databricks736_Project_cc

```
#Imported libraries
#
import pandas as pd
from pyspark.sql.functions import *
```

```
#Accessing Datasource
#
df1 =
spark.createDataFrame(pd.read_csv("https://www.dropbox.com/s/zsbmh4vfknify4n/In
tegrated_Library_System__ILS__Data_Dictionary.csv?raw=1"))
df2 =
spark.createDataFrame(pd.read_csv("https://www.dropbox.com/s/f6a5dgy0c02t9zq/Ch
eckouts_By_Title_Data_Lens_2017.csv?raw=1")) #1
df3 =
spark.createDataFrame(pd.read_csv("https://www.dropbox.com/s/fj9h4xyrmw1n4n2/Ch
eckouts_By_Title_Data_Lens_2016.csv?raw=1")) #2
df4 =
spark.createDataFrame(pd.read_csv("https://www.dropbox.com/s/24qgdfxfdg831is/Ch
eckouts_By_Title_Data_Lens_2015.csv?raw=1")) #3
df5 =
spark.createDataFrame(pd.read_csv("https://www.dropbox.com/s/zsbmh4vfknify4n/In
tegrated_Library_System__ILS__Data_Dictionary.csv?raw=1"))
df6 =
spark.createDataFrame(pd.read_csv("https://www.dropbox.com/s/xfueukaamczjc91/Li
brary_Collection_Inventory.csv?raw=1"))
```

```
df_2005 = sqlContext.read \
    .csv("s3://projectsplheckoutrecords/Checkouts_By_Title_Data_Lens_2005.csv",
header=True)
df_2006 = sqlContext.read \
    .csv("s3://projectsplheckoutrecords/Checkouts_By_Title_Data_Lens_2006.csv",
header=True)
df_2007 = sqlContext.read \
    .csv("s3://projectsplheckoutrecords/Checkouts_By_Title_Data_Lens_2007.csv",
header=True)
df_2008 = sqlContext.read \
    .csv("s3://projectsplheckoutrecords/Checkouts_By_Title_Data_Lens_2008.csv",
header=True)
df_2009 = sqlContext.read \
    .csv("s3://projectsplheckoutrecords/Checkouts_By_Title_Data_Lens_2009.csv",
header=True)
df_2010 = sqlContext.read \
    .csv("s3://projectsplheckoutrecords/Checkouts_By_Title_Data_Lens_2010.csv",
header=True)
df_2011 = sqlContext.read \
    .csv("s3://projectsplheckoutrecords/Checkouts_By_Title_Data_Lens_2011.csv",
header=True)
df_2012 = sqlContext.read \
    .csv("s3://projectsplheckoutrecords/Checkouts_By_Title_Data_Lens_2012.csv",
header=True)
df_2013 = sqlContext.read \
    .csv("s3://projectsplheckoutrecords/Checkouts_By_Title_Data_Lens_2013.csv",
header=True)
df_2014 = sqlContext.read \
    .csv("s3://projectsplheckoutrecords/Checkouts_By_Title_Data_Lens_2014.csv",
header=True)
df_2015 = sqlContext.read \
    .csv("s3://projectsplheckoutrecords/Checkouts_By_Title_Data_Lens_2015.csv",
header=True)
df_2016 = sqlContext.read \
    .csv("s3://projectsplheckoutrecords/Checkouts_By_Title_Data_Lens_2016.csv",
header=True)
df_2017 = sqlContext.read \
    .csv("s3://projectsplheckoutrecords/Checkouts_By_Title_Data_Lens_2017.csv",
header=True)
df_DD = sqlContext.read \

.csv("s3://projectsplheckoutrecords/Integrated_Library_System__ILS__Data_Dictio
nary.csv", header=True)
df_IV = sqlContext.read \
    .csv("s3://projectsplheckoutrecords/Library_Collection_Inventory.csv",
header=True)
```

```
#example data content
#
df_2005.limit(10).display()
```

	BibNumber	ItemBarcode	ItemType	Collection	CallNumber
1	1842225	0010035249209	acbk	namys	MYSTERY ELKINS19
2	1928264	0010037335444	jcbk	ncpic	E TABACK
3	1982511	0010039952527	jcvhs	ncvidnf	VHS J796.2 KNOW_Y
4	2026467	0010040985615	accd	nacd	CD 782.421642 Y71T
5	2174698	0010047696215	jcbk	ncpic	E KROSOCZ
6	1602768	0010028318730	jcbk	ncpic	E BLACK
7	2285195	0010053424767	accd	cacd	CD 782.42166 F19R

Showing all 10 rows.

```
#since we have separate files for all years in the data, joining the files with
a union to append columns to accomodate for all the data spanning \
# all of 12 years from 2017 to 2005 is the plan: since they have the same
columns, it makes it easier for data preparation
#
df_full = df_2005.union(df_2006)
df_full =df_full.union(df_2007)
df_full =df_full.union(df_2008)
df_full =df_full.union(df_2009)
df_full =df_full.union(df_2010)
df_full =df_full.union(df_2011)
df_full =df_full.union(df_2012)
df_full =df_full.union(df_2013)
df_full =df_full.union(df_2014)
df_full =df_full.union(df_2015)
df_full =df_full.union(df_2016)
df_full =df_full.union(df_2017)
```

```
#example display from unified data
#
df_full.limit(20).display()
```

	BibNumber	ItemBarcode	ItemType	Collection	CallNumber
1	1842225	0010035249209	acbk	namys	MYSTERY ELKINS19
2	1928264	0010037335444	jcbk	ncpic	E TABACK
3					

4	2026467	0010040985615	accd	nacd	CD 782.421642 Y71T
5	2174698	0010047696215	jcbk	ncpic	E KROSOCZ
6	1602768	0010028318730	jcbk	ncpic	E BLACK
7	2285195	0010053424767	accd	cacd	CD 782.42166 F19R

Showing all 20 rows.

```
#checking to see that our join operation was successful
#
w=df_2005.count()+df_2006.count()+df_2007.count()+df_2008.count()+df_2009.count
()+df_2010.count()+df_2011.count()+df_2012.count()+df_2013.count()+df_2014.coun
t()+df_2015.count()+df_2016.count()+df_2017.count()
df_full.count() == w
```

```
Out[29]: True
```

```
#Describing the full data
#
print("Total number of instances: ",df_full.count())
```

```
Total number of instances:  91980693
```

```
#check for NANs
df_full.select([count(when(isnan(c), c)).alias(c) for c in
df_full.columns]).show()
```

```
+---------+-----------+--------+----------+----------+----------------+
|BibNumber|ItemBarcode|ItemType|Collection|CallNumber|CheckoutDateTime|
+---------+-----------+--------+----------+----------+----------------+
|        0|          0|       0|         0|         0|               0|
+---------+-----------+--------+----------+----------+----------------+
```

```
#check for NULL values
#
df_full.select([count(when(col(c).isNull(), c)).alias(c) for c in
df_full.columns]).show()
```

```
+---------+-----------+--------+----------+----------+----------------+
|BibNumber|ItemBarcode|ItemType|Collection|CallNumber|CheckoutDateTime|
+---------+-----------+--------+----------+----------+----------------+
|        0|          0|       0|         0|     13909|               0|
+---------+-----------+--------+----------+----------+----------------+
#drop the Call number column since it will not be relevant to planned insight
df_full = df_full.drop("CallNumber")
```

```
#check for duplicate rows
#
df_full.groupBy("BibNumber","ItemBarcode","ItemType", "Collection",
"CheckoutDateTime").count().filter("count > 1").count()
```

```
Out[34]: 61971
```

```
#data de-duplication
df_full = df_full.dropDuplicates()
```

```
#describing the datatypes in the full merged data
df_full.dtypes
```

```
Out[36]: [('BibNumber', 'string'),
 ('ItemBarcode', 'string'),
 ('ItemType', 'string'),
 ('Collection', 'string'),
 ('CheckoutDateTime', 'string')]
```

```
#df_full = df_full.withColumn("CheckoutDateTime",
from_unixtime(unix_timestamp("CheckoutDateTime",'MM/dd/yyyy hh:mm:ss
aa'),'MM/dd/yyyy HH:mm:ss'))
```

```
df_full.limit(5).display()
```

	BibNumber	ItemBarcode	ItemType	Collection	CheckoutDateTime
1	3172300	10087522552	acbk	namys	01/02/2017 08:13:00 A
2	3211526	10089643810	accd	nacd	01/02/2017 08:33:00 A
3	3199718	10088153514	acdvd	nadvdnf	01/02/2017 08:33:00 A
4	2543647	10063298235	accd	nacd	01/02/2017 08:13:00 A
5	2393405	10054483200	acbk	camys	01/02/2017 08:24:00 A

Showing all 5 rows.

```
df_full.dtypes
```

```
Out[44]: [('BibNumber', 'bigint'),
 ('ItemBarcode', 'bigint'),
 ('ItemType', 'string'),
 ('Collection', 'string'),
 ('CheckoutDateTime', 'string')]
```

```
"""
>>>>>>>>>>>>>>>Library collection inventory data<<<<<<<<<<<<<<<<<<<<<<
"""
#inspecting Library collection inventory data
df_IV.display()
```

	BibNum	Title
1	3011076	A tale of two friends / adapted by Ellie O'Ryan ; illustrated by Tom Caulfield, Fred
2	2248846	Naruto. Vol. 1, Uzumaki Naruto / story and art by Masashi Kishimoto ; [English ad
3	3209270	Peace, love & Wi-Fi : a ZITS treasury / by Jerry Scott and Jim Borgman.
4	1907265	The Paris pilgrims : a novel / Clancy Carlile.
5	1644616	Erotic by nature : a celebration of life, of love, and of our wonderful bodies / edite
6	1736505	Children of Cambodia's killing fields : memoirs by survivors / compiled by Dith Pr DePaul.
7	1749492	Anti-Zionism : analytical reflections / editors: Roselle Tekiner, Samir Abed-Rabbo
	3270562	Hard-hearted Highlander / Julia London

Truncated results, showing first 1000 rows.

```
#Describing the Library Inventory data
#
print("Total number of instances: ",df_IV.count())
print(df_IV.dtypes)
```

```
Total number of instances:  2687149
[('BibNum', 'string'), ('Title', 'string'), ('Author', 'string'), ('ISBN', 'str
ing'), ('PublicationYear', 'string'), ('Publisher', 'string'), ('Subjects', 'st
ring'), ('ItemType', 'string'), ('ItemCollection', 'string'), ('FloatingItem',
'string'), ('ItemLocation', 'string'), ('ReportDate', 'string'), ('ItemCount',
'string')]
```

```
#check for NANs
df_IV.select([count(when(isnan(c), c)).alias(c) for c in df_IV.columns]).show()
```

```
+------+-----+------+----+---------------+---------+--------+--------+---------
-----+------------+------------+----------+---------+
|BibNum|Title|Author|ISBN|PublicationYear|Publisher|Subjects|ItemType|ItemColle
ction|FloatingItem|ItemLocation|ReportDate|ItemCount|
+------+-----+------+----+---------------+---------+--------+--------+---------
-----+------------+------------+----------+---------+
|     0|    0|     0|   0|              0|        0|       0|       0|
0|           0|           0|         0|        0|
+------+-----+------+----+---------------+---------+--------+--------+---------
-----+------------+------------+----------+---------+
```

```
#check for NULL values
#
df_IV.select([count(when(col(c).isNull(), c)).alias(c) for c in
df_IV.columns]).show()
```

```
+------+-----+------+------+---------------+---------+--------+--------+-------
-------+------------+------------+----------+---------+
|BibNum|Title|Author|  ISBN|PublicationYear|Publisher|Subjects|ItemType|ItemCol
lection|FloatingItem|ItemLocation|ReportDate|ItemCount|
+------+-----+------+------+---------------+---------+--------+--------+-------
-------+------------+------------+----------+---------+
|     0|14324|426238|587225|          32376|    37939|   65835|    1017|
826|         783|         758|       754|      746|
+------+-----+------+------+---------------+---------+--------+--------+-------
-------+------------+------------+----------+---------+
```

```
df_IV = sqlContext.read \
    .csv("s3://projectsplheckoutrecords/Library_Collection_Inventory.csv",
header=True)
```

```
#drop rows having nulls in Author and Title columns as this will affect the
planned insight
df_IV_clean = df_IV.na.drop(subset=["Title","Author"])
```

```
df_IV_clean.where(col("Author").isNull() & col("Title").isNull()).display()
```

Query returned no results

```
#since to remove null from author information will affect the insight, as some
titles have no authors, but still have some insight \
#based on their title or publisher, null values in the table will be replaced
by *undetermined* \
#same applies to nulls in Title, Publisher, Publication year. They still have
some insights based on a non null associated column
df_IV_clean = df_IV_clean.na.fill("undetermined")
```

```
df_IV_clean.filter((col("Author")=="undetermined")&
(col("Title")=="undetermined")).display()
```

Query returned no results

```
#example data
df_IV_clean.limit(10).display()
```

	BibNum	Title
1	3011076	A tale of two friends / adapted by Ellie O'Ryan ; illustrated by Tom Caulfield, Frec
2	2248846	Naruto. Vol. 1, Uzumaki Naruto / story and art by Masashi Kishimoto ; [English a
3	3209270	Peace, love & Wi-Fi : a ZITS treasury / by Jerry Scott and Jim Borgman.
4	1907265	The Paris pilgrims : a novel / Clancy Carlile.
5	1644616	Erotic by nature : a celebration of life, of love, and of our wonderful bodies / edite
6	1736505	Children of Cambodia's killing fields : memoirs by survivors / compiled by Dith Pr DePaul.
7	1749492	Anti-Zionism : analytical reflections / editors: Roselle Tekiner, Samir Abed-Rabbo
	3270562	Hard-hearted Highlander / Julia London

Showing all 10 rows.

```
#check for duplicates
df_IV_clean.groupBy("BibNum","Title","Author", "ISBN",
"PublicationYear","Publisher","Subjects","ItemType","ItemCollection","FloatingI
tem", \
               "ItemLocation","ReportDate","ItemCount").count().filter("count >
1").count()
```

```
Out[33]: 0
```

```
#drop duplicates
df_IV_clean = df_IV_clean.dropDuplicates()
```

```
#inspecting Integrated_Library_System__ILS__Data_Dictionary data
df_DD.display()
```

	Code	Description	Code Type	Forı
1	pkbknh	Peak Picks Book	ItemType	null
2				

3	acbk	Book: Adult/YA	ItemType	Prin
4	accas	Audio Tape: Adult/YA	ItemType	Mec
5	accd	CD: Adult/YA	ItemType	Mec
6	accdrom	CD-ROM: Adult/YA	ItemType	Mec
7	acdisk	Diskette: Adult/YA	ItemType	Mec

Showing all 555 rows.

```
#Describing the full data
#
print("Total number of instances: ",df_DD.count())
print(df_DD.dtypes)
```

```
Total number of instances:  555
[('Code', 'string'), ('Description', 'string'), ('Code Type', 'string'), ('Form
at Group', 'string'), ('Format Subgroup', 'string'), ('Category Group', 'strin
g'), ('Category Subgroup', 'string')]
```

```
#check for NANs
df_DD.select([count(when(isnan(c), c)).alias(c) for c in df_DD.columns]).show()
```

```
+----+-----------+---------+------------+---------------+--------------+-------
----------+
|Code|Description|Code Type|Format Group|Format Subgroup|Category Group|Categor
y Subgroup|
+----+-----------+---------+------------+---------------+--------------+-------
----------+
|   0|          0|        0|           0|              0|             0|
0|
+----+-----------+---------+------------+---------------+--------------+-------
----------+
```

```
#check for NULL values
#
df_DD.select([count(when(col(c).isNull(), c)).alias(c) for c in
df_DD.columns]).show()
```

```
+----+-----------+---------+------------+---------------+--------------+-------
----------+
|Code|Description|Code Type|Format Group|Format Subgroup|Category Group|Categor
y Subgroup|
+----+-----------+---------+------------+---------------+--------------+-------
----------+
|   0|          0|        0|          29|             78|           297|
519|
+----+-----------+---------+-----------+-------------+------------+-------
----------+
```

```
#replacing Null values with undetermined to maintain some valuable insights
df_DD_clean = df_DD.na.fill("undetermined")
```

```
#check for duplicates
df_DD_clean.groupBy("Code","Description", "Code Type", "Format Group","Category
Group","Category Subgroup").count().filter("count > 1").count()
```

```
Out[51]: 1
```

```
#dropping the duplicate found
df_DD_clean = df_DD_clean.dropDuplicates()
```

```
#example data
df_DD_clean.limit(10).display()
```

	Code	Description	Code Type	Format Group	For
1	acart	Framed Art: Adult/YA	ItemType	Media	Art
2	accdrom	CD-ROM: Adult/YA	ItemType	Media	Data
3	acdvd	DVD: Adult/YA	ItemType	Media	Vide
4	aceq	Equipment: Adult/YA	ItemType	Equipment	und
5	acdisk	Diskette: Adult/YA	ItemType	Media	Data
6	acbk	Book: Adult/YA	ItemType	Print	Boo
7	pkbknh	Peak Picks Book	ItemType	undetermined	und

Showing all 10 rows.

```
#replacing spaces in column names
clm = [col(column).alias(column.replace(' ', '_')) for column in
df_DD_clean.columns]
df_DD_clean_ = df_DD_clean.select(clm)
```

```
#further inspection of Integrated_Library_System__ILS__Data_Dictionary data
#data is made up of different categories having one primary key column that is
better effective to split
df_DD_clean_distinct = df_DD.select("Code Type").distinct()
df_DD_clean_distinct.display()
```

	Code Type
1	ItemLocation
2	ItemType
3	ItemCollection

Showing all 3 rows.

```
#splitting the data into more manageable parts for better querying
df_DD_location = df_DD_clean_.filter(col("Code Type")=="ItemLocation")
df_DD_itemType = df_DD_clean_.filter(col("Code Type")=="ItemType")
df_DD_itemCollection = df_DD_clean_.filter(col("Code Type")=="ItemCollection")
```

```
#some parts of the data has null, but this will not affect the insight, as they
are effectively master data
print("Item Locations")
df_DD_location.display()
print("Item Types")
df_DD_itemType.limit(5).display()
print("Item Collection")
df_DD_itemCollection.display()
```

```
Item Locations
```

	Code	Description	Code_Type	For
1	mag	Magnolia, 2801 34TH AV W	ItemLocation	und
2	mon	Montlake, 2401 24TH AV E	ItemLocation	und
3	rbe	Rainier Beach, 9125 RAINIER AV S	ItemLocation	und
4	bal	Ballard, 5614 22ND AV NW	ItemLocation	und
5	col	Columbia, 4721 RAINIER AV S	ItemLocation	und
6	nga	Northgate, 10548 FIFTH AV NE	ItemLocation	und
7	lcy	Lake City, 12501 28TH AV NE	ItemLocation	und

Showing all 27 rows.

```
Item Types
```

	Code	Description	Code_Type	Format_Group	Format_Su
1	acart	Framed Art: Adult/YA	ItemType	Media	Art
2	acbk	Book: Adult/YA	ItemType	Print	Book
3	pkbknh	Peak Picks Book	ItemType	undetermined	undetermin
4	accas	Audio Tape: Adult/YA	ItemType	Media	Audio Tape
5	accd	CD: Adult/YA	ItemType	Media	Audio Disc

Showing all 5 rows.

```
Item Collection
```

	Code	Description	Code_Type	Format_Gro
1	caesla	CA1-ESL Advanced	ItemCollection	Print
2	nadocl	NA-Local Docs	ItemCollection	Print
3	caaerop	CA7-AERO Periodicals	ItemCollection	Print
4	caspec	CA7-Specifications	ItemCollection	Print
5	ncseas	NC--Children's Seasonal	ItemCollection	Print
6	naesli	NA-Eng Second Lang (Int)	ItemCollection	Print
7	namys	NA-Mysteries	ItemCollection	Print

Showing all 420 rows.

```
df_IV_clean.filter((col("Author")=="undetermined")&
(col("Title")=="undetermined")).display()
```

Query returned no results

```
#create table view for Library collection inventory data
df_IV_clean.createOrReplaceTempView("inventoryTable")


#create table view for Integrated_Library_System__ILS__Data_Dictionary data
df_DD_location.createOrReplaceTempView("LocationTable")
df_DD_itemType.createOrReplaceTempView("ItemTypeTable")
df_DD_itemCollection.createOrReplaceTempView("ItemCollectionTable")


#create table view for merged checkout records
df_full.createOrReplaceTempView("checkoutTable")
```

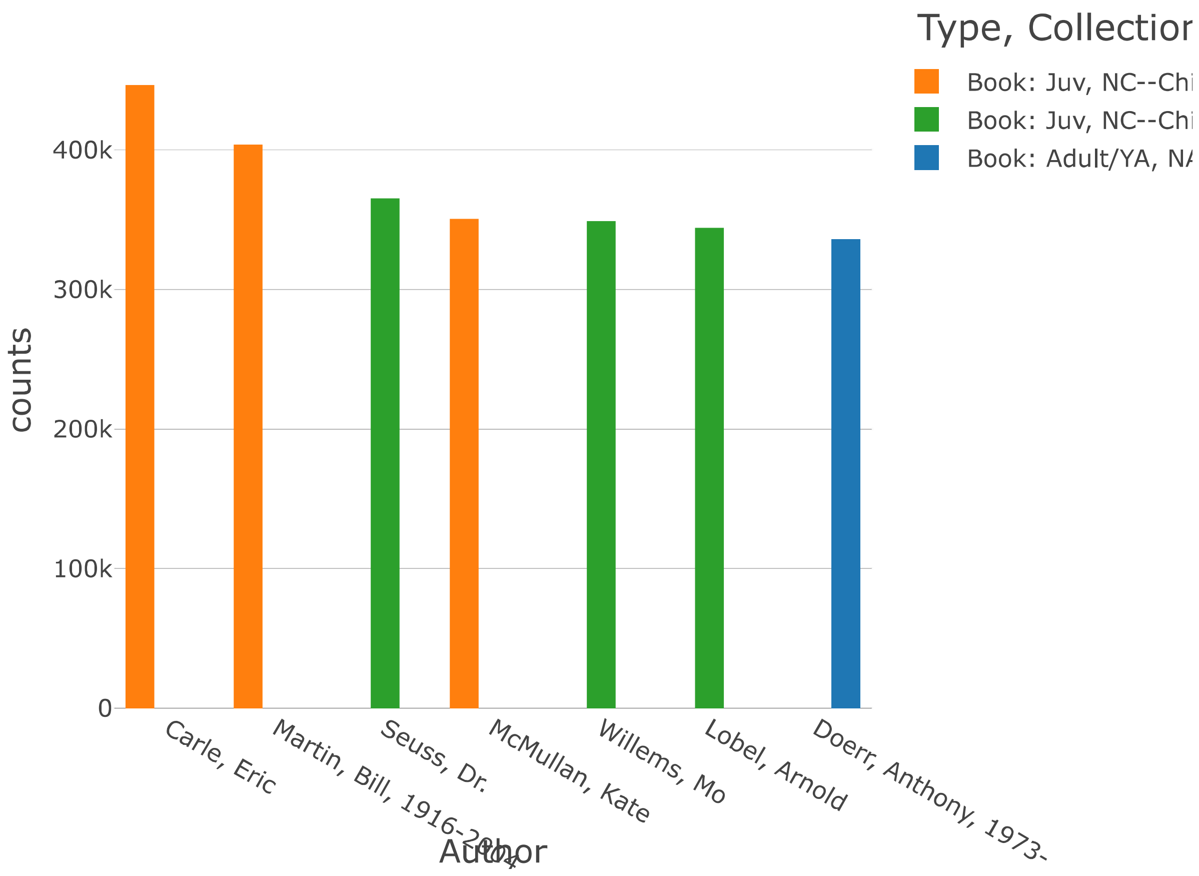
```
  Cancelled
```

```
#Queries
"""
1.) What author/Title got checked out the most historically?
2.) How does the most checked out author historically compare with that of
2017?
3.) Which months of the year are the busiest with the highest number of
checkouts historically?
4.) Which day of the week do library users checkout the most historically?
5.) What author /book/genre is most likely to be checked out using prediction?
6.) What are the busiest library locations historically?
7.) What are the busiest library locations in 2017?
8.) What are the top inventory counts by Author, Title, Subject?
9.) What are the top inventory types?
10.) What are the most checked out types historically?
11.) What are the most checked out types in 2017?
"""
```

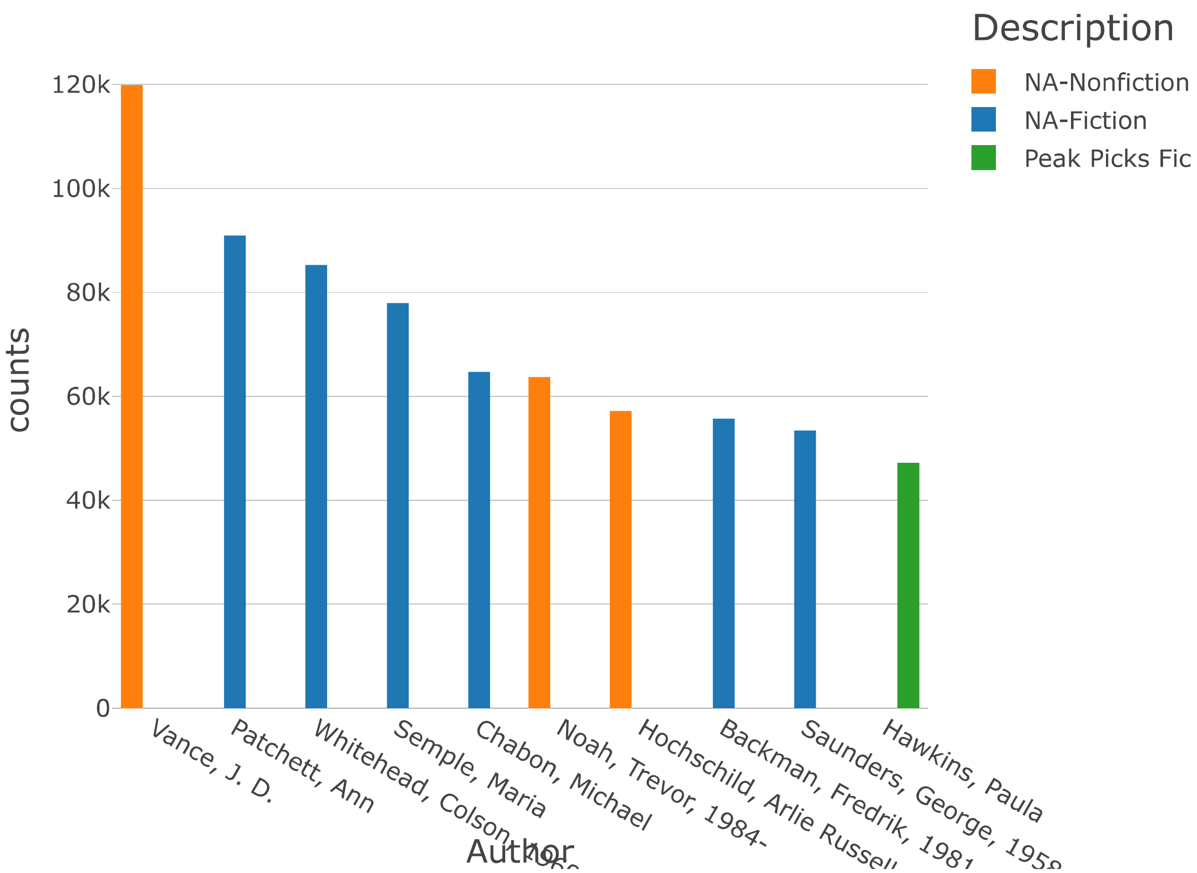
```
spark.sql("set spark.sql.legacy.timeParserPolicy=LEGACY")
```

```
Out[15]: DataFrame[key: string, value: string]
```

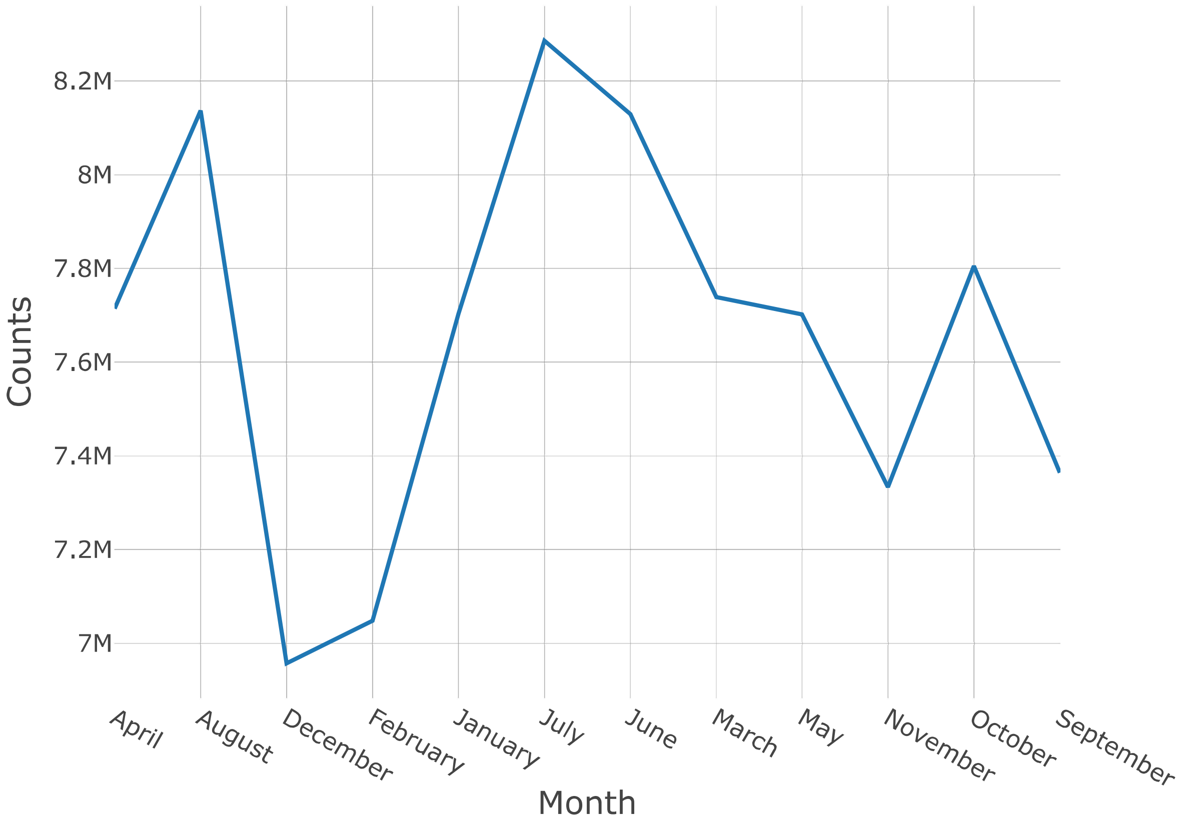
```
"""
1.) What author/Title got checked out the most historically?
"""
most_author = sqlContext.sql("""
 SELECT Author, Title, t.Description as Type, n.Description as Collection,
count(BibNumber) as counts
 FROM checkoutTable as c, inventoryTable as i, ItemTypeTable as t,
ItemCollectionTable as n
 WHERE i.BibNum == c.BibNumber
 AND t.Code == c.ItemType
 AND n.Code == c.Collection
 GROUP BY Author, Title, t.Description, n.Description
 ORDER BY counts desc
""")
display(most_author.take(10))
```



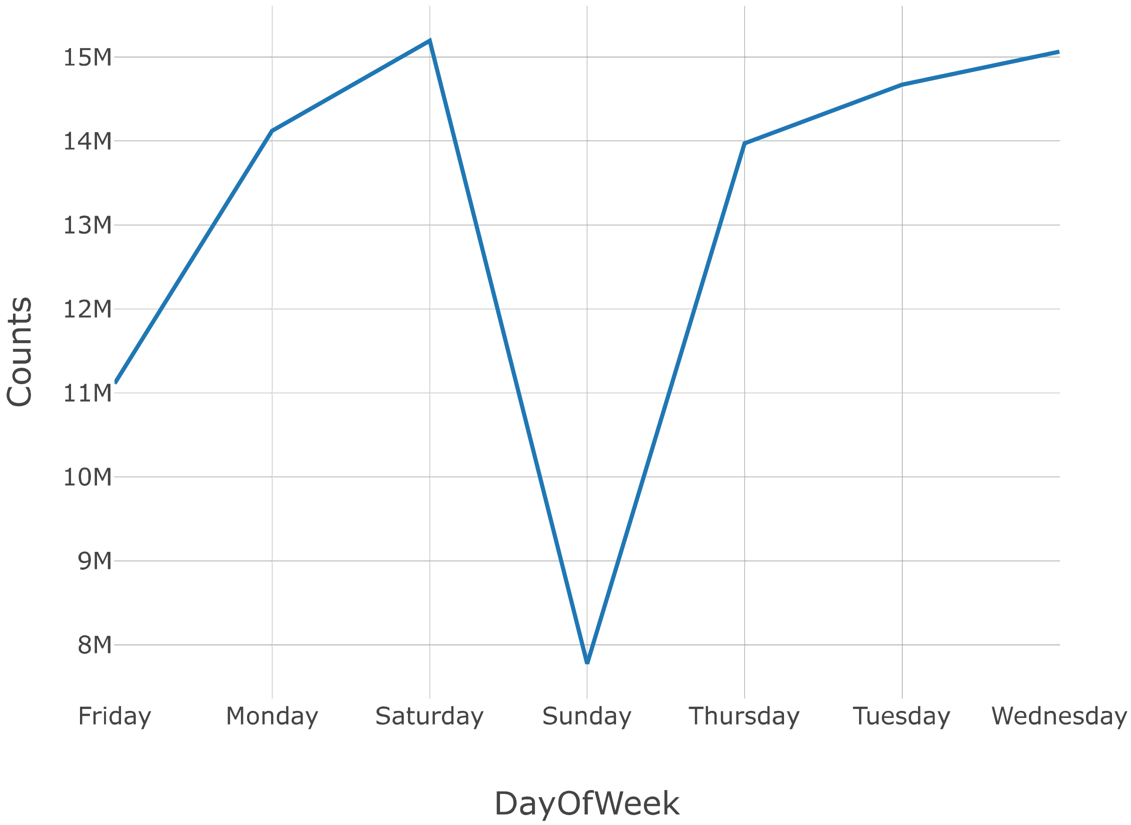
```
"""
2.) How does the most checked out author historically compare with that of
2017?
"""
most_author_recent = sqlContext.sql("""
 SELECT Author, Title, t.Description, n.Description, count(BibNumber) as counts
 FROM checkoutTable as c, inventoryTable as i, ItemTypeTable as t,
ItemCollectionTable as n
 WHERE i.BibNum == c.BibNumber
 AND t.Code == i.ItemType
 AND n.Code == c.Collection
 AND date_format((to_date(CheckoutDateTime, 'M/d/yy H:m')),'YYYY') == 2017
 GROUP BY Author, Title, t.Description, n.Description
 ORDER BY counts desc
""")
display(most_author_recent.take(10))
```



```
"""
3.) Which months of the year are the busiest with the highest number of
checkouts historically?
"""
spark.sql("set spark.sql.legacy.timeParserPolicy=LEGACY")
Top_Month = sqlContext.sql("""
 SELECT date_format((to_date(CheckoutDateTime, 'M/d/yy H:m')),'MMMM') as Month,
count(BibNumber) as Counts
 From checkoutTable
 GROUP BY Month
 ORDER BY Counts desc
""")
display(Top_Month.collect())
```

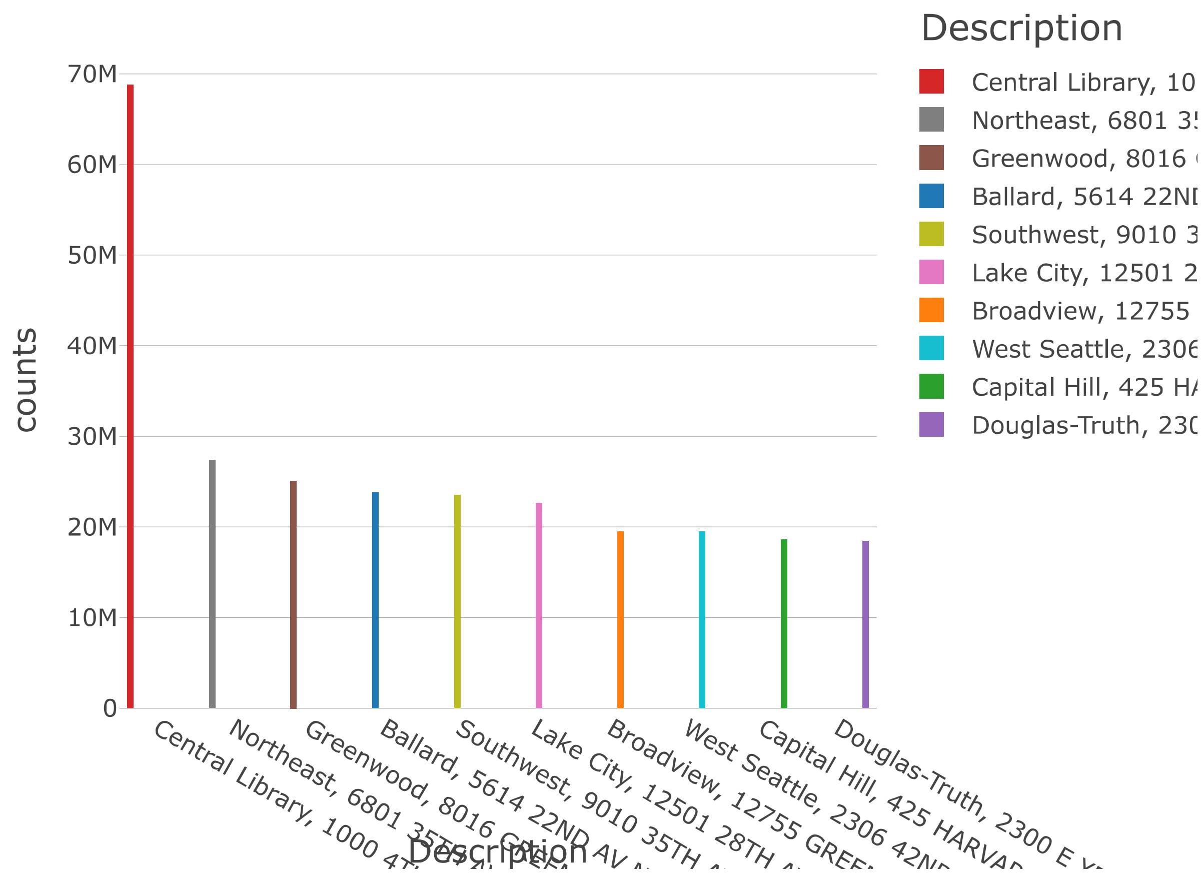


```
"""
4.) Which day of the week do library users checkout the most historically?
"""
spark.sql("set spark.sql.legacy.timeParserPolicy=LEGACY")
Top_Day = sqlContext.sql("""
 SELECT date_format((to_date(CheckoutDateTime, 'M/d/yy H:m')),'EEEE') as
DayOfWeek, count(BibNumber) as Counts
 From checkoutTable
 GROUP BY DayOfWeek
 ORDER BY Counts desc
""")
display(Top_Day.collect())
```



```
"""
5.) What author /book/genre is most likely to be checked out using prediction?
"""
```

```
"""
6.) What are the busiest library locations historically?
"""
Top_Location = sqlContext.sql("""
 SELECT  l.Description, count(BibNumber) as counts
 FROM checkoutTable as c, inventoryTable as i, LocationTable as l
 WHERE i.BibNum == c.BibNumber
 AND l.Code == i.ItemLocation
 GROUP BY l.Description
 ORDER BY counts desc
""")
display(Top_Location.take(10))
```



```
"""
7.) What are the busiest library locations in 2017?
"""
Top_Location_recent = sqlContext.sql("""
 SELECT  l.Description, count(BibNumber) as counts
 FROM checkoutTable as c, inventoryTable as i, LocationTable as l
 WHERE i.BibNum == c.BibNumber
 AND l.Code == i.ItemLocation
 AND date_format((to_date(CheckoutDateTime, 'M/d/yy H:m')),'YYYY') == 2017
 GROUP BY l.Description
 ORDER BY counts desc
""")
display(Top_Location_recent.take(10))
```

	Description	counts
1	Central Library, 1000 4TH AV	6652537
2	Northeast, 6801 35TH AV NE	3769600
3	Ballard, 5614 22ND AV NW	3398897
4		

5	Southwest, 9010 35TH AV SW	3034972
6	Lake City, 12501 28TH AV NE	2839532
7	Broadview, 12755 GREENWOOD AV N	2684434

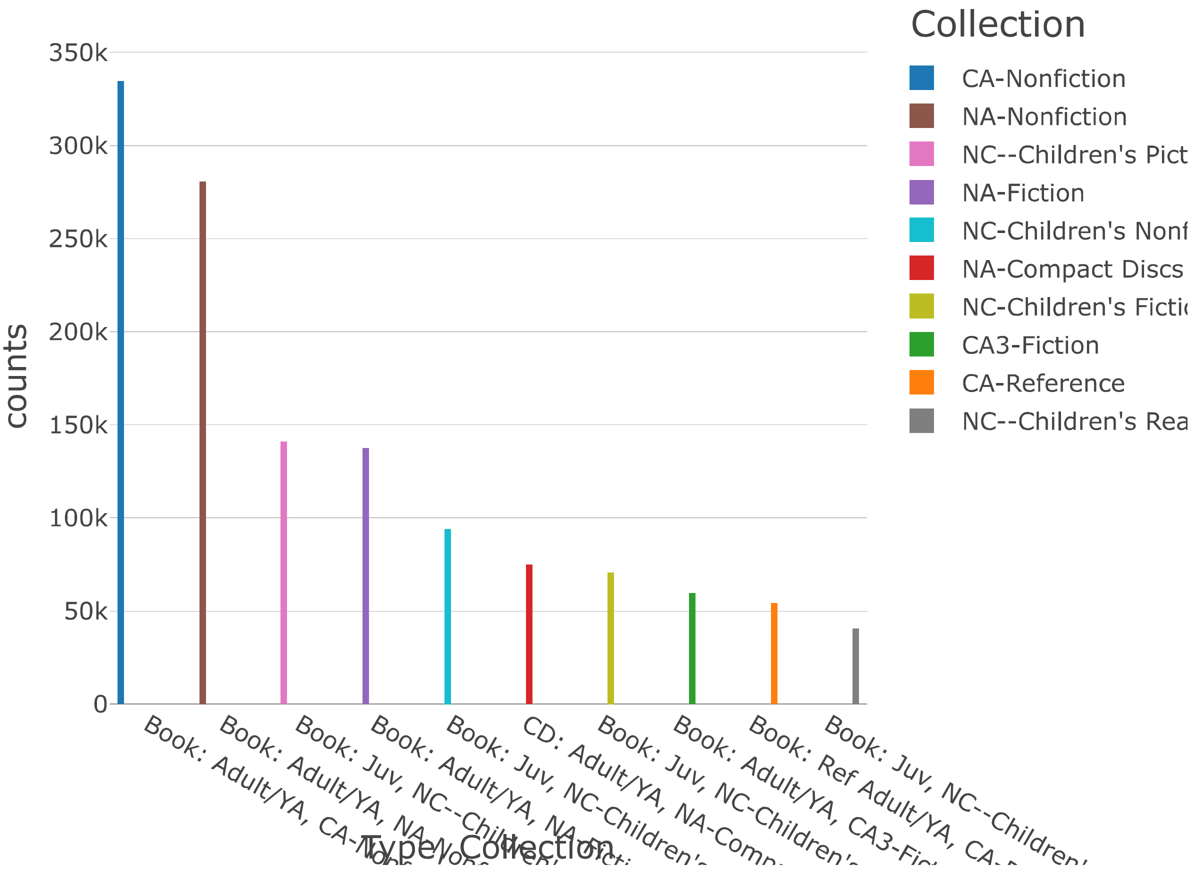
Showing all 10 rows.

```
"""
8.) What are the top inventory counts by Author, Title, Subject?
"""
Top_Inventory = sqlContext.sql("""
 SELECT Author, Title, Subjects, count(BibNum) as counts
 FROM inventoryTable as i
 GROUP BY Author, Title, Subjects
 ORDER BY counts desc
""")
display(Top_Inventory.take(10))
```

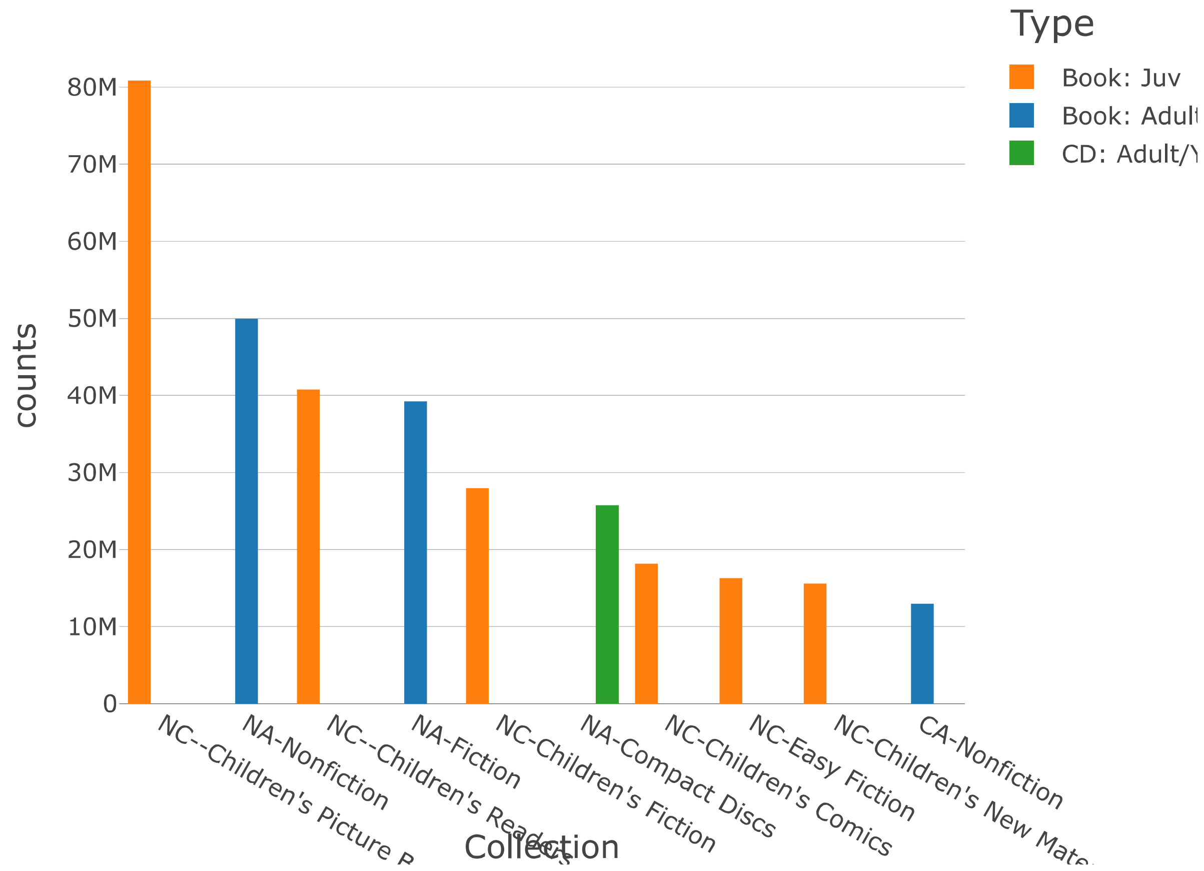
	Author	Title
1	Rand McNally and Company	Road atlas: United States, Canada, and Mexico.
2	Kalanithi, Paul.	When breath becomes air / Paul Kalanithi ; foreword by Abraha
3	Mbue, Imbolo.	Behold the dreamers : a novel / Imbolo Mbue.
4	Gaiman, Neil	Norse mythology / Neil Gaiman.
5	Lehane, Dennis	Since we fell / Dennis Lehane.
6	Nguyen, Viet Thanh, 1971-	The refugees / Viet Thanh Nguyen.
7	Child, Lee	No middle name : the complete collected Jack Reacher short s
8	Backman, Fredrik, 1981-	A man called Ove : a novel / Fredrik Backman ; [translation, He

Showing all 10 rows.

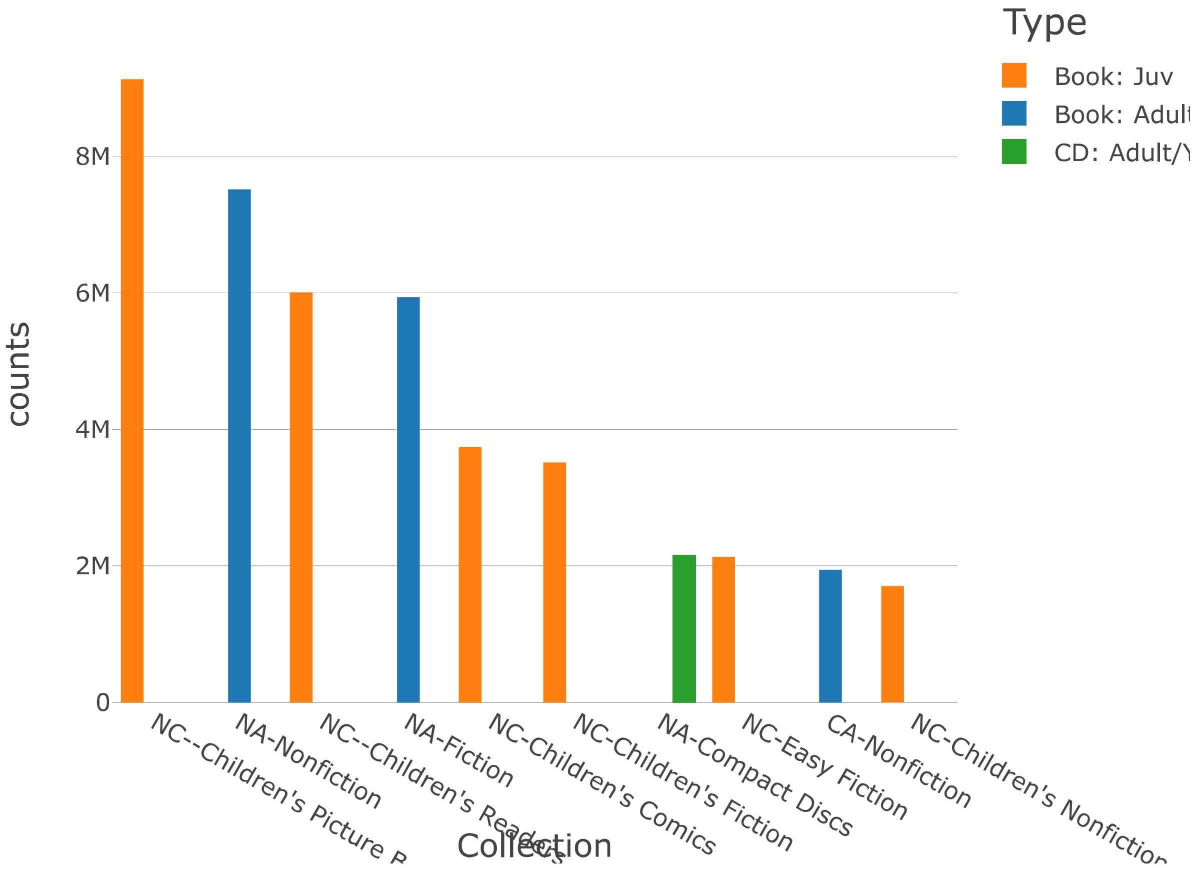
```
"""
9.) What are the top inventory types?
"""
Most_Type_count = sqlContext.sql("""
 SELECT t.Description as Type, n.Description as Collection, count(BibNum) as
counts
 FROM inventoryTable as i, ItemTypeTable as t, ItemCollectionTable as n
 WHERE t.Code == i.ItemType
 AND n.Code == i.ItemCollection
 GROUP BY n.Description, t.Description
 order by counts desc
 """)
display(Most_Type_count.take(10))
```



```
"""
10.) What are the most checked out types historically?
"""
most_type_hist = sqlContext.sql("""
 SELECT t.Description as Type, n.Description as Collection, count(BibNumber) as
counts
 FROM checkoutTable as c, inventoryTable as i, ItemTypeTable as t,
ItemCollectionTable as n
 WHERE i.BibNum == c.BibNumber
 AND t.Code == i.ItemType
 AND n.Code == c.Collection
 GROUP BY n.Description, t.Description
 order by counts desc
 """)
display(most_type_hist.take(10))
```



```
"""
11.) What are the most checked out types in 2017?
"""
most_type_recent = sqlContext.sql("""
 SELECT t.Description as Type, n.Description as Collection,
date_format((to_date(CheckoutDateTime, 'M/d/yy H:m')),'yyyy') as Year,
count(BibNumber) as counts
 FROM checkoutTable as c, inventoryTable as i, ItemTypeTable as t,
ItemCollectionTable as n
 WHERE i.BibNum == c.BibNumber
 AND t.Code == i.ItemType
 AND n.Code == c.Collection
 AND date_format((to_date(CheckoutDateTime, 'M/d/yy H:m')),'yyyy') == 2017
 GROUP BY n.Description, t.Description, Year
 order by counts desc
 """)
display(most_type_recent.take(10))
```


Type
Book: Juv
Book: Adul
CD: Adult/Y
counts
8M
6M
4M
2M
NC--Children's Picture B
NA-Nonfiction
NC--Children's Readers
NA-Fiction
NC-Children's Comics
NC-Children's Fiction
NA-Compact Discs
NC-Easy Fiction
CA-Nonfiction
NC-Children's Nonfiction
Collection