delivery no', 'Description',␣
 ↪'Consignment details', 'Value'])].index, inplace=True)
```

[ ]:
```
bill[bill['Sales ord'] == 101192139]
```

[ ]:
```
   Bill.Doc.   Material code  Required quantity  Billed Quantity  Value  \
0   90025694  7861-93-2330I.                  2                2  11759

   delivery doc  Sales ord  Plnt  Bill date  Delivery no  delivery Item  \
0      81223222  101192139   930 2016-05-26     81226681             10

         Material delivery date     Description Delivery date  \
0  7861-93-2330I.    2016-06-06  SENSOR (PC300)    2016-06-06

           Consignment details    GC date Recpt date
0  806869210 TCI 09-06-2016 GD 2016-06-06 2016-06-17
```

3.1.1 Customer

[ ]:
```
customer[customer.duplicated(['PTNO', 'customer code', 'SONO', 'Price',␣
 ↪'ITEM'], keep=False)]
```

```
Empty DataFrame
Columns: [customer code, Name, PostalCode, Region, Industry, SONO, ITEM, PTNO,
DESC, DATE, ORD_QTY, PLNT, Price, customer PO ref, PO date]
Index: []
```

3.1.2 material_master

```
material_master.head(2)
```

```
   Material code            Material Description  Model  safety stock  Demand
0  01010-61435I.             BOLT (01010-51435I)  PC450             8      30
1  01010-61455I.  BOLT (D65E-12) (01010-31455I.)    D65             1       2
```

```
material_master[material_master.duplicated(['Material code'], keep=False)]
```

```
Empty DataFrame
Columns: [Material code, Material Description, Model, safety stock, Demand]
Index: []
```

3.1.3 stock master

```
stock_master.head(2)
```

```
        Material  DocumentNo  Itm    D/C  Amount  Quantity BUn Pstng Date
0  01010-61435I.   920006718    2  Recpt  167.56         4  EA 2016-04-22
1  01010-61435I.   920006757   21   Issu  167.56         4  EA 2016-04-22
```

```
stock_master[stock_master.duplicated(['Material', 'Itm', 'DocumentNo'],␣
↪keep=False)]
```

```
Empty DataFrame
Columns: [Material, DocumentNo, Itm, D/C, Amount, Quantity, BUn, Pstng Date]
Index: []
```

3.1.4 sales_past_demand

```
sales_past_demand.head(2)
```

```
   Material code  DEM36  DEM35  DEM34  DEM33  DEM32  DEM31  DEM30  DEM29  \
0  01010-61435I.      6      0      0      0      0      0      5      0
1  01010-61455I.      0      0      0      0      0      0      0      0

   DEM28  …  DEM10  DEM9  DEM8  DEM7  DEM6  DEM5  DEM4  DEM3  DEM2  DEM1
0      0  …      0     4     0     0    16    12     2     0     0     0
1      0  …      0     0     0     0     0     0     0     2     0     0
```

```
[2 rows x 37 columns]
```

[ ]:
```
sales_past_demand[sales_past_demand.duplicated(['Material code'], keep=False)]
```

[ ]:
```
Empty DataFrame
Columns: [Material code, DEM36, DEM35, DEM34, DEM33, DEM32, DEM31, DEM30, DEM29,
DEM28, DEM27, DEM26, DEM25, DEM24, DEM23, DEM22, DEM21, DEM20, DEM19, DEM18,
DEM17, DEM16, DEM15, DEM14, DEM13, DEM12, DEM11, DEM10, DEM9, DEM8, DEM7, DEM6,
DEM5, DEM4, DEM3, DEM2, DEM1]
Index: []

[0 rows x 37 columns]
```

4 End of Pre-Processing

Now the data has been cleaned up and duplicates have been removed. We've also merged relavant data together to get new df's.

Currently we have the following DF's: 1. bill 2. customer 3. sales_past_demand 4. stock_master 5. material master

5 Data Analysis

6 Order to Delivery reports

The order to delivery analysis comprises finding the following: 1. Order to delivery note generation 2. Delivery to invoice generation 3. Invoice to consignment 4. Consignment to reaching customers(Recpt date)

6.1 1.Order to delivery note generation

[ ]:
```
#We consider Customer and Delivery data table and map them using (SONO + Item).
#We then find days taken between the date of order(PO date) and date of␣
 ↪delivery note generation(date)

customer.head(2)
```

[ ]:
```
  customer code                    Name  PostalCode          Region  \
0         A1283              ACC LIMITED      442502     Maharashtra
1         A1284  ASSOCIATED CEMENT CO LTD      483880  Madhya Pradesh

  Industry       SONO  ITEM            PTNO  \
0   Mining  101191774    10  6261-71-4112I.
1   Cement  101196605    10  702-21-57700I.

                                      DESC       DATE  ORD_QTY   PLNT  Price  \
```

```
0           TUBE (D155A-6) (6261-71-4111I.) 2016-05-11        1  930.0   4017
1  PILOT VALVE (PC2000-8R) (702-21-55800I.) 2016-09-22        2  930.0  15344

      customer PO ref    PO date
0  Req of Mr. Bhattacha 2016-05-11
1  1200620728 & 1200620 2016-09-16
```

[ ]:
```
bill.head(2)
```

[ ]:
```
   Bill.Doc.   Material code  Required quantity  Billed Quantity  Value  \
0   90025694  7861-93-2330I.                  2                2  11759
3   90025696  207-32-61260I.                 12                9   1599

   delivery doc  Sales ord  Plnt  Bill date  Delivery no  delivery Item  \
0      81223222  101192139   930 2016-05-26     81226681             10
3      81224013  101192000   930 2016-05-26     81227687             60

         Material delivery date                  Description Delivery date  \
0  7861-93-2330I.    2016-06-06               SENSOR (PC300)    2016-06-06
3  207-32-61260I.    2016-06-08  SEAL (PC300) (150-32-11260I)    2016-06-08

           Consignment details    GC date Recpt date
0  806869210 TCI 09-06-2016 GD 2016-06-06 2016-06-17
3    442177440 GATI 10-06-2016 2016-06-08 2016-06-15
```

[ ]:
```
# Bill is missing the PO Date, which we need, lets add that

tmp = customer[['SONO', 'PO date', 'ITEM']]
bill = bill.merge(tmp, how='left', left_on=['Sales ord', 'delivery Item'],␣
 ↪right_on=['SONO', 'ITEM'])
bill.drop(['SONO', 'ITEM'], axis=1, inplace=True)
```

[ ]:
```
bill.head(2)
```

[ ]:
```
   Bill.Doc.   Material code  Required quantity  Billed Quantity  Value  \
0   90025694  7861-93-2330I.                  2                2  11759
1   90025696  207-32-61260I.                 12                9   1599

   delivery doc  Sales ord  Plnt  Bill date  Delivery no  delivery Item  \
0      81223222  101192139   930 2016-05-26     81226681             10
1      81224013  101192000   930 2016-05-26     81227687             60

         Material delivery date                  Description Delivery date  \
0  7861-93-2330I.    2016-06-06               SENSOR (PC300)    2016-06-06
1  207-32-61260I.    2016-06-08  SEAL (PC300) (150-32-11260I)    2016-06-08

           Consignment details    GC date Recpt date    PO date
```

```
0  806869210 TCI 09-06-2016 GD 2016-06-06 2016-06-17 2016-05-16
1    442177440 GATI 10-06-2016 2016-06-08 2016-06-15 2016-05-13
```

[ ]:
```
order_to_delivery = bill[['PO date', 'Delivery date']]
order_to_delivery.loc[:, 'ORD_to_DEL'] = order_to_delivery['Delivery date'] -␣
↪order_to_delivery['PO date']
```

```
/usr/lib/python2.7/site-packages/pandas/core/indexing.py:297:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/indexing.html#indexing-view-versus-copy
  self.obj[key] = _infer_fill_value(value)
/usr/lib/python2.7/site-packages/pandas/core/indexing.py:477:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/indexing.html#indexing-view-versus-copy
  self.obj[item] = s
```

[ ]:
```
order_to_delivery.head(2)
```

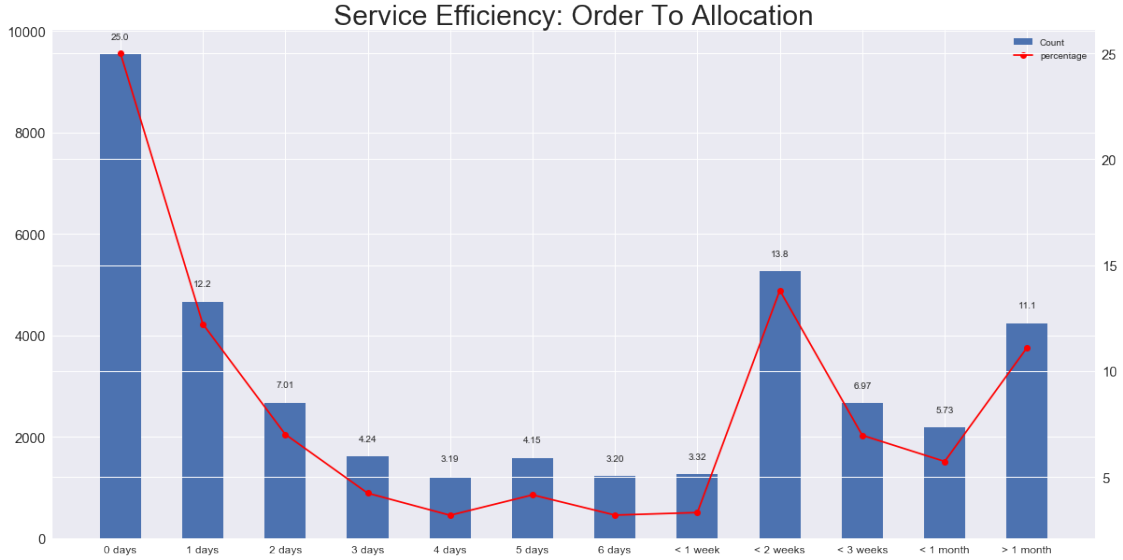
[ ]:
```
   PO date Delivery date  ORD_to_DEL
0 2016-05-16    2016-06-06     21 days
1 2016-05-13    2016-06-08     26 days
```

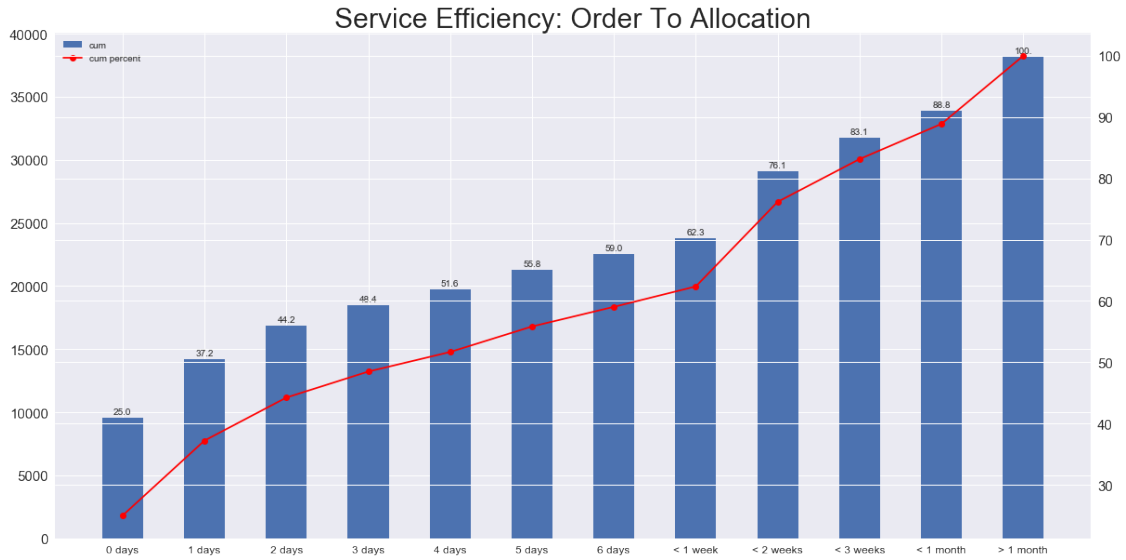
[ ]:
```
tmp = order_to_delivery['ORD_to_DEL'].value_counts().reset_index()
tmp.columns = ['Days', 'Count']
tmp['Days'] = tmp['Days'].apply(lambda x: x.days)
tmp.sort_values(by=['Days'], inplace=True)
tmp = tmp.reset_index().drop('index', axis=1)
```

[ ]:
```
def display_days_difference(tmp, title):

    df = pd.DataFrame(columns=['Days', 'cum'])
    for i in xrange(7):
        df.loc[i] = [str(i) + ' days', tmp[tmp['Days'] <= i]['Count'].sum()]
    df.loc[7] = ['< 1 week', tmp[tmp['Days'] <= 7]['Count'].sum()]
    df.loc[8] = ['< 2 weeks', tmp[tmp['Days'] <= 14]['Count'].sum()]
    df.loc[9] = ['< 3 weeks', tmp[tmp['Days'] <= 21]['Count'].sum()]
    df.loc[10] = ['< 1 month', tmp[tmp['Days'] <= 30]['Count'].sum()]
    df.loc[11] = ['> 1 month', tmp['Count'].sum()]
```

```
    df['Count'] = df['cum']
    for i in xrange(11, 0, -1):
        df.loc[i, 'Count'] = df.loc[i, 'Count'] - df.loc[i - 1, 'Count']

    df['percentage'] = 100*df.Count/df.Count.sum()
    df['cum percent'] = df.percentage.cumsum()

    #setting font size
    plt.rc('axes', titlesize=30)     # fontsize of the axes title
    plt.rc('axes', labelsize=15)    # fontsize of the x and y labels
    plt.rc('xtick', labelsize=12)    # fontsize of the tick labels
    plt.rc('ytick', labelsize=15)    # fontsize of the tick labels

    for (i, j) in [('Count', 'percentage'), ('cum', 'cum percent')]:
        fig, ax1 = plt.subplots()
        ax2 = ax1.twinx()
        ax1.bar([x for x in xrange(len(df))], df[i], width=.5, label=i)
        ax2.plot([x for x in xrange(len(df))], df[j], color='red', marker='o')
        for k in xrange(len(df)):
            ax1.text(k, df[i][k] + 300, str(float(df[j][k]))[:4],␣
↪horizontalalignment='center')
        plt.xticks([x for x in xrange(len(df))], df['Days'])
        h1, l1 = ax1.get_legend_handles_labels()
        h2, l2 = ax2.get_legend_handles_labels()
        ax1.legend(h1+h2, l1+l2, loc=0)
        plt.title(title)
        plt.show()
```

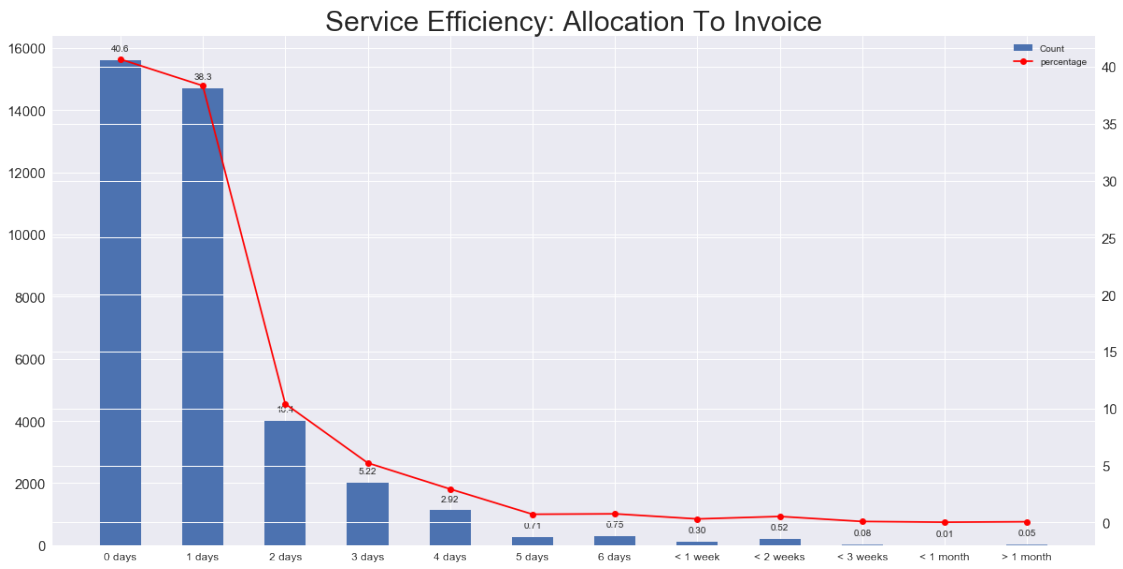
[ ]:
```
display_days_difference(tmp, 'Service Efficiency: Order To Allocation')
```

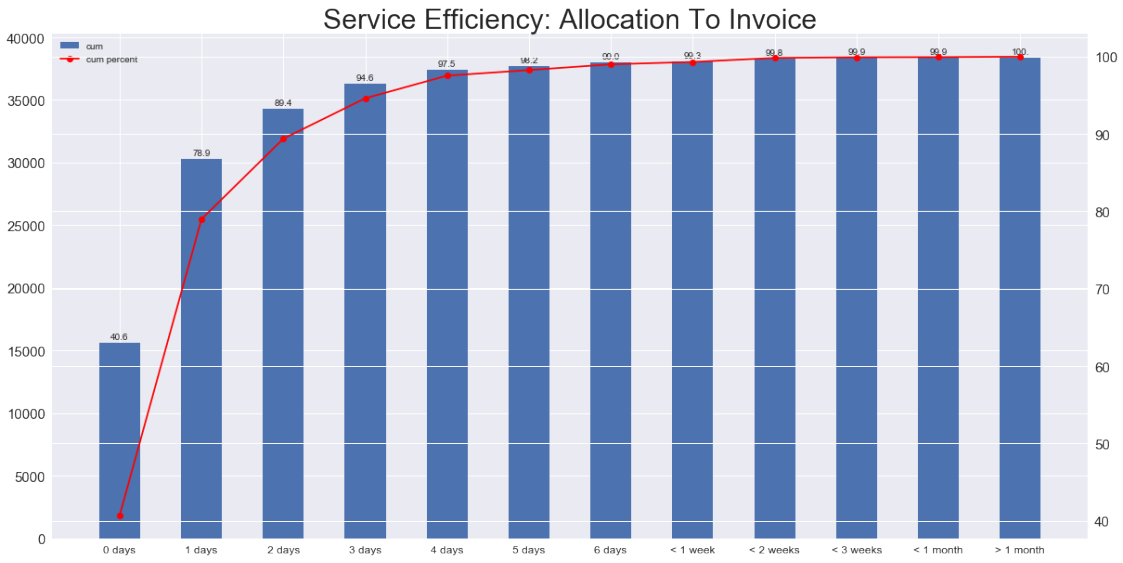




6.2 2. Delivery to invoice generation.

[ ]:
```
delivery_to_bill = bill[['Bill date', 'Delivery date']]
delivery_to_bill.loc[:, 'DEL_TO_BILL'] = delivery_to_bill['Bill date'] -␣
 ↪delivery_to_bill['Delivery date']
tmp = delivery_to_bill['DEL_TO_BILL'].value_counts().reset_index()
tmp.columns = ['Days', 'Count']
tmp['Days'] = tmp['Days'].apply(lambda x: x.days)
tmp.sort_values(by=['Days'], inplace=True)
tmp = tmp.reset_index().drop('index', axis=1)
display_days_difference(tmp, 'Service Efficiency: Allocation To Invoice')
```





6.3 3. Invoice to consignment generation

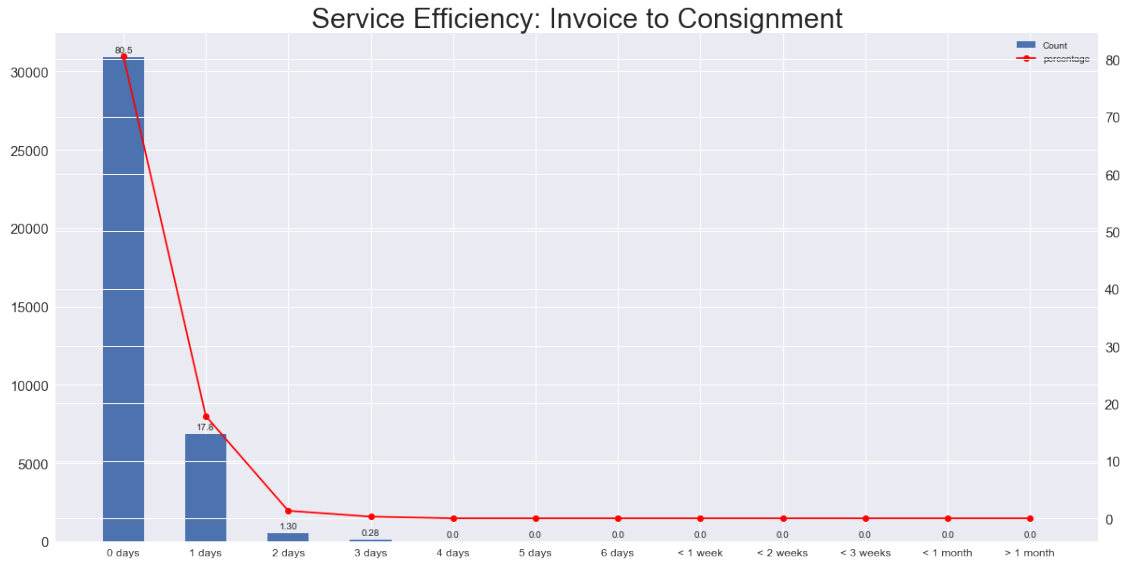
[ ]:
```
bill_to_GC = bill[['Bill date', 'GC date']]
bill_to_GC.loc[:, 'BILL_TO_GC'] = bill_to_GC['GC date'] - bill_to_GC['Bill␣
↪date']
bill_to_GC.loc[:, 'BILL_TO_GC'] = bill_to_GC['BILL_TO_GC'].dt.components.days
```

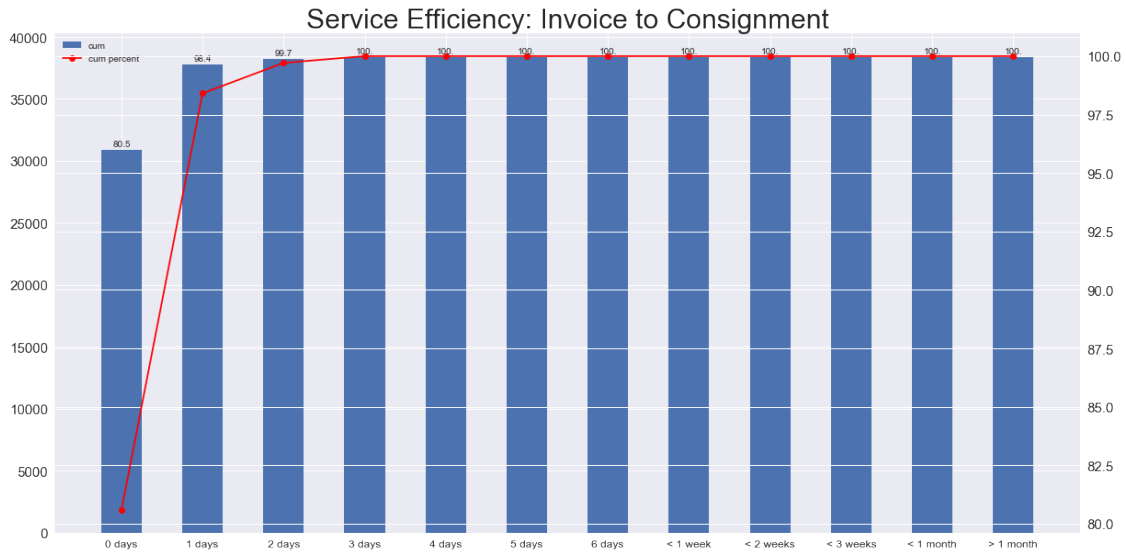
[ ]:
```
def test2 (row):
    if row['BILL_TO_GC'] < -10  :
```

```
        return 3
    if row['BILL_TO_GC'] < -5   :
        return 2
    if row['BILL_TO_GC'] < -1   :
        return 1

    return 0
```

[ ]:
```
bill_to_GC.loc[:, 'BILL_TO_GC'] = bill_to_GC.apply(lambda row: test2 (row),axis
 ↪=1)
bill_to_GC.loc[:, 'BILL_TO_GC'] = pd.to_timedelta(bill_to_GC['BILL_TO_GC'],
 ↪unit='D')
```

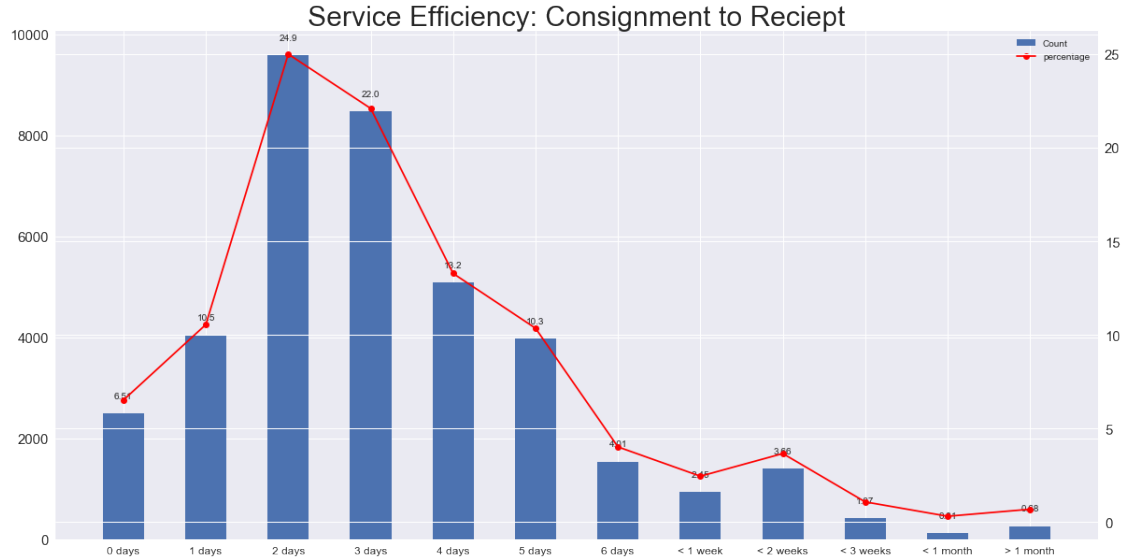
[ ]:
```
tmp = bill_to_GC['BILL_TO_GC'].value_counts().reset_index()
tmp.columns = ['Days', 'Count']
tmp['Days'] = tmp['Days'].apply(lambda x: x.days)
tmp.sort_values(by=['Days'], inplace=True)
tmp = tmp.reset_index().drop('index', axis=1)
display_days_difference(tmp, 'Service Efficiency: Invoice to Consignment')
```

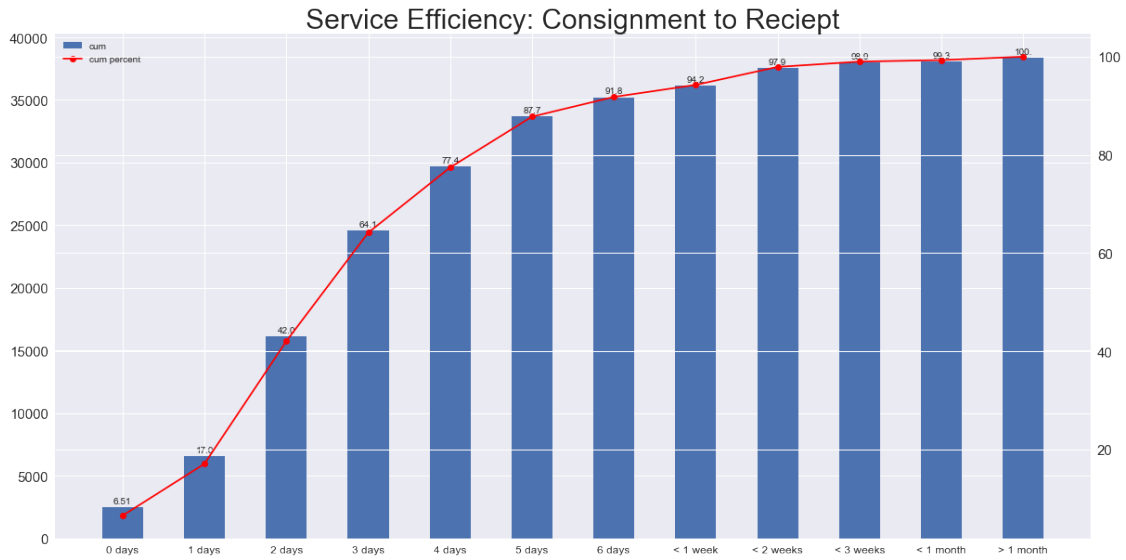




6.4 4. Consignment to receipt

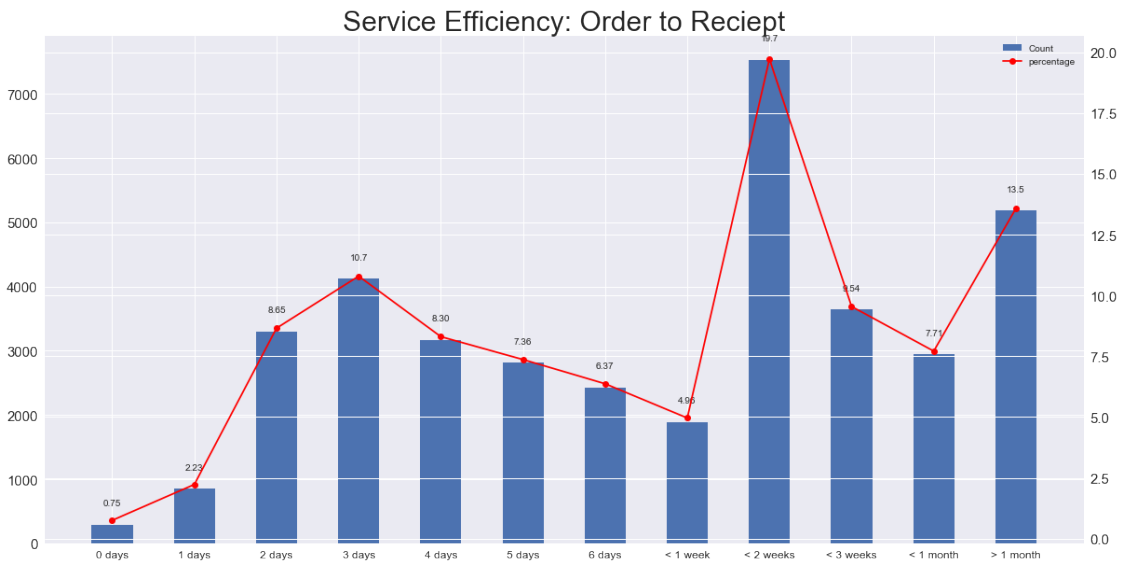
```
[ ]: GC_to_recpt = bill[['GC date', 'Recpt date']]
GC_to_recpt.loc[:, 'GC_TO_RECPT'] = GC_to_recpt['Recpt date'] - GC_to_recpt['GC␣
  ↪date']
tmp = GC_to_recpt['GC_TO_RECPT'].value_counts().reset_index()
tmp.columns = ['Days', 'Count']
tmp['Days'] = tmp['Days'].apply(lambda x: x.days)
tmp.sort_values(by=['Days'], inplace=True)
tmp = tmp.reset_index().drop('index', axis=1)
display_days_difference(tmp, 'Service Efficiency: Consignment to Reciept')
```

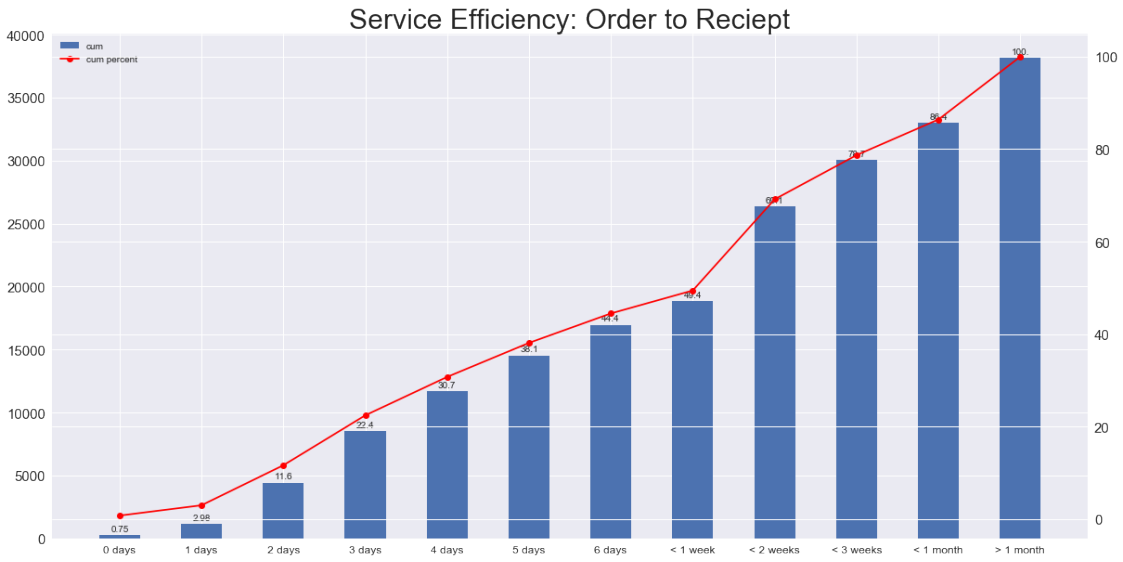




6.5 TOTAL TIME

[ ]:
```
PO_to_recpt = bill[['PO date', 'Recpt date']]
PO_to_recpt.loc[:, 'PO_TO_RECPT'] = PO_to_recpt['Recpt date'] - PO_to_recpt['PO␣
 ↪date']
tmp = PO_to_recpt['PO_TO_RECPT'].value_counts().reset_index()
tmp.columns = ['Days', 'Count']
tmp['Days'] = tmp['Days'].apply(lambda x: x.days)
tmp.sort_values(by=['Days'], inplace=True)
tmp = tmp.reset_index().drop('index', axis=1)
display_days_difference(tmp, 'Service Efficiency: Order to Reciept')
```





7 Customer wise sales

[ ]:
```
customer.head(2)
```

[ ]:
```
  customer code                      Name  PostalCode          Region  \
0         A1283                ACC LIMITED      442502     Maharashtra
1         A1284  ASSOCIATED CEMENT CO LTD      483880  Madhya Pradesh

  Industry       SONO  ITEM           PTNO  \
```

```
0   Mining  101191774    10  6261-71-4112I.
1   Cement  101196605    10  702-21-57700I.

                                      DESC       DATE  ORD_QTY   PLNT  Price  \
0           TUBE (D155A-6) (6261-71-4111I.) 2016-05-11        1  930.0   4017
1  PILOT VALVE (PC2000-8R) (702-21-55800I.) 2016-09-22        2  930.0  15344

        customer PO ref    PO date
0  Req of Mr. Bhattacha 2016-05-11
1  1200620728 & 1200620 2016-09-16
```

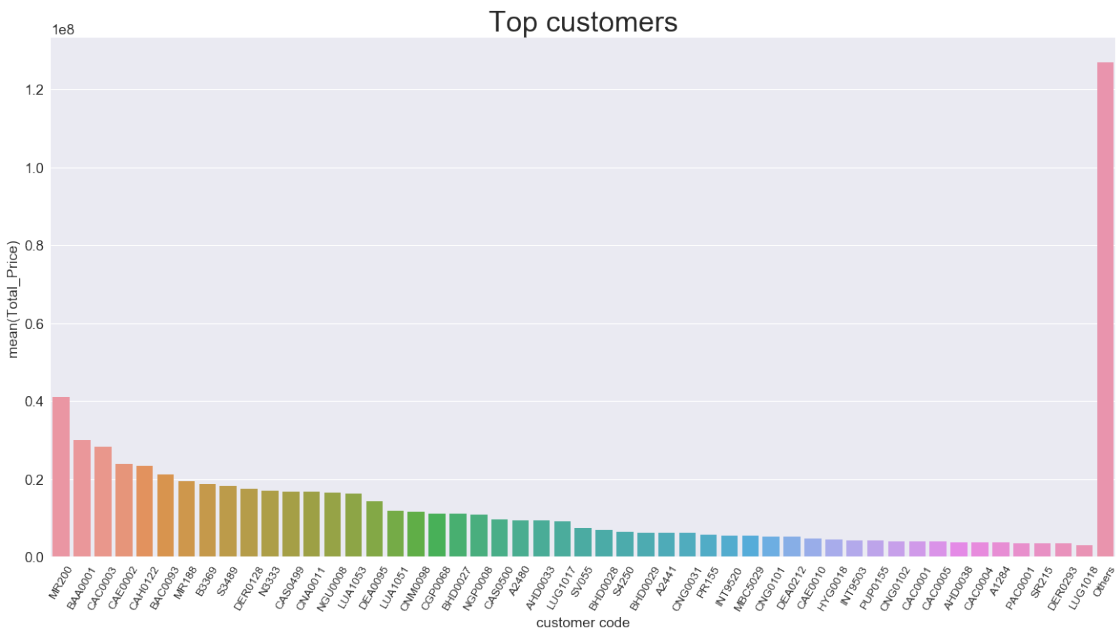
[ ]:
```
#Finding the total price corresponding to each customer

customer['Total_Price'] = customer['ORD_QTY'] * customer['Price']
customer.head(1)
```

[ ]:
```
  customer code         Name  PostalCode       Region Industry       SONO  \
0         A1283  ACC LIMITED      442502  Maharashtra   Mining  101191774

   ITEM            PTNO                                DESC       DATE  ORD_QTY  \
0    10  6261-71-4112I.  TUBE (D155A-6) (6261-71-4111I.) 2016-05-11        1

    PLNT  Price       customer PO ref    PO date  Total_Price
0  930.0   4017  Req of Mr. Bhattacha 2016-05-11         4017
```

[ ]:
```
tmp = customer[['customer code', 'Total_Price']]
tmp = tmp.groupby(['customer code']).sum().reset_index()
tmp.sort_values(by='Total_Price', ascending=False, inplace=True)
tmp = tmp.reset_index(drop=True)

tmp2 = tmp.loc[50:]
tmp = tmp.loc[:49]
tmp.loc[50] = ['Others', tmp2['Total_Price'].sum()]
```

[ ]:
```
sns.barplot(x='customer code', y='Total_Price', data=tmp)
plt.xticks(rotation=60)
plt.title("Top customers")
plt.plot()
```

[ ]:
```
[]
```



8 Region wise Sales

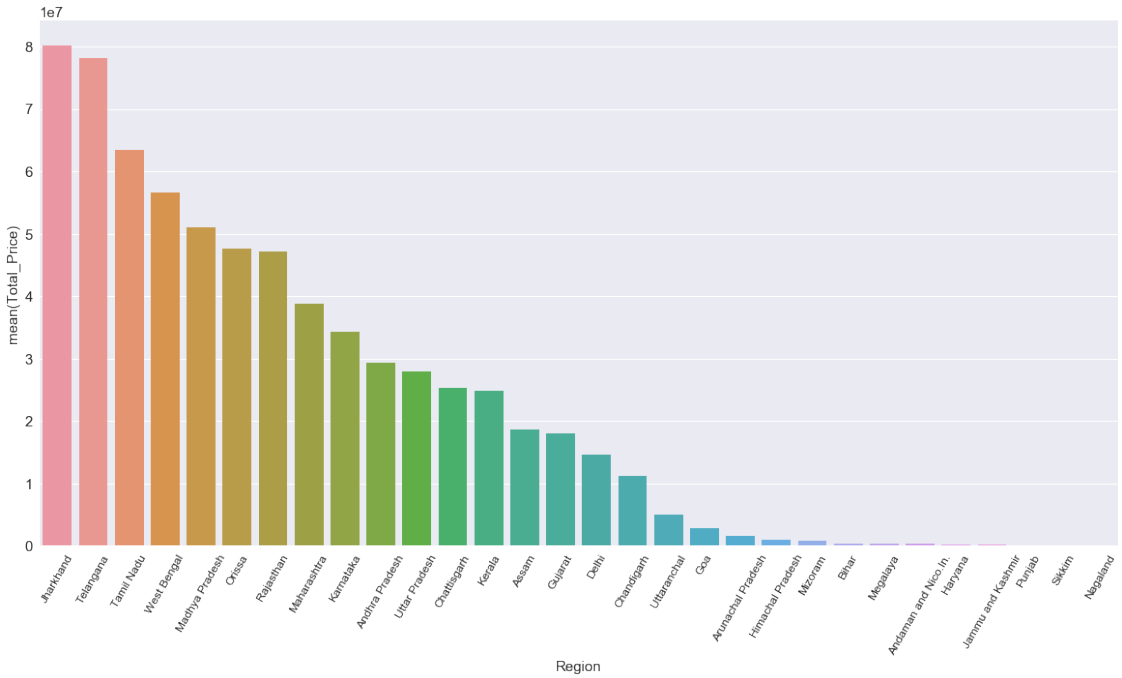
[ ]:
```
customer.head(2)
```

[ ]:
```
  customer code                      Name  PostalCode          Region  \
0         A1283                ACC LIMITED      442502     Maharashtra
1         A1284  ASSOCIATED CEMENT CO LTD      483880  Madhya Pradesh

  Industry       SONO  ITEM            PTNO  \
0   Mining  101191774    10  6261-71-4112I.
1   Cement  101196605    10  702-21-57700I.

                                      DESC       DATE  ORD_QTY   PLNT  Price  \
0           TUBE (D155A-6) (6261-71-4111I.) 2016-05-11        1  930.0   4017
1  PILOT VALVE (PC2000-8R) (702-21-55800I.) 2016-09-22        2  930.0  15344

        customer PO ref    PO date  Total_Price
0  Req of Mr. Bhattacha 2016-05-11         4017
1  1200620728 & 1200620 2016-09-16        30688
```

[ ]:
```
df = customer.groupby('Region')['Total_Price'].sum().reset_index()
# sort by income
df = df.sort_values(by='Total_Price', ascending=False)
df = df.reset_index(drop=True)
```

[ ]:
```
#Plotting the region wise profit graph

sns.barplot(x = 'Region', y= 'Total_Price', data=df)
plt.xticks(rotation=60)
plt.plot()
```

[ ]: []



9 Material Wise Profit

[ ]:
```
customer.head(2)
```

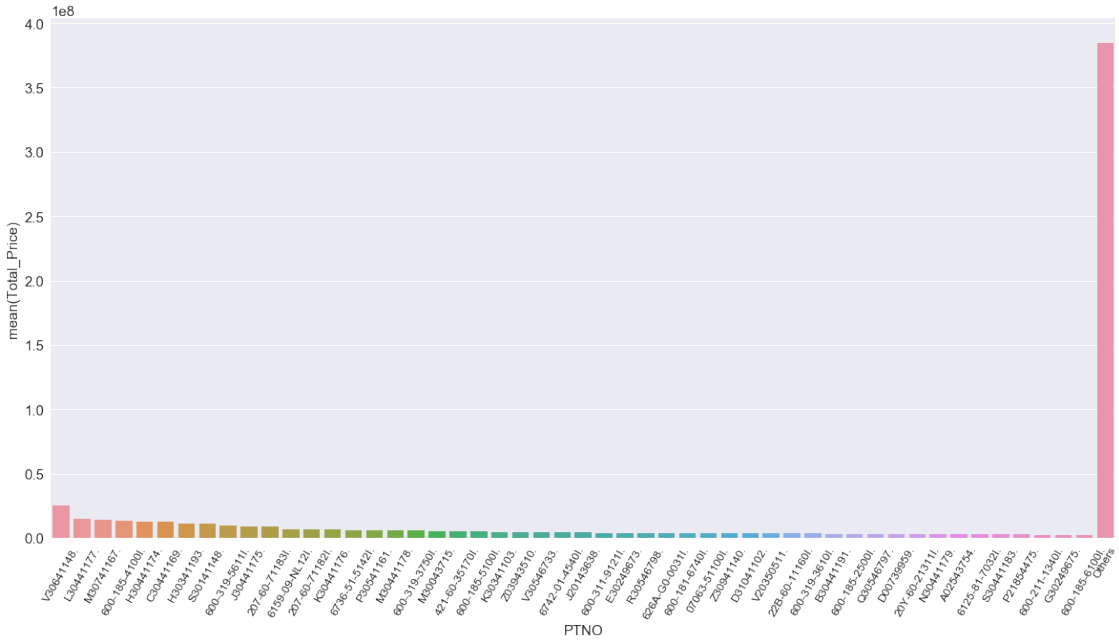
[ ]:
```
  customer code                      Name  PostalCode          Region  \
0         A1283                ACC LIMITED      442502     Maharashtra
1         A1284  ASSOCIATED CEMENT CO LTD      483880  Madhya Pradesh

  Industry       SONO  ITEM            PTNO  \
0   Mining  101191774    10  6261-71-4112I.
1   Cement  101196605    10  702-21-57700I.

                                    DESC       DATE  ORD_QTY   PLNT  Price  \
0           TUBE (D155A-6) (6261-71-4111I.) 2016-05-11        1  930.0   4017
1  PILOT VALVE (PC2000-8R) (702-21-55800I.) 2016-09-22        2  930.0  15344
```

```
        customer PO ref    PO date  Total_Price
0  Req of Mr. Bhattacha 2016-05-11         4017
1  1200620728 & 1200620 2016-09-16        30688
```

[ ]:
```
df = customer.groupby('PTNO')['Total_Price'].sum().reset_index()
# sort by income
df = df.sort_values(by='Total_Price', ascending=False)
df = df.reset_index(drop=True)

tmp = df.loc[50:]
df = df.loc[:49]
df.loc[50] = ['Others', tmp['Total_Price'].sum()]
```

[ ]:
```
#Plotting the region wise profit graph

sns.barplot(x = 'PTNO', y= 'Total_Price', data=df)
plt.xticks(rotation=60)
plt.plot()
```

[ ]: []



10 Model wise demand

[ ]:
```
material_master.head()
```

[ ]:

	Material code	Material Description	Model	safety stock	Demand
0	01010-61435I.	BOLT (01010-51435I)	PC450	8	30
1	01010-61455I.	BOLT (D65E-12) (01010-31455I.)	D65	1	2
2	01010-61635I.	BOLT (GD511)	GD511	1	12
3	01010-61645I.	BOLT (01010-81645I.)	D475	8	32
4	01010-61650I.	BOLT (HD465) (01010-81650I.)	HD465	5	18

[ ]:
```
model_demand = material_master
model_demand = model_demand.groupby('Model').sum().reset_index()
model_demand.drop(['safety stock'], axis=1, inplace=True)
model = model_demand.sort_values(by='Demand', ascending=False)
model = model.reset_index(drop=True)
model.head(15)
```

[ ]:

	Model	Demand
0	PC200	129526
1	PC300	61608
2	PC71	44963
3	PC210	34785
4	D155	32093
5	300CK	29608
6	PC130	26286
7	PC450	21973
8	Others	14919
9	OTHER	13158
10	D475	9353
11	90CK	8794
12	GD511	7975
13	D65	6908
14	PC600	3769

[ ]:
```
# 8, 9 is others, lets combine and remove them
tmp = model.loc[8:9]
model = model.drop([8, 9])
tmp
```

[ ]:

	Model	Demand
8	Others	14919
9	OTHER	13158

[ ]:
```
model = model.reset_index(drop=True)
tmp = tmp.reset_index(drop=True)
```

[ ]:
```
tmp
```

[ ]:

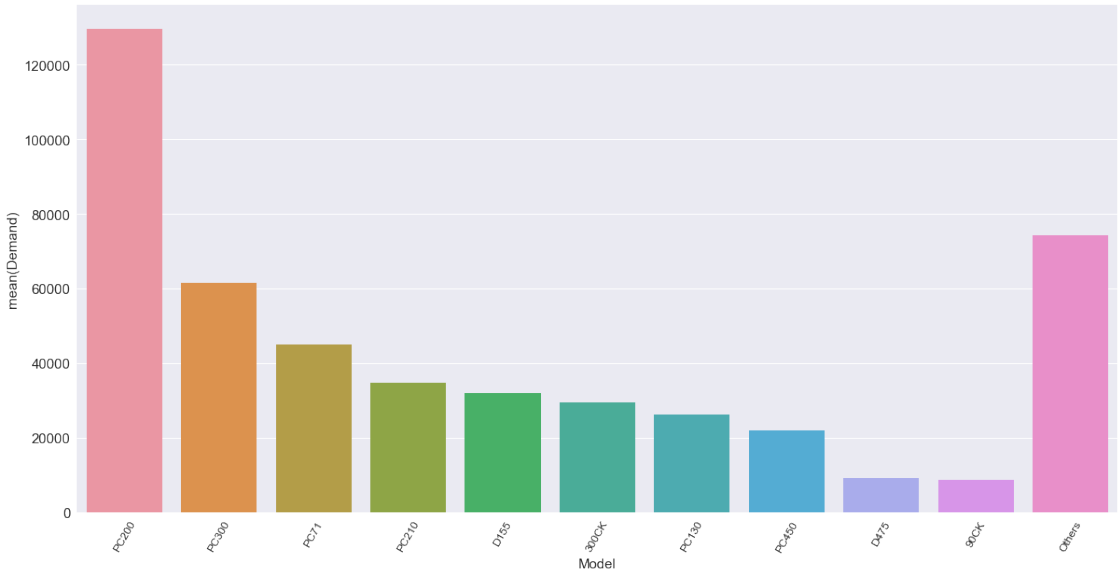
	Model	Demand
0	Others	14919

```
1    OTHER    13158
```

[ ]:
```
tmp = pd.concat([tmp, model.loc[10:]])
model = model.loc[:9]
model.loc[10] = ['Others', tmp['Demand'].sum()]
```

[ ]:
```
sns.barplot(x = 'Model', y= 'Demand', data=model)
plt.xticks(rotation=60)
plt.plot()
```

[ ]:
```
[]
```



11 Pie chart to be displayed : Out of stock or stocks needed immediately

[ ]:
```
stock_df = material_master.merge(stock_master,left_on='Material code',right_on␣
↪= 'Material')
stock_df.drop(['Material Description','DocumentNo','D/C','Amount','BUn','Pstng␣
↪Date','Material'],axis=1, inplace=True)
stock_df.tail()
```

[ ]:

	Material code	Model	safety stock	Demand	Itm	Quantity
68550	Z30941140.	PC200	4	14	1	1
68551	Z30941140.	PC200	4	14	1	1
68552	Z30941140.	PC200	4	14	1	1
68553	Z30941140.	PC200	4	14	1	1

```
68554    Z30941140.  PC200             4      14    1         1
```

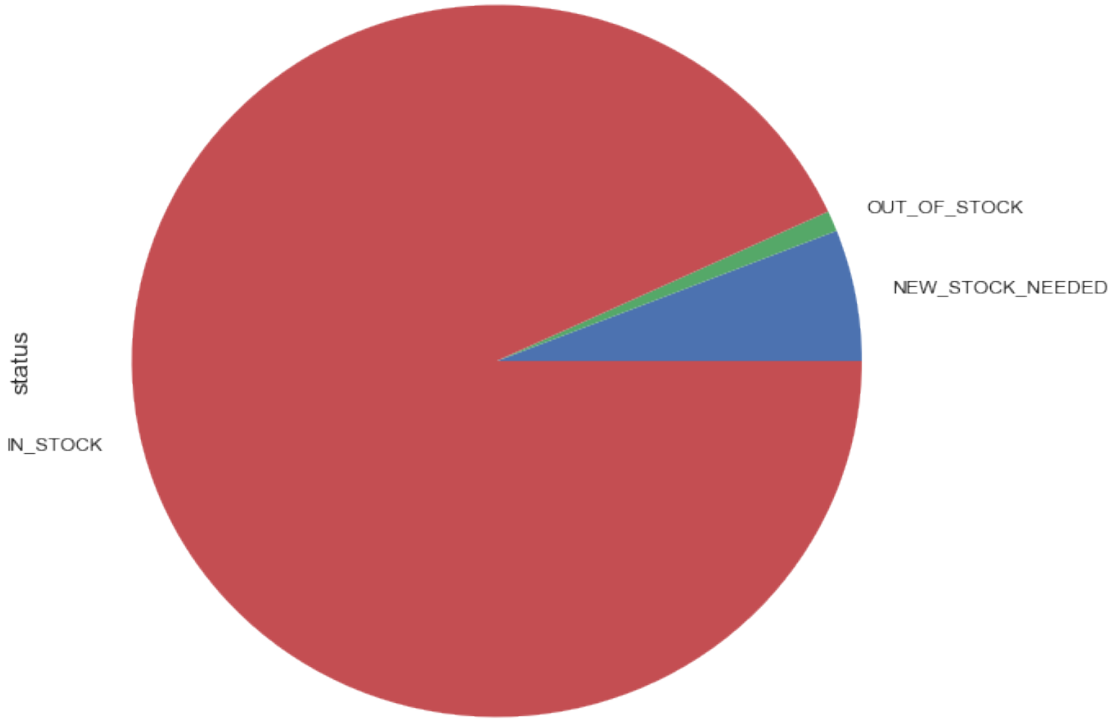
[ ]:
```
stock_df = stock_df.groupby(['Material code','Model','safety␣
 ↪stock'])['Quantity'].sum().reset_index()
stock_df.head()
```

[ ]:
```
   Material code  Model  safety stock  Quantity
0  01010-61435I.  PC450             8        86
1  01010-61455I.    D65             1         4
2  01010-61635I.  GD511             1        34
3  01010-61645I.   D475             8         4
4  01010-61650I.  HD465             5         8
```

[ ]:
```
# checking for various conditions
def test (row):
   if row['safety stock'] < row['Quantity']  :
      return 'IN_STOCK'
   if row['safety stock'] > row['Quantity']  :
      return 'OUT_OF_STOCK'
   if row['safety stock'] == row['Quantity']  :
      return 'NEW_STOCK_NEEDED'

   return 'Other'
```

[ ]:
```
stock_df['status'] = stock_df.apply (lambda row: test (row),axis=1)
```

[ ]:
```
#pie chart(using quantity and safety_stock)

plt.figure(figsize=(10, 10))
stock_df.status.value_counts(sort=False).plot(kind='pie')
plt.show()
```



12 Season wise demand forecast

Here we try to find out how the demand for various products in the company vary with the seasons

First we split the order dates for each material according to seasons We consider 5 seasons:
1. Winter - November to Feb (11,12,1,2) 2. Spring - March to April (3,4)
3. Summer - May to June (5,6) 4. Monsoon - July to August (7,8)
5. Autumn - September to October (9,10)

[ ]:
```
customer.head(2)
```

[ ]:
```
  customer code                      Name  PostalCode          Region  \
0         A1283                ACC LIMITED      442502     Maharashtra
1         A1284  ASSOCIATED CEMENT CO LTD      483880  Madhya Pradesh

  Industry       SONO  ITEM            PTNO  \
0   Mining  101191774    10  6261-71-4112I.
1   Cement  101196605    10  702-21-57700I.
```

```
                                        DESC       DATE  ORD_QTY   PLNT  Price  \
0           TUBE (D155A-6) (6261-71-4111I.) 2016-05-11        1  930.0   4017
1  PILOT VALVE (PC2000-8R) (702-21-55800I.) 2016-09-22        2  930.0  15344

         customer PO ref    PO date  Total_Price
0  Req of Mr. Bhattacha 2016-05-11         4017
1  1200620728 & 1200620 2016-09-16        30688
```

Lets get the month from 'PO date' and assign it to a new df

[ ]:
```
df = customer[['PO date', 'Total_Price']]
df.loc[:, 'month'] = df['PO date'].apply(lambda x: x.month)
df = df.drop(['PO date'], axis=1)
```

[ ]:
```
df = df.groupby(['month']).sum().reset_index()
```

[ ]:
```
# checking for various conditions
def test_s (row):
   if (row['month'] ==1)|(row['month']==2)   :
      return 'winter'

   if (row['month'] ==3)|(row['month'] ==4)   :
      return 'spring'
   if (row['month'] ==5)|(row['month'] ==6)  :
      return 'summer'
   if (row['month']==7)|(row['month']==8) :
      return 'mansoon'
   if (row['month'] ==9)|(row['month'] ==10)   :
      return 'autumn'
   if (row['month'] ==11)|(row['month'] ==12)   :
      return 'winter'
```

[ ]:
```
df['season'] = df.apply (lambda row: test_s (row),axis=1)
df = df.drop(['month'], axis=1)
```

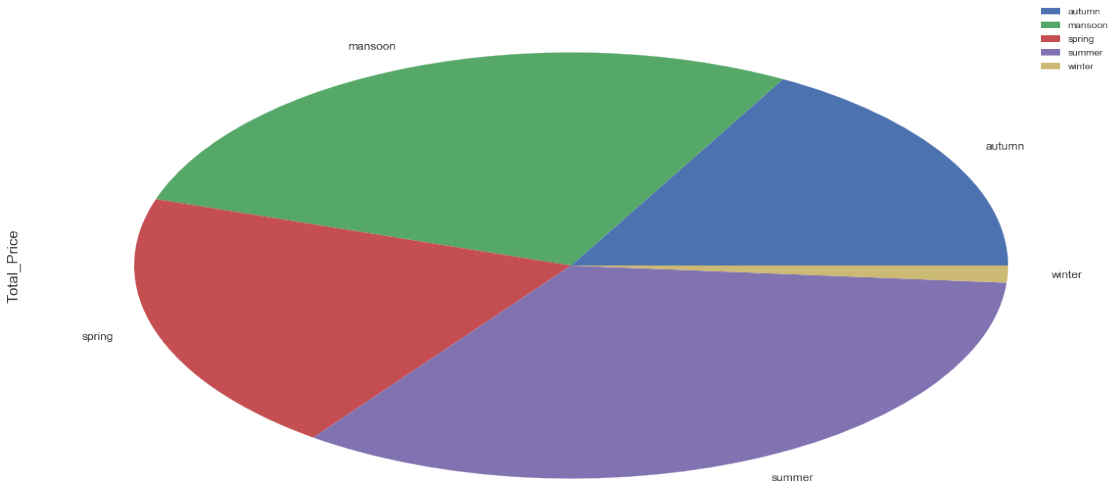
[ ]:
```
df = df.groupby('season').sum().reset_index()
df
```

[ ]:
```
    season  Total_Price
0   autumn    115282947
1  mansoon    190583361
2   spring    136098629
3   summer    229683886
4   winter      8753860
```

[ ]:
```
plt.figure(figsize=(10, 10))
df.plot(kind='pie', y='Total_Price', labels=df['season'])
```

```
plt.show()
```

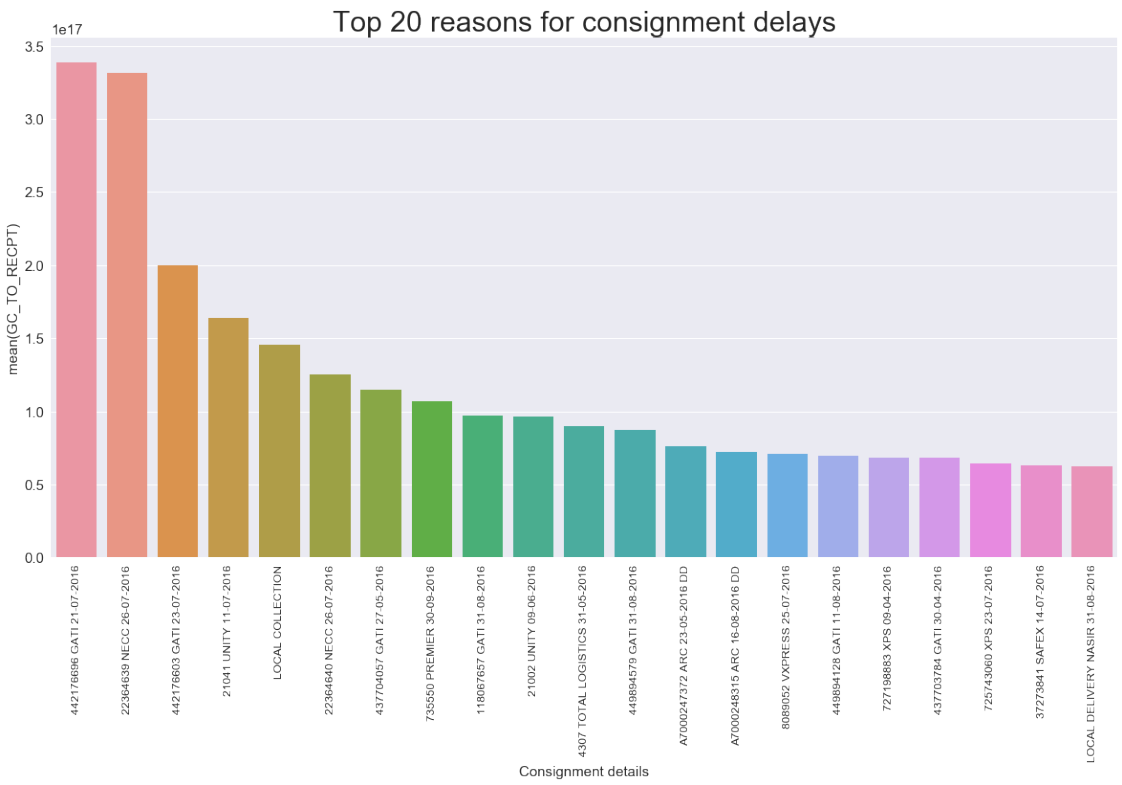
```
<matplotlib.figure.Figure at 0x7f557ec62ad0>
```



13 Checking possible reason for delay from consignment to delivery

[ ]:
```
GC_to_recpt = bill[['GC date', 'Recpt date', 'Consignment details']]
GC_to_recpt.loc[:, 'GC_TO_RECPT'] = GC_to_recpt['Recpt date'] - GC_to_recpt['GC␣
 ↪date']
GC_to_recpt = GC_to_recpt.sort_values(by='GC_TO_RECPT', ascending=False)
GC_to_recpt = GC_to_recpt.groupby('Consignment details').sum().reset_index().
 ↪sort_values(by='GC_TO_RECPT', ascending=False).reset_index(drop=True)
```

[ ]:
```
# Top 20 reasons for consignment delays

GC_to_recpt = GC_to_recpt.loc[:20]
sns.barplot(y = 'GC_TO_RECPT', x= 'Consignment details', data=GC_to_recpt)
plt.xticks(rotation=90)
plt.title('Top 20 reasons for consignment delays')
plt.show()
```

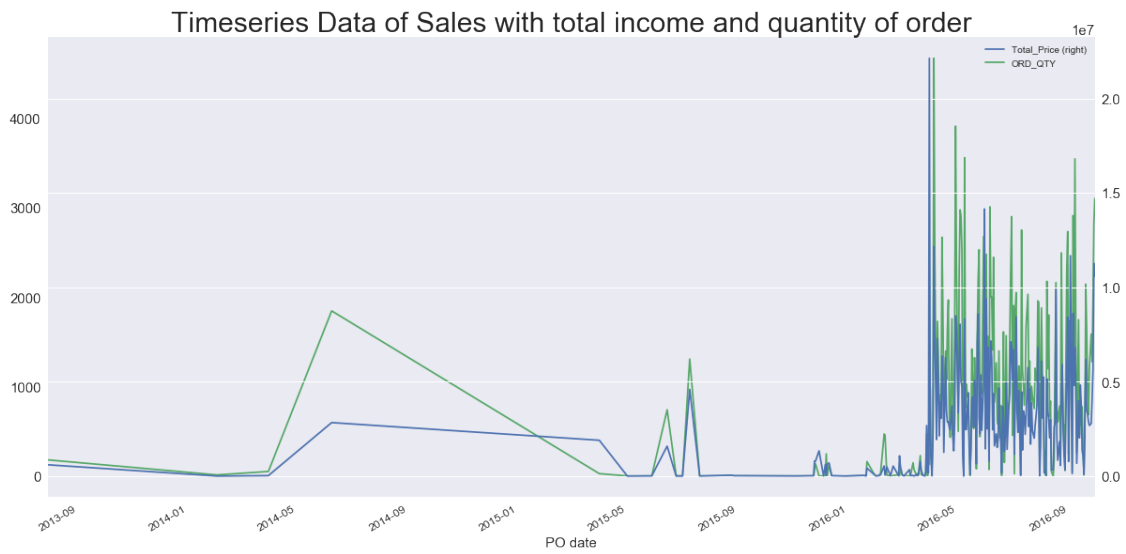


14 Query Drill down

[ ]:
```
# Lets see if the orders fluctuate as per the date, you know if there
# is a season for most orders. Maybe end of fiscal year or start of new
# fiscal year, this would show optimal time to pump up production of parts

# df is a new dataframe created using the customer_order table, we only need
# to put emphasis on the date of orders and the Price.
df = customer[['PO date', 'Total_Price', 'ORD_QTY']]
```

[ ]:
```
df = df.groupby(['PO date']).sum().reset_index()
df = df.sort_values(by=['PO date'])
df['PO date'] = pd.to_datetime(df['PO date'])
```

[ ]:
```
ax = df.plot(x='PO date', y='Total_Price', secondary_y=True)
df.plot(x='PO date', y='ORD_QTY', ax=ax)
plt.title('Timeseries Data of Sales with total income and quantity of order')
plt.show()
```



14.0.1 Month of the year

[ ]:
```
df = customer[['PO date', 'Total_Price', 'ORD_QTY']]
df.loc[:,'month'] = df['PO date'].apply(lambda x: x.month)
df = df.drop(['PO date'], axis=1)
df = df.groupby(['month']).sum().reset_index()
```

[ ]:
```
df
```

[ ]:

	month	Total_Price	ORD_QTY
0	1	534144	191
1	2	3298831	1099
2	3	27143439	1439
3	4	108955190	38685
4	5	120713266	38275
5	6	108970620	38315
6	7	104511662	34957
7	8	86071699	29733
8	9	115282947	35748
9	11	2934394	215
10	12	1986491	378

[ ]:
```
ax = df.plot(x='month', y='Total_Price', secondary_y=True, marker='o')
df.plot(x='month', y='ORD_QTY', marker='o', ax=ax)
plt.xticks([x for x in xrange(1, 13)], ['Jan', 'Feb', 'Mar', 'Apr', 'May',␣
 ↪'Jun', 'July', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec'])
plt.title('Income and Order Quantity per month')
plt.show()
```

14.0.2 Day of the month

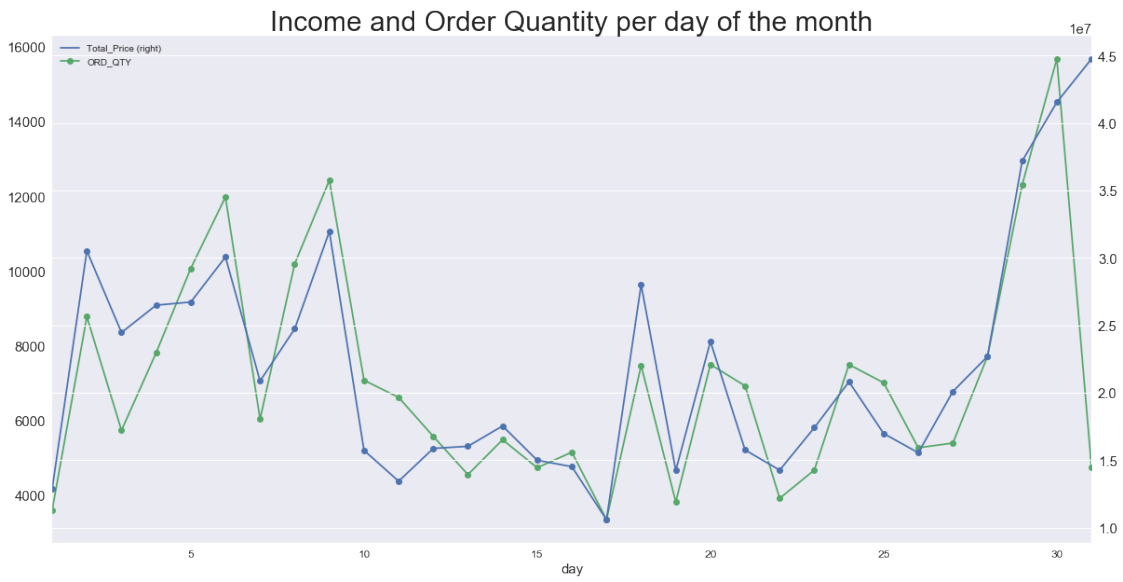
[ ]:
```
df = customer[['PO date', 'Total_Price', 'ORD_QTY']]
df.loc[:,'day'] = df['PO date'].apply(lambda x: x.day)
df = df.drop(['PO date'], axis=1)
df = df.groupby(['day']).sum().reset_index()
```

[ ]:
```
df.head()
```

[ ]:

	day	Total_Price	ORD_QTY
0	1	12893564	3604
1	2	30516498	8774
2	3	24459472	5732
3	4	26489072	7810
4	5	26719262	10069

[ ]:
```
ax = df.plot(x='day', y='Total_Price', secondary_y=True, marker='o')
df.plot(x='day', y='ORD_QTY', marker='o', ax=ax)
plt.title('Income and Order Quantity per day of the month')
plt.show()
```



14.0.3 Machine Learning

[ ]:
```
sales_past_demand.head()
```

[ ]:

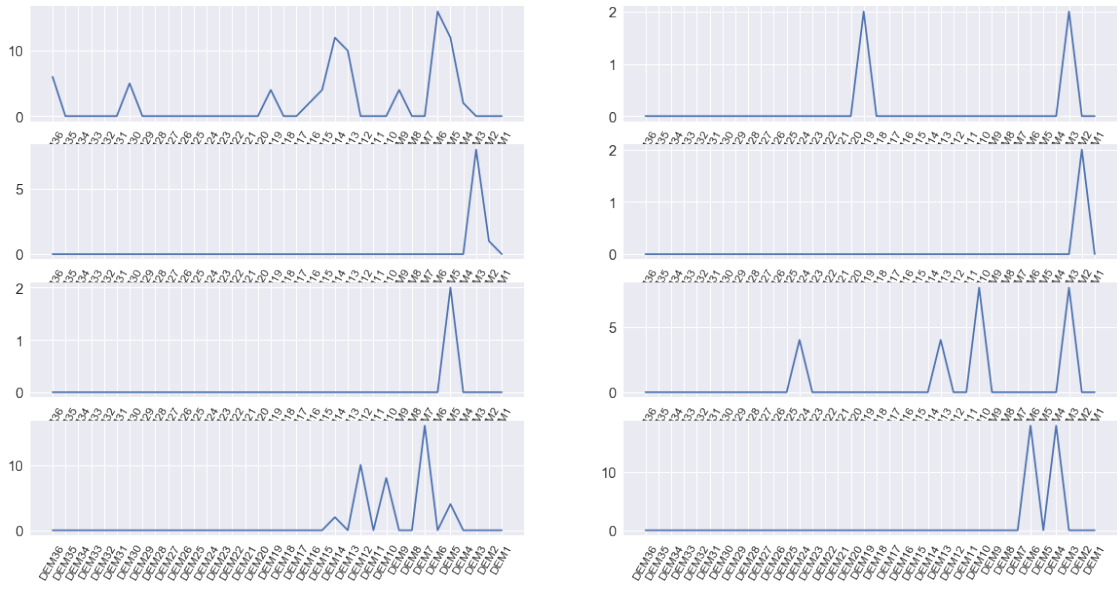
	Material code	DEM36	DEM35	DEM34	DEM33	DEM32	DEM31	DEM30	DEM29	\
0	01010-61435I.	6	0	0	0	0	0	5	0	
1	01010-61455I.	0	0	0	0	0	0	0	0	
2	01010-61635I.	0	0	0	0	0	0	0	0	
3	01010-61645I.	0	0	0	0	0	0	0	0	
4	01010-61650I.	0	0	0	0	0	0	0	0	

	DEM28	…	DEM10	DEM9	DEM8	DEM7	DEM6	DEM5	DEM4	DEM3	DEM2	DEM1
0	0	…	0	4	0	0	16	12	2	0	0	0
1	0	…	0	0	0	0	0	0	0	2	0	0
2	0	…	0	0	0	0	0	0	0	8	1	0
3	0	…	0	0	0	0	0	0	0	0	2	0
4	0	…	0	0	0	0	0	2	0	0	0	0

[5 rows x 37 columns]

[ ]:
```
tmp = sales_past_demand.drop(['Material code'], axis=1)
```

[ ]:
```
for i in xrange(0, 8):
    plt.subplot(4, 2, i + 1)
    plt.plot([x for x in range(0, 36)], tmp.loc[i].values)
    plt.xticks([x for x in range(0, 36)], tmp.loc[i].index, rotation=60)
plt.show()
```

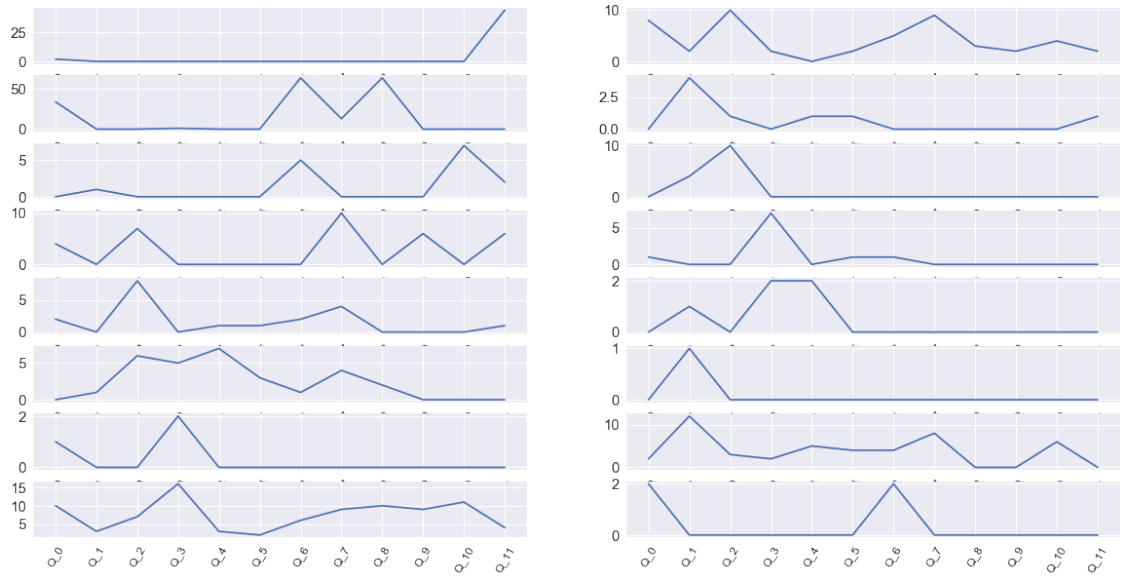


There is no periodicity in this whatsoever, perhaps we need to find other factors which influence these purchases? Or maybe we could try and represent data in some other form?

Instead of data per month, divide the data as to have data per 3 months. This allows us to predict the demand for the next three months which would be aggregated better than data per month.

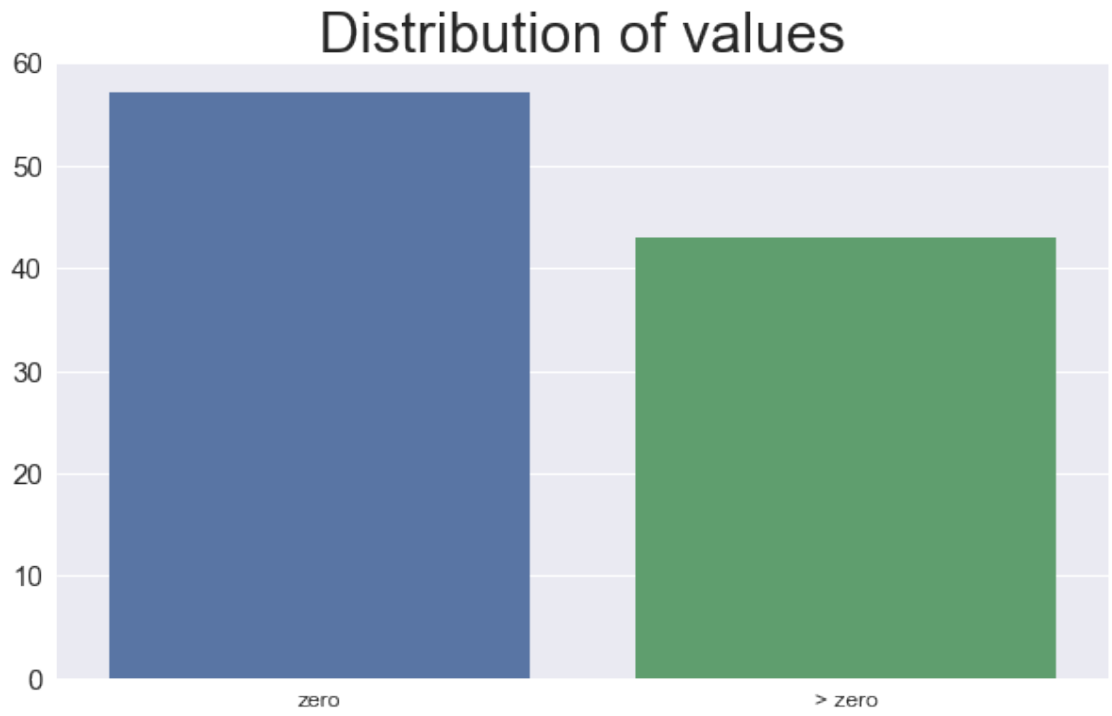
[ ]:
```
df = pd.DataFrame()
df['Material code'] = sales_past_demand['Material code']
for i in range(1, 37, 3):
    df['Q_' + str((i-1)/3)] = sales_past_demand[['DEM' + str(x) for x in␣
 ↪range(i, i + 3)]].sum(axis=1)
```

[ ]:
```
tmp = df.drop(['Material code'], axis=1)
for i in xrange(0, 16):
    plt.subplot(8, 2, i + 1)
    plt.plot([x for x in range(0, 12)], tmp.loc[i+100].values)
    plt.xticks([x for x in range(0, 12)], tmp.loc[i+100].index, rotation=60)
plt.show()
```



Here, we can some sort of patterns. this could be predicted well. First lets see the prediction value, if we predict 0 for Q_11 by default.

[ ]:
```
x = len(tmp[tmp['Q_11'] == 0])/float(len(tmp))
y = 1 - x
x = x * 100
y = y * 100
plt.figure(figsize=(10, 6))
plt.title('Distribution of values')
sns.barplot(y = [x, y], x = ['zero', '> zero'])
plt.show()
print 'If we predict the demand to be zero every time, our accuracy would␣
 ↪therefore be:', x, '%'
```




```
If we predict the demand to be zero every time, our accuracy would therefore be:
57.1072733311 %
```

So are prediction should at minimum perform better than 57%. Lets drop the 'Material code' since we're training our model to predict the demand for any given material Quaters.

So we've split the data into quarters (3 months). Since we have data of 36 months, this gives us 12 quarters. So the idea is to train the model on 11 quarters so that it is able to predict the 12th quarter.

[ ]:
```
df = df.drop(['Material code'], axis=1)
df.columns
```

[ ]:
```
Index([u'Q_0', u'Q_1', u'Q_2', u'Q_3', u'Q_4', u'Q_5', u'Q_6', u'Q_7', u'Q_8',
       u'Q_9', u'Q_10', u'Q_11'],
      dtype='object')
```

14.1 Regression

Now we have our desired inputs and desired outputs. But it wouldn't make sense to train the machine learning algorithm and test it on the same data, so we'll now split our data into training and tests sets (70% - 30%).

[ ]:
```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(df.drop('Q_11', axis=1), df.
↪Q_11, test_size=.3, random_state=42)
```

Now given the Training and testing set, we can use GridSearchCV to find best model for the given data.

ElasticNet

[ ]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import ElasticNet

param = {'alpha': [1.0, 2, 5, 10, 50, 100, 1000], 'normalize': [True, False]}
reg = GridSearchCV(ElasticNet(), param)
reg.fit(X_train, y_train)
reg.score(X_test, y_test)
```

[ ]:
```
0.86167075983978569
```

[ ]:
```
reg.best_params_
```

[ ]:
```
{'alpha': 100, 'normalize': False}
```

Lasso

[ ]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import Lasso

param = {'alpha': [1.0, 2, 5, 10, 50, 100, 1000], 'normalize': [True, False]}
reg = GridSearchCV(Lasso(), param)
reg.fit(X_train, y_train)
reg.score(X_test, y_test)
```

[ ]:
```
0.84968164809059832
```

[ ]:
```
reg.best_params_
```

[ ]:
```
{'alpha': 100, 'normalize': False}
```

Ridge

[ ]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import Ridge

param = {'alpha': [1.0, 10, 100], 'normalize': [True, False], 'solver' :␣
↪['auto', 'svd', 'cholesky', 'lsqr', 'sparse_cg', 'sag']}
reg = GridSearchCV(Ridge(), param)
reg.fit(X_train, y_train)
reg.score(X_test, y_test)
```

[ ]:
```
0.87287512860147776
```

[ ]:
```
reg.best_params_
```

[ ]:
```
{'alpha': 100, 'normalize': False, 'solver': 'sag'}
```

AdaboostRegressor

[ ]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import AdaBoostRegressor

param = {'n_estimators': [50, 100, 500], 'loss': ['linear', 'square',␣
 ↪'exponential']}
reg = GridSearchCV(AdaBoostRegressor(), param)
reg.fit(X_train, y_train)
reg.score(X_test, y_test)
```

[ ]:
```
0.74447372512362997
```

[ ]:
```
reg.best_params_
```

[ ]:
```
{'loss': 'linear', 'n_estimators': 50}
```

14.1.1 Picking the best model

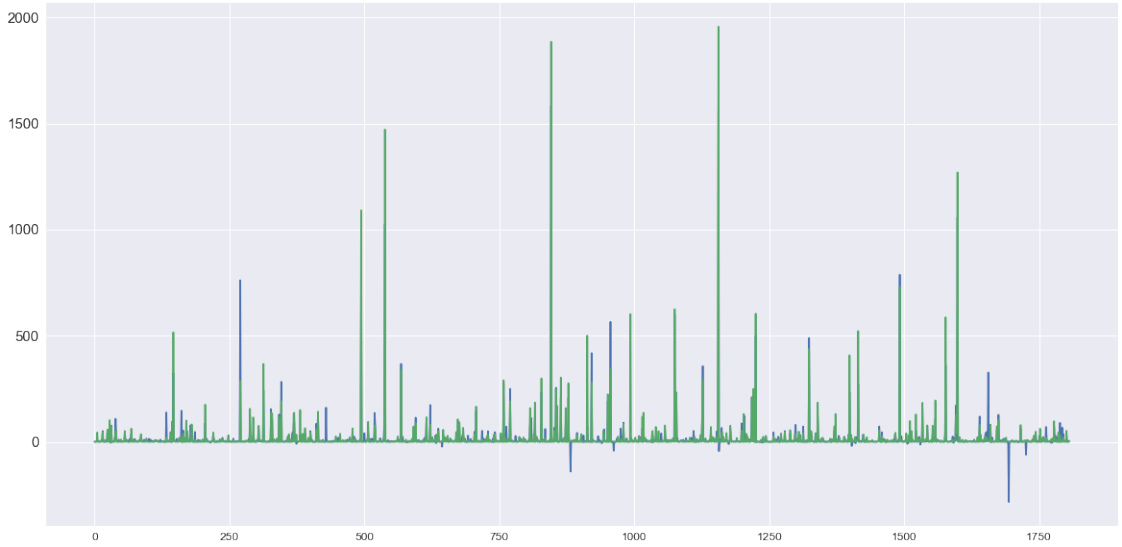
Since we got the highest score with the Ridge model, we'll use it to do our predictions.

[ ]:
```
from sklearn.linear_model import Ridge

reg = Ridge(alpha=100, normalize=False, solver='sag')
reg.fit(X_train, y_train)
reg.score(X_test, y_test)
```

[ ]:
```
0.87211240115639543
```

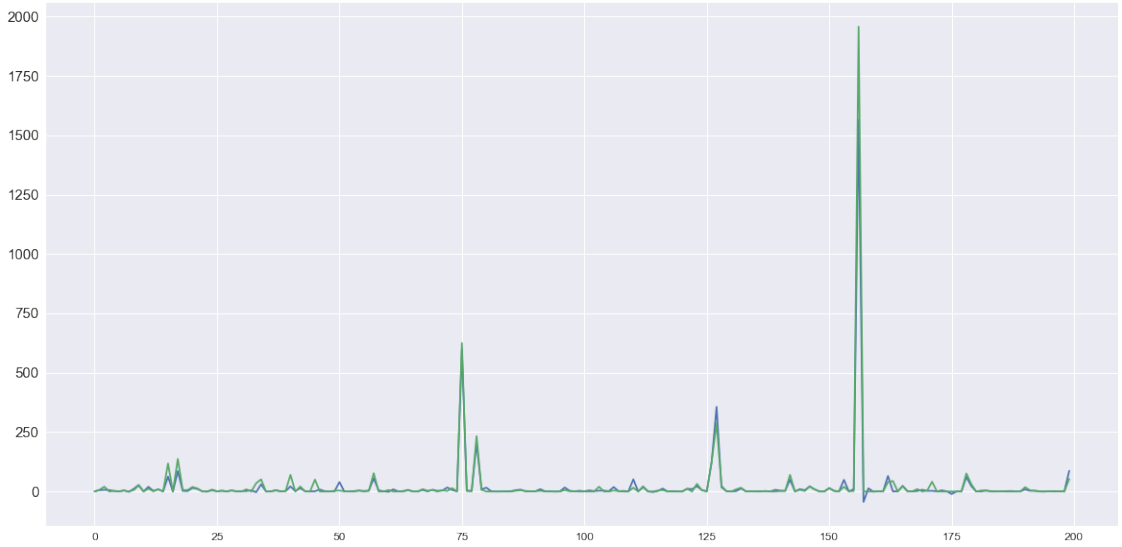
Lets compare the prediction with the actual values

[ ]:
```
pred = reg.predict(X_test)
plt.plot(pred)
plt.plot(y_test.values)
plt.show()
```

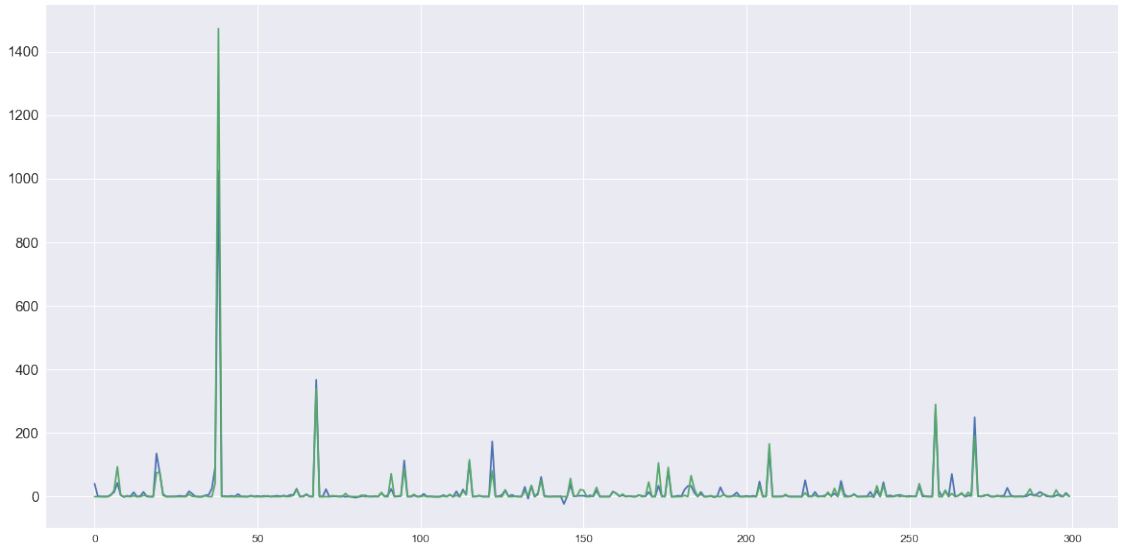


Zooming in a little:

[ ]:
```
plt.plot(pred[1000:1200])
plt.plot(y_test.values[1000:1200])
plt.show()
```



[ ]:
```
plt.plot(pred[500:800])
plt.plot(y_test.values[500:800])
plt.show()
```


1400
1200
1000
800
600
400
200
0
0
50
100
150
200
250
300