SHU CONSULTANCY

CRIME STATISTICS REPORT 2020-2022

GROUP 6

TEAM MEMBERS

- **Faiz Ud Din (31027622),** *Coordinator*
- **Harshitha Surendra Reddy (31044171),** *Complete Finisher*
- **Jagadeesan Rajalakshmi Vellaichamy (31018584),** *Specialist*
- **Tenzin Rabyang (30017118),** *Resource Investigator*
- **Vijaykumar Uppari (31048691),**

USE CASE

- The UK police department are mainly focusing improving their resource allocation and funding for the county with the highest crime rate.
- SHU Group analysis team approaches to address the business intelligence issues of the police department with five business questions and the appropriate data sources by using the Big data tools and techniques. A crime statistics report will be prepared and submitted to the police department which includes graphs, charts, and findings of each business question for the period of January 2020 till February 2022.
- The insights or information from the report can further be used by the police team in building a predictive crime model and in conducting any campaigns for spreading awareness to the public about safety measures of certain crimes.

OVERVIEW OF THE BUSINESS QUESTIONS

1

Provide the monthly breakdown of the overall UK crime rate

2

Provide the monthly breakdown of the Yorkshire & Humber crime rate by crime-type

According to a latest report [1] the county Yorkshire & Humber in England has the highest crime rate per 1,000 population with 91.9% over the time period 2020-2021.

3 **Provide the monthly breakdown of crime rate & unemployment rate by crime type**

4 **Provide the monthly breakdown of stop & search crime rate by ethnicity**

5 **Provide the monthly breakdown of the latest crime outcome rates by outcome type**

January 2020 – February 2022

[1] *UK crime rate 2020, by region*. (n.d.). Statista. https://www.statista.com/statistics/1254571/uk-crime-rate-by-region/

OVERVIEW OF THE BUSINESS QUESTIONS

3

Provide the monthly breakdown of crime rate and unemployment rate and by crime-type

Socio-economic factors add an impact to the crime rate in the society. Unemployment rate of the population is one such factor that affects the crime rate. Eventually, the police resource allocation and funding plans change as the crime rate fluctuates.

4

Provide the monthly breakdown of stop & search crime rate by ethnicity

According to a research [2], certain ethnicities are stopped and searched more frequently than the others. It is responsibility of the police department to assign a fair police team for the operation.

5

Provide a breakdown of latest crime outcome rates by outcome type, by month

The outcome rate of a crime category depends on how efficiently the forces are working with the given resources. The result helps the department in understanding the amount of police force to be deployed for working on a particular crime.

[2] HMICFRS. (2021). *Disproportionate use of police powers A spotlight on stop and search and the use of force i*. https://www.justiceinspectorates.gov.uk/hmicfrs/wp-content/uploads/disproportionate-use-of-police-powers-spotlight-on-stop-search-and-use-of-force.pdf (stop and search by ethnicity)

APPROACH TO FINDING DATA SETS

- Open datasets
- Based on last updated date
- Based on time frame of the data needed 2020 – 2022
- Based on business intelligence requirements
- Based on granularity considered for framing the business questions – by month, by country crime rates, Yorkshire & Humber detailed crime data
- Based on the data service: through API or csv

DIFFERENT DATA SETS LOOKED AT

- UK Crime Visualizations from Kaggle (Last updated year - 2018) [3] ⊗
- Crime in England and Wales (Gives Yearly data – doesn't meet the granularity level of month) [4] ⊗

[3] *Recorded Crime Data at the Police Force Area Level*. (n.d.). Www.kaggle.com. Retrieved April 28, 2022, from https://www.kaggle.com/datasets/r3w0p4/recorded-crime-data-at-police-force-area-level?resource=download

[4] Gregory, K., Khalsa, S. J., Michener, W. K., Psomopoulos, F. E., de Waard, A., & Wu, M. (2018). Eleven quick tips for finding research data. *PLOS Computational Biology*, *14*(4), e1006038. https://doi.org/10.1371/journal.pcbi.1006038

CHOSEN DATA SETS

1 **Crime rates of countries in UK** (*monthly data from January 2020 – February 2022*)

- England & Wales
  - (CSV) Edition: Year Ending September 2021 edition (***Monthly Table tab***) (Jan 2020 to Sep 2020). Link: https://cy.ons.gov.uk/peoplepopulationandcommunity/crimeandjustice/datasets/crimeinenglandandwalesquarterlydatatables
  - (Data Table) All Crime Types and ASB Totals section (Oct 2020 to Feb 2022). Link: https://www.ukcrimestats.com/National_Picture/
- Northern Ireland
  - (CSV) Police recorded monthly crime (Jan 2020 to Dec 2021). Link: https://www.opendatani.gov.uk/dataset/police-recorded-crime-in-northern-ireland
  - (Data Table) Total all crimes in Northern Ireland for Jan 2022 & Feb 2022. Link: https://www.ukcrimestats.com/
- Scotland
  - (Articles) Scraping data from the Scotland monthly official statistics publications (**Summary & Main Findings**) (Jan 2020 to Feb 2022). Link: https://www.gov.scot/collections/recorded-crime-in-scotland/

DATA FROM THE ABOVE DATA SOURCES ARE COMBINED INTO A SINGLE FILE

CHOSEN DATA SETS

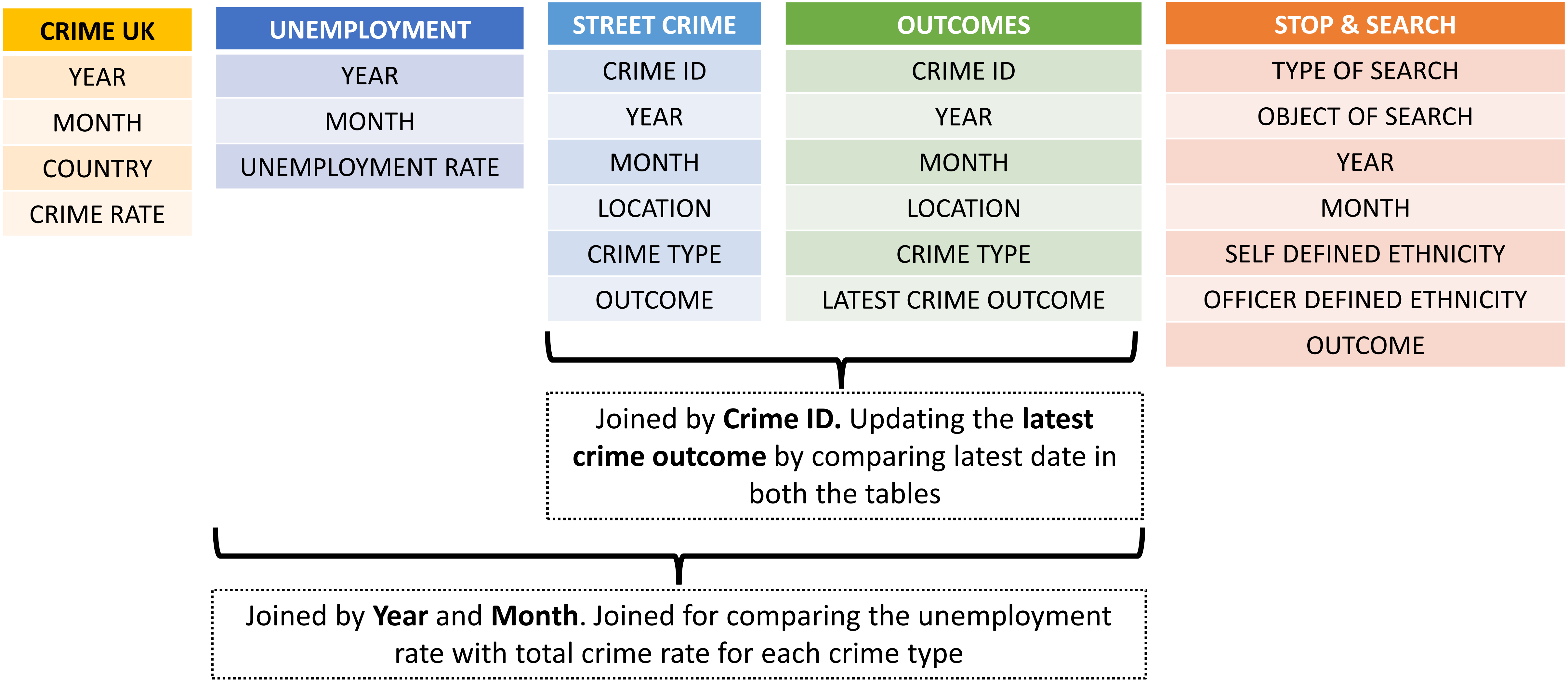
2 **Street-level crime, Outcome, and Stop and Search information for Yorkshire and Humber by police force** (*monthly data from January 2020 – February 2022*)

- 4 police forces under Yorkshire and Humber county:
  - South Yorkshire police
  - North Yorkshire police
  - West Yorkshire police
  - Humberside police
- Custom download data as **CSV** with date range from **January 2020 to February 2022** and all the above **4 police forces** from https://data.police.uk/data/

3 **Unemployment rates in Yorkshire and Humber** (*monthly data from January 2020 – February 2022*)

- https://www.ons.gov.uk/employmentandlabourmarket/peoplenotinwork/unemployment/timeseries/ycne/lms

KEY FEATURES OF THE DATA SETS



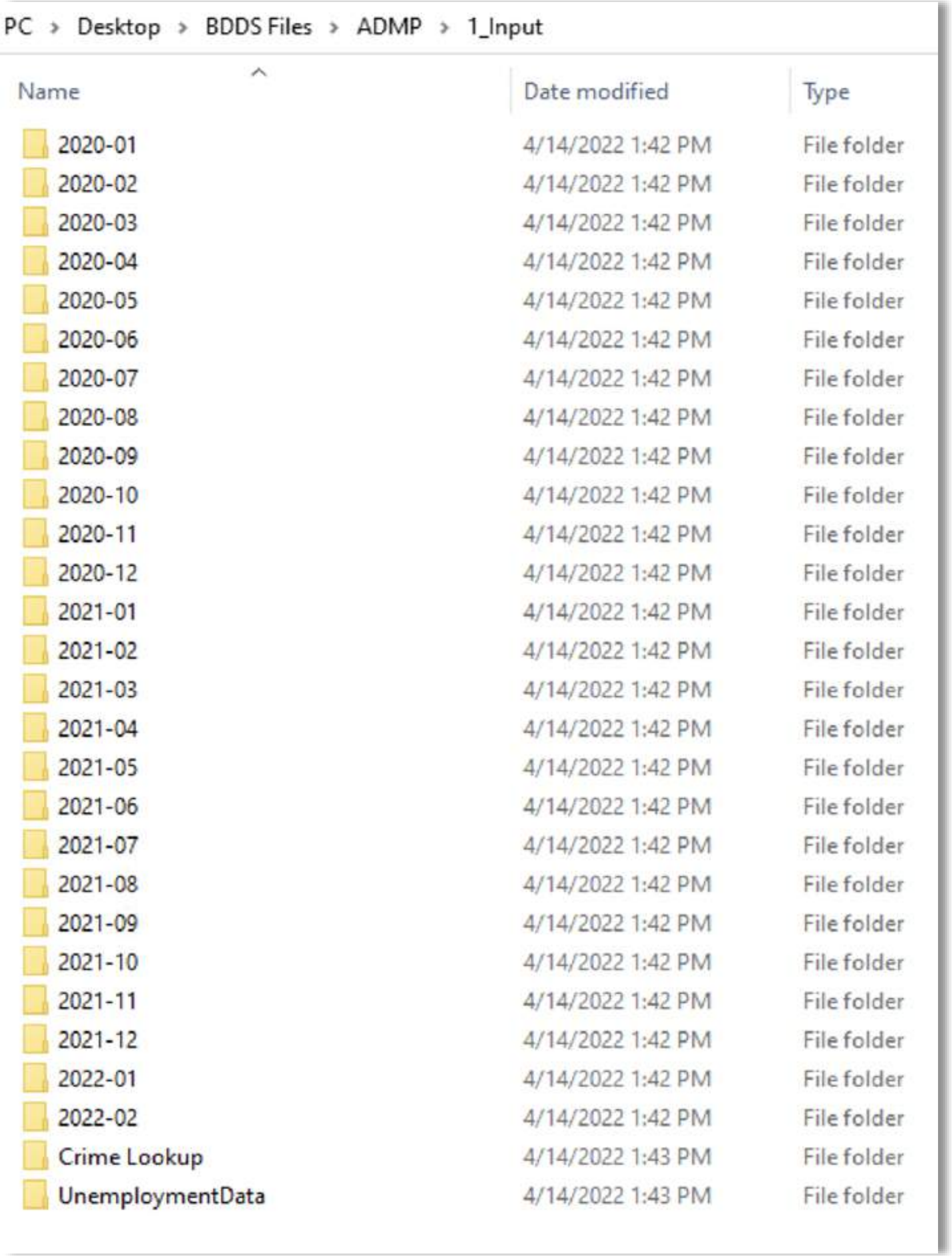
CRIME UK	UNEMPLOYMENT	STREET CRIME	OUTCOMES	STOP & SEARCH
YEAR	YEAR	CRIME ID	CRIME ID	TYPE OF SEARCH
MONTH	MONTH	YEAR	YEAR	OBJECT OF SEARCH
COUNTRY	UNEMPLOYMENT RATE	MONTH	MONTH	YEAR
CRIME RATE		LOCATION	LOCATION	MONTH
		CRIME TYPE	CRIME TYPE	SELF DEFINED ETHNICITY
		OUTCOME	LATEST CRIME OUTCOME	OFFICER DEFINED ETHNICITY
				OUTCOME

Joined by **Crime ID.** Updating the **latest crime outcome** by comparing latest date in both the tables

Joined by **Year** and **Month**. Joined for comparing the unemployment rate with total crime rate for each crime type

- The key features are unemployment rate, number of crimes derived from street-crime dataset and outcomes dataset, number of stop searches derived from stop search dataset
- To derive the business question crime type, ethnicity, outcomes, and object of search are key categorical variables.

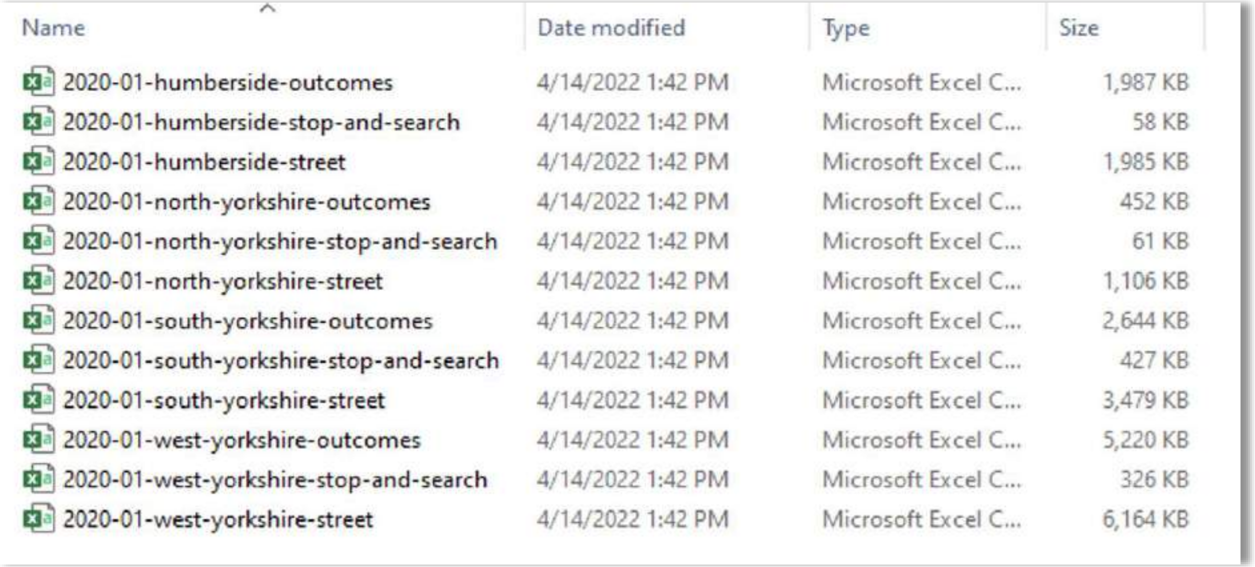
FOLDER STRUCTURE OF THE DATA SETS



PC › Desktop › BDDS Files › ADMP › 1_Input

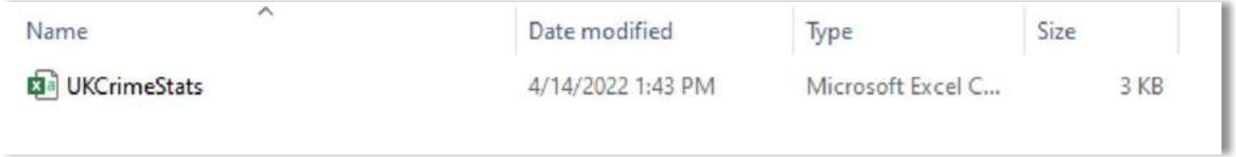
Name	Date modified	Type
2020-01	4/14/2022 1:42 PM	File folder
2020-02	4/14/2022 1:42 PM	File folder
2020-03	4/14/2022 1:42 PM	File folder
2020-04	4/14/2022 1:42 PM	File folder
2020-05	4/14/2022 1:42 PM	File folder
2020-06	4/14/2022 1:42 PM	File folder
2020-07	4/14/2022 1:42 PM	File folder
2020-08	4/14/2022 1:42 PM	File folder
2020-09	4/14/2022 1:42 PM	File folder
2020-10	4/14/2022 1:42 PM	File folder
2020-11	4/14/2022 1:42 PM	File folder
2020-12	4/14/2022 1:42 PM	File folder
2021-01	4/14/2022 1:42 PM	File folder
2021-02	4/14/2022 1:42 PM	File folder
2021-03	4/14/2022 1:42 PM	File folder
2021-04	4/14/2022 1:42 PM	File folder
2021-05	4/14/2022 1:42 PM	File folder
2021-06	4/14/2022 1:42 PM	File folder
2021-07	4/14/2022 1:42 PM	File folder
2021-08	4/14/2022 1:42 PM	File folder
2021-09	4/14/2022 1:42 PM	File folder
2021-10	4/14/2022 1:42 PM	File folder
2021-11	4/14/2022 1:42 PM	File folder
2021-12	4/14/2022 1:42 PM	File folder
2022-01	4/14/2022 1:42 PM	File folder
2022-02	4/14/2022 1:42 PM	File folder
Crime Lookup	4/14/2022 1:43 PM	File folder
UnemploymentData	4/14/2022 1:43 PM	File folder

- Each folder with the naming convention “yyyy-mm” has street crime, outcomes, and stop search files for each of the 4 regions



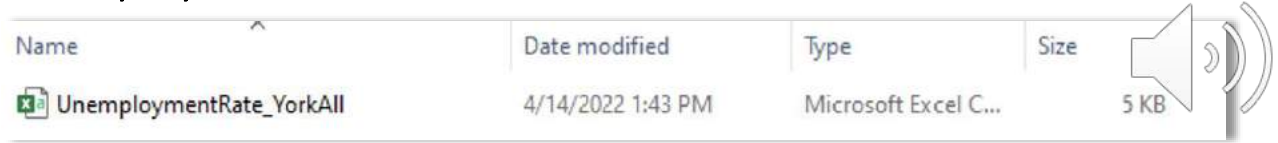
Name	Date modified	Type	Size
2020-01-humberside-outcomes	4/14/2022 1:42 PM	Microsoft Excel C...	1,987 KB
2020-01-humberside-stop-and-search	4/14/2022 1:42 PM	Microsoft Excel C...	58 KB
2020-01-humberside-street	4/14/2022 1:42 PM	Microsoft Excel C...	1,985 KB
2020-01-north-yorkshire-outcomes	4/14/2022 1:42 PM	Microsoft Excel C...	452 KB
2020-01-north-yorkshire-stop-and-search	4/14/2022 1:42 PM	Microsoft Excel C...	61 KB
2020-01-north-yorkshire-street	4/14/2022 1:42 PM	Microsoft Excel C...	1,106 KB
2020-01-south-yorkshire-outcomes	4/14/2022 1:42 PM	Microsoft Excel C...	2,644 KB
2020-01-south-yorkshire-stop-and-search	4/14/2022 1:42 PM	Microsoft Excel C...	427 KB
2020-01-south-yorkshire-street	4/14/2022 1:42 PM	Microsoft Excel C...	3,479 KB
2020-01-west-yorkshire-outcomes	4/14/2022 1:42 PM	Microsoft Excel C...	5,220 KB
2020-01-west-yorkshire-stop-and-search	4/14/2022 1:42 PM	Microsoft Excel C...	326 KB
2020-01-west-yorkshire-street	4/14/2022 1:42 PM	Microsoft Excel C...	6,164 KB

- Crime Lookup



Name	Date modified	Type	Size
UKCrimeStats	4/14/2022 1:43 PM	Microsoft Excel C...	3 KB

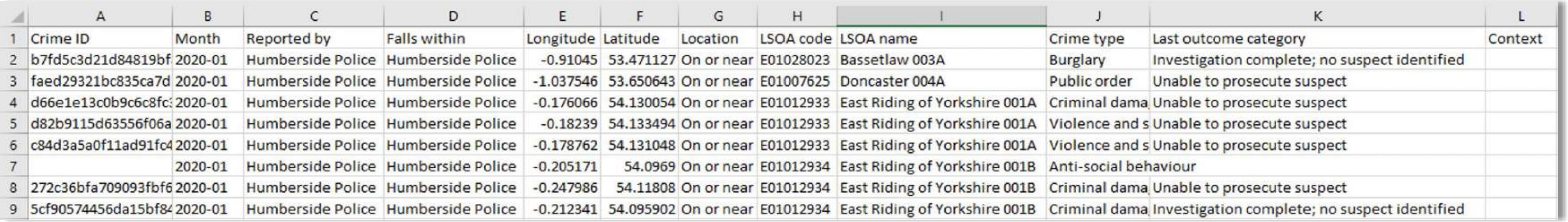
- Unemployment Data



Name	Date modified	Type	Size
UnemploymentRate_YorkAll	4/14/2022 1:43 PM	Microsoft Excel C...	5 KB

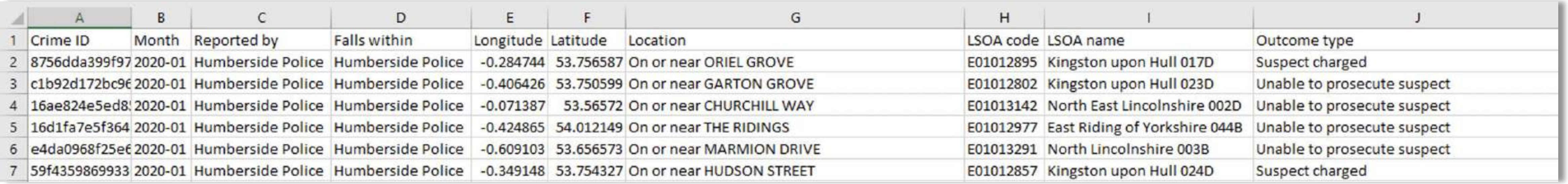
SNIPPETS OF THE DATA SETS

➢ Street-Crime Final



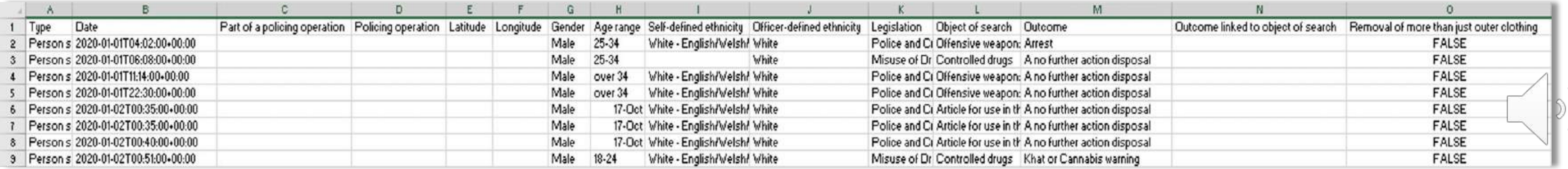
	A	B	C	D	E	F	G	H	I	J	K	L
1	Crime ID	Month	Reported by	Falls within	Longitude	Latitude	Location	LSOA code	LSOA name	Crime type	Last outcome category	Context
2	b7fd5c3d21d84819bf	2020-01	Humberside Police	Humberside Police	-0.91045	53.471127	On or near	E01028023	Bassetlaw 003A	Burglary	Investigation complete; no suspect identified	
3	faed29321bc835ca7d	2020-01	Humberside Police	Humberside Police	-1.037546	53.650643	On or near	E01007625	Doncaster 004A	Public order	Unable to prosecute suspect	
4	d66e1e13c0b9c6c8fc:	2020-01	Humberside Police	Humberside Police	-0.176066	54.130054	On or near	E01012933	East Riding of Yorkshire 001A	Criminal dama	Unable to prosecute suspect	
5	d82b9115d63556f06a	2020-01	Humberside Police	Humberside Police	-0.18239	54.133494	On or near	E01012933	East Riding of Yorkshire 001A	Violence and s	Unable to prosecute suspect	
6	c84d3a5a0f11ad91fc4	2020-01	Humberside Police	Humberside Police	-0.178762	54.131048	On or near	E01012933	East Riding of Yorkshire 001A	Violence and s	Unable to prosecute suspect	
7		2020-01	Humberside Police	Humberside Police	-0.205171	54.0969	On or near	E01012934	East Riding of Yorkshire 001B	Anti-social behaviour		
8	272c36bfa709093fbf6	2020-01	Humberside Police	Humberside Police	-0.247986	54.11808	On or near	E01012934	East Riding of Yorkshire 001B	Criminal dama	Unable to prosecute suspect	
9	5cf90574456da15bf84	2020-01	Humberside Police	Humberside Police	-0.212341	54.095902	On or near	E01012934	East Riding of Yorkshire 001B	Criminal dama	Investigation complete; no suspect identified	

➢ Outcomes Final



	A	B	C	D	E	F	G	H	I	J
1	Crime ID	Month	Reported by	Falls within	Longitude	Latitude	Location	LSOA code	LSOA name	Outcome type
2	8756dda399f97	2020-01	Humberside Police	Humberside Police	-0.284744	53.756587	On or near ORIEL GROVE	E01012895	Kingston upon Hull 017D	Suspect charged
3	c1b92d172bc96	2020-01	Humberside Police	Humberside Police	-0.406426	53.750599	On or near GARTON GROVE	E01012802	Kingston upon Hull 023D	Unable to prosecute suspect
4	16ae824e5ed8:	2020-01	Humberside Police	Humberside Police	-0.071387	53.56572	On or near CHURCHILL WAY	E01013142	North East Lincolnshire 002D	Unable to prosecute suspect
5	16d1fa7e5f364	2020-01	Humberside Police	Humberside Police	-0.424865	54.012149	On or near THE RIDINGS	E01012977	East Riding of Yorkshire 044B	Unable to prosecute suspect
6	e4da0968f25e6	2020-01	Humberside Police	Humberside Police	-0.609103	53.656573	On or near MARMION DRIVE	E01013291	North Lincolnshire 003B	Unable to prosecute suspect
7	59f4359869933	2020-01	Humberside Police	Humberside Police	-0.349148	53.754327	On or near HUDSON STREET	E01012857	Kingston upon Hull 024D	Suspect charged

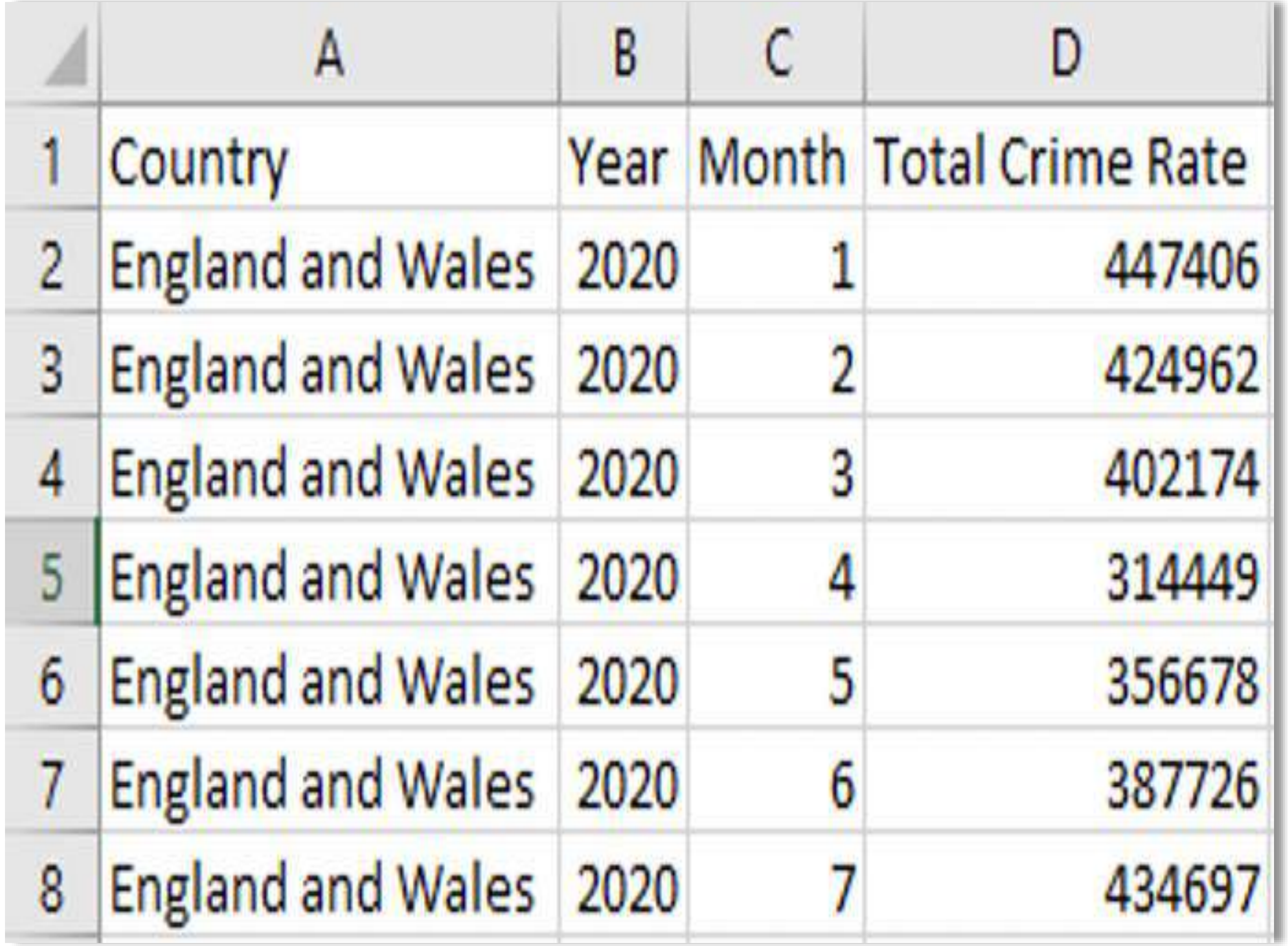
➢ Stop & Search Final



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Type	Date	Part of a policing operation	Policing operation	Latitude	Longitude	Gender	Age range	Self-defined ethnicity	Officer-defined ethnicity	Legislation	Object of search	Outcome	Outcome linked to object of search	Removal of more than just outer clothing
2	Person s	2020-01-01T04:02:00+00:00					Male	25-34	White - English/Welsh/	White	Police and Cı	Offensive weapon:	Arrest		FALSE
3	Person s	2020-01-01T06:08:00+00:00					Male	25-34		White	Misuse of Dr	Controlled drugs	A no further action disposal		FALSE
4	Person s	2020-01-01T11:14:00+00:00					Male	over 34	White - English/Welsh/	White	Police and Cı	Offensive weapon:	A no further action disposal		FALSE
5	Person s	2020-01-01T22:30:00+00:00					Male	over 34	White - English/Welsh/	White	Police and Cı	Offensive weapon:	A no further action disposal		FALSE
6	Person s	2020-01-02T00:35:00+00:00					Male	17-Oct	White - English/Welsh/	White	Police and Cı	Article for use in tł	A no further action disposal		FALSE
7	Person s	2020-01-02T00:35:00+00:00					Male	17-Oct	White - English/Welsh/	White	Police and Cı	Article for use in tł	A no further action disposal		FALSE
8	Person s	2020-01-02T00:40:00+00:00					Male	17-Oct	White - English/Welsh/	White	Police and Cı	Article for use in tł	A no further action disposal		FALSE
9	Person s	2020-01-02T00:51:00+00:00					Male	18-24	White - English/Welsh/	White	Misuse of Dr	Controlled drugs	Khat or Cannabis warning		FALSE

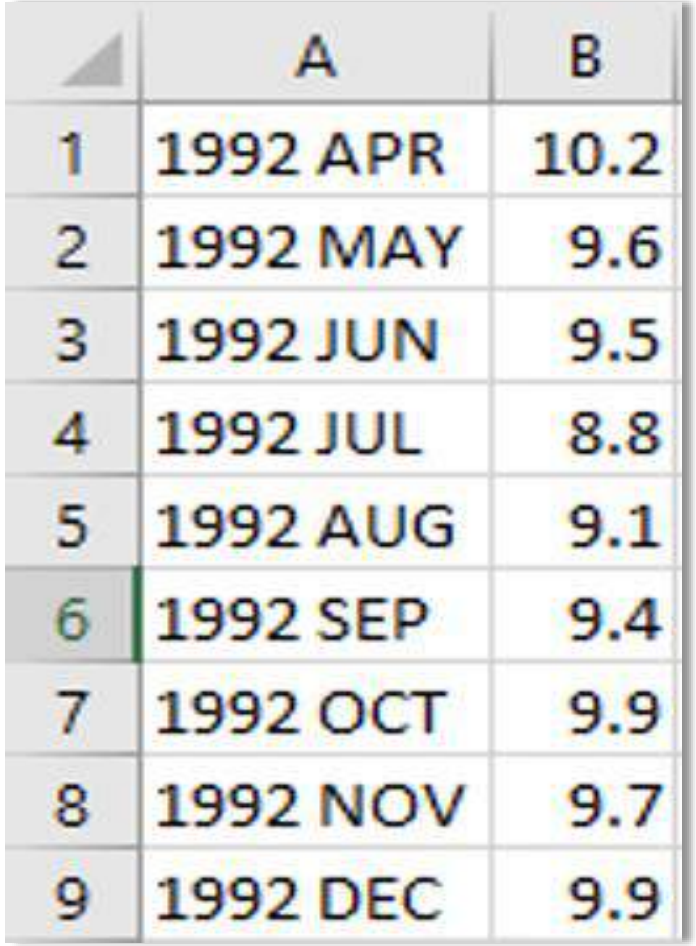
SNIPPETS OF THE DATA SETS

- **Crime Rate UK Stats**



	A	B	C	D
1	Country	Year	Month	Total Crime Rate
2	England and Wales	2020	1	447406
3	England and Wales	2020	2	424962
4	England and Wales	2020	3	402174
5	England and Wales	2020	4	314449
6	England and Wales	2020	5	356678
7	England and Wales	2020	6	387726
8	England and Wales	2020	7	434697

- **Unemployment rate**



	A	B
1	1992 APR	10.2
2	1992 MAY	9.6
3	1992 JUN	9.5
4	1992 JUL	8.8
5	1992 AUG	9.1
6	1992 SEP	9.4
7	1992 OCT	9.9
8	1992 NOV	9.7
9	1992 DEC	9.9

RECOMMENDED SOFTWARE SOLUTION

- Data Extraction: CRAN R / R Studio is used to extract from data source.
- ETL: Zeppelin SparkR, a component of Hadoop environment is used to apply data transformations and data validity checks.
- Data marts: Hive is used for querying the historical data stored as facts and dimensions.
- Business Intelligence: A visual analytics platform Tableau is used for each of the business question in the form of Charts/Dashboard.

FUTURE DEPLOYMENT

- Instead of Zeppelin we can migrate to **Azure Databricks** for enterprise-wide deployment – Big data analytics and AI with optimized Apache Spark
  - The performance of entire ETL Process can be boosted
  - Production-ready - integrations for CI/CD pipeline and monitoring.
- **Snowflake**
  - High storage capacity
  - Supports multi cloud

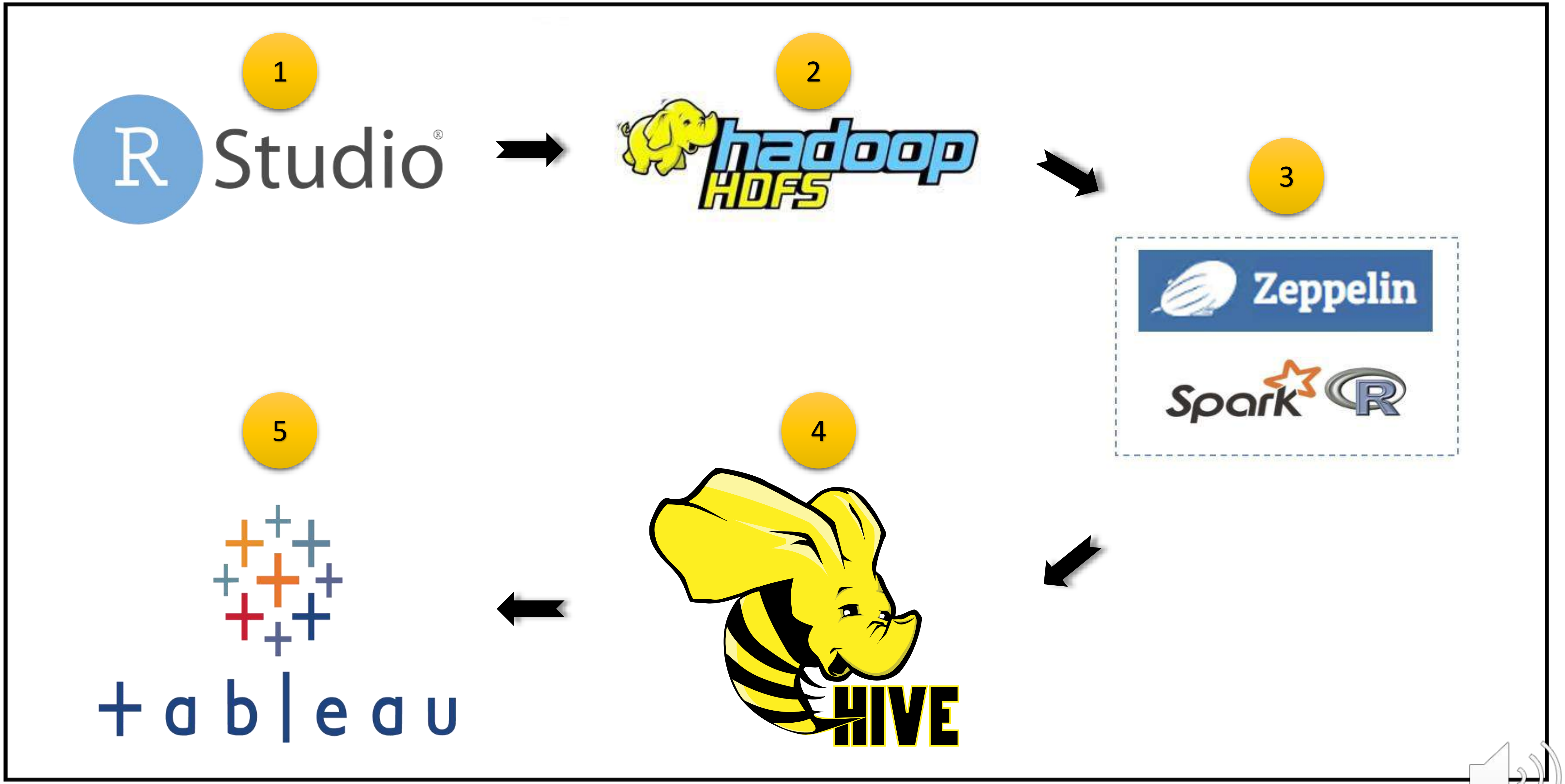
DEVELOPMENT METHODOLOGIES

METHOD	JUSTIFICATION
DEVELOPMENT METHOD: **WATERFALL**	Reasons for choosing waterfall model: • Business requirements are very clear prior to the start of the project • A short period project of one month • A single and stable end-product – business intelligence report Requirements collection -> Design -> Implementation -> Integration & Testing -> Deployment -> Maintenance
DATA WAREHOUSE DESIGN: **KIMBALL'S BOTTOM-UP APPROACH**	In Kimball's bottom-up approach the first step is to have the data marts ready. Later these marts can be used to build the data warehouse. Kimball's approach is best suitable for the below reasons: • The data marts are sufficient to answer the business questions for the current use case • Initial set-up takes very less time and is suitable for the short-term project. Identifying business questions -> Identify lowest granularity -> Identify metrics -> Define dimension tables-> Define fact tables

BDDS AND DI TECHNIQUES

(BDDS) Data Storage - HDFS	Hadoop Distributed File System supports distributed processing of high data volume with high data velocity for variety of data.
(DI/BDDS) Staging – SPARK (HDFS)	The staging location by default is user’s home directory /user/maria_dev/ in HDFS filesystem.
(DI/BDDS) Extract, Transform and Load Zeppelin SparkR Interpreter	Speed, Ease of use, Run applications in Hadoop clusters up to x100 in memory and x10 on disk, In-memory storage (as much possible, and spill over to disk), Near real-time processing, Several times faster than other Big Data technologies and used in Tech Giants like Netflix, Uber etc
(DI/BDDS) Data Quality - Zeppelin SparkR Interpreter	Handles Incomplete, Incorrect, Incomprehensible and Inconsistent data with variety of functions which delivers distributed processing with high velocity
Data Warehousing Design Approaches Bottom-Up The Kimball Method	In Kimball’s bottom-up approach the first step is to have the data marts ready. Later these marts can be used to build the data warehouse. • Kimball’s approach is best suitable for the below reasons: • The data marts are sufficient to answer the business questions for the current use case Initial set-up takes very less time and is suitable for the short-term project.
Date Warehouse - OLAP HIVE	Supports historical data store with huge volume of data. Data – summarized Access – Many Records querying decision support systems faster
Presenting Information - Tableau	Its in leader quadrant in Gartner chart for top business intelligence platform Custom Query build, Mostly relationships between tables are auto detected and connected Aesthetic BI Dashboards with many advanced and complicated charts can be produced

PREREQUISITES



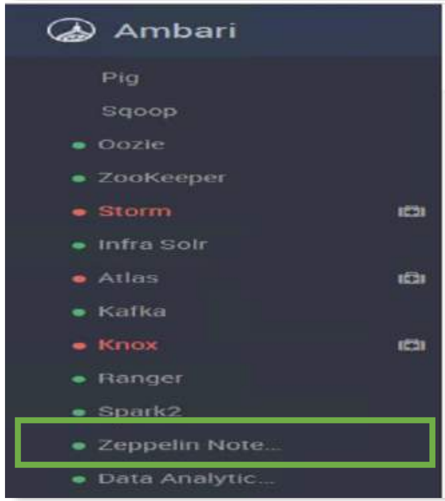
RStudio

- **RStudio** is used to for extracting the data and append all the street-crime, outcomes, and stop & search files from all the folders into single files

Zeppelin SparkR

- SparkR (*%spark2.r*) in Zeppelin has the direct connectivity with the HDFS location. It is used for performing the **data extraction, and the data transformation**
- Installing SparkR interpreter in Zeppelin: Run the following commands in Putty

```
1) yum install R R-devel libcurl-devel
openssl-devel
2) + devtools with `R -e
"install.packages('devtools', repos =
'http://cran.us.r-project.org’)”`
3) + knitr with `R -e
"install.packages('knitr', repos =
'http://cran.us.r-project.org')"`
```

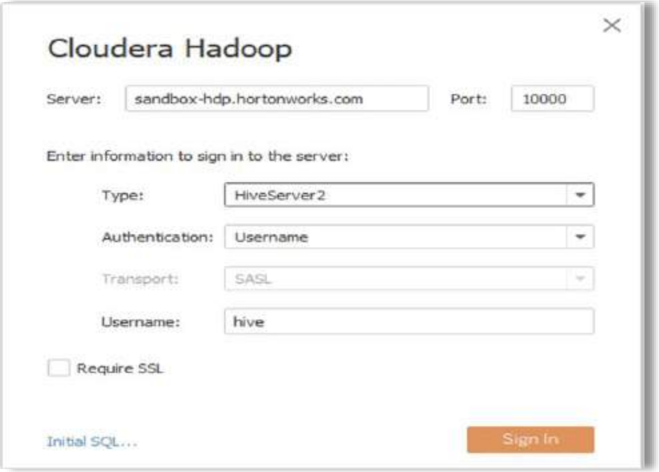


Hive

- JDBC Hive (*%jdbc(hive)*) in Zeppelin is used for creating the fact and dimension tables and loading the data into hive tables

Tableau

- Tableau is used for the analyzing the transformed data and to generate visualizations of the data trends for each of the business questions
- Hive to Tableau connection details



Cloudera Hadoop

Server: sandbox-hdp.hortonworks.com Port: 10000

Enter information to sign in to the server:

Type: HiveServer2

Authentication: Username

Transport: SASL

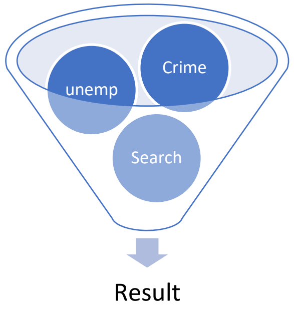
Username: hive

□ Require SSL

ETL PROCESS

1. Data Extract From HDFS

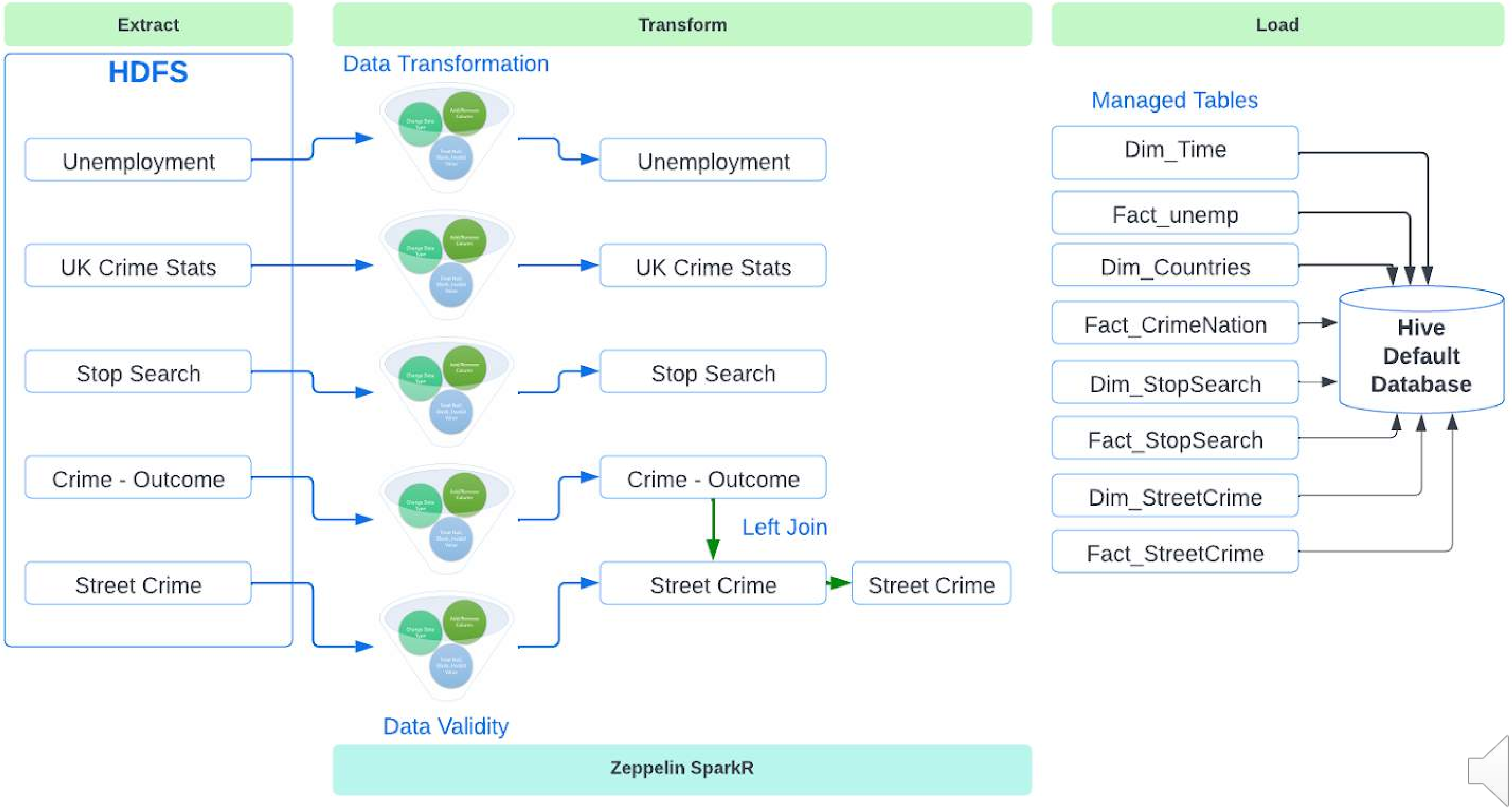
2. Transformation Stage



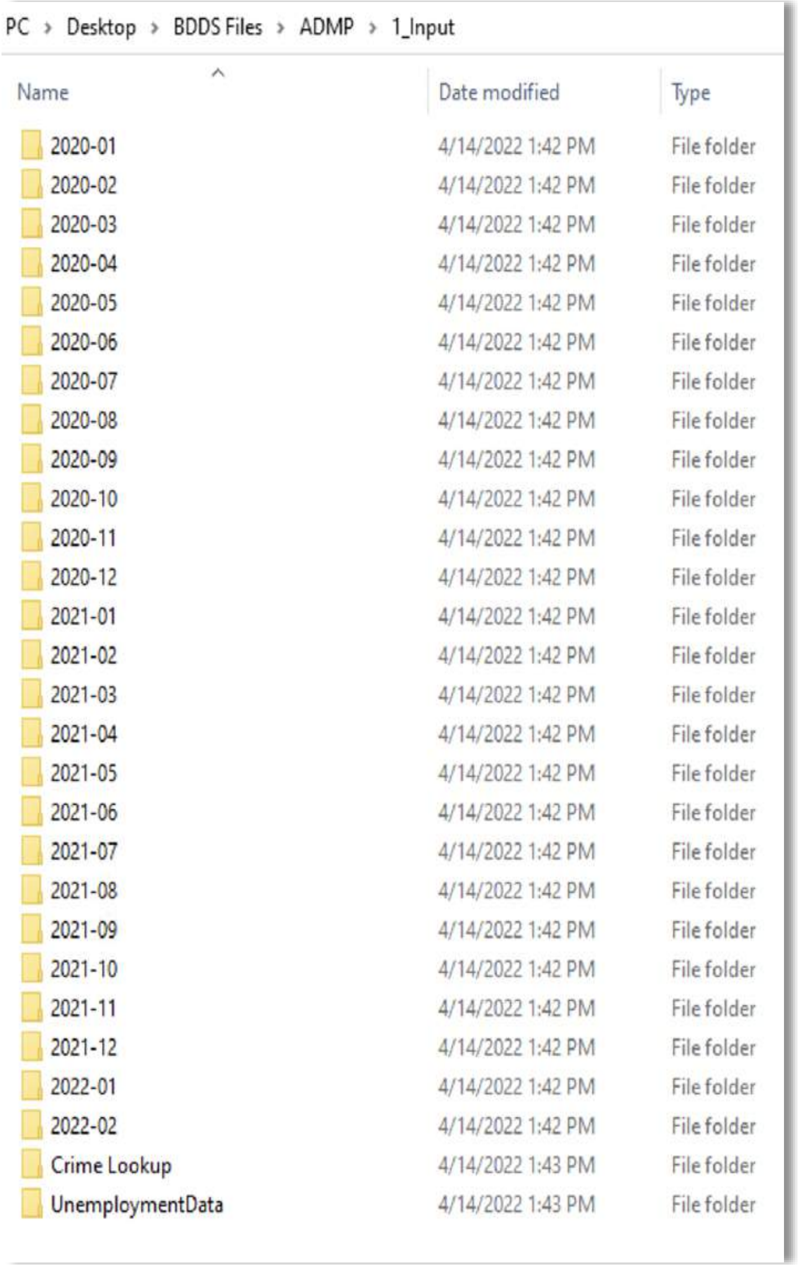
3. Data Load to HIVE (FACT/DIMENSION)

Steps	Data Validity – SparkR Zeppelin	Data Transformation – SParkR Zeppelin
1	Rows and Columns Count before Transformation	-
2	-	New Columns Derivation
3	-	Remove unwanted columns
4	Rows and Columns count after columns removal	-
5		Calculating aggregates based on group by variables
6	Rows and Columns count after aggregating by variables	
7	Data Type Check on each column – Before	Convert to appropriate column datatype (CAST)
8	Data Type Check on each column – After conversion	
9	-	Data filter >= 2020
10	Rows and Columns Count after filter conditions applied	-
11	Frequency distribution on each columns to identify Nulls, Etc.	11.1 Treatment: • Numeric Column = Null to -99 value change • Character column = Null to “Undefined” • Delete ID columns if NULL and Crime ID length != 64 characters 11.2 Treatment: • Character column = ‘NA’ to “Undefined” • Character column = Invalid values to “Undefined” 11.3 Treatment: Character column = Blank values to “Undefined”
12	Frequency distribution on each columns to check columns value treatments applied	-
13	Duplicate Records - Check	Remove Duplicate Records
14	Rows and Columns Count After dropping duplicates	-
15	Final Check on Rows and Columns before creating Fact and Dimension tables and Loading Stage (Hive)	-
16	-	Create Fact and Dimension tables
17	Rows and Columns Count after Fact/Dimension Split	

ETL PLAN


Extract
Transform
Load
HDFS
Unemployment
UK Crime Stats
Stop Search
Crime - Outcome
Street Crime
Data Transformation
Unemployment
UK Crime Stats
Stop Search
Crime - Outcome
Left Join
Street Crime
Street Crime
Data Validity
Managed Tables
Dim_Time
Fact_unemp
Dim_Countries
Fact_CrimeNation
Dim_StopSearch
Fact_StopSearch
Dim_StreetCrime
Fact_StreetCrime
Hive Default Database
Zeppelin SparkR

DATA STAGING – EXTRACTION PROCESS

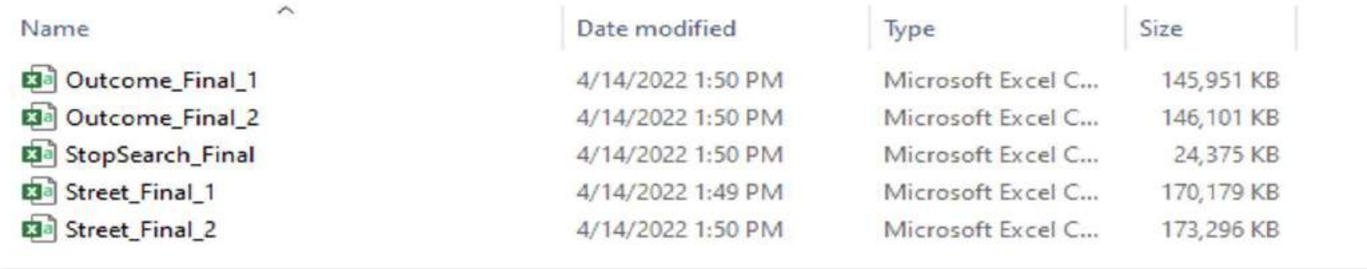


PC › Desktop › BDDS Files › ADMP › 1_Input

Name	Date modified	Type
2020-01	4/14/2022 1:42 PM	File folder
2020-02	4/14/2022 1:42 PM	File folder
2020-03	4/14/2022 1:42 PM	File folder
2020-04	4/14/2022 1:42 PM	File folder
2020-05	4/14/2022 1:42 PM	File folder
2020-06	4/14/2022 1:42 PM	File folder
2020-07	4/14/2022 1:42 PM	File folder
2020-08	4/14/2022 1:42 PM	File folder
2020-09	4/14/2022 1:42 PM	File folder
2020-10	4/14/2022 1:42 PM	File folder
2020-11	4/14/2022 1:42 PM	File folder
2020-12	4/14/2022 1:42 PM	File folder
2021-01	4/14/2022 1:42 PM	File folder
2021-02	4/14/2022 1:42 PM	File folder
2021-03	4/14/2022 1:42 PM	File folder
2021-04	4/14/2022 1:42 PM	File folder
2021-05	4/14/2022 1:42 PM	File folder
2021-06	4/14/2022 1:42 PM	File folder
2021-07	4/14/2022 1:42 PM	File folder
2021-08	4/14/2022 1:42 PM	File folder
2021-09	4/14/2022 1:42 PM	File folder
2021-10	4/14/2022 1:42 PM	File folder
2021-11	4/14/2022 1:42 PM	File folder
2021-12	4/14/2022 1:42 PM	File folder
2022-01	4/14/2022 1:42 PM	File folder
2022-02	4/14/2022 1:42 PM	File folder
Crime Lookup	4/14/2022 1:43 PM	File folder
UnemploymentData	4/14/2022 1:43 PM	File folder

1

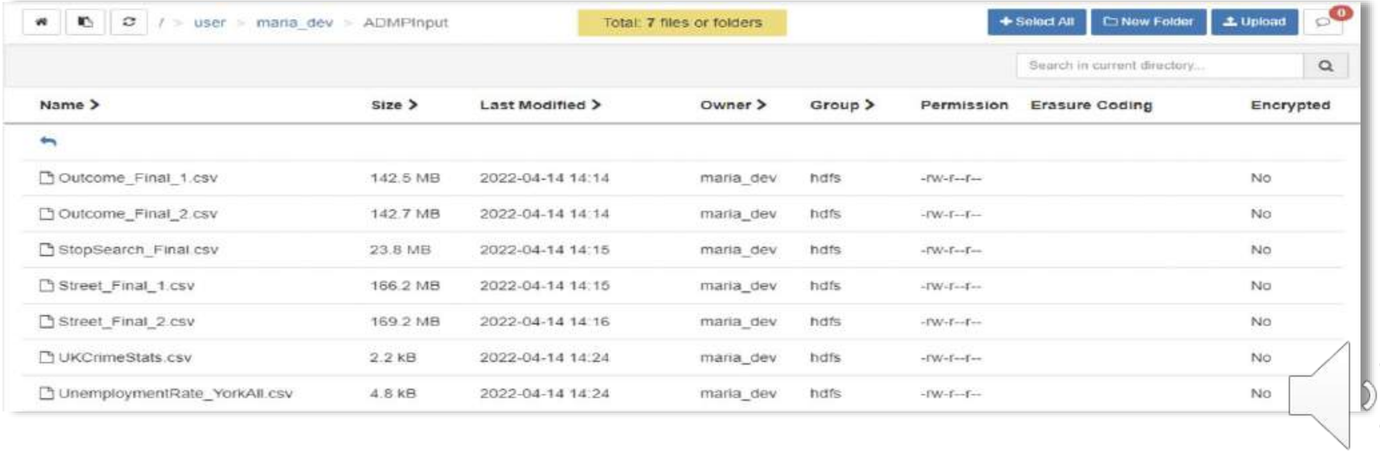
- Files with size less than **200MB** only are accepted in the HDFS accessible location - /user/maria_dev/ADMPInput
- In order to avoid the maximum file limit issue, the outcome and street-crime files are divided into two files each in **RStudio**



Name	Date modified	Type	Size
Outcome_Final_1	4/14/2022 1:50 PM	Microsoft Excel C...	145,951 KB
Outcome_Final_2	4/14/2022 1:50 PM	Microsoft Excel C...	146,101 KB
StopSearch_Final	4/14/2022 1:50 PM	Microsoft Excel C...	24,375 KB
Street_Final_1	4/14/2022 1:49 PM	Microsoft Excel C...	170,179 KB
Street_Final_2	4/14/2022 1:50 PM	Microsoft Excel C...	173,296 KB

2

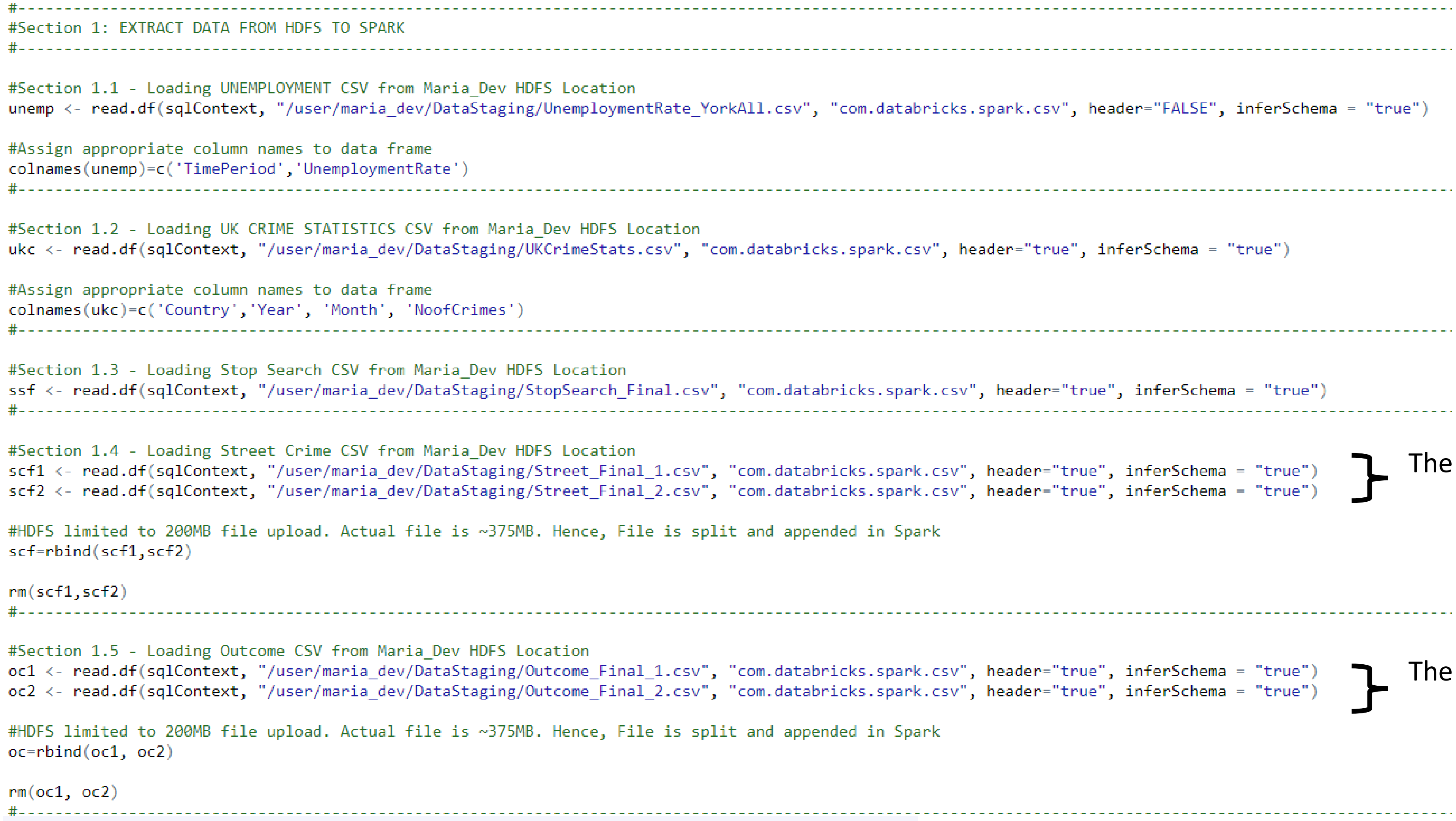
Uploading the files into HDFS location



/ > user > maria_dev > ADMPInput
Total: 7 files or folders
+ Select All
New Folder
Upload
Search in current directory...

Name >	Size >	Last Modified >	Owner >	Group >	Permission	Erasure Coding	Encrypted
Outcome_Final_1.csv	142.5 MB	2022-04-14 14:14	maria_dev	hdfs	-rw-r--r--		No
Outcome_Final_2.csv	142.7 MB	2022-04-14 14:14	maria_dev	hdfs	-rw-r--r--		No
StopSearch_Final.csv	23.8 MB	2022-04-14 14:15	maria_dev	hdfs	-rw-r--r--		No
Street_Final_1.csv	166.2 MB	2022-04-14 14:15	maria_dev	hdfs	-rw-r--r--		No
Street_Final_2.csv	169.2 MB	2022-04-14 14:16	maria_dev	hdfs	-rw-r--r--		No
UKCrimeStats.csv	2.2 kB	2022-04-14 14:24	maria_dev	hdfs	-rw-r--r--		No
UnemploymentRate_YorkAll.csv	4.8 kB	2022-04-14 14:24	maria_dev	hdfs	-rw-r--r--		No

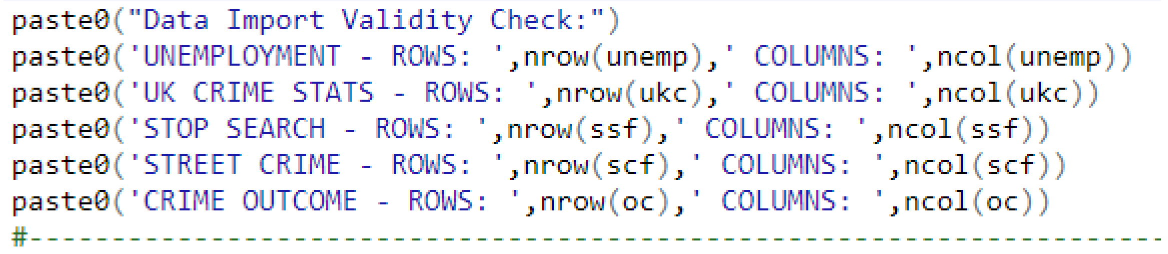
DATA STAGING – EXTRACTION PROCESS



```
#----------------------------------------------------------------------------------------------------------------------------------
#Section 1: EXTRACT DATA FROM HDFS TO SPARK
#----------------------------------------------------------------------------------------------------------------------------------

#Section 1.1 - Loading UNEMPLOYMENT CSV from Maria_Dev HDFS Location
unemp <- read.df(sqlContext, "/user/maria_dev/DataStaging/UnemploymentRate_YorkAll.csv", "com.databricks.spark.csv", header="FALSE", inferSchema = "true")

#Assign appropriate column names to data frame
colnames(unemp)=c('TimePeriod','UnemploymentRate')
#----------------------------------------------------------------------------------------------------------------------------------

#Section 1.2 - Loading UK CRIME STATISTICS CSV from Maria_Dev HDFS Location
ukc <- read.df(sqlContext, "/user/maria_dev/DataStaging/UKCrimeStats.csv", "com.databricks.spark.csv", header="true", inferSchema = "true")

#Assign appropriate column names to data frame
colnames(ukc)=c('Country','Year', 'Month', 'NoofCrimes')
#----------------------------------------------------------------------------------------------------------------------------------

#Section 1.3 - Loading Stop Search CSV from Maria_Dev HDFS Location
ssf <- read.df(sqlContext, "/user/maria_dev/DataStaging/StopSearch_Final.csv", "com.databricks.spark.csv", header="true", inferSchema = "true")
#----------------------------------------------------------------------------------------------------------------------------------

#Section 1.4 - Loading Street Crime CSV from Maria_Dev HDFS Location
scf1 <- read.df(sqlContext, "/user/maria_dev/DataStaging/Street_Final_1.csv", "com.databricks.spark.csv", header="true", inferSchema = "true")
scf2 <- read.df(sqlContext, "/user/maria_dev/DataStaging/Street_Final_2.csv", "com.databricks.spark.csv", header="true", inferSchema = "true")

#HDFS limited to 200MB file upload. Actual file is ~375MB. Hence, File is split and appended in Spark
scf=rbind(scf1,scf2)

rm(scf1,scf2)
#----------------------------------------------------------------------------------------------------------------------------------

#Section 1.5 - Loading Outcome CSV from Maria_Dev HDFS Location
oc1 <- read.df(sqlContext, "/user/maria_dev/DataStaging/Outcome_Final_1.csv", "com.databricks.spark.csv", header="true", inferSchema = "true")
oc2 <- read.df(sqlContext, "/user/maria_dev/DataStaging/Outcome_Final_2.csv", "com.databricks.spark.csv", header="true", inferSchema = "true")

#HDFS limited to 200MB file upload. Actual file is ~375MB. Hence, File is split and appended in Spark
oc=rbind(oc1, oc2)

rm(oc1, oc2)
#----------------------------------------------------------------------------------------------------------------------------------
```

} The split files are merged into a single data frame

} The split files are merged into a single data frame



```
paste0("Data Import Validity Check:")
paste0('UNEMPLOYMENT - ROWS: ',nrow(unemp),' COLUMNS: ',ncol(unemp))
paste0('UK CRIME STATS - ROWS: ',nrow(ukc),' COLUMNS: ',ncol(ukc))
paste0('STOP SEARCH - ROWS: ',nrow(ssf),' COLUMNS: ',ncol(ssf))
paste0('STREET CRIME - ROWS: ',nrow(scf),' COLUMNS: ',ncol(scf))
paste0('CRIME OUTCOME - ROWS: ',nrow(oc),' COLUMNS: ',ncol(oc))
#-------------------------------------------------------------------
```

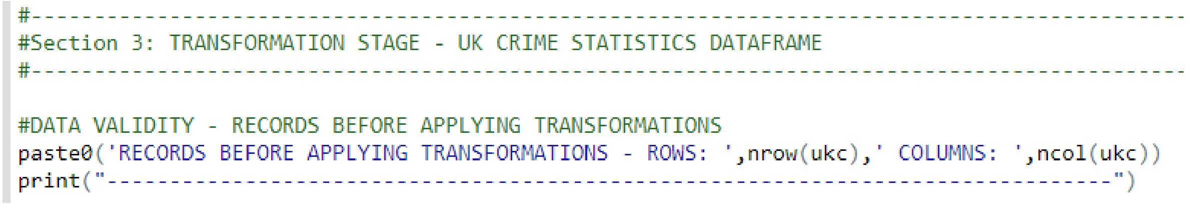
Output →

```
[1] "Data Import Validity Check:"
[1] "UNEMPLOYMENT - ROWS: 359 COLUMNS: 2"
[1] "UK CRIME STATS - ROWS: 78 COLUMNS: 4"
[1] "STOP SEARCH - ROWS: 96163 COLUMNS: 16"
[1] "STREET CRIME - ROWS: 1429456 COLUMNS: 12"
[1] "CRIME OUTCOME - ROWS: 1262104 COLUMNS: 10"
```

DATA STAGING – TRANSFORMATION STAGE

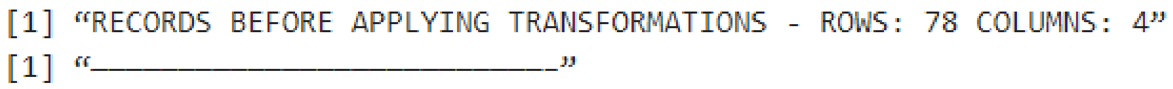
1. Data Validity - Rows and Columns Count before Transformation

Code ⍰



```
#-------------------------------------------------------------------------------------
#Section 3: TRANSFORMATION STAGE - UK CRIME STATISTICS DATAFRAME
#-------------------------------------------------------------------------------------

#DATA VALIDITY - RECORDS BEFORE APPLYING TRANSFORMATIONS
paste0('RECORDS BEFORE APPLYING TRANSFORMATIONS - ROWS: ',nrow(ukc),' COLUMNS: ',ncol(ukc))
print("-------------------------------------------------------------------------------")
```

Output ⍰



```
[1] “RECORDS BEFORE APPLYING TRANSFORMATIONS - ROWS: 78 COLUMNS: 4”
[1] “______________________________”
```

2. Data Transformation - New Columns Derivation

Note: No New columns needed in dataset, hence this stage is skipped

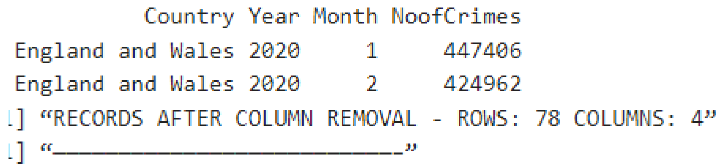
3. Data Transformation - Remove unwanted columns

Note: No unwanted columns in dataset, hence this stage is skipped

4. Data Validity - Rows and Columns count after columns removal

Note: Step-3 skipped, hence this stage is skipped, and Row/Column count of Step-1 holds good

Output ⍰



```
                  Country Year Month NoofCrimes
England and Wales 2020     1     447406
England and Wales 2020     2     424962
1] “RECORDS AFTER COLUMN REMOVAL - ROWS: 78 COLUMNS: 4”
1] “___________________________”
```

5. Data Transformation – Calculating aggregates based on group by variables

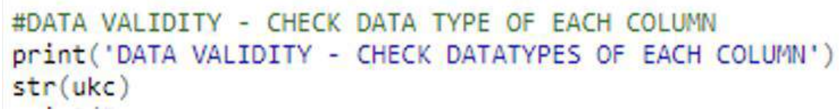
Note: No aggregation is needed in dataset, hence this stage is skipped

6. Data Validity - Rows and Columns count after aggregating by variables

Note: No aggregation happened in dataset, hence this stage is skipped

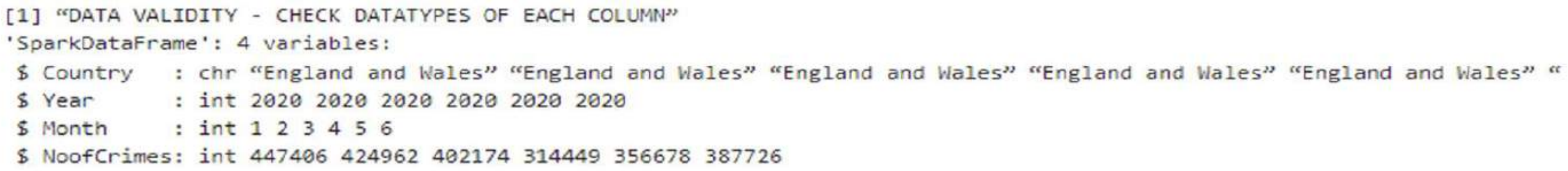
7. Data Validity - Data Type Check on each column

Code ⍰



```
#DATA VALIDITY - CHECK DATA TYPE OF EACH COLUMN
print('DATA VALIDITY - CHECK DATATYPES OF EACH COLUMN')
str(ukc)
```

Output ⍰

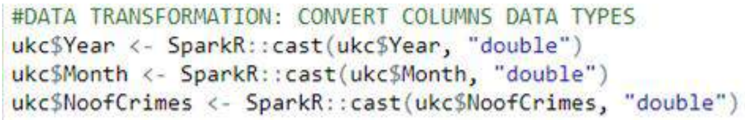


```
[1] “DATA VALIDITY - CHECK DATATYPES OF EACH COLUMN”
'SparkDataFrame': 4 variables:
 $ Country   : chr “England and Wales” “England and Wales” “England and Wales” “England and Wales” “England and Wales” “
 $ Year      : int 2020 2020 2020 2020 2020 2020
 $ Month     : int 1 2 3 4 5 6
 $ NoofCrimes: int 447406 424962 402174 314449 356678 387726
```

DATA STAGING – TRANSFORMATION STAGE

7.1 Data Transformation - Convert to appropriate column datatype (CAST)

Code ⍰

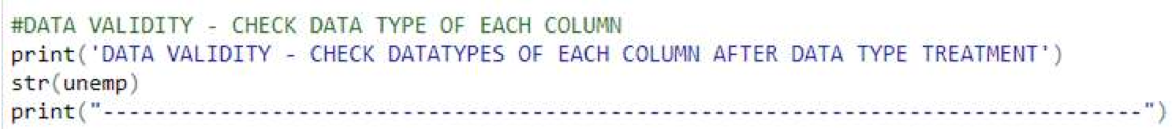


```
#DATA TRANSFORMATION: CONVERT COLUMNS DATA TYPES
ukc$Year <- SparkR::cast(ukc$Year, "double")
ukc$Month <- SparkR::cast(ukc$Month, "double")
ukc$NoofCrimes <- SparkR::cast(ukc$NoofCrimes, "double")
```

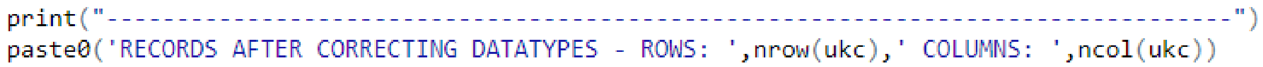
Note: All columns have correct data type but, this step is to capture and treat any variables have issues in future

8. Data Validity - Data Type Check on each column – After conversion

Code ⍰

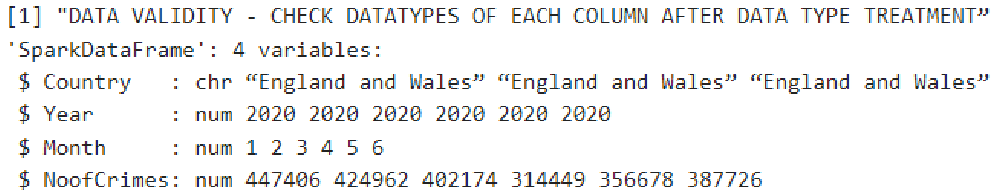


```
#DATA VALIDITY - CHECK DATA TYPE OF EACH COLUMN
print('DATA VALIDITY - CHECK DATATYPES OF EACH COLUMN AFTER DATA TYPE TREATMENT')
str(unemp)
print("--------------------------------------------------------------------------")
```

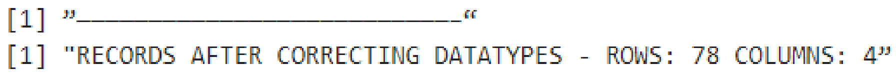


```
print("--------------------------------------------------------------------------")
paste0('RECORDS AFTER CORRECTING DATATYPES - ROWS: ',nrow(ukc),' COLUMNS: ',ncol(ukc))
```

Output ⍰



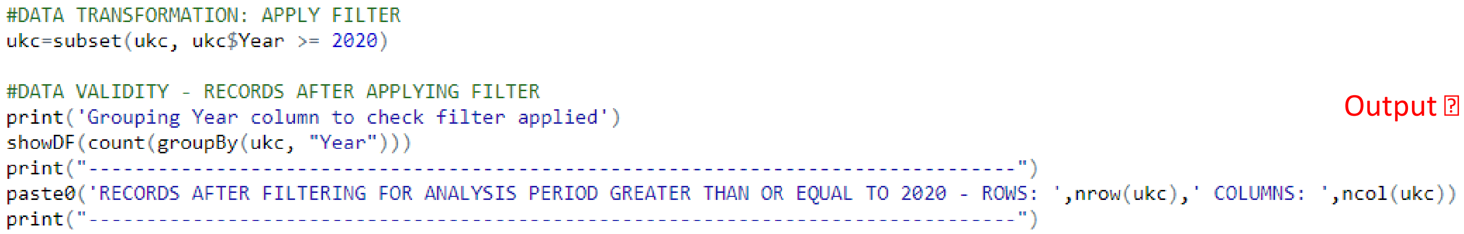
```
[1] "DATA VALIDITY - CHECK DATATYPES OF EACH COLUMN AFTER DATA TYPE TREATMENT”
'SparkDataFrame': 4 variables:
 $ Country   : chr “England and Wales” “England and Wales” “England and Wales”
 $ Year      : num 2020 2020 2020 2020 2020 2020
 $ Month     : num 1 2 3 4 5 6
 $ NoofCrimes: num 447406 424962 402174 314449 356678 387726
```



```
[1] ”_____________________________“
[1] "RECORDS AFTER CORRECTING DATATYPES - ROWS: 78 COLUMNS: 4”
```

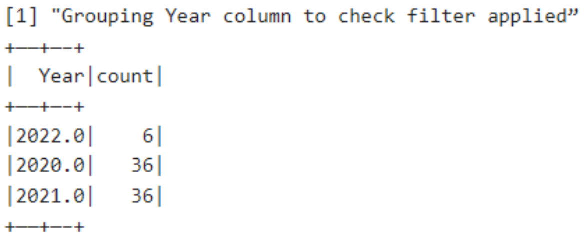
9. Data Transformation - Data filter >= 2020

Code ⍰



```
#DATA TRANSFORMATION: APPLY FILTER
ukc=subset(ukc, ukc$Year >= 2020)

#DATA VALIDITY - RECORDS AFTER APPLYING FILTER
print('Grouping Year column to check filter applied')
showDF(count(groupBy(ukc, "Year")))
print("--------------------------------------------------------------------------")
paste0('RECORDS AFTER FILTERING FOR ANALYSIS PERIOD GREATER THAN OR EQUAL TO 2020 - ROWS: ',nrow(ukc),' COLUMNS: ',ncol(ukc))
print("--------------------------------------------------------------------------")
```

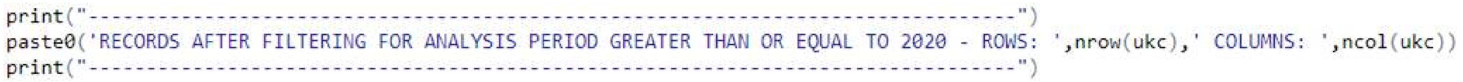
Output ⍰



```
[1] "Grouping Year column to check filter applied”
+——+——+
|  Year|count|
+——+——+
|2022.0|    6|
|2020.0|   36|
|2021.0|   36|
+——+——+
```

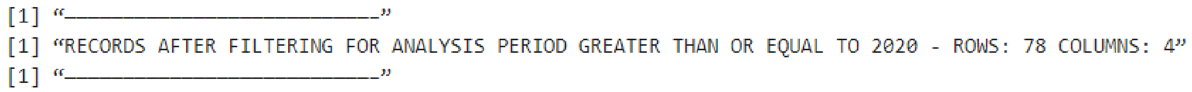
10. Data Validity - Rows and Columns Count after filter conditions applied

Code ⍰



```
print("--------------------------------------------------------------------------")
paste0('RECORDS AFTER FILTERING FOR ANALYSIS PERIOD GREATER THAN OR EQUAL TO 2020 - ROWS: ',nrow(ukc),' COLUMNS: ',ncol(ukc))
print("--------------------------------------------------------------------------")
```

Output ⍰



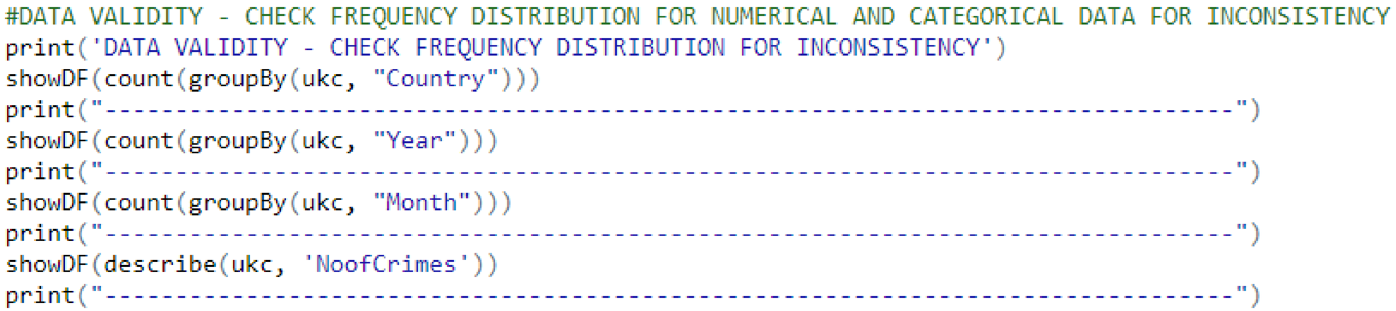
```
[1] “_________________________”
[1] “RECORDS AFTER FILTERING FOR ANALYSIS PERIOD GREATER THAN OR EQUAL TO 2020 - ROWS: 78 COLUMNS: 4”
[1] “_________________________”
```

DATA STAGING – TRANSFORMATION STAGE

UK Crime Statistics Data

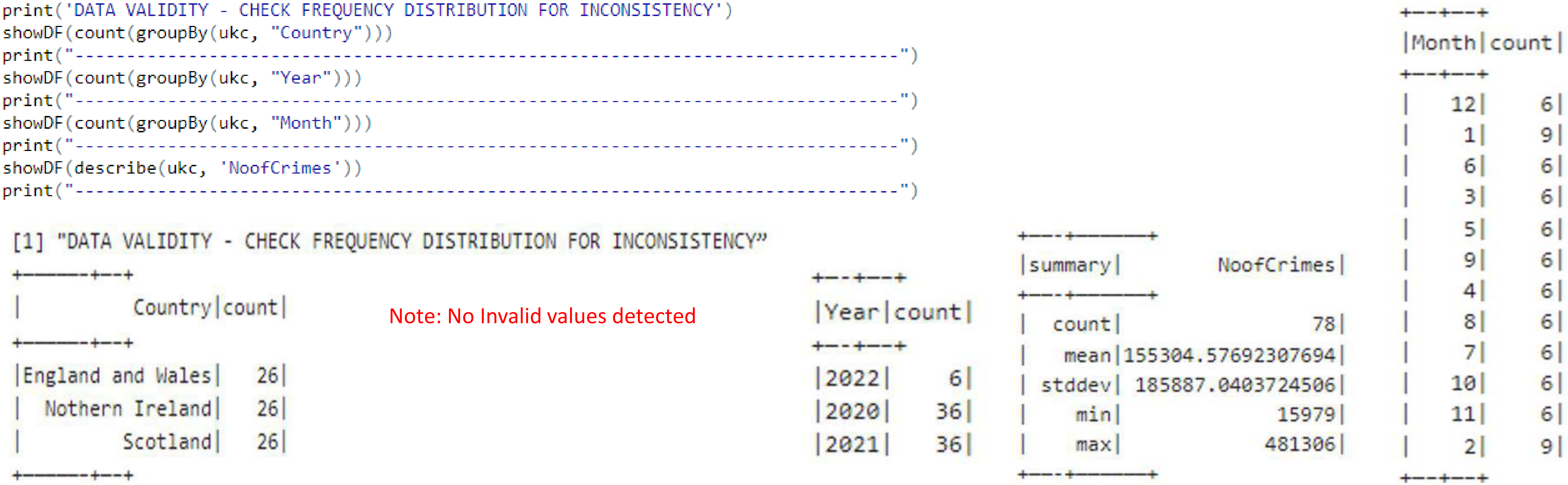
11. Data Validity - Frequency distribution on each columns to identify Nulls, Invalid values, Blanks

Code ⍰



```
#DATA VALIDITY - CHECK FREQUENCY DISTRIBUTION FOR NUMERICAL AND CATEGORICAL DATA FOR INCONSISTENCY
print('DATA VALIDITY - CHECK FREQUENCY DISTRIBUTION FOR INCONSISTENCY')
showDF(count(groupBy(ukc, "Country")))
print("-----------------------------------------------------------------------------")
showDF(count(groupBy(ukc, "Year")))
print("-----------------------------------------------------------------------------")
showDF(count(groupBy(ukc, "Month")))
print("-----------------------------------------------------------------------------")
showDF(describe(ukc, 'NoofCrimes'))
print("-----------------------------------------------------------------------------")
```

Output ⍰



```
[1] "DATA VALIDITY - CHECK FREQUENCY DISTRIBUTION FOR INCONSISTENCY"
+----------------+-----+
|         Country|count|
+----------------+-----+
|England and Wales|   26|
|  Nothern Ireland|   26|
|         Scotland|   26|
+----------------+-----+
```

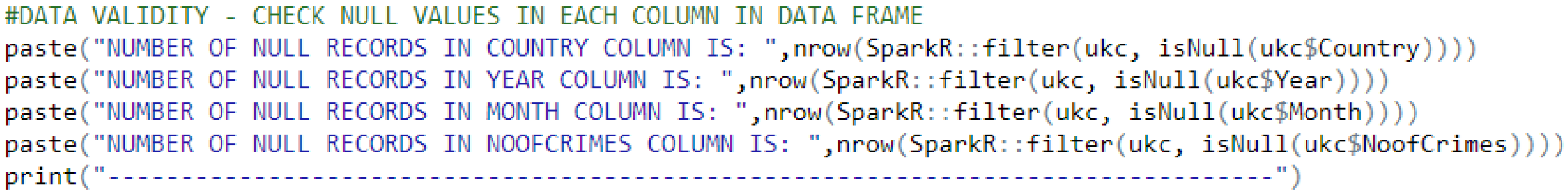
Note: No Invalid values detected

```
+----+-----+
|Year|count|
+----+-----+
|2022|    6|
|2020|   36|
|2021|   36|
+----+-----+
```

```
+-------+-----------------+
|summary|       NoofCrimes|
+-------+-----------------+
|  count|               78|
|   mean|155304.57692307694|
| stddev| 185887.0403724506|
|    min|            15979|
|    max|           481306|
+-------+-----------------+
```

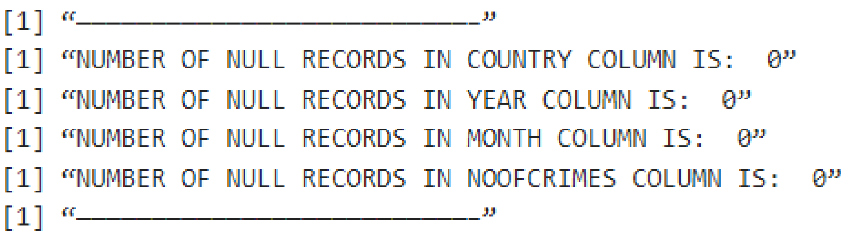
```
+-----+-----+
|Month|count|
+-----+-----+
|   12|    6|
|    1|    9|
|    6|    6|
|    3|    6|
|    5|    6|
|    9|    6|
|    4|    6|
|    8|    6|
|    7|    6|
|   10|    6|
|   11|    6|
|    2|    9|
+-----+-----+
```

Code ⍰



```
#DATA VALIDITY - CHECK NULL VALUES IN EACH COLUMN IN DATA FRAME
paste("NUMBER OF NULL RECORDS IN COUNTRY COLUMN IS: ",nrow(SparkR::filter(ukc, isNull(ukc$Country))))
paste("NUMBER OF NULL RECORDS IN YEAR COLUMN IS: ",nrow(SparkR::filter(ukc, isNull(ukc$Year))))
paste("NUMBER OF NULL RECORDS IN MONTH COLUMN IS: ",nrow(SparkR::filter(ukc, isNull(ukc$Month))))
paste("NUMBER OF NULL RECORDS IN NOOFCRIMES COLUMN IS: ",nrow(SparkR::filter(ukc, isNull(ukc$NoofCrimes))))
print("-----------------------------------------------------------------------------")
```

Output ⍰



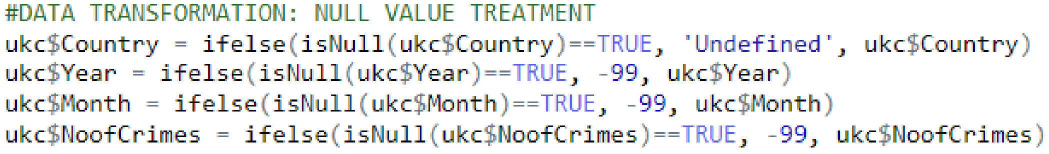
```
[1] "_____________________________"
[1] "NUMBER OF NULL RECORDS IN COUNTRY COLUMN IS:  0"
[1] "NUMBER OF NULL RECORDS IN YEAR COLUMN IS:  0"
[1] "NUMBER OF NULL RECORDS IN MONTH COLUMN IS:  0"
[1] "NUMBER OF NULL RECORDS IN NOOFCRIMES COLUMN IS:  0"
[1] "_____________________________"
```

DATA STAGING – TRANSFORMATION STAGE

11.1 Data Transformation - Treatment:

- Numeric Column = Null to -99 value change
- Character column = Null to “Undefined”

Code ⍰

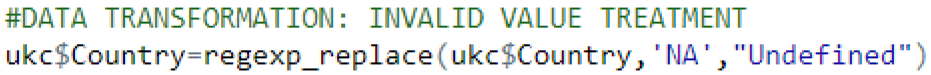


```
#DATA TRANSFORMATION: NULL VALUE TREATMENT
ukc$Country = ifelse(isNull(ukc$Country)==TRUE, 'Undefined', ukc$Country)
ukc$Year = ifelse(isNull(ukc$Year)==TRUE, -99, ukc$Year)
ukc$Month = ifelse(isNull(ukc$Month)==TRUE, -99, ukc$Month)
ukc$NoofCrimes = ifelse(isNull(ukc$NoofCrimes)==TRUE, -99, ukc$NoofCrimes)
```

11.2 Data Transformation - Treatment:

- Character column = ‘NA’ to “Undefined”
- Character column = Invalid values to “Undefined”

Code ⍰

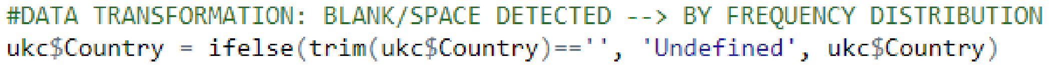


```
#DATA TRANSFORMATION: INVALID VALUE TREATMENT
ukc$Country=regexp_replace(ukc$Country,'NA',"Undefined")
```

11.3 Data Transformation - Treatment:

- Character column = Blank values to “Undefined”

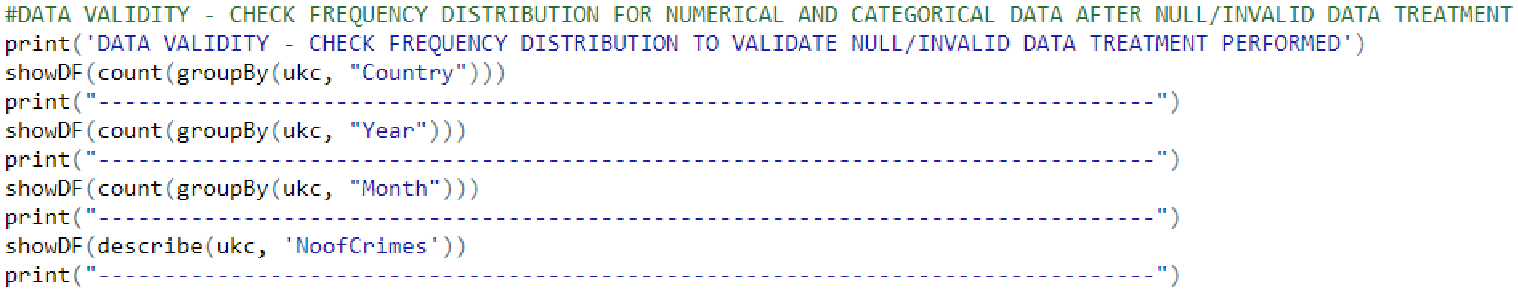
Code ⍰



```
#DATA TRANSFORMATION: BLANK/SPACE DETECTED --> BY FREQUENCY DISTRIBUTION
ukc$Country = ifelse(trim(ukc$Country)=='', 'Undefined', ukc$Country)
```

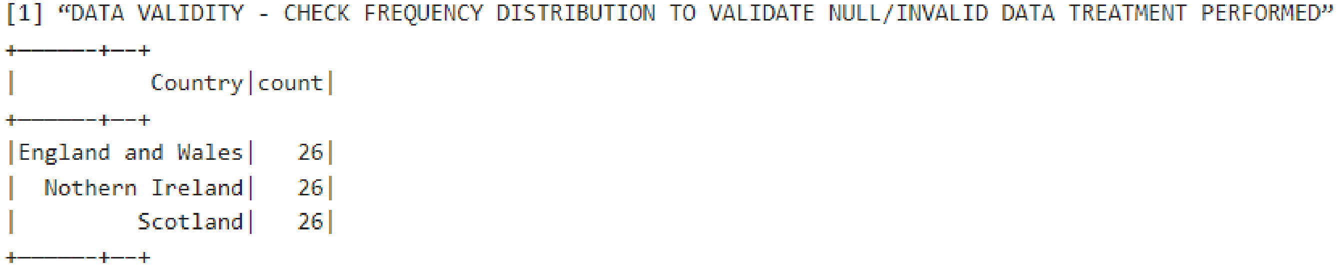
12. Data Validity - Frequency distribution on each columns to check columns value treatments applied

Code ⍰

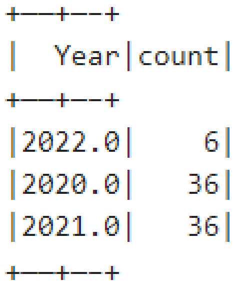


```
#DATA VALIDITY - CHECK FREQUENCY DISTRIBUTION FOR NUMERICAL AND CATEGORICAL DATA AFTER NULL/INVALID DATA TREATMENT
print('DATA VALIDITY - CHECK FREQUENCY DISTRIBUTION TO VALIDATE NULL/INVALID DATA TREATMENT PERFORMED')
showDF(count(groupBy(ukc, "Country")))
print("---------------------------------------------------------------------------")
showDF(count(groupBy(ukc, "Year")))
print("---------------------------------------------------------------------------")
showDF(count(groupBy(ukc, "Month")))
print("---------------------------------------------------------------------------")
showDF(describe(ukc, 'NoofCrimes'))
print("---------------------------------------------------------------------------")
```

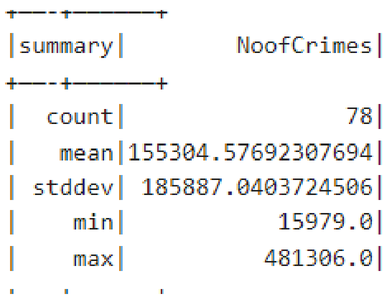
Output ⍰



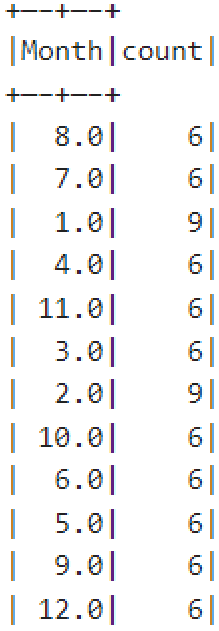
```
[1] “DATA VALIDITY - CHECK FREQUENCY DISTRIBUTION TO VALIDATE NULL/INVALID DATA TREATMENT PERFORMED”
+--------+---+
|         Country|count|
+--------+---+
|England and Wales|   26|
|  Nothern Ireland|   26|
|         Scotland|   26|
+--------+---+
```



```
+---+---+
|  Year|count|
+---+---+
|2022.0|    6|
|2020.0|   36|
|2021.0|   36|
+---+---+
```



```
+----+--------+
|summary|       NoofCrimes|
+----+--------+
|  count|                78|
|   mean|155304.57692307694|
| stddev| 185887.0403724506|
|    min|           15979.0|
|    max|          481306.0|
```



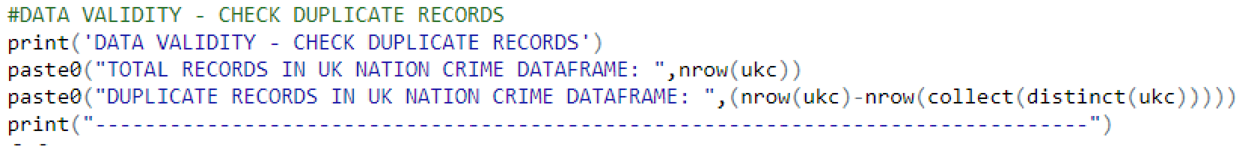
```
+---+---+
|Month|count|
+---+---+
|  8.0|    6|
|  7.0|    6|
|  1.0|    9|
|  4.0|    6|
| 11.0|    6|
|  3.0|    6|
|  2.0|    9|
| 10.0|    6|
|  6.0|    6|
|  5.0|    6|
|  9.0|    6|
| 12.0|    6|
```

DATA STAGING – TRANSFORMATION STAGE

UK Crime Statistics Data

13. Data Validity - Duplicate Records - Check

Code ⍰



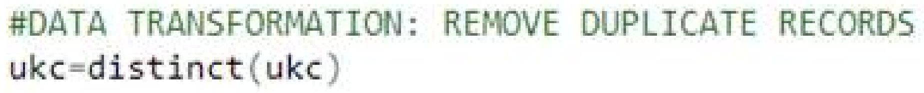
```
#DATA VALIDITY - CHECK DUPLICATE RECORDS
print('DATA VALIDITY - CHECK DUPLICATE RECORDS')
paste0("TOTAL RECORDS IN UK NATION CRIME DATAFRAME: ",nrow(ukc))
paste0("DUPLICATE RECORDS IN UK NATION CRIME DATAFRAME: ",(nrow(ukc)-nrow(collect(distinct(ukc)))))
print("------------------------------------------------------------------------------")
- -
```

Output ⍰

```
[1] “DATA VALIDITY - CHECK DUPLICATE RECORDS”
[1] “TOTAL RECORDS IN UK NATION CRIME DATAFRAME: 78”
[1] “DUPLICATE RECORDS IN UK NATION CRIME DATAFRAME: 0”
[1] “___________________________”
```

13.1 Data Transformation - Remove Duplicate Records

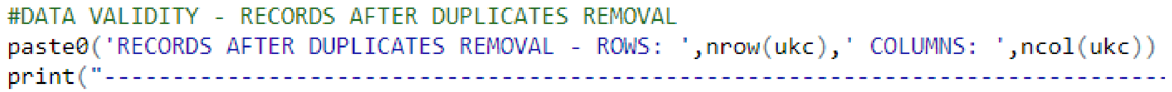
Code ⍰



```
#DATA TRANSFORMATION: REMOVE DUPLICATE RECORDS
ukc=distinct(ukc)
```

14. Data Validity - Rows and Columns Count After dropping duplicates

Code ⍰



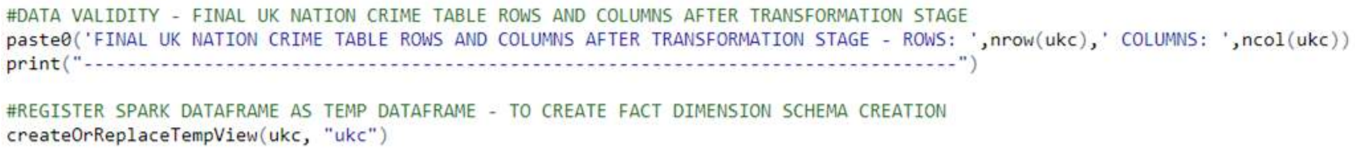
```
#DATA VALIDITY - RECORDS AFTER DUPLICATES REMOVAL
paste0('RECORDS AFTER DUPLICATES REMOVAL - ROWS: ',nrow(ukc),' COLUMNS: ',ncol(ukc))
print("----------------------------------------------------------------------------
```

Output ⍰

```
“_____________________________”

“RECORDS AFTER DUPLICATES REMOVAL - ROWS: 78 COLUMNS: 4”
“_____________________________”
```

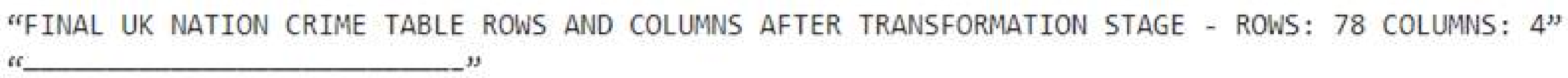
15. Data Validity - Final Check on Rows and Columns before creating Fact and Dimension tables and Loading Stage (Hive)

Code ⍰



```
#DATA VALIDITY - FINAL UK NATION CRIME TABLE ROWS AND COLUMNS AFTER TRANSFORMATION STAGE
paste0('FINAL UK NATION CRIME TABLE ROWS AND COLUMNS AFTER TRANSFORMATION STAGE - ROWS: ',nrow(ukc),' COLUMNS: ',ncol(ukc))
print("----------------------------------------------------------------------------")

#REGISTER SPARK DATAFRAME AS TEMP DATAFRAME - TO CREATE FACT DIMENSION SCHEMA CREATION
createOrReplaceTempView(ukc, "ukc")
```

Output ⍰



```
“FINAL UK NATION CRIME TABLE ROWS AND COLUMNS AFTER TRANSFORMATION STAGE - ROWS: 78 COLUMNS: 4”
“___________________________________”
```

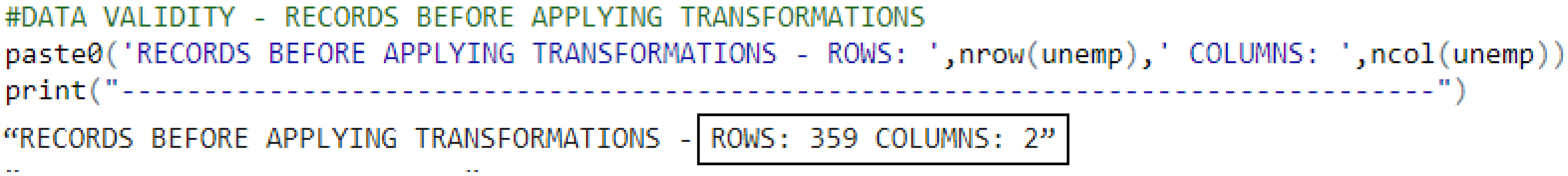
DATA STAGING – TRANSFORMATION STAGE

UNEMPLOYMENT DATA

1. Data Validity - Rows and Columns Count before Transformation

Code →

Output →

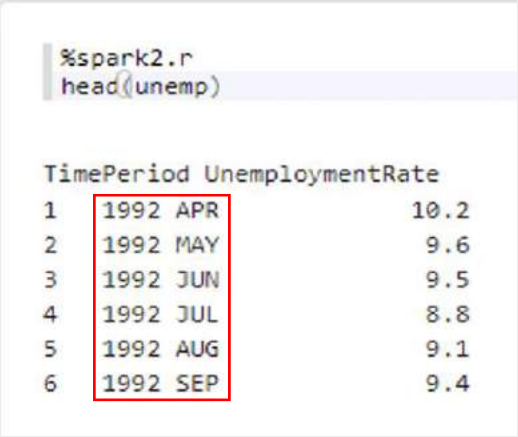


```
#DATA VALIDITY - RECORDS BEFORE APPLYING TRANSFORMATIONS
paste0('RECORDS BEFORE APPLYING TRANSFORMATIONS - ROWS: ',nrow(unemp),' COLUMNS: ',ncol(unemp))
print("----------------------------------------------------------------------------------")
```

```
“RECORDS BEFORE APPLYING TRANSFORMATIONS - ROWS: 359 COLUMNS: 2”
```

2. Data Transformation - New Columns Derivation

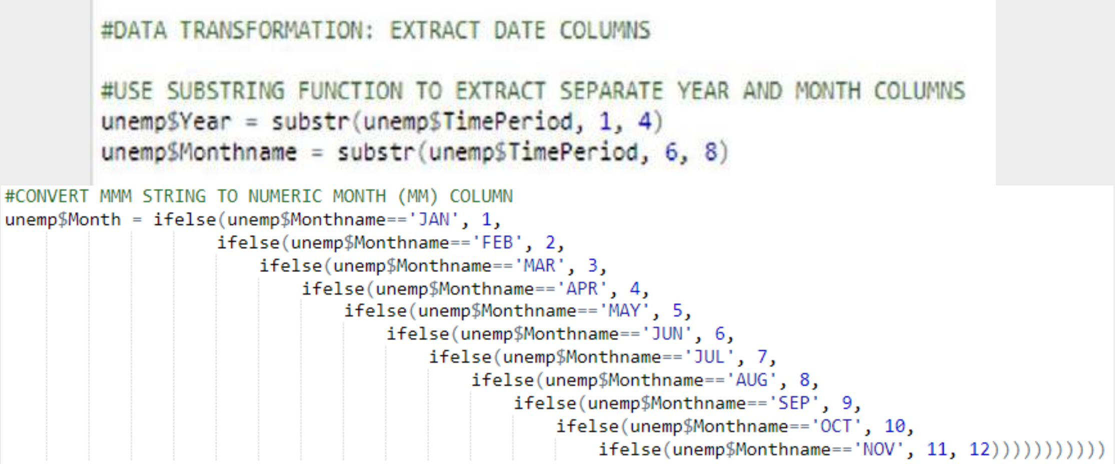
BEFORE



```
%spark2.r
head(unemp)
```

```
TimePeriod UnemploymentRate
1   1992 APR          10.2
2   1992 MAY           9.6
3   1992 JUN           9.5
4   1992 JUL           8.8
5   1992 AUG           9.1
6   1992 SEP           9.4
```

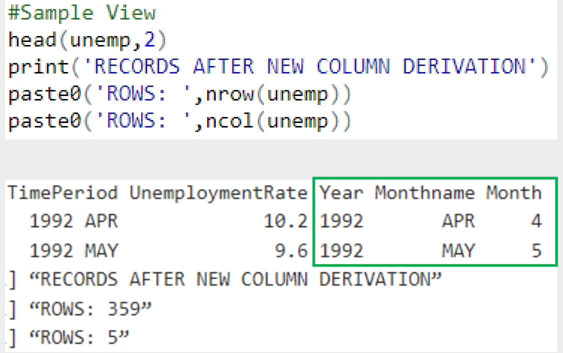
CODE



```
#DATA TRANSFORMATION: EXTRACT DATE COLUMNS

#USE SUBSTRING FUNCTION TO EXTRACT SEPARATE YEAR AND MONTH COLUMNS
unemp$Year = substr(unemp$TimePeriod, 1, 4)
unemp$Monthname = substr(unemp$TimePeriod, 6, 8)
```

```
#CONVERT MMM STRING TO NUMERIC MONTH (MM) COLUMN
unemp$Month = ifelse(unemp$Monthname=='JAN', 1,
                     ifelse(unemp$Monthname=='FEB', 2,
                        ifelse(unemp$Monthname=='MAR', 3,
                           ifelse(unemp$Monthname=='APR', 4,
                              ifelse(unemp$Monthname=='MAY', 5,
                                 ifelse(unemp$Monthname=='JUN', 6,
                                    ifelse(unemp$Monthname=='JUL', 7,
                                       ifelse(unemp$Monthname=='AUG', 8,
                                          ifelse(unemp$Monthname=='SEP', 9,
                                             ifelse(unemp$Monthname=='OCT', 10,
                                                ifelse(unemp$Monthname=='NOV', 11, 12)))))))))))
```

AFTER

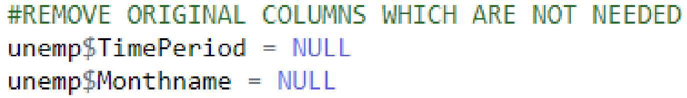


```
#Sample View
head(unemp,2)
print('RECORDS AFTER NEW COLUMN DERIVATION')
paste0('ROWS: ',nrow(unemp))
paste0('ROWS: ',ncol(unemp))
```

```
TimePeriod UnemploymentRate Year Monthname Month
  1992 APR             10.2 1992       APR     4
  1992 MAY              9.6 1992       MAY     5
] “RECORDS AFTER NEW COLUMN DERIVATION”
] “ROWS: 359”
] “ROWS: 5”
```

3. Data Transformation - Remove unwanted columns

Code →

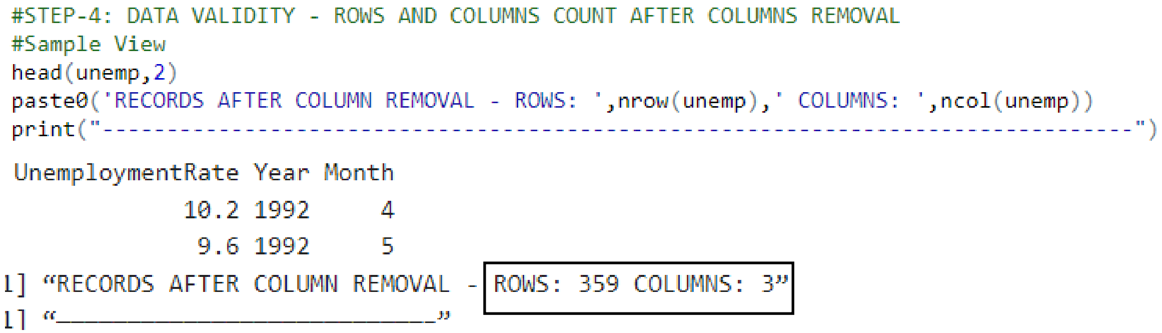


```
#REMOVE ORIGINAL COLUMNS WHICH ARE NOT NEEDED
unemp$TimePeriod = NULL
unemp$Monthname = NULL
```

4. Data Validity - Rows and Columns count after columns removal

Code →

Output →



```
#STEP-4: DATA VALIDITY - ROWS AND COLUMNS COUNT AFTER COLUMNS REMOVAL
#Sample View
head(unemp,2)
paste0('RECORDS AFTER COLUMN REMOVAL - ROWS: ',nrow(unemp),' COLUMNS: ',ncol(unemp))
print("----------------------------------------------------------------------------------")
```

```
UnemploymentRate Year Month
            10.2 1992     4
             9.6 1992     5
1] “RECORDS AFTER COLUMN REMOVAL - ROWS: 359 COLUMNS: 3”
1] “______________________________”
```

DATA STAGING – TRANSFORMATION STAGE

5. Data Transformation – Calculating aggregates based on group by variables

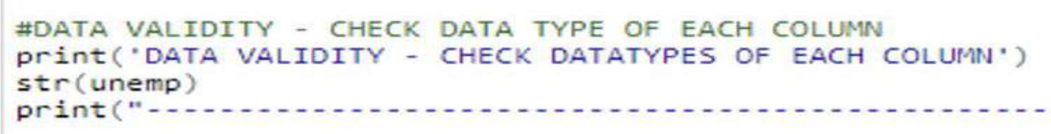
Note: No aggregation is needed in dataset, hence this stage is skipped

6. Data Validity - Rows and Columns count after aggregating by variables

Note: No Action needed since No aggregation is performed, hence this stage is skipped

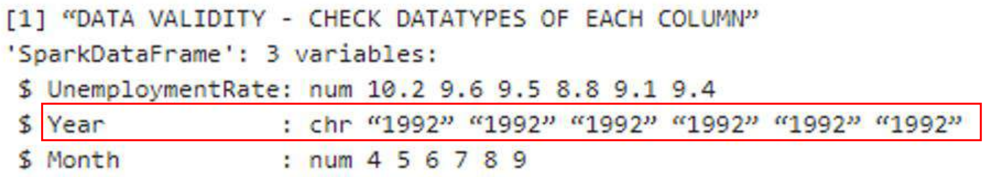
7. Data Validity - Data Type Check on each column - Before

Code →



```
#DATA VALIDITY - CHECK DATA TYPE OF EACH COLUMN
print('DATA VALIDITY - CHECK DATATYPES OF EACH COLUMN')
str(unemp)
print("-----------------------------------------------
```

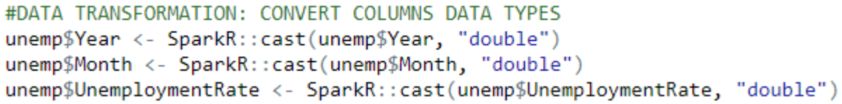
Output →



```
[1] “DATA VALIDITY - CHECK DATATYPES OF EACH COLUMN”
'SparkDataFrame': 3 variables:
 $ UnemploymentRate: num 10.2 9.6 9.5 8.8 9.1 9.4
 $ Year            : chr “1992” “1992” “1992” “1992” “1992” “1992”
 $ Month           : num 4 5 6 7 8 9
```

7.1 Data Transformation - Convert to appropriate column datatype (CAST)

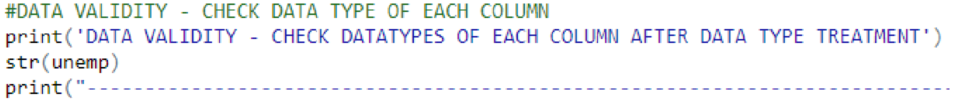
Code →



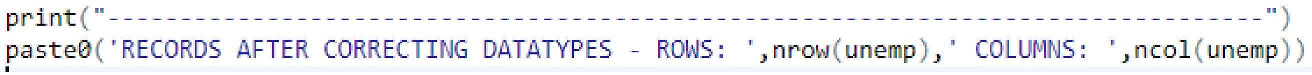
```
#DATA TRANSFORMATION: CONVERT COLUMNS DATA TYPES
unemp$Year <- SparkR::cast(unemp$Year, "double")
unemp$Month <- SparkR::cast(unemp$Month, "double")
unemp$UnemploymentRate <- SparkR::cast(unemp$UnemploymentRate, "double")
```

8. Data Validity - Data Type Check on each column – After conversion

Code →



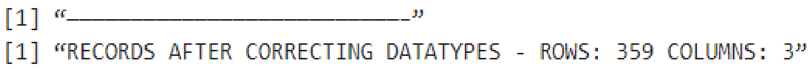
```
#DATA VALIDITY - CHECK DATA TYPE OF EACH COLUMN
print('DATA VALIDITY - CHECK DATATYPES OF EACH COLUMN AFTER DATA TYPE TREATMENT')
str(unemp)
print("----------------------------------------------------------------------
```



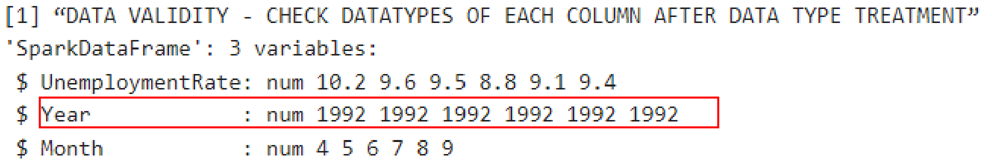
```
print("--------------------------------------------------------------------------------")
paste0('RECORDS AFTER CORRECTING DATATYPES - ROWS: ',nrow(unemp),' COLUMNS: ',ncol(unemp))
```

Few columns have correct data type, Step-7.1 will handle if any future data is having data type issue

Output →



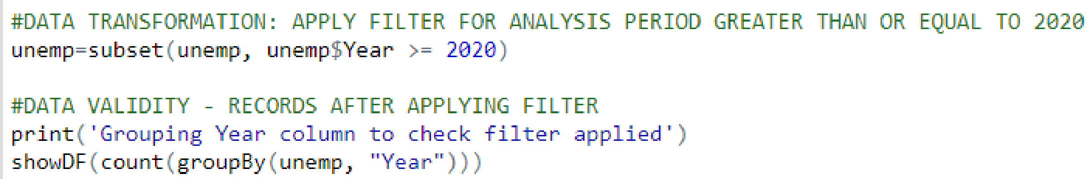
```
[1] “___________________________”
[1] “RECORDS AFTER CORRECTING DATATYPES - ROWS: 359 COLUMNS: 3”
```



```
[1] “DATA VALIDITY - CHECK DATATYPES OF EACH COLUMN AFTER DATA TYPE TREATMENT”
'SparkDataFrame': 3 variables:
 $ UnemploymentRate: num 10.2 9.6 9.5 8.8 9.1 9.4
 $ Year            : num 1992 1992 1992 1992 1992 1992
 $ Month           : num 4 5 6 7 8 9
```

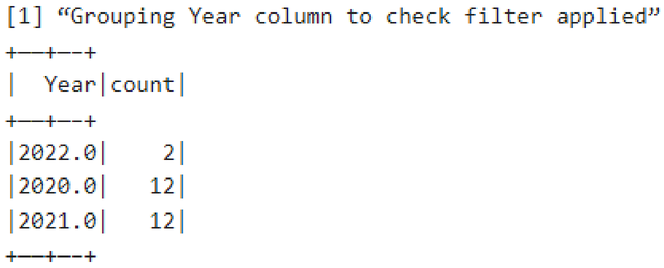
9. Data Transformation - Data filter >= 2020

Code →



```
#DATA TRANSFORMATION: APPLY FILTER FOR ANALYSIS PERIOD GREATER THAN OR EQUAL TO 2020
unemp=subset(unemp, unemp$Year >= 2020)

#DATA VALIDITY - RECORDS AFTER APPLYING FILTER
print('Grouping Year column to check filter applied')
showDF(count(groupBy(unemp, "Year")))
```

Output →



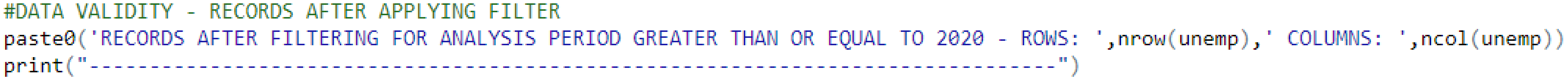
```
[1] “Grouping Year column to check filter applied”
+---+---+
|  Year|count|
+---+---+
|2022.0|    2|
|2020.0|   12|
|2021.0|   12|
+---+---+
```

DATA STAGING – TRANSFORMATION STAGE

UNEMPLOYMENT DATA

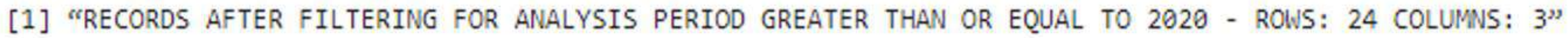
10. Data Validity - Rows and Columns Count after filter conditions applied

Code →



```
#DATA VALIDITY - RECORDS AFTER APPLYING FILTER
paste0('RECORDS AFTER FILTERING FOR ANALYSIS PERIOD GREATER THAN OR EQUAL TO 2020 - ROWS: ',nrow(unemp),' COLUMNS: ',ncol(unemp))
print("------------------------------------------------------------------------")
```

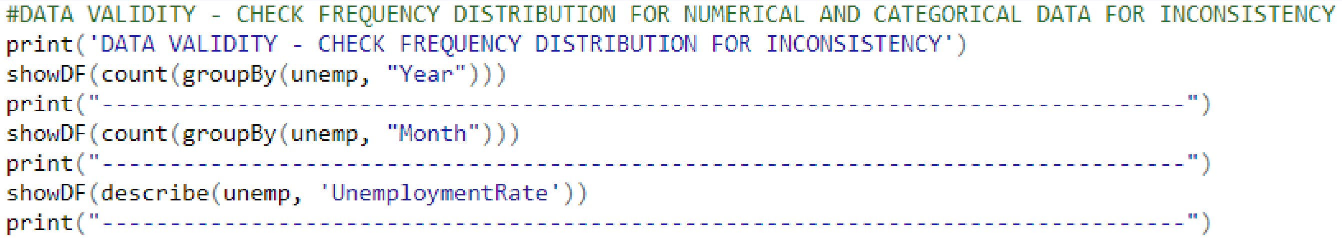
Output →



```
[1] “RECORDS AFTER FILTERING FOR ANALYSIS PERIOD GREATER THAN OR EQUAL TO 2020 - ROWS: 24 COLUMNS: 3”
```

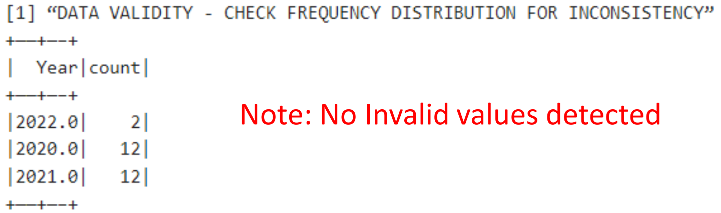
11. Data Validity - Frequency distribution on each columns to identify Nulls, Invalid values, Blanks

Code →



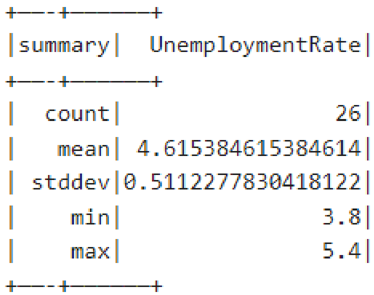
```
#DATA VALIDITY - CHECK FREQUENCY DISTRIBUTION FOR NUMERICAL AND CATEGORICAL DATA FOR INCONSISTENCY
print('DATA VALIDITY - CHECK FREQUENCY DISTRIBUTION FOR INCONSISTENCY')
showDF(count(groupBy(unemp, "Year")))
print("------------------------------------------------------------------------")
showDF(count(groupBy(unemp, "Month")))
print("------------------------------------------------------------------------")
showDF(describe(unemp, 'UnemploymentRate'))
print("------------------------------------------------------------------------")
```

Output →

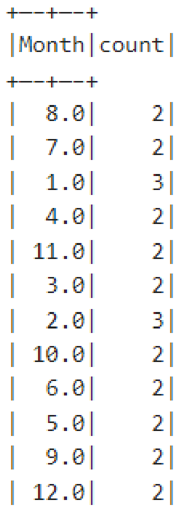


```
[1] “DATA VALIDITY - CHECK FREQUENCY DISTRIBUTION FOR INCONSISTENCY”
+----+-----+
|  Year|count|
+----+-----+
|2022.0|    2|
|2020.0|   12|
|2021.0|   12|
+----+-----+
```

Note: No Invalid values detected

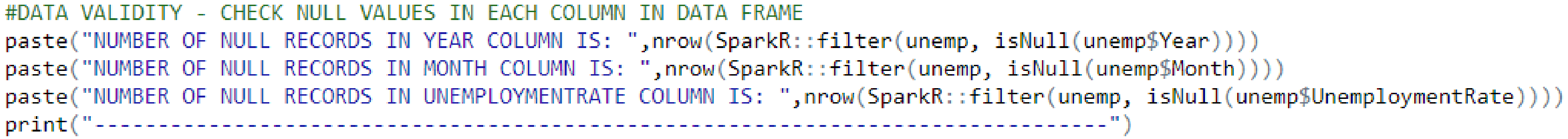


```
+-------+----------------+
|summary|  UnemploymentRate|
+-------+----------------+
|  count|                26|
|   mean| 4.615384615384614|
| stddev|0.5112277830418122|
|    min|               3.8|
|    max|               5.4|
+-------+----------------+
```



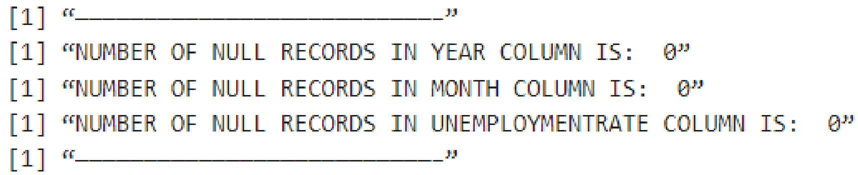
```
+-----+-----+
|Month|count|
+-----+-----+
|  8.0|    2|
|  7.0|    2|
|  1.0|    3|
|  4.0|    2|
| 11.0|    2|
|  3.0|    2|
|  2.0|    3|
| 10.0|    2|
|  6.0|    2|
|  5.0|    2|
|  9.0|    2|
| 12.0|    2|
```

Code →



```
#DATA VALIDITY - CHECK NULL VALUES IN EACH COLUMN IN DATA FRAME
paste("NUMBER OF NULL RECORDS IN YEAR COLUMN IS: ",nrow(SparkR::filter(unemp, isNull(unemp$Year))))
paste("NUMBER OF NULL RECORDS IN MONTH COLUMN IS: ",nrow(SparkR::filter(unemp, isNull(unemp$Month))))
paste("NUMBER OF NULL RECORDS IN UNEMPLOYMENTRATE COLUMN IS: ",nrow(SparkR::filter(unemp, isNull(unemp$UnemploymentRate))))
print("------------------------------------------------------------------------")
```

Output →



```
[1] “___________________________”
[1] “NUMBER OF NULL RECORDS IN YEAR COLUMN IS:  0”
[1] “NUMBER OF NULL RECORDS IN MONTH COLUMN IS:  0”
[1] “NUMBER OF NULL RECORDS IN UNEMPLOYMENTRATE COLUMN IS:  0”
[1] “___________________________”
```

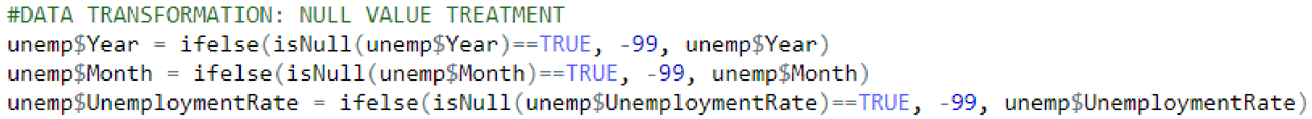
DATA STAGING – TRANSFORMATION STAGE

UNEMPLOYMENT DATA

11.1 Data Transformation - Treatment:

- Numeric Column = Null to -99 value change
- Character column = Null to “Undefined”

Code →



```
#DATA TRANSFORMATION: NULL VALUE TREATMENT
unemp$Year = ifelse(isNull(unemp$Year)==TRUE, -99, unemp$Year)
unemp$Month = ifelse(isNull(unemp$Month)==TRUE, -99, unemp$Month)
unemp$UnemploymentRate = ifelse(isNull(unemp$UnemploymentRate)==TRUE, -99, unemp$UnemploymentRate)
```

11.2 Data Transformation - Treatment:

- Character column = ‘NA’ to “Undefined”
- Character column = Invalid values to “Undefined”

Note: No Character column in dataset, hence this stage is skipped

11.3 Data Transformation - Treatment:

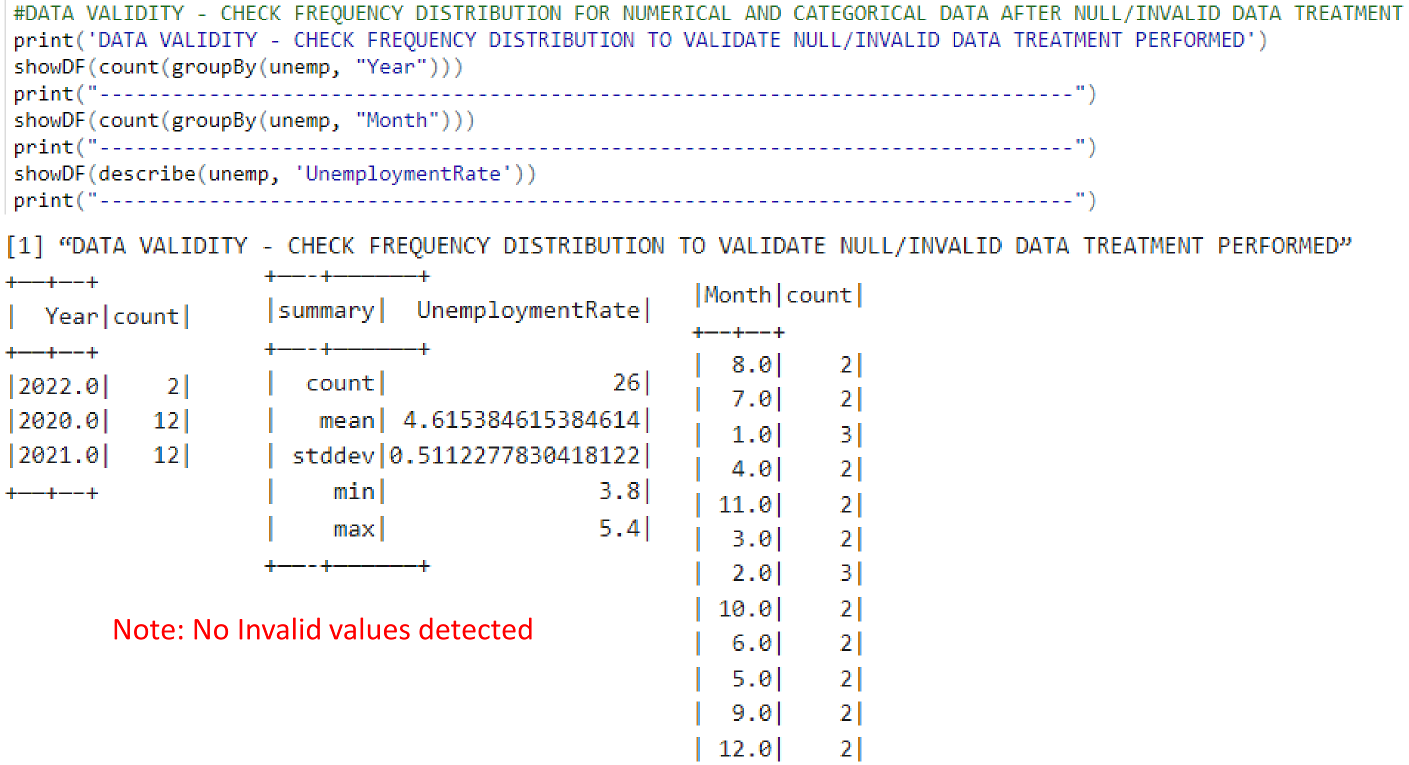
- Character column = Blank values to “Undefined”

Note: No Character column in dataset, hence this stage is skipped

12. Data Validity - Frequency distribution on each columns to check columns value treatments applied

Code →

Output →



```
#DATA VALIDITY - CHECK FREQUENCY DISTRIBUTION FOR NUMERICAL AND CATEGORICAL DATA AFTER NULL/INVALID DATA TREATMENT
print('DATA VALIDITY - CHECK FREQUENCY DISTRIBUTION TO VALIDATE NULL/INVALID DATA TREATMENT PERFORMED')
showDF(count(groupBy(unemp, "Year")))
print("--------------------------------------------------------------------------------")
showDF(count(groupBy(unemp, "Month")))
print("--------------------------------------------------------------------------------")
showDF(describe(unemp, 'UnemploymentRate'))
print("--------------------------------------------------------------------------------")
```

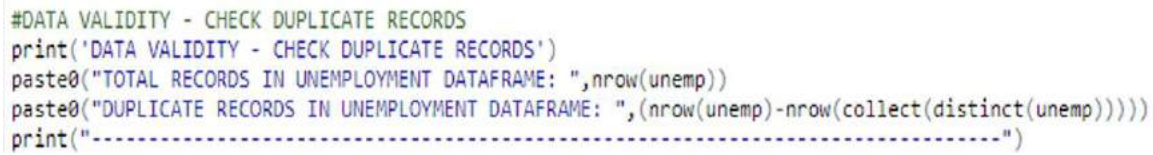
```
[1] “DATA VALIDITY - CHECK FREQUENCY DISTRIBUTION TO VALIDATE NULL/INVALID DATA TREATMENT PERFORMED”
+---+---+
|  Year|count|
+---+---+
|2022.0|    2|
|2020.0|   12|
|2021.0|   12|
+---+---+

+----+--------+
|summary|  UnemploymentRate|
+----+--------+
|  count|                26|
|   mean| 4.615384615384614|
| stddev|0.5112277830418122|
|    min|               3.8|
|    max|               5.4|
+----+--------+

|Month|count|
+---+---+
|  8.0|    2|
|  7.0|    2|
|  1.0|    3|
|  4.0|    2|
| 11.0|    2|
|  3.0|    2|
|  2.0|    3|
| 10.0|    2|
|  6.0|    2|
|  5.0|    2|
|  9.0|    2|
| 12.0|    2|
```

Note: No Invalid values detected

DATA STAGING – TRANSFORMATION STAGE

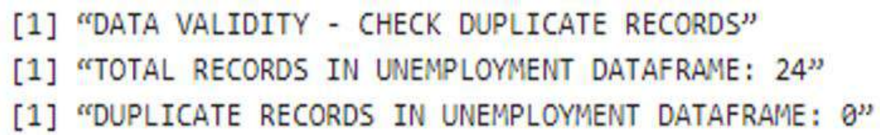
13. Data Validity - Duplicate Records - Check

Code →



```
#DATA VALIDITY - CHECK DUPLICATE RECORDS
print('DATA VALIDITY - CHECK DUPLICATE RECORDS')
paste0("TOTAL RECORDS IN UNEMPLOYMENT DATAFRAME: ",nrow(unemp))
paste0("DUPLICATE RECORDS IN UNEMPLOYMENT DATAFRAME: ",(nrow(unemp)-nrow(collect(distinct(unemp)))))
print("-------------------------------------------------------------------------------")
```

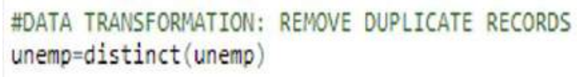
Output →



```
[1] “DATA VALIDITY - CHECK DUPLICATE RECORDS”
[1] “TOTAL RECORDS IN UNEMPLOYMENT DATAFRAME: 24”
[1] “DUPLICATE RECORDS IN UNEMPLOYMENT DATAFRAME: 0”
```

13.1 Data Transformation - Remove Duplicate Records

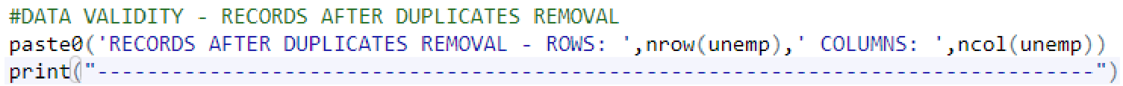
Code →



```
#DATA TRANSFORMATION: REMOVE DUPLICATE RECORDS
unemp=distinct(unemp)
```

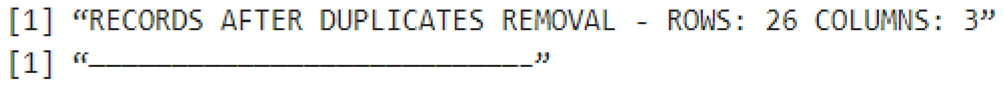
14. Data Validity - Rows and Columns Count After dropping duplicates

Code →



```
#DATA VALIDITY - RECORDS AFTER DUPLICATES REMOVAL
paste0('RECORDS AFTER DUPLICATES REMOVAL - ROWS: ',nrow(unemp),' COLUMNS: ',ncol(unemp))
print("--------------------------------------------------------------------------------")
```

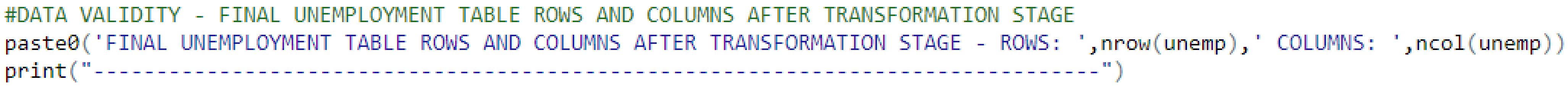
Output →



```
[1] “RECORDS AFTER DUPLICATES REMOVAL - ROWS: 26 COLUMNS: 3”
[1] “______________________________”
```

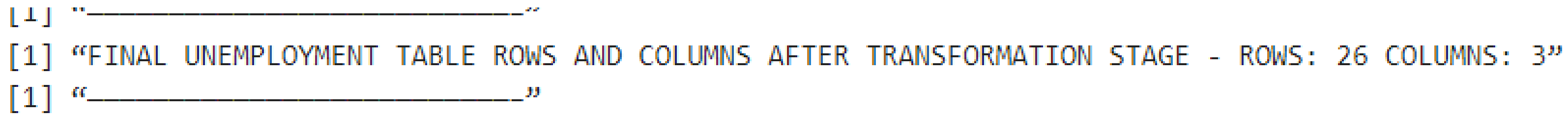
15. Data Validity - Final Check on Rows and Columns before creating Fact and Dimension tables and Loading Stage (Hive)

Code →



```
#DATA VALIDITY - FINAL UNEMPLOYMENT TABLE ROWS AND COLUMNS AFTER TRANSFORMATION STAGE
paste0('FINAL UNEMPLOYMENT TABLE ROWS AND COLUMNS AFTER TRANSFORMATION STAGE - ROWS: ',nrow(unemp),' COLUMNS: ',ncol(unemp))
print("------------------------------------------------------------------------")
```

Output →



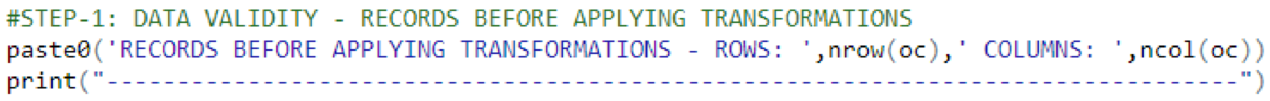
```
[1] "______________________________"
[1] “FINAL UNEMPLOYMENT TABLE ROWS AND COLUMNS AFTER TRANSFORMATION STAGE - ROWS: 26 COLUMNS: 3”
[1] “______________________________”
```

DATA STAGING – TRANSFORMATION STAGE

CRIME - OUTCOME DATA

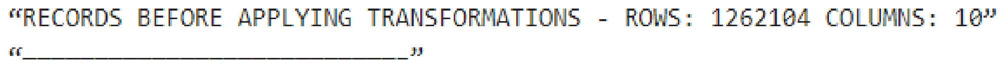
1. Data Validity - Rows and Columns Count before Transformation

Code →



```
#STEP-1: DATA VALIDITY - RECORDS BEFORE APPLYING TRANSFORMATIONS
paste0('RECORDS BEFORE APPLYING TRANSFORMATIONS - ROWS: ',nrow(oc),' COLUMNS: ',ncol(oc))
print("------------------------------------------------------------------------------")
```

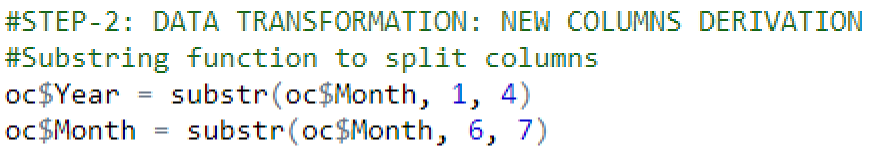
Output →



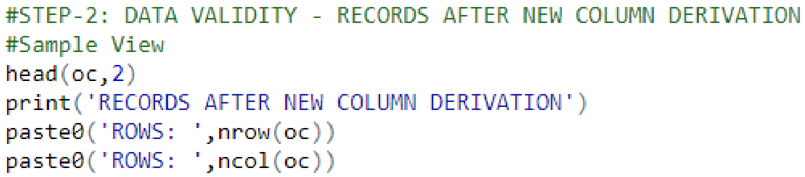
```
“RECORDS BEFORE APPLYING TRANSFORMATIONS - ROWS: 1262104 COLUMNS: 10”
“____________________________”
```

2. Data Transformation - New Columns Derivation

Code →

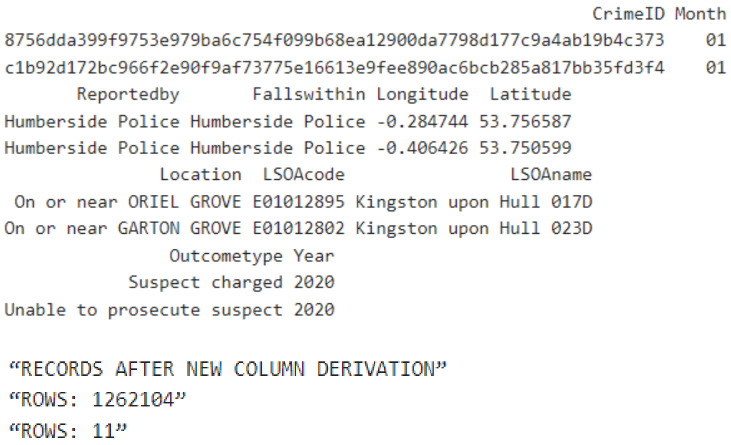


```
#STEP-2: DATA TRANSFORMATION: NEW COLUMNS DERIVATION
#Substring function to split columns
oc$Year = substr(oc$Month, 1, 4)
oc$Month = substr(oc$Month, 6, 7)
```



```
#STEP-2: DATA VALIDITY - RECORDS AFTER NEW COLUMN DERIVATION
#Sample View
head(oc,2)
print('RECORDS AFTER NEW COLUMN DERIVATION')
paste0('ROWS: ',nrow(oc))
paste0('ROWS: ',ncol(oc))
```

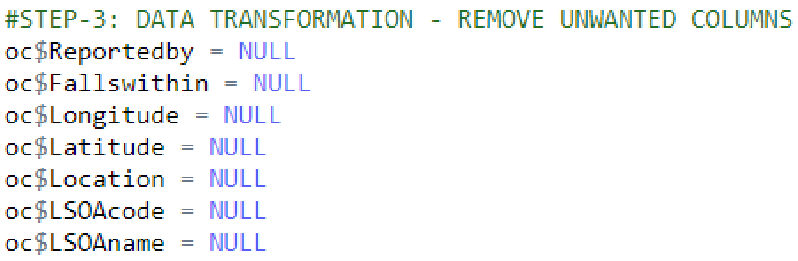
Output →



```
                                                          CrimeID Month
8756dda399f9753e979ba6c754f099b68ea12900da7798d177c9a4ab19b4c373    01
c1b92d172bc966f2e90f9af73775e16613e9fee890ac6bcb285a817bb35fd3f4    01
       Reportedby        Fallswithin Longitude  Latitude
Humberside Police Humberside Police -0.284744 53.756587
Humberside Police Humberside Police -0.406426 53.750599
               Location  LSOAcode                LSOAname
 On or near ORIEL GROVE E01012895 Kingston upon Hull 017D
On or near GARTON GROVE E01012802 Kingston upon Hull 023D
               Outcometype Year
           Suspect charged 2020
Unable to prosecute suspect 2020

“RECORDS AFTER NEW COLUMN DERIVATION”
“ROWS: 1262104”
“ROWS: 11”
```

3. Data Transformation - Remove unwanted columns

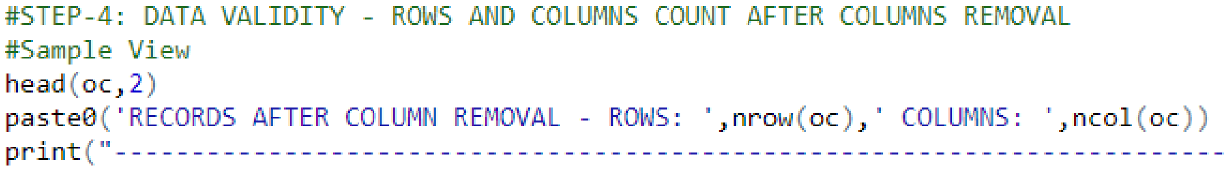
Code →



```
#STEP-3: DATA TRANSFORMATION - REMOVE UNWANTED COLUMNS
oc$Reportedby = NULL
oc$Fallswithin = NULL
oc$Longitude = NULL
oc$Latitude = NULL
oc$Location = NULL
oc$LSOAcode = NULL
oc$LSOAname = NULL
```

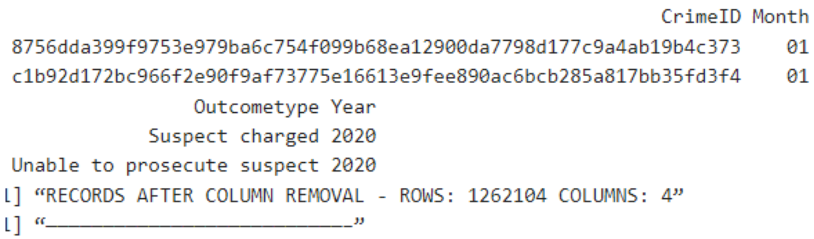
4. Data Validity - Rows and Columns count after columns removal

Code →



```
#STEP-4: DATA VALIDITY - ROWS AND COLUMNS COUNT AFTER COLUMNS REMOVAL
#Sample View
head(oc,2)
paste0('RECORDS AFTER COLUMN REMOVAL - ROWS: ',nrow(oc),' COLUMNS: ',ncol(oc))
print("-----------------------------------------------------------------------
```

Output →



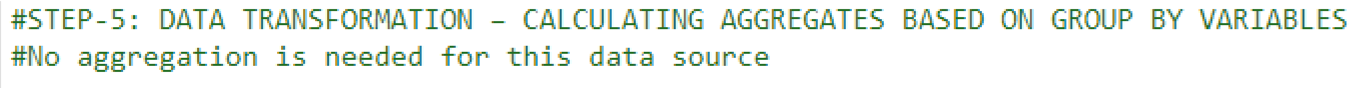
```
                                                          CrimeID Month
 8756dda399f9753e979ba6c754f099b68ea12900da7798d177c9a4ab19b4c373    01
 c1b92d172bc966f2e90f9af73775e16613e9fee890ac6bcb285a817bb35fd3f4    01
               Outcometype Year
           Suspect charged 2020
 Unable to prosecute suspect 2020
l] “RECORDS AFTER COLUMN REMOVAL - ROWS: 1262104 COLUMNS: 4”
l] “_______________________”
```

DATA STAGING – TRANSFORMATION STAGE

CRIME - OUTCOME DATA

5. Data Transformation – Calculating aggregates based on group by variables

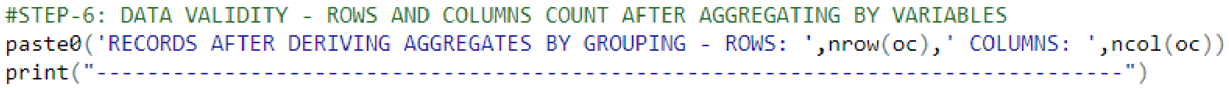
Code →



```
#STEP-5: DATA TRANSFORMATION – CALCULATING AGGREGATES BASED ON GROUP BY VARIABLES
#No aggregation is needed for this data source
```

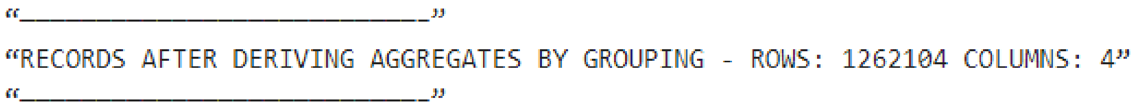
6. Data Validity - Rows and Columns count after aggregating by variables

Code →



```
#STEP-6: DATA VALIDITY - ROWS AND COLUMNS COUNT AFTER AGGREGATING BY VARIABLES
paste0('RECORDS AFTER DERIVING AGGREGATES BY GROUPING - ROWS: ',nrow(oc),' COLUMNS: ',ncol(oc))
print("-----------------------------------------------------------------------------")
```

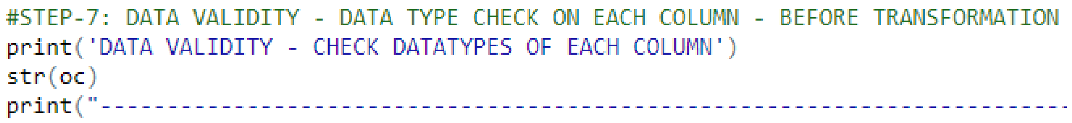
Output →



```
“________________________________”
“RECORDS AFTER DERIVING AGGREGATES BY GROUPING - ROWS: 1262104 COLUMNS: 4”
“________________________________”
```

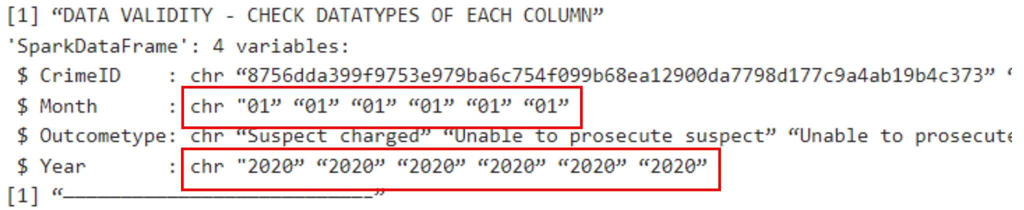
7. Data Validity - Data Type Check on each column - Before

Code →



```
#STEP-7: DATA VALIDITY - DATA TYPE CHECK ON EACH COLUMN - BEFORE TRANSFORMATION
print('DATA VALIDITY - CHECK DATATYPES OF EACH COLUMN')
str(oc)
print("--------------------------------------------------------------------------
```

Output →

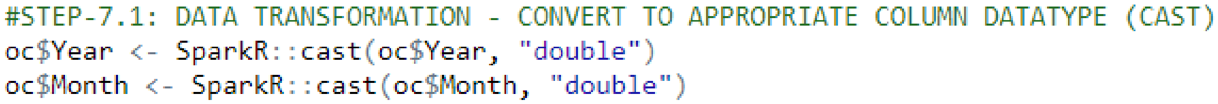


```
[1] “DATA VALIDITY - CHECK DATATYPES OF EACH COLUMN”
'SparkDataFrame': 4 variables:
 $ CrimeID    : chr “8756dda399f9753e979ba6c754f099b68ea12900da7798d177c9a4ab19b4c373” ‘
 $ Month      : chr "01” “01” “01” “01” “01” “01”
 $ Outcometype: chr “Suspect charged” “Unable to prosecute suspect” “Unable to prosecute
 $ Year       : chr "2020” “2020” “2020” “2020” “2020” “2020”
[1] “__________________________”
```

Few columns have correct data type, Step-7.1 will handle if any future data is having data type issue

7.1 Data Transformation - Convert to appropriate column datatype (CAST)

Code →



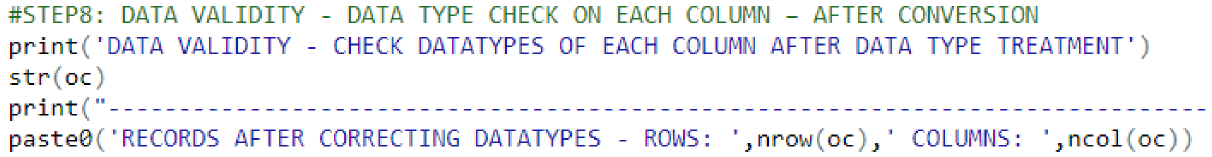
```
#STEP-7.1: DATA TRANSFORMATION - CONVERT TO APPROPRIATE COLUMN DATATYPE (CAST)
oc$Year <- SparkR::cast(oc$Year, "double")
oc$Month <- SparkR::cast(oc$Month, "double")
```

DATA STAGING – TRANSFORMATION STAGE

CRIME - OUTCOME DATA

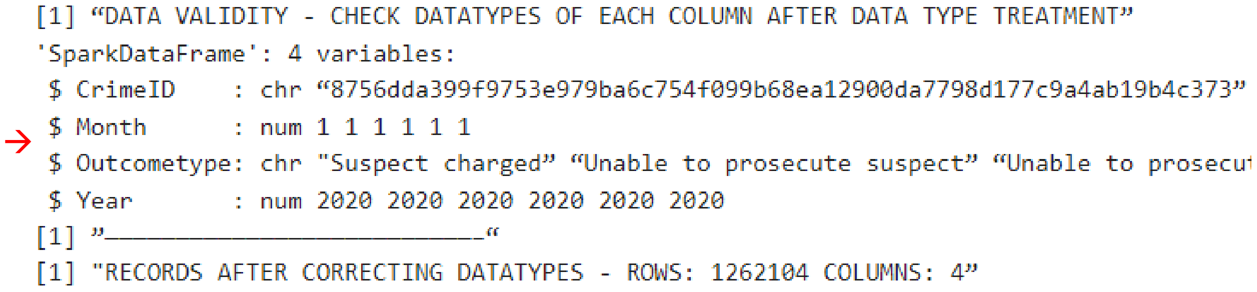
8. Data Validity - Data Type Check on each column – After conversion

Code →



```
#STEP8: DATA VALIDITY - DATA TYPE CHECK ON EACH COLUMN – AFTER CONVERSION
print('DATA VALIDITY - CHECK DATATYPES OF EACH COLUMN AFTER DATA TYPE TREATMENT')
str(oc)
print("-----------------------------------------------------------------------------
paste0('RECORDS AFTER CORRECTING DATATYPES - ROWS: ',nrow(oc),' COLUMNS: ',ncol(oc))
```

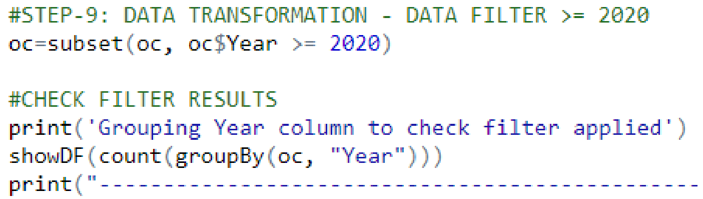
Output →



```
[1] “DATA VALIDITY - CHECK DATATYPES OF EACH COLUMN AFTER DATA TYPE TREATMENT”
'SparkDataFrame': 4 variables:
 $ CrimeID    : chr “8756dda399f9753e979ba6c754f099b68ea12900da7798d177c9a4ab19b4c373”
 $ Month      : num 1 1 1 1 1 1
 $ Outcometype: chr "Suspect charged” “Unable to prosecute suspect” “Unable to prosecut
 $ Year       : num 2020 2020 2020 2020 2020 2020
[1] ”___________________________“
[1] "RECORDS AFTER CORRECTING DATATYPES - ROWS: 1262104 COLUMNS: 4”
```

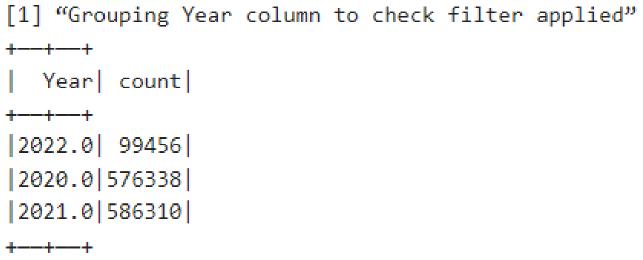
9. Data Transformation - Data filter >= 2020

Code →



```
#STEP-9: DATA TRANSFORMATION - DATA FILTER >= 2020
oc=subset(oc, oc$Year >= 2020)

#CHECK FILTER RESULTS
print('Grouping Year column to check filter applied')
showDF(count(groupBy(oc, "Year")))
print("---------------------------------------------
```

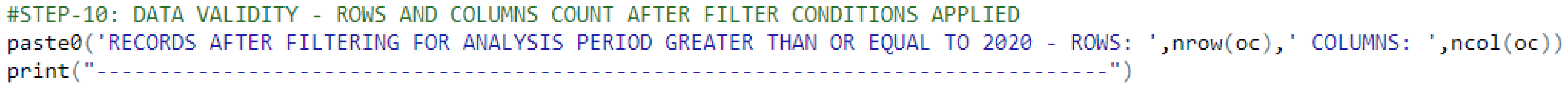
Output →



```
[1] “Grouping Year column to check filter applied”
+——+——+
|  Year| count|
+——+——+
|2022.0| 99456|
|2020.0|576338|
|2021.0|586310|
+——+——+
```

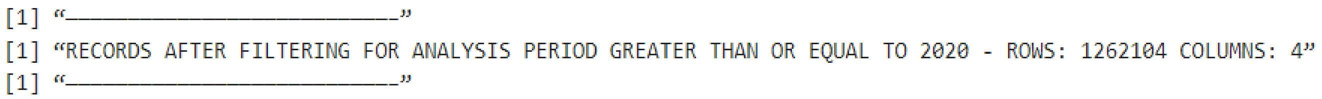
10. Data Validity - Rows and Columns Count after filter conditions applied

Code →



```
#STEP-10: DATA VALIDITY - ROWS AND COLUMNS COUNT AFTER FILTER CONDITIONS APPLIED
paste0('RECORDS AFTER FILTERING FOR ANALYSIS PERIOD GREATER THAN OR EQUAL TO 2020 - ROWS: ',nrow(oc),' COLUMNS: ',ncol(oc))
print("--------------------------------------------------------------------------------")
```

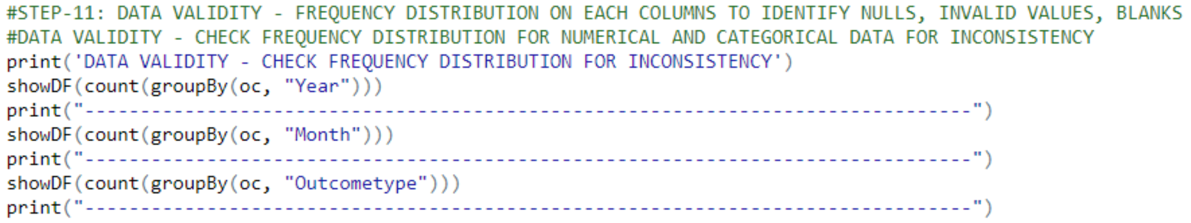
Output →



```
[1] “_________________________”
[1] “RECORDS AFTER FILTERING FOR ANALYSIS PERIOD GREATER THAN OR EQUAL TO 2020 - ROWS: 1262104 COLUMNS: 4”
[1] “_________________________”
```

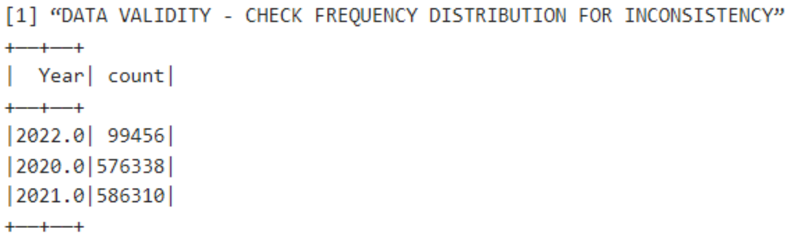
11. Data Validity - Frequency distribution on each columns to identify Nulls, Invalid values, Blanks

Code →



```
#STEP-11: DATA VALIDITY - FREQUENCY DISTRIBUTION ON EACH COLUMNS TO IDENTIFY NULLS, INVALID VALUES, BLANKS
#DATA VALIDITY - CHECK FREQUENCY DISTRIBUTION FOR NUMERICAL AND CATEGORICAL DATA FOR INCONSISTENCY
print('DATA VALIDITY - CHECK FREQUENCY DISTRIBUTION FOR INCONSISTENCY')
showDF(count(groupBy(oc, "Year")))
print("--------------------------------------------------------------------------------")
showDF(count(groupBy(oc, "Month")))
print("--------------------------------------------------------------------------------")
showDF(count(groupBy(oc, "Outcometype")))
print("--------------------------------------------------------------------------------")
```

Output →



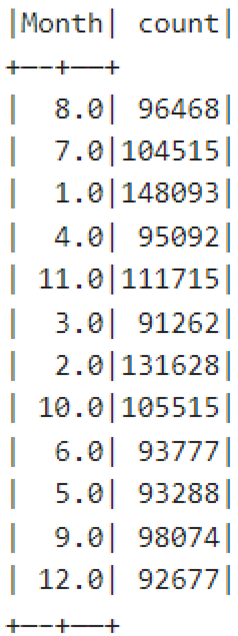
```
[1] “DATA VALIDITY - CHECK FREQUENCY DISTRIBUTION FOR INCONSISTENCY”
+——+——+
|  Year| count|
+——+——+
|2022.0| 99456|
|2020.0|576338|
|2021.0|586310|
+——+——+
```

To be continued

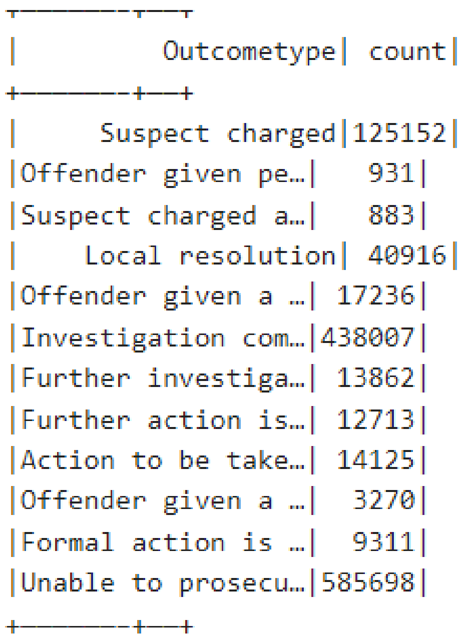
DATA STAGING – TRANSFORMATION STAGE

CRIME - OUTCOME DATA

Output →



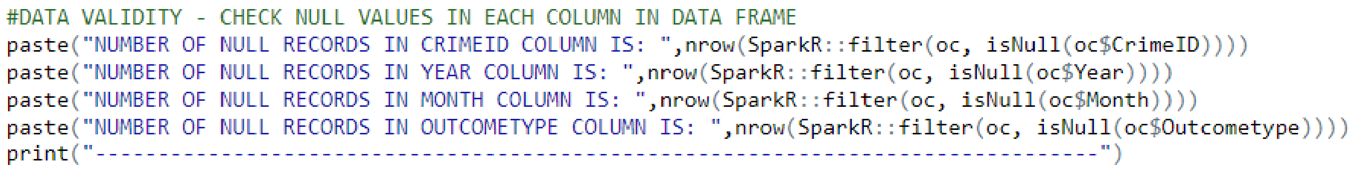
```
|Month| count|
+---+---+
|  8.0| 96468|
|  7.0|104515|
|  1.0|148093|
|  4.0| 95092|
| 11.0|111715|
|  3.0| 91262|
|  2.0|131628|
| 10.0|105515|
|  6.0| 93777|
|  5.0| 93288|
|  9.0| 98074|
| 12.0| 92677|
+---+---+
```



```
+--------+---+
|         Outcometype| count|
+--------+---+
|     Suspect charged|125152|
|Offender given pe…|   931|
|Suspect charged a…|   883|
|    Local resolution| 40916|
|Offender given a …| 17236|
|Investigation com…|438007|
|Further investiga…| 13862|
|Further action is…| 12713|
|Action to be take…| 14125|
|Offender given a …|  3270|
|Formal action is …|  9311|
|Unable to prosecu…|585698|
+--------+---+
```

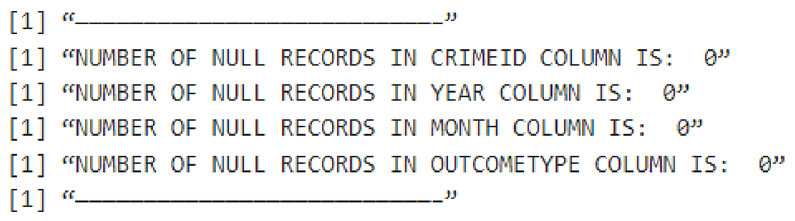
Note: No invalid data detected

Code →



```
#DATA VALIDITY - CHECK NULL VALUES IN EACH COLUMN IN DATA FRAME
paste("NUMBER OF NULL RECORDS IN CRIMEID COLUMN IS: ",nrow(SparkR::filter(oc, isNull(oc$CrimeID))))
paste("NUMBER OF NULL RECORDS IN YEAR COLUMN IS: ",nrow(SparkR::filter(oc, isNull(oc$Year))))
paste("NUMBER OF NULL RECORDS IN MONTH COLUMN IS: ",nrow(SparkR::filter(oc, isNull(oc$Month))))
paste("NUMBER OF NULL RECORDS IN OUTCOMETYPE COLUMN IS: ",nrow(SparkR::filter(oc, isNull(oc$Outcometype))))
print("-----------------------------------------------------------------------------------")
```

Output →

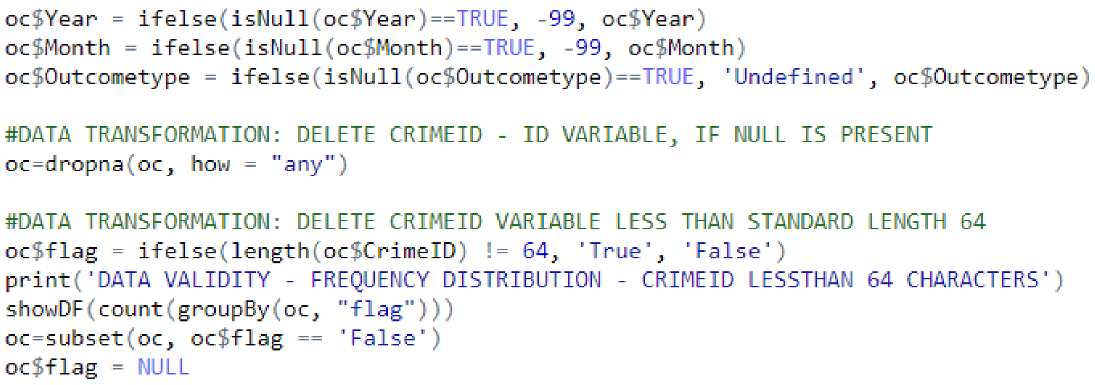


```
[1] "____________________________"
[1] "NUMBER OF NULL RECORDS IN CRIMEID COLUMN IS:  0"
[1] "NUMBER OF NULL RECORDS IN YEAR COLUMN IS:  0"
[1] "NUMBER OF NULL RECORDS IN MONTH COLUMN IS:  0"
[1] "NUMBER OF NULL RECORDS IN OUTCOMETYPE COLUMN IS:  0"
[1] "____________________________"
```

11.1 Data Transformation - Treatment:

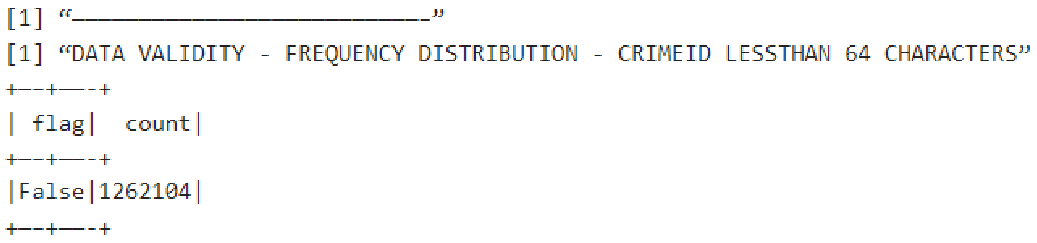
- **Numeric Column = Null to -99 value change**
- **Character column = Null to “Undefined”**
- **Delete ID columns if NULL and Crime ID length != 64 characters**

Code →



```
oc$Year = ifelse(isNull(oc$Year)==TRUE, -99, oc$Year)
oc$Month = ifelse(isNull(oc$Month)==TRUE, -99, oc$Month)
oc$Outcometype = ifelse(isNull(oc$Outcometype)==TRUE, 'Undefined', oc$Outcometype)

#DATA TRANSFORMATION: DELETE CRIMEID - ID VARIABLE, IF NULL IS PRESENT
oc=dropna(oc, how = "any")

#DATA TRANSFORMATION: DELETE CRIMEID VARIABLE LESS THAN STANDARD LENGTH 64
oc$flag = ifelse(length(oc$CrimeID) != 64, 'True', 'False')
print('DATA VALIDITY - FREQUENCY DISTRIBUTION - CRIMEID LESSTHAN 64 CHARACTERS')
showDF(count(groupBy(oc, "flag")))
oc=subset(oc, oc$flag == 'False')
oc$flag = NULL
```

Output →



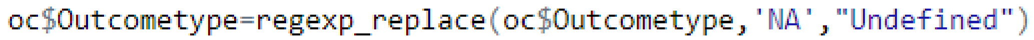
```
[1] "____________________________"
[1] "DATA VALIDITY - FREQUENCY DISTRIBUTION - CRIMEID LESSTHAN 64 CHARACTERS"
+---+----+
| flag|  count|
+---+----+
|False|1262104|
+---+----+
```

Note: No invalid data detected but still data treatment steps are in place to handle any data issues in future

DATA STAGING – TRANSFORMATION STAGE

CRIME - OUTCOME DATA

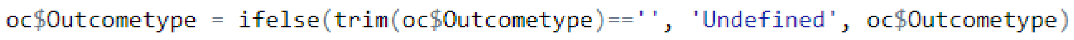
11.2 Data Transformation - Treatment:
- **Character column = ‘NA’ to “Undefined”**
- **Character column = Invalid values to “Undefined”**



```
oc$Outcometype=regexp_replace(oc$Outcometype,'NA',"Undefined")
```

Note: No invalid data detected but still data treatment steps are in place to handle any data issues in future

11.3 Data Transformation - Treatment:
- **Character column = Blank values to “Undefined”**

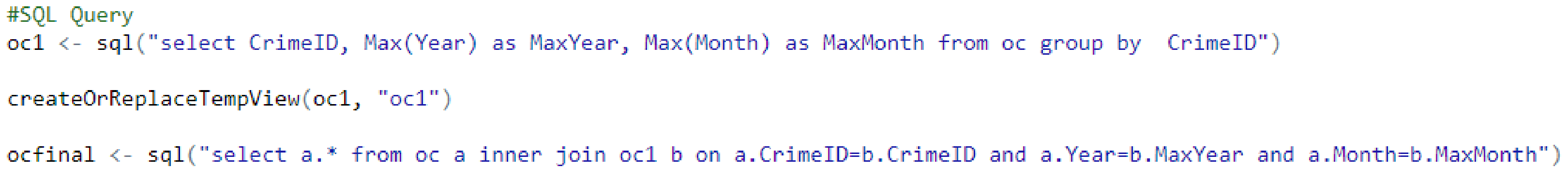


```
oc$Outcometype = ifelse(trim(oc$Outcometype)=='', 'Undefined', oc$Outcometype)
```

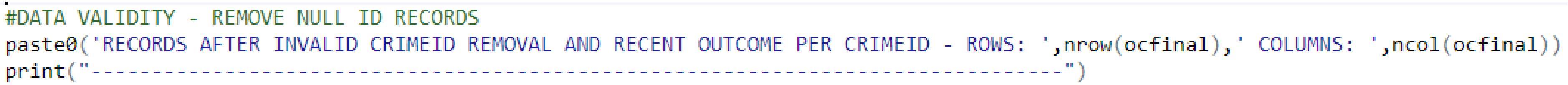
Note: No invalid data detected but still data treatment steps are in place to handle any data issues in future

11.4 Data Transformation – Additional Business Logic:
RETAIN MOST RECENT OUTCOME FOR A CRIME (Multiple outcomes are present over time period like year or month)

Code →



```
#SQL Query
oc1 <- sql("select CrimeID, Max(Year) as MaxYear, Max(Month) as MaxMonth from oc group by  CrimeID")

createOrReplaceTempView(oc1, "oc1")

ocfinal <- sql("select a.* from oc a inner join oc1 b on a.CrimeID=b.CrimeID and a.Year=b.MaxYear and a.Month=b.MaxMonth")
```



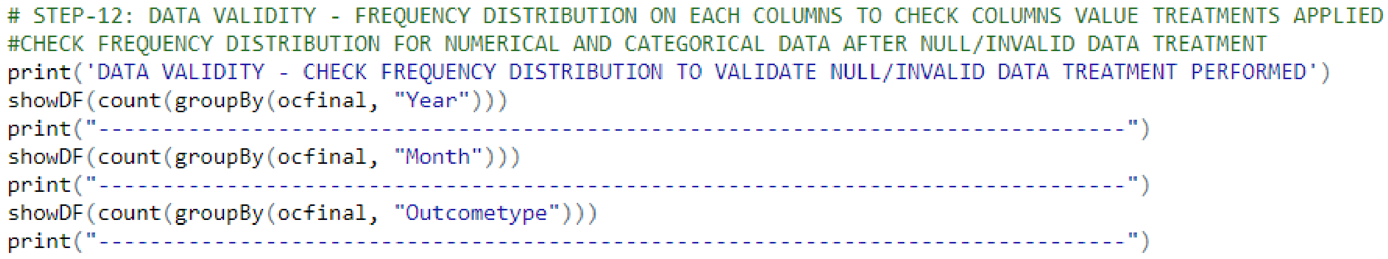
```
#DATA VALIDITY - REMOVE NULL ID RECORDS
paste0('RECORDS AFTER INVALID CRIMEID REMOVAL AND RECENT OUTCOME PER CRIMEID - ROWS: ',nrow(ocfinal),' COLUMNS: ',ncol(ocfinal))
print("------------------------------------------------------------------------------------")
```

Output →

```
[1] “RECORDS AFTER INVALID CRIMEID REMOVAL AND RECENT OUTCOME PER CRIMEID - ROWS: 1131460 COLUMNS: 4”
[1] “_____________________________”
```

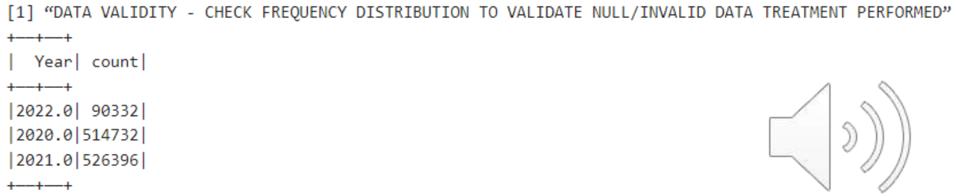
12. Data Validity - Frequency distribution on each columns to check columns value treatments applied

Code →



```
# STEP-12: DATA VALIDITY - FREQUENCY DISTRIBUTION ON EACH COLUMNS TO CHECK COLUMNS VALUE TREATMENTS APPLIED
#CHECK FREQUENCY DISTRIBUTION FOR NUMERICAL AND CATEGORICAL DATA AFTER NULL/INVALID DATA TREATMENT
print('DATA VALIDITY - CHECK FREQUENCY DISTRIBUTION TO VALIDATE NULL/INVALID DATA TREATMENT PERFORMED')
showDF(count(groupBy(ocfinal, "Year")))
print("------------------------------------------------------------------------------------")
showDF(count(groupBy(ocfinal, "Month")))
print("------------------------------------------------------------------------------------")
showDF(count(groupBy(ocfinal, "Outcometype")))
print("------------------------------------------------------------------------------------")
```

Output →

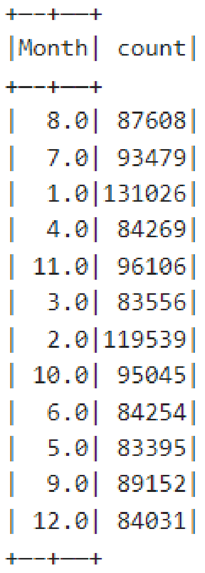


```
[1] “DATA VALIDITY - CHECK FREQUENCY DISTRIBUTION TO VALIDATE NULL/INVALID DATA TREATMENT PERFORMED”
+---+---+
|  Year| count|
+---+---+
|2022.0| 90332|
|2020.0|514732|
|2021.0|526396|
+---+---+
```

DATA STAGING – TRANSFORMATION STAGE

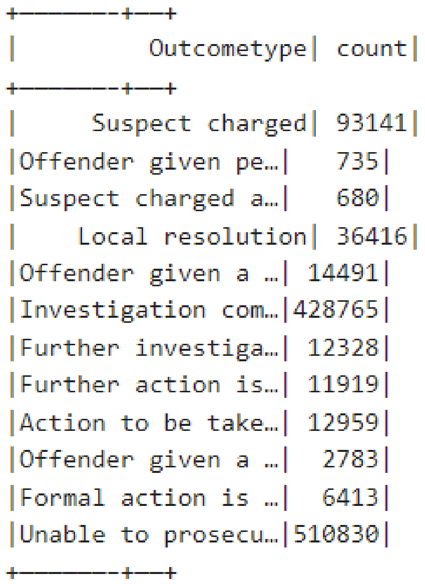
CRIME - OUTCOME DATA

Output →



```
+--+---+
|Month| count|
+--+---+
|  8.0| 87608|
|  7.0| 93479|
|  1.0|131026|
|  4.0| 84269|
| 11.0| 96106|
|  3.0| 83556|
|  2.0|119539|
| 10.0| 95045|
|  6.0| 84254|
|  5.0| 83395|
|  9.0| 89152|
| 12.0| 84031|
+--+---+
```

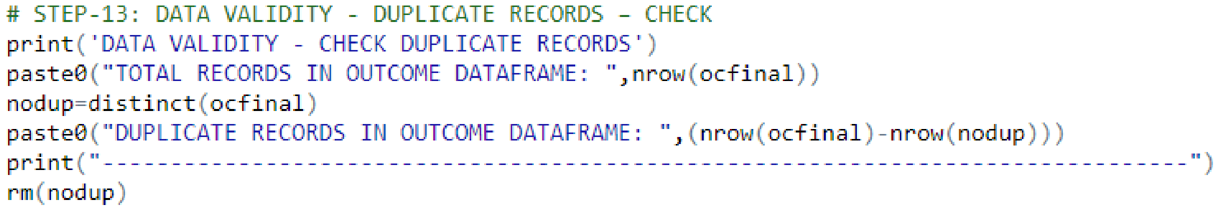
Note: No invalid data detected



```
+--------+---+
|         Outcometype| count|
+--------+---+
|     Suspect charged| 93141|
|Offender given pe…|   735|
|Suspect charged a…|   680|
|    Local resolution| 36416|
|Offender given a …| 14491|
|Investigation com…|428765|
|Further investiga…| 12328|
|Further action is…| 11919|
|Action to be take…| 12959|
|Offender given a …|  2783|
|Formal action is …|  6413|
|Unable to prosecu…|510830|
+--------+---+
```

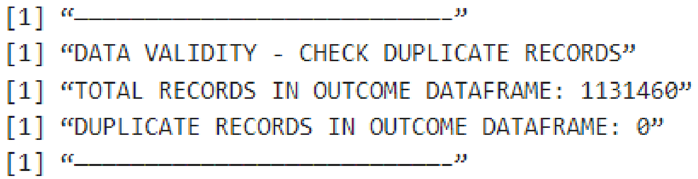
13. Data Validity - Duplicate Records - Check

Code →



```
# STEP-13: DATA VALIDITY - DUPLICATE RECORDS - CHECK
print('DATA VALIDITY - CHECK DUPLICATE RECORDS')
paste0("TOTAL RECORDS IN OUTCOME DATAFRAME: ",nrow(ocfinal))
nodup=distinct(ocfinal)
paste0("DUPLICATE RECORDS IN OUTCOME DATAFRAME: ",(nrow(ocfinal)-nrow(nodup)))
print("-------------------------------------------------------------------------------")
rm(nodup)
```

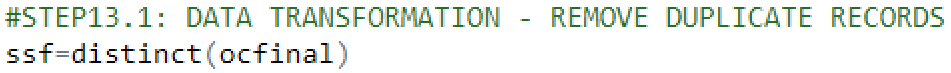
Output →



```
[1] "_____________________________"
[1] "DATA VALIDITY - CHECK DUPLICATE RECORDS"
[1] "TOTAL RECORDS IN OUTCOME DATAFRAME: 1131460"
[1] "DUPLICATE RECORDS IN OUTCOME DATAFRAME: 0"
[1] "_____________________________"
```

13.1 Data Transformation - Remove Duplicate Records

Code →



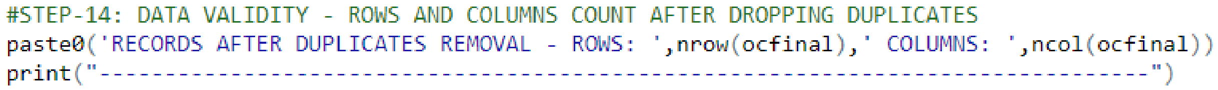
```
#STEP13.1: DATA TRANSFORMATION - REMOVE DUPLICATE RECORDS
ssf=distinct(ocfinal)
```

DATA STAGING – TRANSFORMATION STAGE

CRIME - OUTCOME DATA

14. Data Validity - Rows and Columns Count After dropping duplicates

Code →



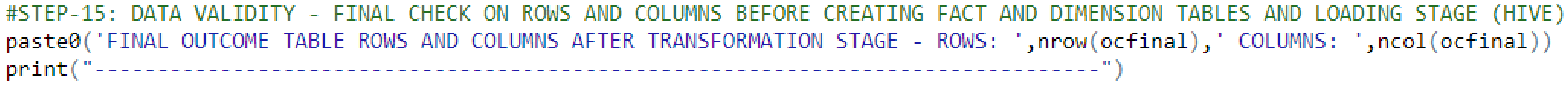
```
#STEP-14: DATA VALIDITY - ROWS AND COLUMNS COUNT AFTER DROPPING DUPLICATES
paste0('RECORDS AFTER DUPLICATES REMOVAL - ROWS: ',nrow(ocfinal),' COLUMNS: ',ncol(ocfinal))
print("--------------------------------------------------------------------------------")
```

Output →

```
[1] “___________________________”
[1] “RECORDS AFTER DUPLICATES REMOVAL - ROWS: 1131460 COLUMNS: 4”
[1] “___________________________”
```

15. Data Validity - Final Check on Rows and Columns before creating Fact and Dimension tables and Loading Stage (Hive)

Code →



```
#STEP-15: DATA VALIDITY - FINAL CHECK ON ROWS AND COLUMNS BEFORE CREATING FACT AND DIMENSION TABLES AND LOADING STAGE (HIVE)
paste0('FINAL OUTCOME TABLE ROWS AND COLUMNS AFTER TRANSFORMATION STAGE - ROWS: ',nrow(ocfinal),' COLUMNS: ',ncol(ocfinal))
print("--------------------------------------------------------------------------")
```

Output →

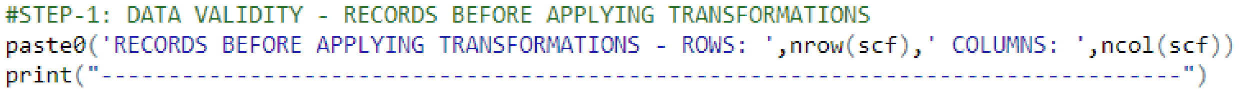
```
[1] “____________________________”
[1] “FINAL OUTCOME TABLE ROWS AND COLUMNS AFTER TRANSFORMATION STAGE - ROWS: 1131460 COLUMNS: 4”
[1] “____________________________”
```

DATA STAGING – TRANSFORMATION STAGE

CRIME - STREET DATA

1. Data Validity - Rows and Columns Count before Transformation

Code →



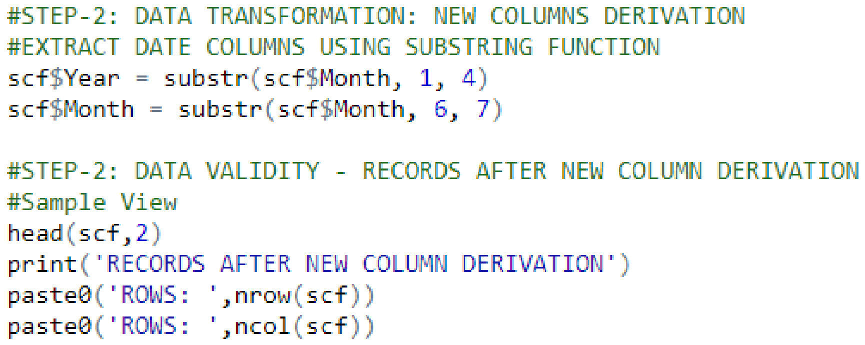
```
#STEP-1: DATA VALIDITY - RECORDS BEFORE APPLYING TRANSFORMATIONS
paste0('RECORDS BEFORE APPLYING TRANSFORMATIONS - ROWS: ',nrow(scf),' COLUMNS: ',ncol(scf))
print("----------------------------------------------------------------------------")
```

Output →

```
[1] “RECORDS BEFORE APPLYING TRANSFORMATIONS - ROWS: 1429456 COLUMNS: 12”
[1] “—————————————————————————”
```

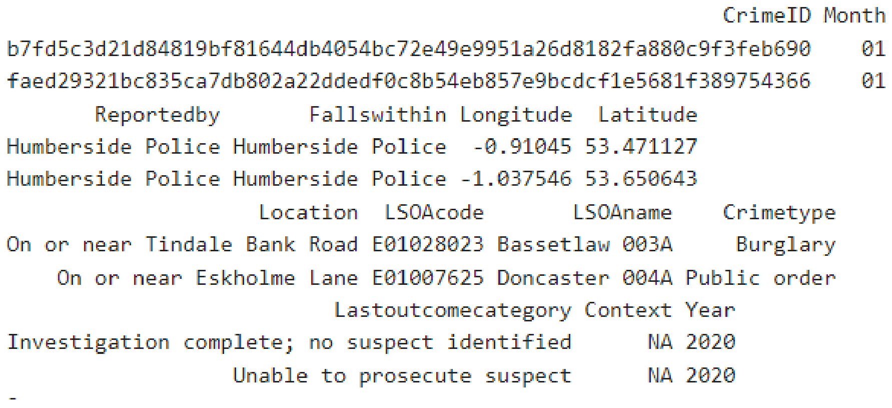
2. Data Transformation - New Columns Derivation

Code →



```
#STEP-2: DATA TRANSFORMATION: NEW COLUMNS DERIVATION
#EXTRACT DATE COLUMNS USING SUBSTRING FUNCTION
scf$Year = substr(scf$Month, 1, 4)
scf$Month = substr(scf$Month, 6, 7)

#STEP-2: DATA VALIDITY - RECORDS AFTER NEW COLUMN DERIVATION
#Sample View
head(scf,2)
print('RECORDS AFTER NEW COLUMN DERIVATION')
paste0('ROWS: ',nrow(scf))
paste0('ROWS: ',ncol(scf))
```

Output →

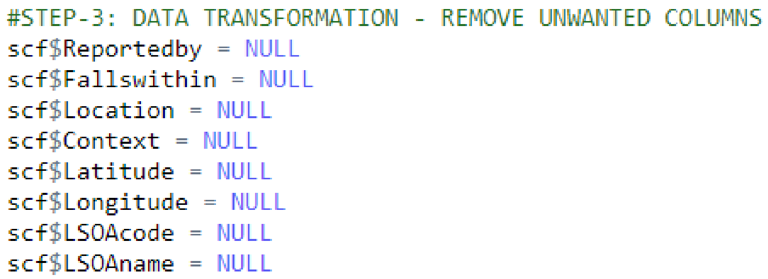


```
                                                             CrimeID Month
b7fd5c3d21d84819bf81644db4054bc72e49e9951a26d8182fa880c9f3feb690    01
faed29321bc835ca7db802a22ddedf0c8b54eb857e9bcdcf1e5681f389754366    01
        Reportedby        Fallswithin Longitude  Latitude
Humberside Police Humberside Police  -0.91045 53.471127
Humberside Police Humberside Police -1.037546 53.650643
                    Location  LSOAcode       LSOAname    Crimetype
On or near Tindale Bank Road E01028023 Bassetlaw 003A     Burglary
    On or near Eskholme Lane E01007625 Doncaster 004A Public order
                         Lastoutcomecategory Context Year
Investigation complete; no suspect identified      NA 2020
                 Unable to prosecute suspect      NA 2020
-
```

```
“RECORDS AFTER NEW COLUMN DERIVATION”
“ROWS: 1429456”
“ROWS: 13”
```

3. Data Transformation - Remove unwanted columns

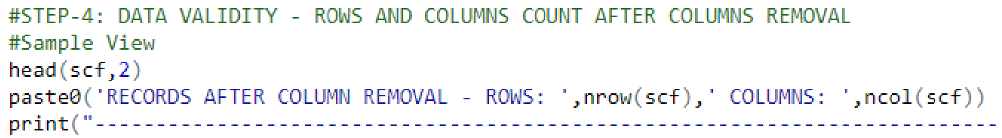
Code →



```
#STEP-3: DATA TRANSFORMATION - REMOVE UNWANTED COLUMNS
scf$Reportedby = NULL
scf$Fallswithin = NULL
scf$Location = NULL
scf$Context = NULL
scf$Latitude = NULL
scf$Longitude = NULL
scf$LSOAcode = NULL
scf$LSOAname = NULL
```

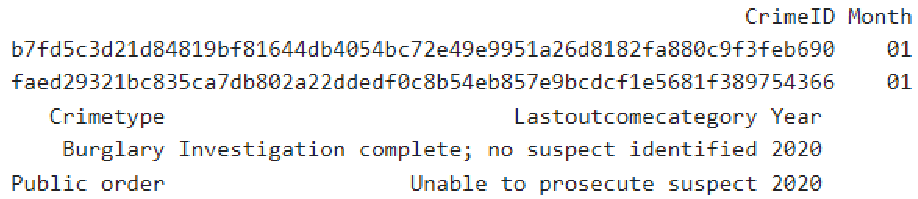
4. Data Validity - Rows and Columns count after columns removal

Code →



```
#STEP-4: DATA VALIDITY - ROWS AND COLUMNS COUNT AFTER COLUMNS REMOVAL
#Sample View
head(scf,2)
paste0('RECORDS AFTER COLUMN REMOVAL - ROWS: ',nrow(scf),' COLUMNS: ',ncol(scf))
print("----------------------------------------------------------------------------
```

Output →



```
                                                             CrimeID Month
b7fd5c3d21d84819bf81644db4054bc72e49e9951a26d8182fa880c9f3feb690    01
faed29321bc835ca7db802a22ddedf0c8b54eb857e9bcdcf1e5681f389754366    01
   Crimetype                           Lastoutcomecategory Year
    Burglary Investigation complete; no suspect identified 2020
Public order                   Unable to prosecute suspect 2020
```

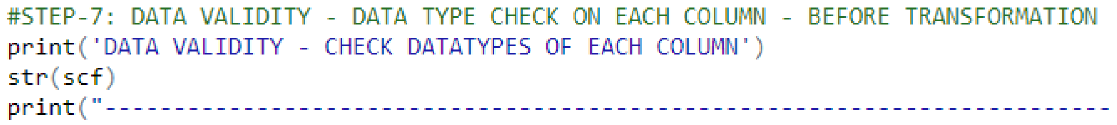
```
1] “RECORDS AFTER COLUMN REMOVAL - ROWS: 1429456 COLUMNS: 5”
1] “—————————————————————————”
```

DATA STAGING – TRANSFORMATION STAGE

CRIME - STREET DATA

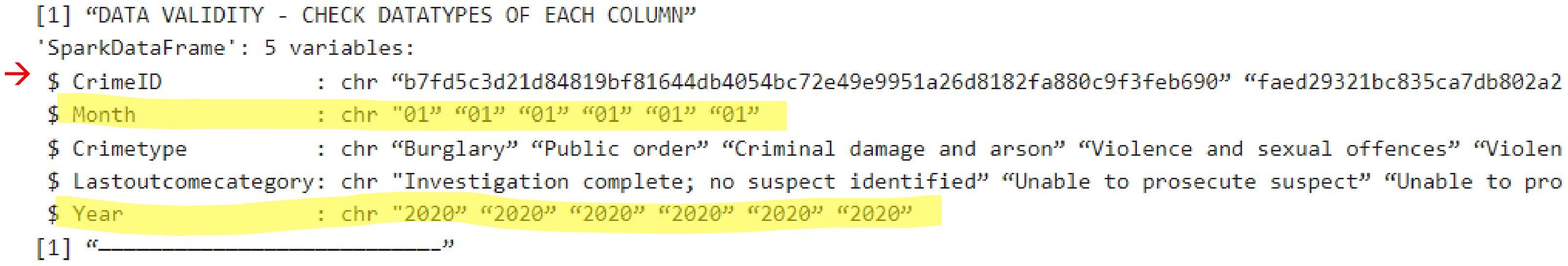
7. Data Validity - Data Type Check on each column - Before

Code →



```
#STEP-7: DATA VALIDITY - DATA TYPE CHECK ON EACH COLUMN - BEFORE TRANSFORMATION
print('DATA VALIDITY - CHECK DATATYPES OF EACH COLUMN')
str(scf)
print("---------------------------------------------------------------------------
```

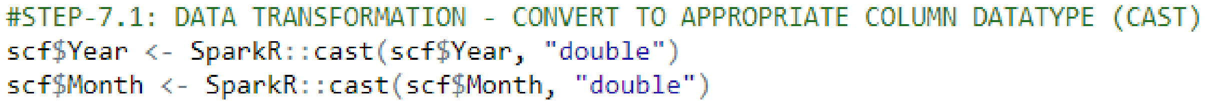
Output →



```
[1] “DATA VALIDITY - CHECK DATATYPES OF EACH COLUMN”
'SparkDataFrame': 5 variables:
 $ CrimeID           : chr “b7fd5c3d21d84819bf81644db4054bc72e49e9951a26d8182fa880c9f3feb690” “faed29321bc835ca7db802a2
 $ Month             : chr "01” “01” “01” “01” “01” “01”
 $ Crimetype         : chr “Burglary” “Public order” “Criminal damage and arson” “Violence and sexual offences” “Violen
 $ Lastoutcomecategory: chr "Investigation complete; no suspect identified” “Unable to prosecute suspect” “Unable to pro
 $ Year              : chr "2020” “2020” “2020” “2020” “2020” “2020”
[1] “————————————————————————————”
```

7.1 Data Transformation - Convert to appropriate column datatype (CAST)

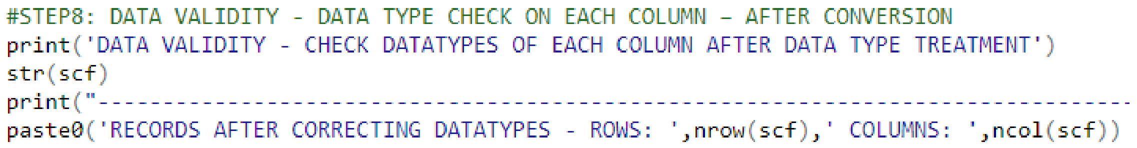
Code →



```
#STEP-7.1: DATA TRANSFORMATION - CONVERT TO APPROPRIATE COLUMN DATATYPE (CAST)
scf$Year <- SparkR::cast(scf$Year, "double")
scf$Month <- SparkR::cast(scf$Month, "double")
```

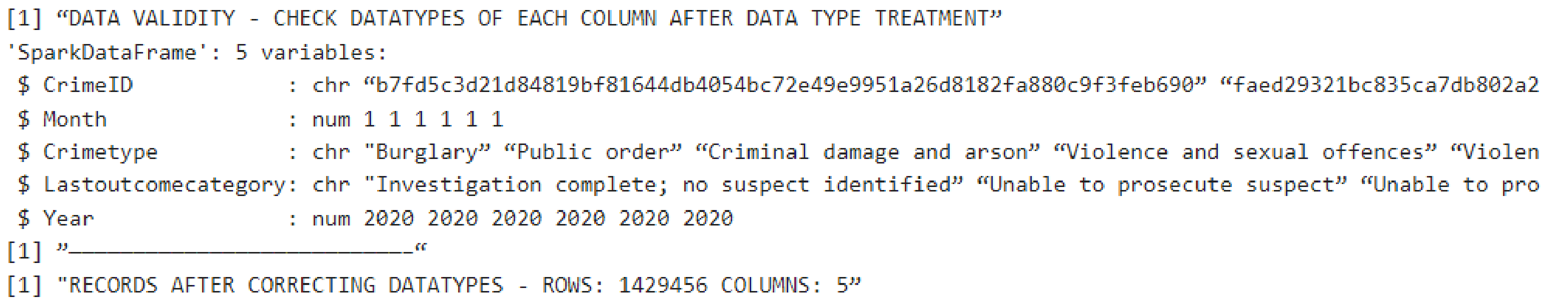
8. Data Validity - Data Type Check on each column – After conversion

Code →



```
#STEP8: DATA VALIDITY - DATA TYPE CHECK ON EACH COLUMN – AFTER CONVERSION
print('DATA VALIDITY - CHECK DATATYPES OF EACH COLUMN AFTER DATA TYPE TREATMENT')
str(scf)
print("-----------------------------------------------------------------------------
paste0('RECORDS AFTER CORRECTING DATATYPES - ROWS: ',nrow(scf),' COLUMNS: ',ncol(scf))
```

Output →



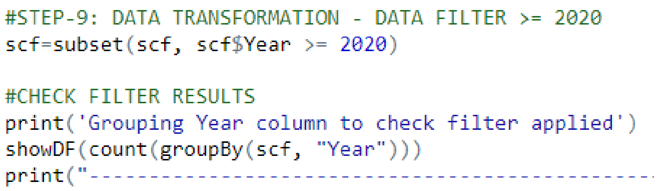
```
[1] “DATA VALIDITY - CHECK DATATYPES OF EACH COLUMN AFTER DATA TYPE TREATMENT”
'SparkDataFrame': 5 variables:
 $ CrimeID           : chr “b7fd5c3d21d84819bf81644db4054bc72e49e9951a26d8182fa880c9f3feb690” “faed29321bc835ca7db802a2
 $ Month             : num 1 1 1 1 1 1
 $ Crimetype         : chr "Burglary” “Public order” “Criminal damage and arson” “Violence and sexual offences” “Violen
 $ Lastoutcomecategory: chr "Investigation complete; no suspect identified" “Unable to prosecute suspect” “Unable to pro
 $ Year              : num 2020 2020 2020 2020 2020 2020
[1] ”————————————————————————————“
[1] "RECORDS AFTER CORRECTING DATATYPES - ROWS: 1429456 COLUMNS: 5”
```

DATA STAGING – TRANSFORMATION STAGE

CRIME - STREET DATA

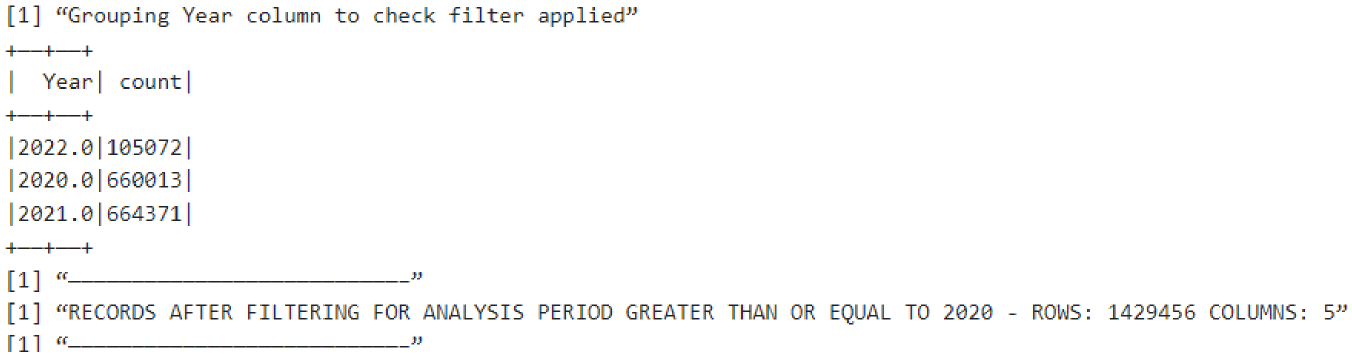
9. Data Transformation - Data filter >= 2020

Code →



```
#STEP-9: DATA TRANSFORMATION - DATA FILTER >= 2020
scf=subset(scf, scf$Year >= 2020)

#CHECK FILTER RESULTS
print('Grouping Year column to check filter applied')
showDF(count(groupBy(scf, "Year")))
print("--------------------------------------------------
```

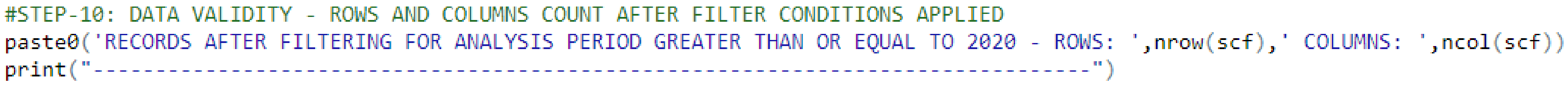
Output →



```
[1] “Grouping Year column to check filter applied”
+—+—+
|  Year| count|
+—+—+
|2022.0|105072|
|2020.0|660013|
|2021.0|664371|
+—+—+
[1] “_____________________________”
[1] “RECORDS AFTER FILTERING FOR ANALYSIS PERIOD GREATER THAN OR EQUAL TO 2020 - ROWS: 1429456 COLUMNS: 5”
[1] “_____________________________”
```

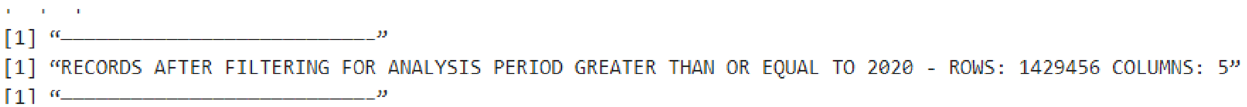
10. Data Validity - Rows and Columns Count after filter conditions applied

Code →



```
#STEP-10: DATA VALIDITY - ROWS AND COLUMNS COUNT AFTER FILTER CONDITIONS APPLIED
paste0('RECORDS AFTER FILTERING FOR ANALYSIS PERIOD GREATER THAN OR EQUAL TO 2020 - ROWS: ',nrow(scf),' COLUMNS: ',ncol(scf))
print("-----------------------------------------------------------------------------")
```

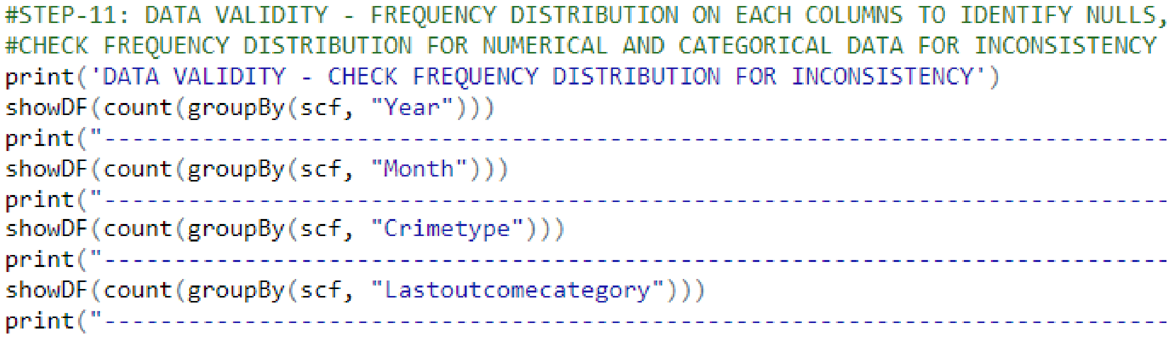
Output →



```
[1] “___________________________”
[1] “RECORDS AFTER FILTERING FOR ANALYSIS PERIOD GREATER THAN OR EQUAL TO 2020 - ROWS: 1429456 COLUMNS: 5”
[1] “___________________________”
```

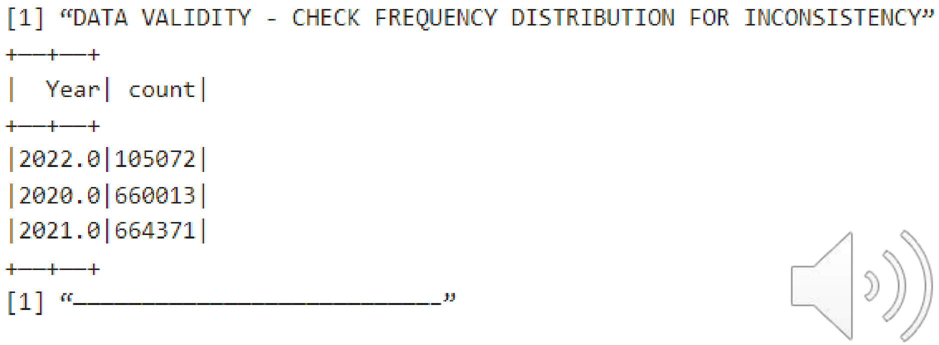
11. Data Validity - Frequency distribution on each columns to identify Nulls, Invalid values, Blanks

Code →



```
#STEP-11: DATA VALIDITY - FREQUENCY DISTRIBUTION ON EACH COLUMNS TO IDENTIFY NULLS,
#CHECK FREQUENCY DISTRIBUTION FOR NUMERICAL AND CATEGORICAL DATA FOR INCONSISTENCY
print('DATA VALIDITY - CHECK FREQUENCY DISTRIBUTION FOR INCONSISTENCY')
showDF(count(groupBy(scf, "Year")))
print("---------------------------------------------------------------------------
showDF(count(groupBy(scf, "Month")))
print("---------------------------------------------------------------------------
showDF(count(groupBy(scf, "Crimetype")))
print("---------------------------------------------------------------------------
showDF(count(groupBy(scf, "Lastoutcomecategory")))
print("---------------------------------------------------------------------------
```

Output →

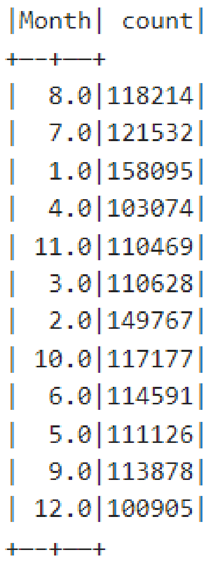


```
[1] “DATA VALIDITY - CHECK FREQUENCY DISTRIBUTION FOR INCONSISTENCY”
+—+—+
|  Year| count|
+—+—+
|2022.0|105072|
|2020.0|660013|
|2021.0|664371|
+—+—+
[1] “___________________________”
```

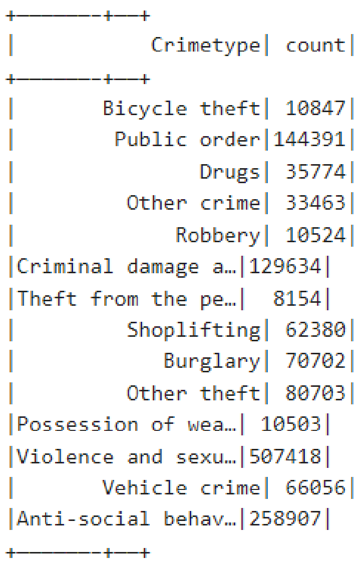
To be continued

DATA STAGING – TRANSFORMATION STAGE

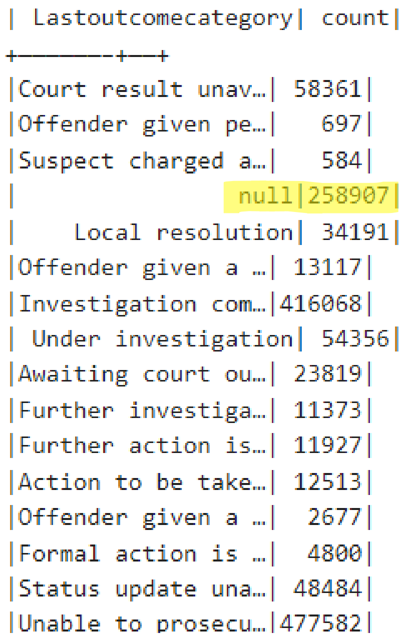
Output →



```
|Month| count|
+--+--+
|  8.0|118214|
|  7.0|121532|
|  1.0|158095|
|  4.0|103074|
| 11.0|110469|
|  3.0|110628|
|  2.0|149767|
| 10.0|117177|
|  6.0|114591|
|  5.0|111126|
|  9.0|113878|
| 12.0|100905|
+--+--+
```

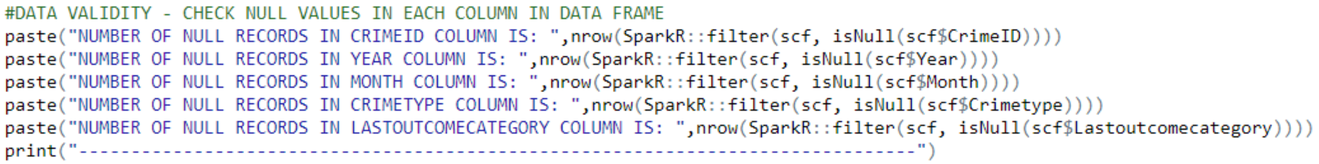


```
+-------+--+
|          Crimetype| count|
+-------+--+
|      Bicycle theft| 10847|
|       Public order|144391|
|              Drugs| 35774|
|        Other crime| 33463|
|            Robbery| 10524|
|Criminal damage a…|129634|
|Theft from the pe…|  8154|
|        Shoplifting| 62380|
|           Burglary| 70702|
|        Other theft| 80703|
|Possession of wea…| 10503|
|Violence and sexu…|507418|
|      Vehicle crime| 66056|
|Anti-social behav…|258907|
+-------+--+
```



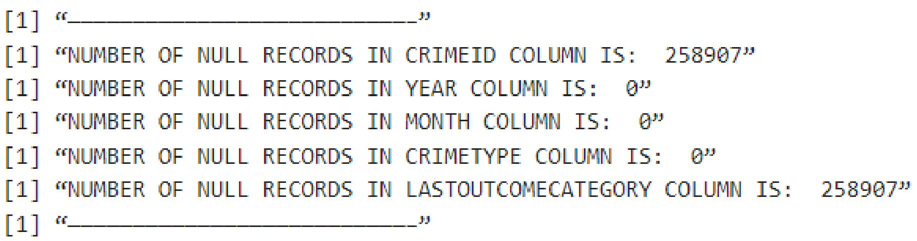
```
| Lastoutcomecategory| count|
+-------+--+
|Court result unav…| 58361|
|Offender given pe…|   697|
|Suspect charged a…|   584|
|                null|258907|
|    Local resolution| 34191|
|Offender given a …| 13117|
|Investigation com…|416068|
| Under investigation| 54356|
|Awaiting court ou…| 23819|
|Further investiga…| 11373|
|Further action is…| 11927|
|Action to be take…| 12513|
|Offender given a …|  2677|
|Formal action is …|  4800|
|Status update una…| 48484|
|Unable to prosecu…|477582|
```

Code →



```
#DATA VALIDITY - CHECK NULL VALUES IN EACH COLUMN IN DATA FRAME
paste("NUMBER OF NULL RECORDS IN CRIMEID COLUMN IS: ",nrow(SparkR::filter(scf, isNull(scf$CrimeID))))
paste("NUMBER OF NULL RECORDS IN YEAR COLUMN IS: ",nrow(SparkR::filter(scf, isNull(scf$Year))))
paste("NUMBER OF NULL RECORDS IN MONTH COLUMN IS: ",nrow(SparkR::filter(scf, isNull(scf$Month))))
paste("NUMBER OF NULL RECORDS IN CRIMETYPE COLUMN IS: ",nrow(SparkR::filter(scf, isNull(scf$Crimetype))))
paste("NUMBER OF NULL RECORDS IN LASTOUTCOMECATEGORY COLUMN IS: ",nrow(SparkR::filter(scf, isNull(scf$Lastoutcomecategory))))
print("--------------------------------------------------------------------------------")
```

Output →

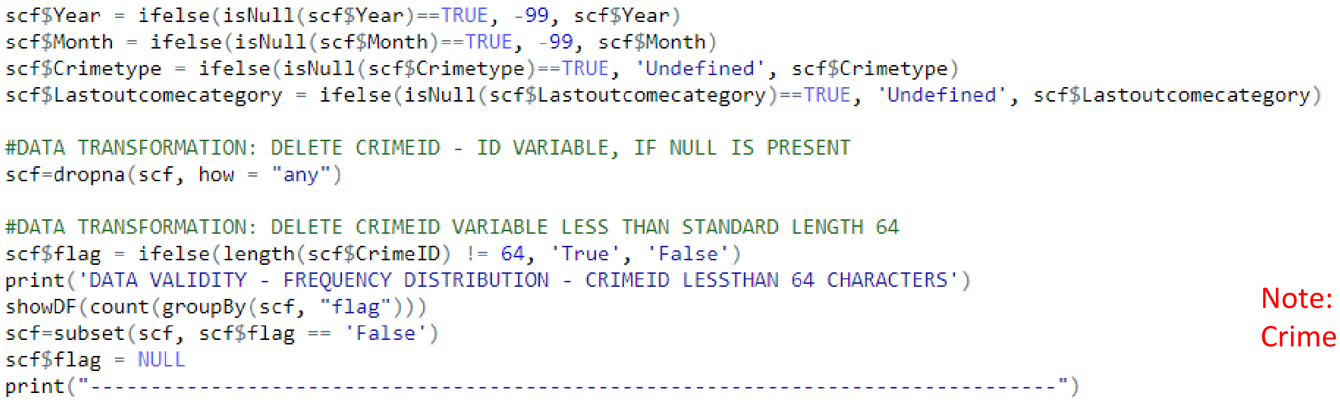


```
[1] “______________________________”
[1] “NUMBER OF NULL RECORDS IN CRIMEID COLUMN IS:  258907”
[1] “NUMBER OF NULL RECORDS IN YEAR COLUMN IS:  0”
[1] “NUMBER OF NULL RECORDS IN MONTH COLUMN IS:  0”
[1] “NUMBER OF NULL RECORDS IN CRIMETYPE COLUMN IS:  0”
[1] “NUMBER OF NULL RECORDS IN LASTOUTCOMECATEGORY COLUMN IS:  258907”
[1] “______________________________”
```

11.1 Data Transformation - Treatment:

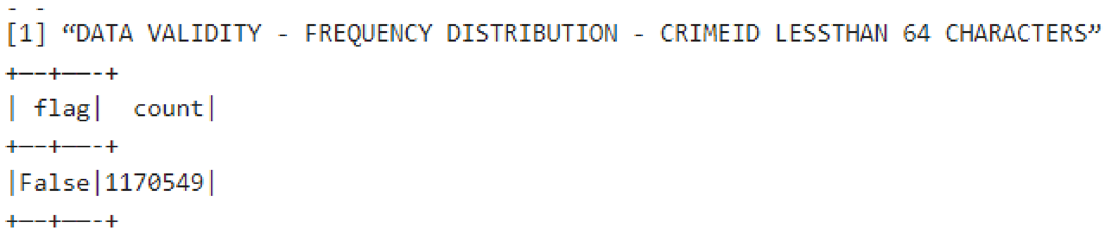
- **Numeric Column = Null to -99 value change**
- **Character column = Null to “Undefined”**
- **Delete ID columns if NULL and Crime ID length != 64 characters**

Code →



```
scf$Year = ifelse(isNull(scf$Year)==TRUE, -99, scf$Year)
scf$Month = ifelse(isNull(scf$Month)==TRUE, -99, scf$Month)
scf$Crimetype = ifelse(isNull(scf$Crimetype)==TRUE, 'Undefined', scf$Crimetype)
scf$Lastoutcomecategory = ifelse(isNull(scf$Lastoutcomecategory)==TRUE, 'Undefined', scf$Lastoutcomecategory)

#DATA TRANSFORMATION: DELETE CRIMEID - ID VARIABLE, IF NULL IS PRESENT
scf=dropna(scf, how = "any")

#DATA TRANSFORMATION: DELETE CRIMEID VARIABLE LESS THAN STANDARD LENGTH 64
scf$flag = ifelse(length(scf$CrimeID) != 64, 'True', 'False')
print('DATA VALIDITY - FREQUENCY DISTRIBUTION - CRIMEID LESSTHAN 64 CHARACTERS')
showDF(count(groupBy(scf, "flag")))
scf=subset(scf, scf$flag == 'False')
scf$flag = NULL
print("--------------------------------------------------------------------------------")
```

Output →



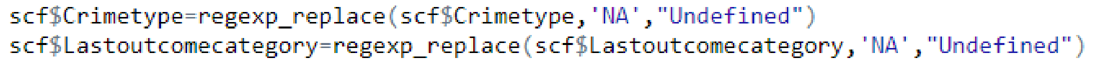
```
[1] “DATA VALIDITY - FREQUENCY DISTRIBUTION - CRIMEID LESSTHAN 64 CHARACTERS”
+--+---+
| flag|  count|
+--+---+
|False|1170549|
+--+---+
```

Note: Crime IDs with NULL has Lastoutcomecategory as NULL. Hence, After trea
CrimeID, we won’t have NULL in Lastoutcomecategory

DATA STAGING – TRANSFORMATION STAGE

11.2 Data Transformation - Treatment:

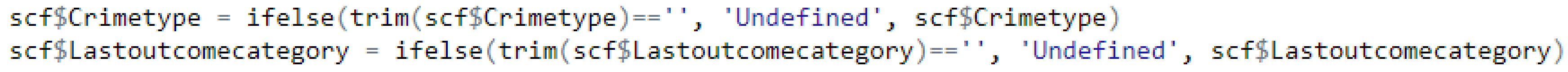
- Character column = ‘NA’ to “Undefined”
- Character column = Invalid values to “Undefined”



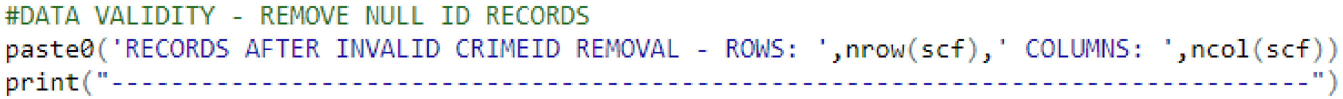
```
scf$Crimetype=regexp_replace(scf$Crimetype,'NA',"Undefined")
scf$Lastoutcomecategory=regexp_replace(scf$Lastoutcomecategory,'NA',"Undefined")
```

11.3 Data Transformation - Treatment:

- Character column = Blank values to “Undefined”

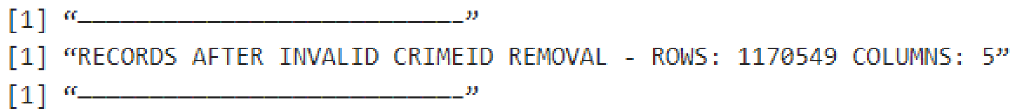


```
scf$Crimetype = ifelse(trim(scf$Crimetype)=='', 'Undefined', scf$Crimetype)
scf$Lastoutcomecategory = ifelse(trim(scf$Lastoutcomecategory)=='', 'Undefined', scf$Lastoutcomecategory)
```



```
#DATA VALIDITY - REMOVE NULL ID RECORDS
paste0('RECORDS AFTER INVALID CRIMEID REMOVAL - ROWS: ',nrow(scf),' COLUMNS: ',ncol(scf))
print("--------------------------------------------------------------------------------")
```

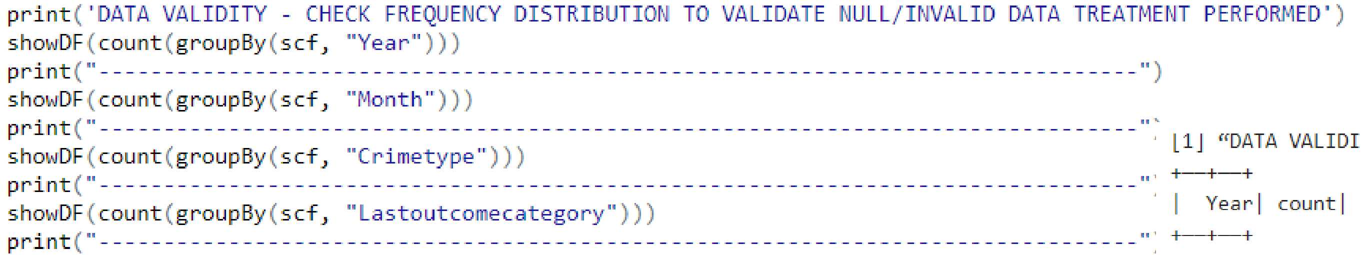
Output →



```
[1] “___________________________”
[1] “RECORDS AFTER INVALID CRIMEID REMOVAL - ROWS: 1170549 COLUMNS: 5”
[1] “___________________________”
```

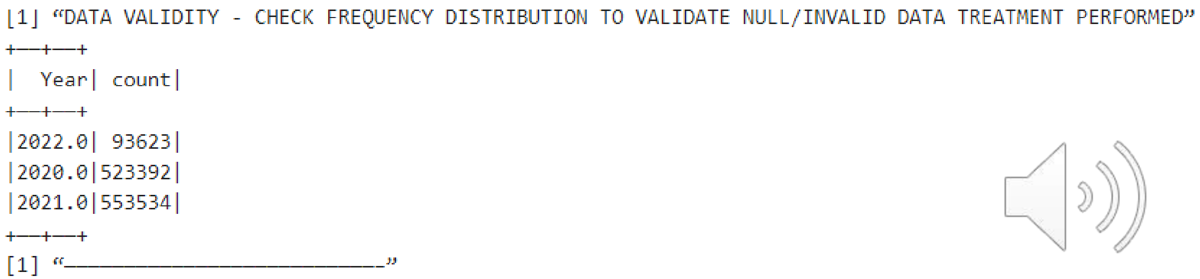
12. Data Validity - Frequency distribution on each columns to check columns value treatments applied

Code →



```
print('DATA VALIDITY - CHECK FREQUENCY DISTRIBUTION TO VALIDATE NULL/INVALID DATA TREATMENT PERFORMED')
showDF(count(groupBy(scf, "Year")))
print("--------------------------------------------------------------------------------")
showDF(count(groupBy(scf, "Month")))
print("--------------------------------------------------------------------------------"
showDF(count(groupBy(scf, "Crimetype")))
print("--------------------------------------------------------------------------------"
showDF(count(groupBy(scf, "Lastoutcomecategory")))
print("--------------------------------------------------------------------------------"
```

Output →

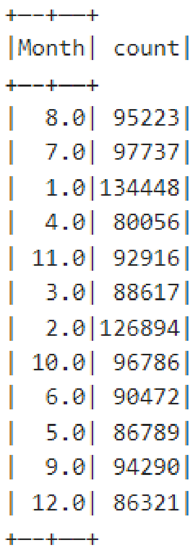


```
[1] “DATA VALIDITY - CHECK FREQUENCY DISTRIBUTION TO VALIDATE NULL/INVALID DATA TREATMENT PERFORMED”
+——+——+
|  Year| count|
+——+——+
|2022.0| 93623|
|2020.0|523392|
|2021.0|553534|
+——+——+
[1] “_________________________”
```

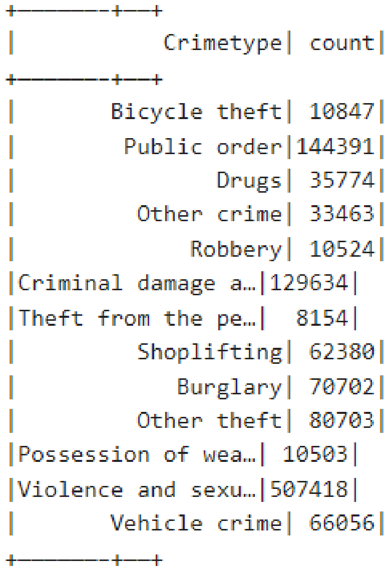
DATA STAGING – TRANSFORMATION STAGE

CRIME - STREET DATA

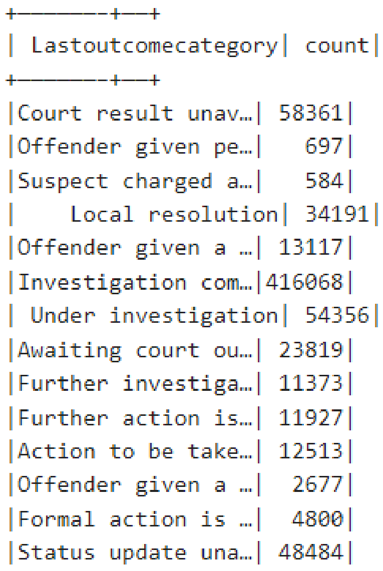
Output →



```
+----+------+
|Month| count|
+----+------+
|  8.0| 95223|
|  7.0| 97737|
|  1.0|134448|
|  4.0| 80056|
| 11.0| 92916|
|  3.0| 88617|
|  2.0|126894|
| 10.0| 96786|
|  6.0| 90472|
|  5.0| 86789|
|  9.0| 94290|
| 12.0| 86321|
+----+------+
```



```
+--------------------+------+
|           Crimetype| count|
+--------------------+------+
|       Bicycle theft| 10847|
|        Public order|144391|
|               Drugs| 35774|
|         Other crime| 33463|
|             Robbery| 10524|
|Criminal damage a...|129634|
|Theft from the pe...|  8154|
|         Shoplifting| 62380|
|            Burglary| 70702|
|         Other theft| 80703|
|Possession of wea...| 10503|
|Violence and sexu...|507418|
|       Vehicle crime| 66056|
+--------------------+------+
```

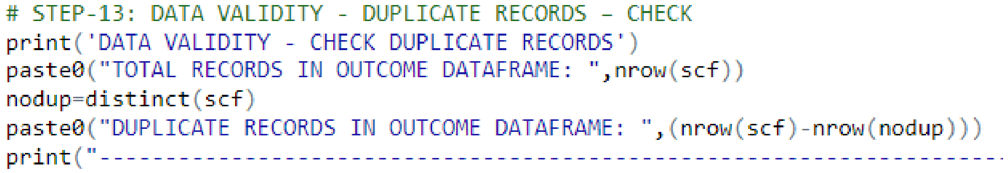


```
+--------------------+------+
| Lastoutcomecategory| count|
+--------------------+------+
|Court result unav...| 58361|
|Offender given pe...|   697|
|Suspect charged a...|   584|
|    Local resolution| 34191|
|Offender given a ...| 13117|
|Investigation com...|416068|
| Under investigation| 54356|
|Awaiting court ou...| 23819|
|Further investiga...| 11373|
|Further action is...| 11927|
|Action to be take...| 12513|
|Offender given a ...|  2677|
|Formal action is ...|  4800|
|Status update una...| 48484|
```

Note: Crime IDs with NULL has Lastoutcomecategory as NULL. Hence, After treating CrimeID, we won’t have NULL in Lastoutcomecategory

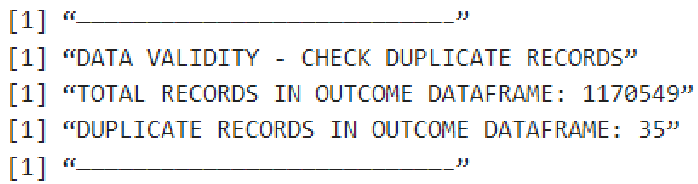
13. Data Validity - Duplicate Records - Check

Code →



```
# STEP-13: DATA VALIDITY - DUPLICATE RECORDS – CHECK
print('DATA VALIDITY - CHECK DUPLICATE RECORDS')
paste0("TOTAL RECORDS IN OUTCOME DATAFRAME: ",nrow(scf))
nodup=distinct(scf)
paste0("DUPLICATE RECORDS IN OUTCOME DATAFRAME: ",(nrow(scf)-nrow(nodup)))
print("------------------------------------------------------------------
```

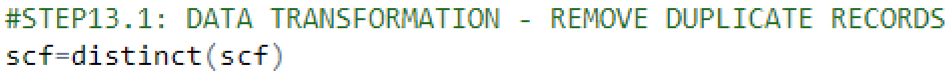
Output →



```
[1] “____________________________”
[1] “DATA VALIDITY - CHECK DUPLICATE RECORDS”
[1] “TOTAL RECORDS IN OUTCOME DATAFRAME: 1170549”
[1] “DUPLICATE RECORDS IN OUTCOME DATAFRAME: 35”
[1] “____________________________”
```

13.1 Data Transformation - Remove Duplicate Records

Code →



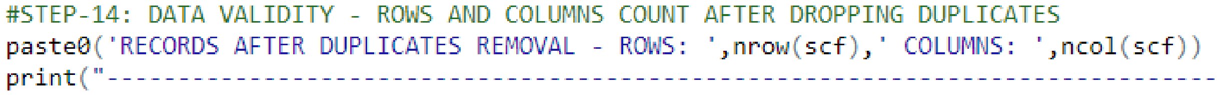
```
#STEP13.1: DATA TRANSFORMATION - REMOVE DUPLICATE RECORDS
scf=distinct(scf)
```

DATA STAGING – TRANSFORMATION STAGE

CRIME - STREET DATA

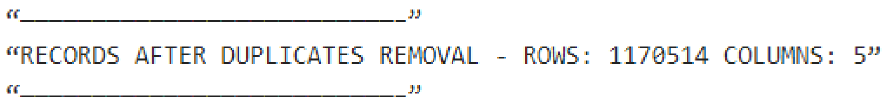
14. Data Validity - Rows and Columns Count After dropping duplicates

Code →



```
#STEP-14: DATA VALIDITY - ROWS AND COLUMNS COUNT AFTER DROPPING DUPLICATES
paste0('RECORDS AFTER DUPLICATES REMOVAL - ROWS: ',nrow(scf),' COLUMNS: ',ncol(scf))
print("--------------------------------------------------------------------------
```

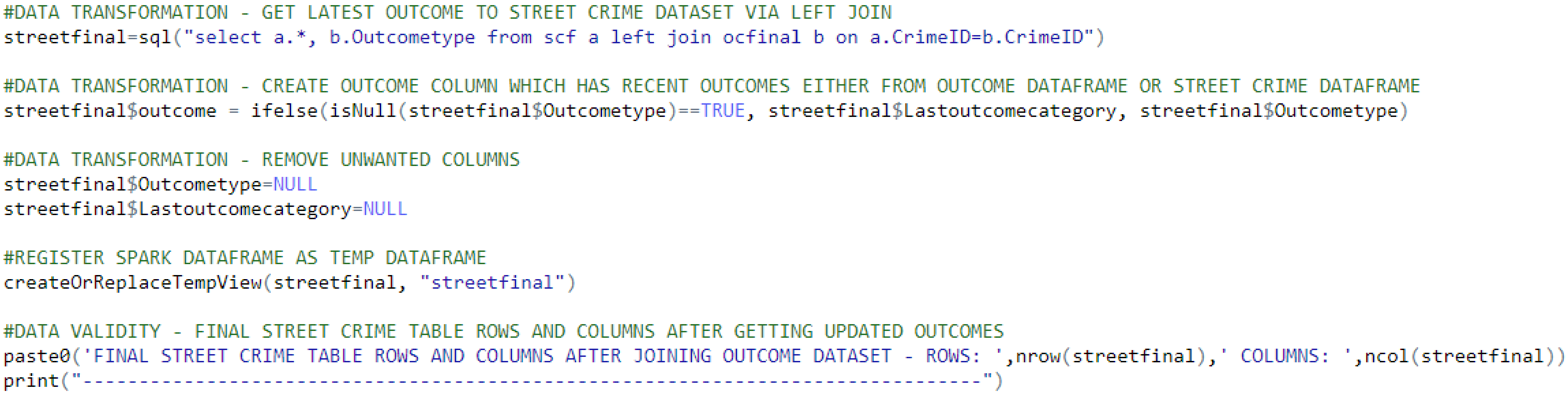
Output →



```
“____________________________”
“RECORDS AFTER DUPLICATES REMOVAL - ROWS: 1170514 COLUMNS: 5”
“____________________________”
```

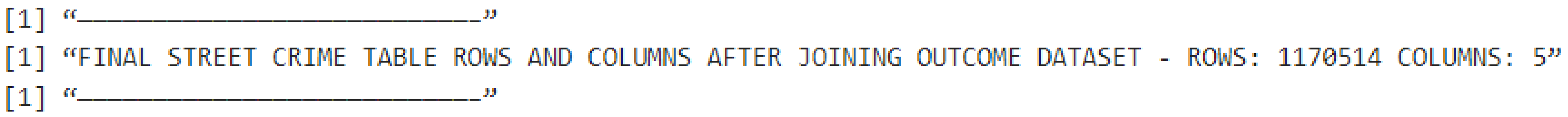
ADDITIONAL STEP: DATA TRANSFORMATION – UPDATING LATEST OUTCOME INFORMATION TO STREET CRIME DATASET FROM OUTCOME DATASET VIA JOINS / DATA VALIDITY – CHECKING ROWS AND COLUMNS

Code →



```
#DATA TRANSFORMATION - GET LATEST OUTCOME TO STREET CRIME DATASET VIA LEFT JOIN
streetfinal=sql("select a.*, b.Outcometype from scf a left join ocfinal b on a.CrimeID=b.CrimeID")

#DATA TRANSFORMATION - CREATE OUTCOME COLUMN WHICH HAS RECENT OUTCOMES EITHER FROM OUTCOME DATAFRAME OR STREET CRIME DATAFRAME
streetfinal$outcome = ifelse(isNull(streetfinal$Outcometype)==TRUE, streetfinal$Lastoutcomecategory, streetfinal$Outcometype)

#DATA TRANSFORMATION - REMOVE UNWANTED COLUMNS
streetfinal$Outcometype=NULL
streetfinal$Lastoutcomecategory=NULL

#REGISTER SPARK DATAFRAME AS TEMP DATAFRAME
createOrReplaceTempView(streetfinal, "streetfinal")

#DATA VALIDITY - FINAL STREET CRIME TABLE ROWS AND COLUMNS AFTER GETTING UPDATED OUTCOMES
paste0('FINAL STREET CRIME TABLE ROWS AND COLUMNS AFTER JOINING OUTCOME DATASET - ROWS: ',nrow(streetfinal),' COLUMNS: ',ncol(streetfinal))
print("--------------------------------------------------------------------------------")
```

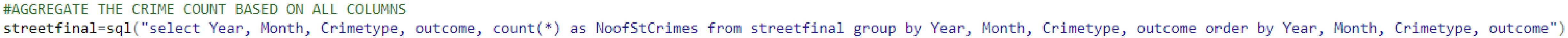
Output →



```
[1] “____________________________”
[1] “FINAL STREET CRIME TABLE ROWS AND COLUMNS AFTER JOINING OUTCOME DATASET - ROWS: 1170514 COLUMNS: 5”
[1] “____________________________”
```

DATA STAGING – TRANSFORMATION STAGE

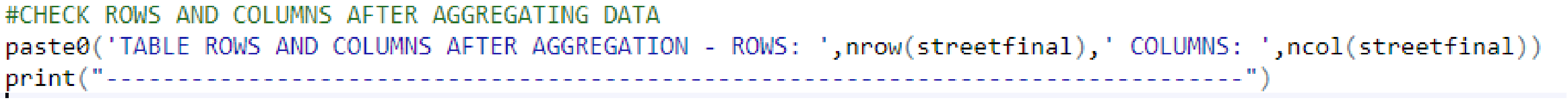
5. Data Transformation – Calculating aggregates based on group by variables



```
#AGGREGATE THE CRIME COUNT BASED ON ALL COLUMNS
streetfinal=sql("select Year, Month, Crimetype, outcome, count(*) as NoofStCrimes from streetfinal group by Year, Month, Crimetype, outcome order by Year, Month, Crimetype, outcome")
```

6. Data Validity - Rows and Columns count after aggregating by variables

Code →



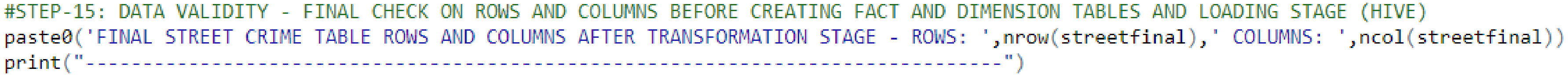
```
#CHECK ROWS AND COLUMNS AFTER AGGREGATING DATA
paste0('TABLE ROWS AND COLUMNS AFTER AGGREGATION - ROWS: ',nrow(streetfinal),' COLUMNS: ',ncol(streetfinal))
print("--------------------------------------------------------------------------")
```

Output →

```
[1] “TABLE ROWS AND COLUMNS AFTER AGGREGATION - ROWS: 3395 COLUMNS: 5”
[1] “___________________________”
```

15. Data Validity - Final Check on Rows and Columns before creating Fact and Dimension tables and Loading Stage (Hive)

Code →



```
#STEP-15: DATA VALIDITY - FINAL CHECK ON ROWS AND COLUMNS BEFORE CREATING FACT AND DIMENSION TABLES AND LOADING STAGE (HIVE)
paste0('FINAL STREET CRIME TABLE ROWS AND COLUMNS AFTER TRANSFORMATION STAGE - ROWS: ',nrow(streetfinal),' COLUMNS: ',ncol(streetfinal))
print("--------------------------------------------------------------------")
```

Output →

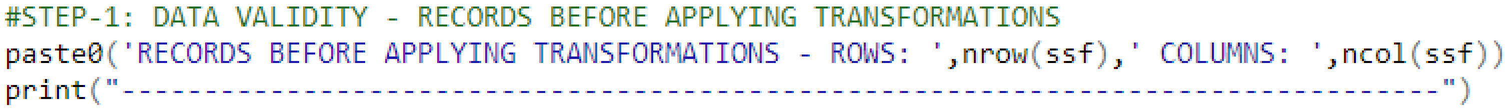
```
[1] “_______________________________”
[1] “FINAL STREET CRIME TABLE ROWS AND COLUMNS AFTER TRANSFORMATION STAGE - ROWS: 3395 COLUMNS: 5”
[1] “_______________________________”
```

DATA STAGING – TRANSFORMATION STAGE

CRIME - STOP SEARCH DATA

1. Data Validity - Rows and Columns Count before Transformation

Code →



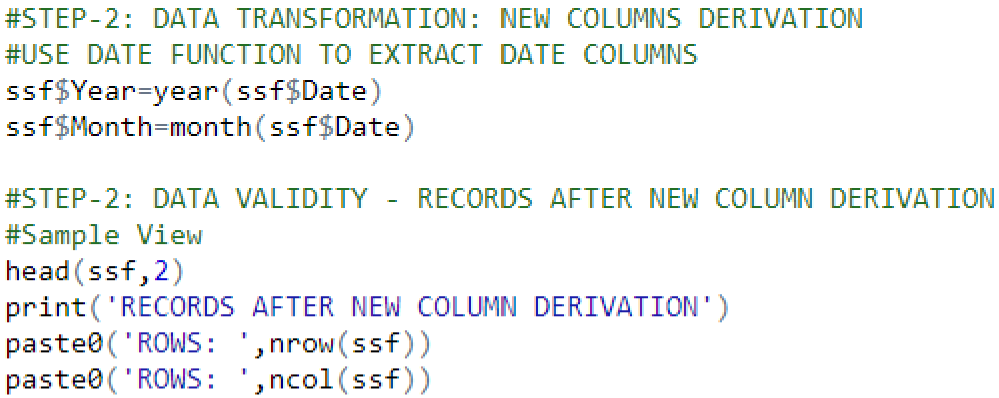
```
#STEP-1: DATA VALIDITY - RECORDS BEFORE APPLYING TRANSFORMATIONS
paste0('RECORDS BEFORE APPLYING TRANSFORMATIONS - ROWS: ',nrow(ssf),' COLUMNS: ',ncol(ssf))
print("----------------------------------------------------------------------------------")
```

Output →

```
"RECORDS BEFORE APPLYING TRANSFORMATIONS - ROWS: 96163 COLUMNS: 16"
"______________________________"
```

2. Data Transformation - New Columns Derivation

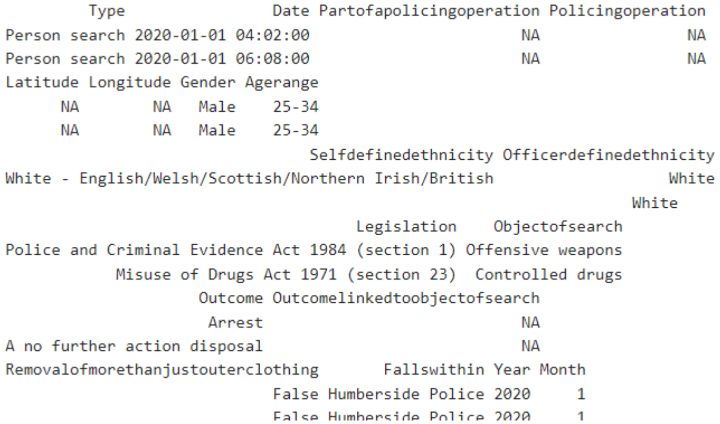
Code →



```
#STEP-2: DATA TRANSFORMATION: NEW COLUMNS DERIVATION
#USE DATE FUNCTION TO EXTRACT DATE COLUMNS
ssf$Year=year(ssf$Date)
ssf$Month=month(ssf$Date)

#STEP-2: DATA VALIDITY - RECORDS AFTER NEW COLUMN DERIVATION
#Sample View
head(ssf,2)
print('RECORDS AFTER NEW COLUMN DERIVATION')
paste0('ROWS: ',nrow(ssf))
paste0('ROWS: ',ncol(ssf))
```

Output →

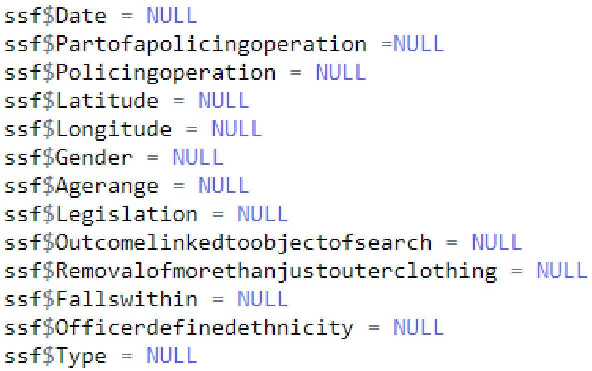
```
"RECORDS AFTER NEW COLUMN DERIVATION"
"ROWS: 96163"
"ROWS: 18"
```



```
         Type                Date Partofapolicingoperation Policingoperation
Person search 2020-01-01 04:02:00                       NA                NA
Person search 2020-01-01 06:08:00                       NA                NA
Latitude Longitude Gender Agerange
      NA        NA   Male    25-34
      NA        NA   Male    25-34
                                 Selfdefinedethnicity Officerdefinedethnicity
White - English/Welsh/Scottish/Northern Irish/British                   White
                                                                White
                                          Legislation    Objectofsearch
Police and Criminal Evidence Act 1984 (section 1) Offensive weapons
            Misuse of Drugs Act 1971 (section 23)  Controlled drugs
                    Outcome Outcomelinkedtoobjectofsearch
                     Arrest                            NA
A no further action disposal                           NA
Removalofmorethanjustouterclothing       Fallswithin Year Month
                              False Humberside Police 2020     1
                              False Humberside Police 2020     1
```

3. Data Transformation - Remove unwanted columns

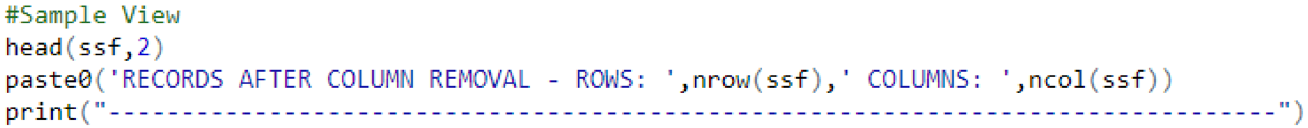
Code →



```
ssf$Date = NULL
ssf$Partofapolicingoperation =NULL
ssf$Policingoperation = NULL
ssf$Latitude = NULL
ssf$Longitude = NULL
ssf$Gender = NULL
ssf$Agerange = NULL
ssf$Legislation = NULL
ssf$Outcomelinkedtoobjectofsearch = NULL
ssf$Removalofmorethanjustouterclothing = NULL
ssf$Fallswithin = NULL
ssf$Officerdefinedethnicity = NULL
ssf$Type = NULL
```

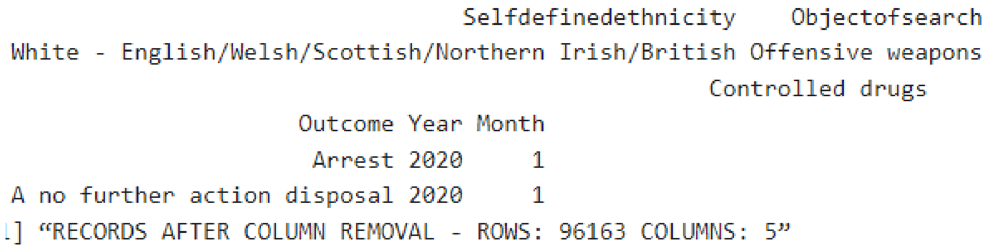
4. Data Validity - Rows and Columns count after columns removal

Code →



```
#Sample View
head(ssf,2)
paste0('RECORDS AFTER COLUMN REMOVAL - ROWS: ',nrow(ssf),' COLUMNS: ',ncol(ssf))
print("----------------------------------------------------------------------------------")
```

Output →



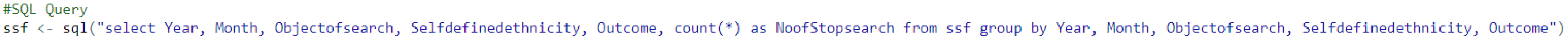
```
                               Selfdefinedethnicity    Objectofsearch
White - English/Welsh/Scottish/Northern Irish/British Offensive weapons
                                                       Controlled drugs
                      Outcome Year Month
                       Arrest 2020     1
A no further action disposal 2020     1
1] "RECORDS AFTER COLUMN REMOVAL - ROWS: 96163 COLUMNS: 5"
```

DATA STAGING – TRANSFORMATION STAGE

CRIME - STOP SEARCH DATA

5. Data Transformation – Calculating aggregates based on group by variables

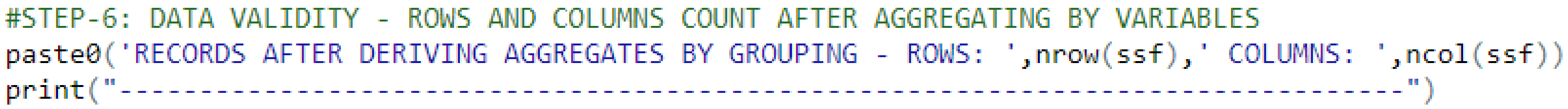
Code →



```
#SQL Query
ssf <- sql("select Year, Month, Objectofsearch, Selfdefinedethnicity, Outcome, count(*) as NoofStopsearch from ssf group by Year, Month, Objectofsearch, Selfdefinedethnicity, Outcome")
```

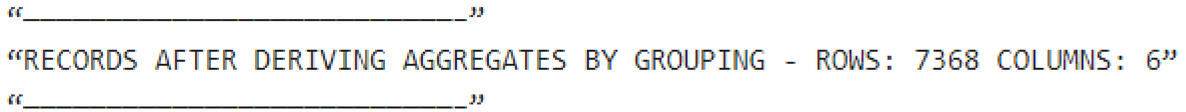
6. Data Validity - Rows and Columns count after aggregating by variables

Code →



```
#STEP-6: DATA VALIDITY - ROWS AND COLUMNS COUNT AFTER AGGREGATING BY VARIABLES
paste0('RECORDS AFTER DERIVING AGGREGATES BY GROUPING - ROWS: ',nrow(ssf),' COLUMNS: ',ncol(ssf))
print("------------------------------------------------------------------------")
```

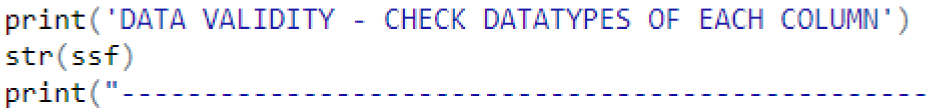
Output →



```
"________________________________"
"RECORDS AFTER DERIVING AGGREGATES BY GROUPING - ROWS: 7368 COLUMNS: 6"
"________________________________"
```

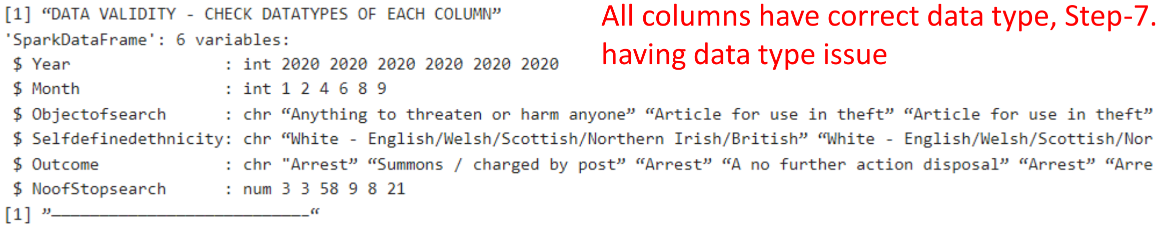
7. Data Validity - Data Type Check on each column - Before

Code →



```
print('DATA VALIDITY - CHECK DATATYPES OF EACH COLUMN')
str(ssf)
print("-----------------------------------------
```

Output →

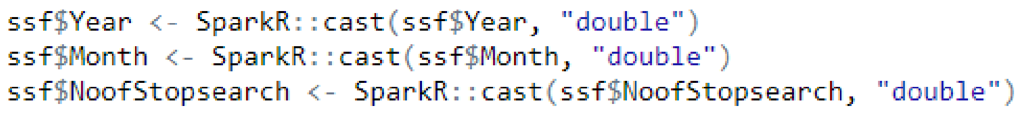


```
[1] "DATA VALIDITY - CHECK DATATYPES OF EACH COLUMN"
'SparkDataFrame': 6 variables:
 $ Year                : int 2020 2020 2020 2020 2020 2020
 $ Month               : int 1 2 4 6 8 9
 $ Objectofsearch      : chr "Anything to threaten or harm anyone" "Article for use in theft" "Article for use in theft"
 $ Selfdefinedethnicity: chr "White - English/Welsh/Scottish/Northern Irish/British" "White - English/Welsh/Scottish/Nor
 $ Outcome             : chr "Arrest" "Summons / charged by post" "Arrest" "A no further action disposal" "Arrest" "Arre
 $ NoofStopsearch      : num 3 3 58 9 8 21
[1] "___________________________"
```

All columns have correct data type, Step-7.1 will handle if any future data is having data type issue

7.1 Data Transformation - Convert to appropriate column datatype (CAST)

Code →



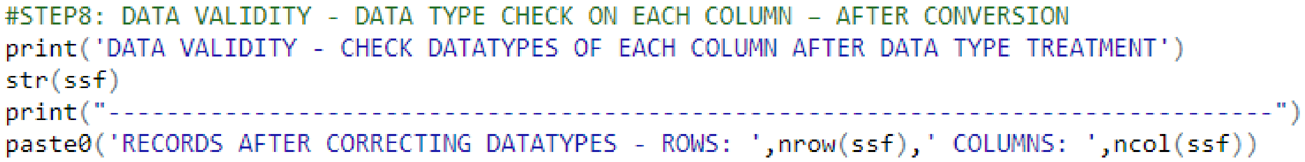
```
ssf$Year <- SparkR::cast(ssf$Year, "double")
ssf$Month <- SparkR::cast(ssf$Month, "double")
ssf$NoofStopsearch <- SparkR::cast(ssf$NoofStopsearch, "double")
```

DATA STAGING – TRANSFORMATION STAGE

CRIME - STOP SEARCH DATA

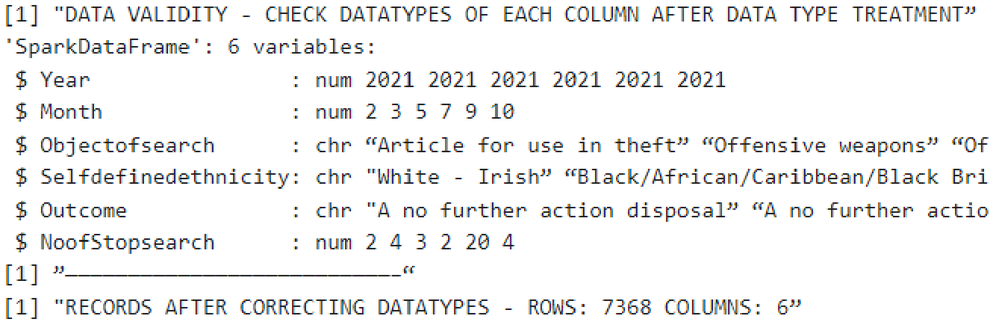
8. Data Validity - Data Type Check on each column – After conversion

Code →



```
#STEP8: DATA VALIDITY - DATA TYPE CHECK ON EACH COLUMN – AFTER CONVERSION
print('DATA VALIDITY - CHECK DATATYPES OF EACH COLUMN AFTER DATA TYPE TREATMENT')
str(ssf)
print("------------------------------------------------------------------------------")
paste0('RECORDS AFTER CORRECTING DATATYPES - ROWS: ',nrow(ssf),' COLUMNS: ',ncol(ssf))
```

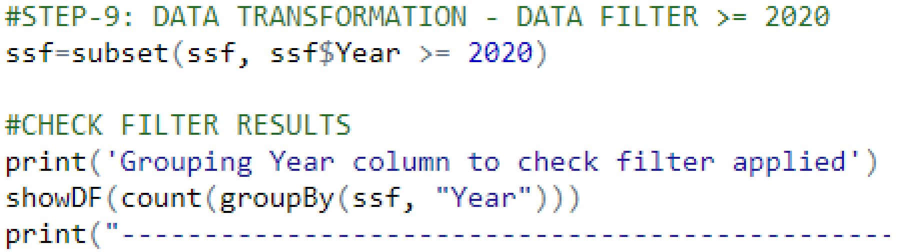
Output →



```
[1] "DATA VALIDITY - CHECK DATATYPES OF EACH COLUMN AFTER DATA TYPE TREATMENT”
'SparkDataFrame': 6 variables:
 $ Year                 : num 2021 2021 2021 2021 2021 2021
 $ Month                : num 2 3 5 7 9 10
 $ Objectofsearch       : chr “Article for use in theft” “Offensive weapons” “Of
 $ Selfdefinedethnicity: chr "White - Irish” “Black/African/Caribbean/Black Bri
 $ Outcome              : chr "A no further action disposal” “A no further actio
 $ NoofStopsearch       : num 2 4 3 2 20 4
[1] ”___________________________“
[1] "RECORDS AFTER CORRECTING DATATYPES - ROWS: 7368 COLUMNS: 6”
```

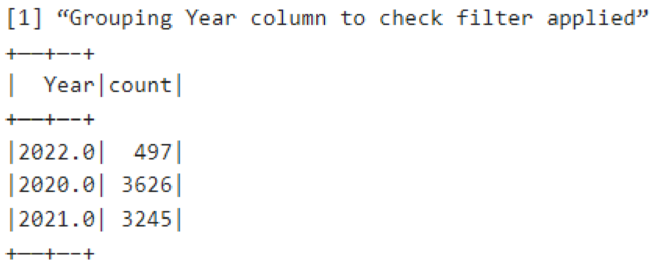
9. Data Transformation - Data filter >= 2020

Code →



```
#STEP-9: DATA TRANSFORMATION - DATA FILTER >= 2020
ssf=subset(ssf, ssf$Year >= 2020)

#CHECK FILTER RESULTS
print('Grouping Year column to check filter applied')
showDF(count(groupBy(ssf, "Year")))
print("----------------------------------------------------
```

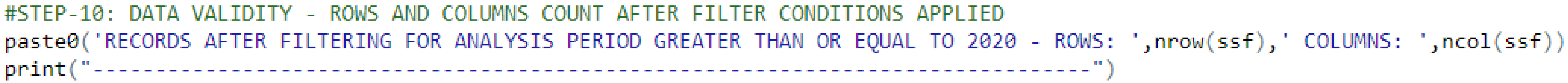
Output →



```
[1] “Grouping Year column to check filter applied”
+——+——+
|  Year|count|
+——+——+
|2022.0|  497|
|2020.0| 3626|
|2021.0| 3245|
+——+——+
```

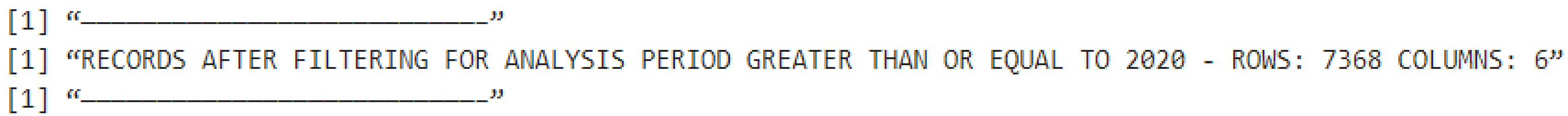
10. Data Validity - Rows and Columns Count after filter conditions applied

Code →



```
#STEP-10: DATA VALIDITY - ROWS AND COLUMNS COUNT AFTER FILTER CONDITIONS APPLIED
paste0('RECORDS AFTER FILTERING FOR ANALYSIS PERIOD GREATER THAN OR EQUAL TO 2020 - ROWS: ',nrow(ssf),' COLUMNS: ',ncol(ssf))
print("--------------------------------------------------------------------")
```

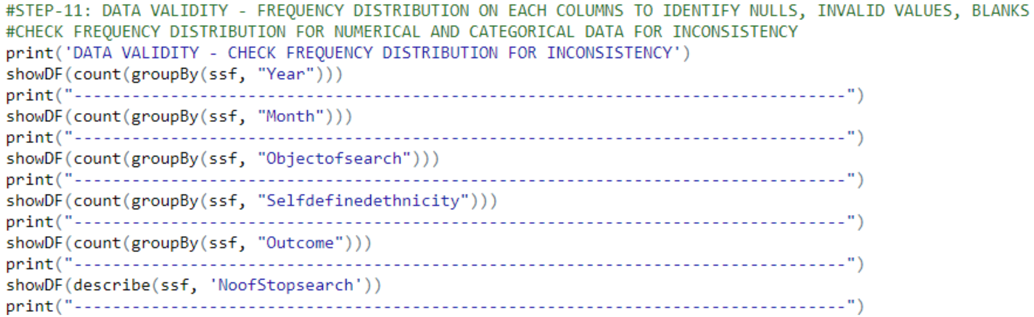
Output →



```
[1] “__________________________”
[1] “RECORDS AFTER FILTERING FOR ANALYSIS PERIOD GREATER THAN OR EQUAL TO 2020 - ROWS: 7368 COLUMNS: 6”
[1] “__________________________”
```

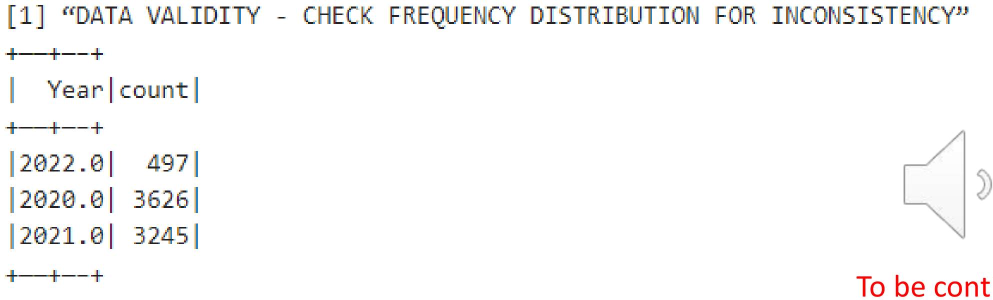
11. Data Validity - Frequency distribution on each columns to identify Nulls, Invalid values, Blanks

Code →



```
#STEP-11: DATA VALIDITY - FREQUENCY DISTRIBUTION ON EACH COLUMNS TO IDENTIFY NULLS, INVALID VALUES, BLANKS
#CHECK FREQUENCY DISTRIBUTION FOR NUMERICAL AND CATEGORICAL DATA FOR INCONSISTENCY
print('DATA VALIDITY - CHECK FREQUENCY DISTRIBUTION FOR INCONSISTENCY')
showDF(count(groupBy(ssf, "Year")))
print("------------------------------------------------------------------------")
showDF(count(groupBy(ssf, "Month")))
print("------------------------------------------------------------------------")
showDF(count(groupBy(ssf, "Objectofsearch")))
print("------------------------------------------------------------------------")
showDF(count(groupBy(ssf, "Selfdefinedethnicity")))
print("------------------------------------------------------------------------")
showDF(count(groupBy(ssf, "Outcome")))
print("------------------------------------------------------------------------")
showDF(describe(ssf, 'NoofStopsearch'))
print("------------------------------------------------------------------------")
```

Output →

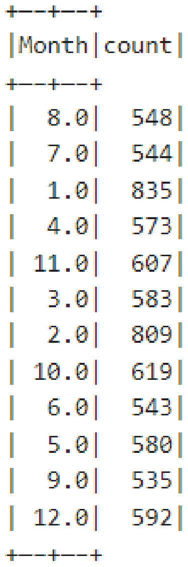


```
[1] “DATA VALIDITY - CHECK FREQUENCY DISTRIBUTION FOR INCONSISTENCY”
+——+——+
|  Year|count|
+——+——+
|2022.0|  497|
|2020.0| 3626|
|2021.0| 3245|
+——+——+
```

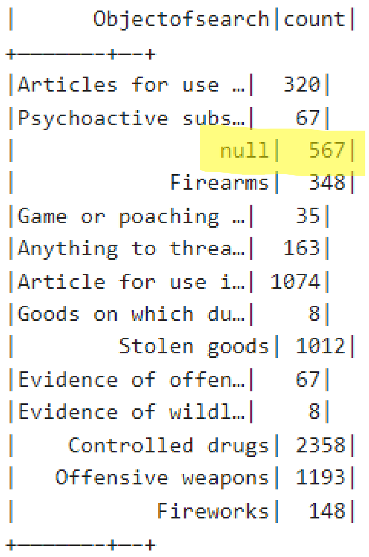
To be continued

DATA STAGING – TRANSFORMATION STAGE

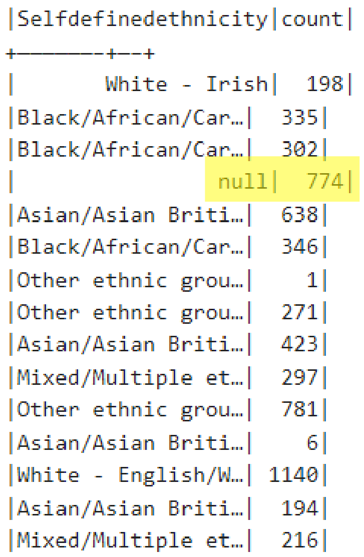
Output →



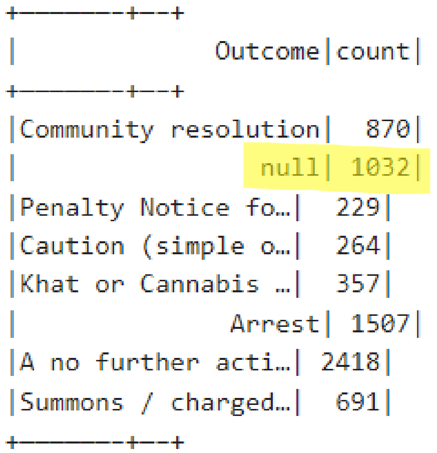
```
+----+-----+
|Month|count|
+----+-----+
|  8.0|  548|
|  7.0|  544|
|  1.0|  835|
|  4.0|  573|
| 11.0|  607|
|  3.0|  583|
|  2.0|  809|
| 10.0|  619|
|  6.0|  543|
|  5.0|  580|
|  9.0|  535|
| 12.0|  592|
+----+-----+
```



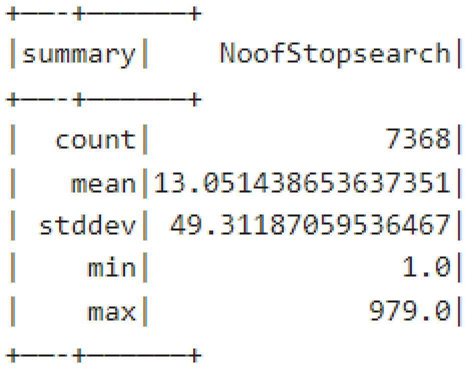
```
|       Objectofsearch|count|
+--------------------+-----+
|Articles for use ...|  320|
|Psychoactive subs...|   67|
|                null|  567|
|            Firearms|  348|
|Game or poaching ...|   35|
|Anything to threa...|  163|
|Article for use i...| 1074|
|Goods on which du...|    8|
|        Stolen goods| 1012|
|Evidence of offen...|   67|
|Evidence of wildl...|    8|
|     Controlled drugs| 2358|
|    Offensive weapons| 1193|
|           Fireworks|  148|
+--------------------+-----+
```



```
|Selfdefinedethnicity|count|
+--------------------+-----+
|      White - Irish|  198|
|Black/African/Car...|  335|
|Black/African/Car...|  302|
|                null|  774|
|Asian/Asian Briti...|  638|
|Black/African/Car...|  346|
|Other ethnic grou...|    1|
|Other ethnic grou...|  271|
|Asian/Asian Briti...|  423|
|Mixed/Multiple et...|  297|
|Other ethnic grou...|  781|
|Asian/Asian Briti...|    6|
|White - English/W...| 1140|
|Asian/Asian Briti...|  194|
|Mixed/Multiple et...|  216|
```

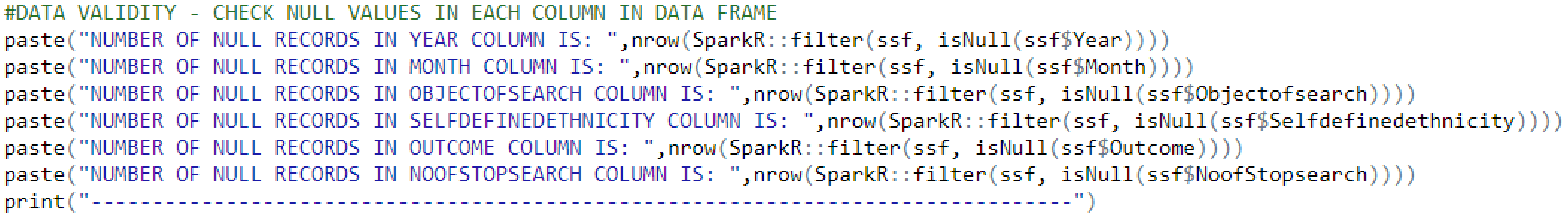


```
+--------------------+-----+
|             Outcome|count|
+--------------------+-----+
|Community resolution|  870|
|                null| 1032|
|Penalty Notice fo...|  229|
|Caution (simple o...|  264|
|Khat or Cannabis ...|  357|
|              Arrest| 1507|
|A no further acti...| 2418|
|Summons / charged...|  691|
+--------------------+-----+
```



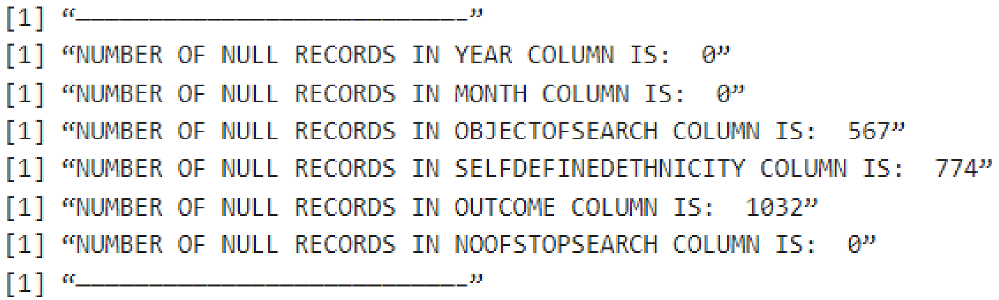
```
+-------+------------------+
|summary|    NoofStopsearch|
+-------+------------------+
|  count|              7368|
|   mean|13.051438653637351|
| stddev| 49.31187059536467|
|    min|               1.0|
|    max|             979.0|
+-------+------------------+
```

Code →



```
#DATA VALIDITY - CHECK NULL VALUES IN EACH COLUMN IN DATA FRAME
paste("NUMBER OF NULL RECORDS IN YEAR COLUMN IS: ",nrow(SparkR::filter(ssf, isNull(ssf$Year))))
paste("NUMBER OF NULL RECORDS IN MONTH COLUMN IS: ",nrow(SparkR::filter(ssf, isNull(ssf$Month))))
paste("NUMBER OF NULL RECORDS IN OBJECTOFSEARCH COLUMN IS: ",nrow(SparkR::filter(ssf, isNull(ssf$Objectofsearch))))
paste("NUMBER OF NULL RECORDS IN SELFDEFINEDETHNICITY COLUMN IS: ",nrow(SparkR::filter(ssf, isNull(ssf$Selfdefinedethnicity))))
paste("NUMBER OF NULL RECORDS IN OUTCOME COLUMN IS: ",nrow(SparkR::filter(ssf, isNull(ssf$Outcome))))
paste("NUMBER OF NULL RECORDS IN NOOFSTOPSEARCH COLUMN IS: ",nrow(SparkR::filter(ssf, isNull(ssf$NoofStopsearch))))
print("-------------------------------------------------------------------------------")
```

Output →



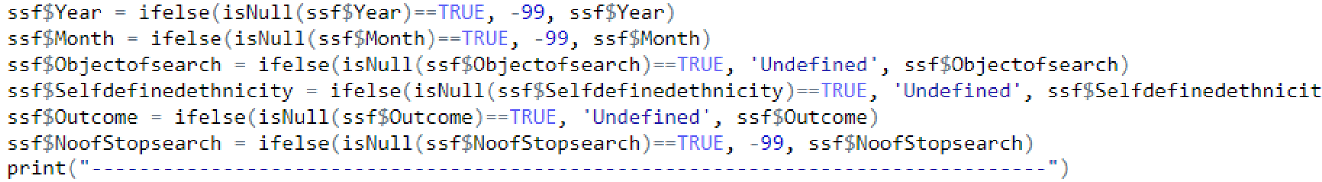
```
[1] "____________________________"
[1] "NUMBER OF NULL RECORDS IN YEAR COLUMN IS:  0"
[1] "NUMBER OF NULL RECORDS IN MONTH COLUMN IS:  0"
[1] "NUMBER OF NULL RECORDS IN OBJECTOFSEARCH COLUMN IS:  567"
[1] "NUMBER OF NULL RECORDS IN SELFDEFINEDETHNICITY COLUMN IS:  774"
[1] "NUMBER OF NULL RECORDS IN OUTCOME COLUMN IS:  1032"
[1] "NUMBER OF NULL RECORDS IN NOOFSTOPSEARCH COLUMN IS:  0"
[1] "____________________________"
```

DATA STAGING – TRANSFORMATION STAGE

CRIME - STOP SEARCH DATA

11.1 Data Transformation - Treatment:

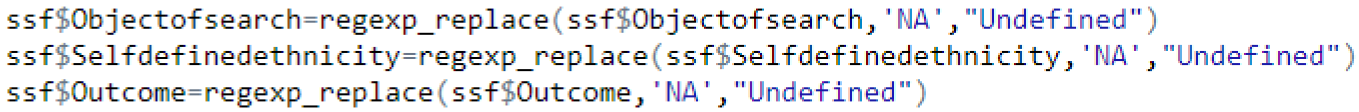
- **Numeric Column = Null to -99 value change**
- **Character column = Null to “Undefined”**
- **Delete ID columns if NULL and Crime ID length != 64 characters**



```
ssf$Year = ifelse(isNull(ssf$Year)==TRUE, -99, ssf$Year)
ssf$Month = ifelse(isNull(ssf$Month)==TRUE, -99, ssf$Month)
ssf$Objectofsearch = ifelse(isNull(ssf$Objectofsearch)==TRUE, 'Undefined', ssf$Objectofsearch)
ssf$Selfdefinedethnicity = ifelse(isNull(ssf$Selfdefinedethnicity)==TRUE, 'Undefined', ssf$Selfdefinedethnicity)
ssf$Outcome = ifelse(isNull(ssf$Outcome)==TRUE, 'Undefined', ssf$Outcome)
ssf$NoofStopsearch = ifelse(isNull(ssf$NoofStopsearch)==TRUE, -99, ssf$NoofStopsearch)
print("-----------------------------------------------------------------------------")
```

11.2 Data Transformation - Treatment:

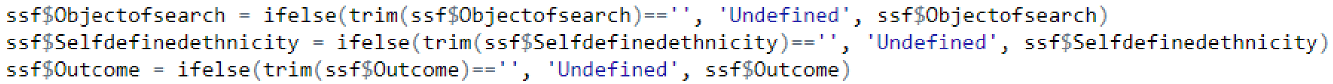
- **Character column = ‘NA’ to “Undefined”**
- **Character column = Invalid values to “Undefined”**



```
ssf$Objectofsearch=regexp_replace(ssf$Objectofsearch,'NA',"Undefined")
ssf$Selfdefinedethnicity=regexp_replace(ssf$Selfdefinedethnicity,'NA',"Undefined")
ssf$Outcome=regexp_replace(ssf$Outcome,'NA',"Undefined")
```

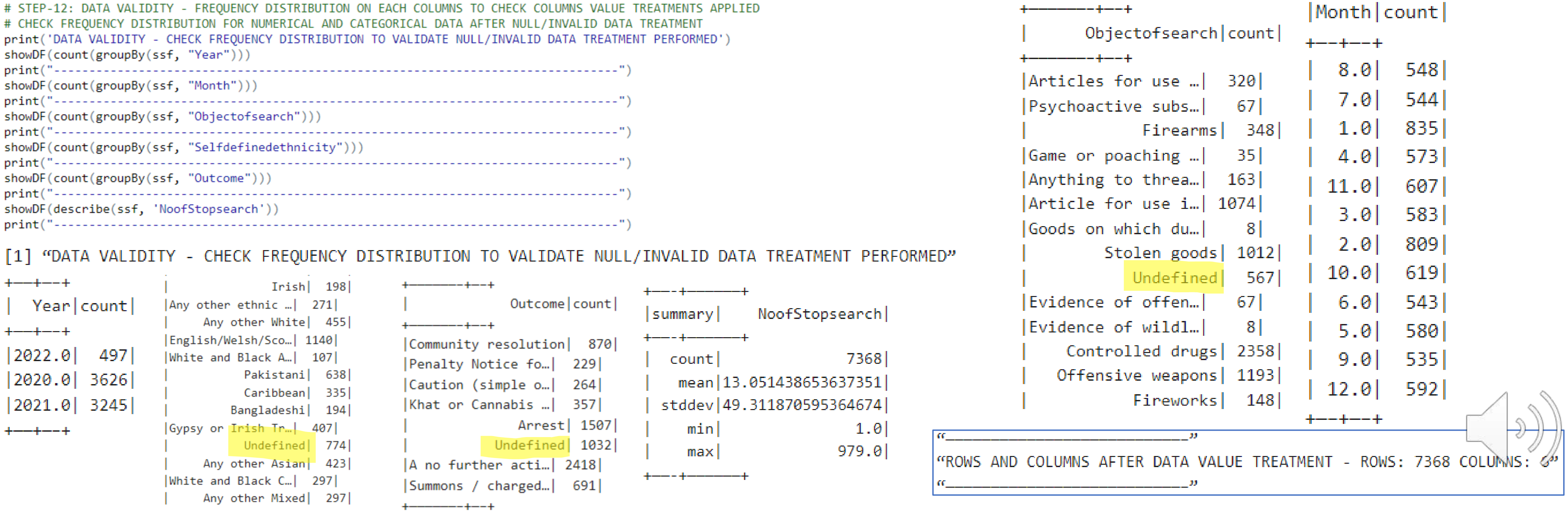
11.3 Data Transformation - Treatment:

- **Character column = Blank values to “Undefined”**



```
ssf$Objectofsearch = ifelse(trim(ssf$Objectofsearch)=='', 'Undefined', ssf$Objectofsearch)
ssf$Selfdefinedethnicity = ifelse(trim(ssf$Selfdefinedethnicity)=='', 'Undefined', ssf$Selfdefinedethnicity)
ssf$Outcome = ifelse(trim(ssf$Outcome)=='', 'Undefined', ssf$Outcome)
```

12. Data Validity - Frequency distribution on each columns to check columns value treatments applied



Code →

```
# STEP-12: DATA VALIDITY - FREQUENCY DISTRIBUTION ON EACH COLUMNS TO CHECK COLUMNS VALUE TREATMENTS APPLIED
# CHECK FREQUENCY DISTRIBUTION FOR NUMERICAL AND CATEGORICAL DATA AFTER NULL/INVALID DATA TREATMENT
print('DATA VALIDITY - CHECK FREQUENCY DISTRIBUTION TO VALIDATE NULL/INVALID DATA TREATMENT PERFORMED')
showDF(count(groupBy(ssf, "Year")))
print("----------------------------------------------------------------------")
showDF(count(groupBy(ssf, "Month")))
print("----------------------------------------------------------------------")
showDF(count(groupBy(ssf, "Objectofsearch")))
print("----------------------------------------------------------------------")
showDF(count(groupBy(ssf, "Selfdefinedethnicity")))
print("----------------------------------------------------------------------")
showDF(count(groupBy(ssf, "Outcome")))
print("----------------------------------------------------------------------")
showDF(describe(ssf, 'NoofStopsearch'))
print("----------------------------------------------------------------------")
```

Output →

```
[1] “DATA VALIDITY - CHECK FREQUENCY DISTRIBUTION TO VALIDATE NULL/INVALID DATA TREATMENT PERFORMED”

+---+---+
|   Year|count|
+---+---+
|2022.0|  497|
|2020.0| 3626|
|2021.0| 3245|
+---+---+
```

```
|               Irish|  198|
|Any other ethnic …|  271|
|     Any other White|  455|
|English/Welsh/Sco…| 1140|
|White and Black A…|  107|
|           Pakistani|  638|
|           Caribbean|  335|
|         Bangladeshi|  194|
|Gypsy or Irish Tr…|  407|
|           Undefined|  774|
|     Any other Asian|  423|
|White and Black C…|  297|
|     Any other Mixed|  297|
```

```
+--------+---+
|             Outcome|count|
+--------+---+
|Community resolution|  870|
|Penalty Notice fo…|  229|
|Caution (simple o…|  264|
|Khat or Cannabis …|  357|
|              Arrest| 1507|
|           Undefined| 1032|
|A no further acti…| 2418|
|Summons / charged…|  691|
+--------+---+
```

```
+---+-------+
|summary|    NoofStopsearch|
+---+-------+
|  count|              7368|
|   mean|13.051438653637351|
| stddev|49.311870595364674|
|    min|               1.0|
|    max|             979.0|
+---+-------+
```

```
+--------+---+
|      Objectofsearch|count|
+--------+---+
|Articles for use …|  320|
|Psychoactive subs…|   67|
|            Firearms|  348|
|Game or poaching …|   35|
|Anything to threa…|  163|
|Article for use i…| 1074|
|Goods on which du…|    8|
|        Stolen goods| 1012|
|           Undefined|  567|
|Evidence of offen…|   67|
|Evidence of wildl…|    8|
|    Controlled drugs| 2358|
|   Offensive weapons| 1193|
|           Fireworks|  148|
```

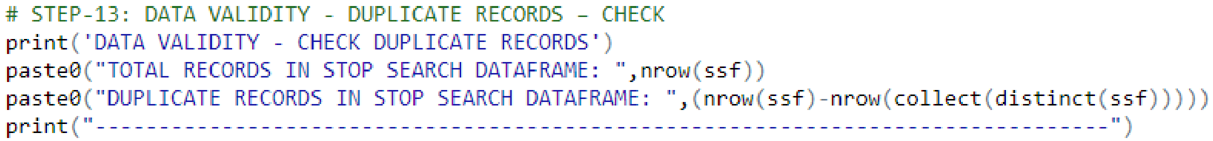
```
|Month|count|
+---+---+
|  8.0|  548|
|  7.0|  544|
|  1.0|  835|
|  4.0|  573|
| 11.0|  607|
|  3.0|  583|
|  2.0|  809|
| 10.0|  619|
|  6.0|  543|
|  5.0|  580|
|  9.0|  535|
| 12.0|  592|
+---+---+
```

```
“_______________________________”
“ROWS AND COLUMNS AFTER DATA VALUE TREATMENT - ROWS: 7368 COLUMNS: 6”
“_______________________________”
```

DATA STAGING – TRANSFORMATION STAGE

13. Data Validity - Duplicate Records - Check

Code →



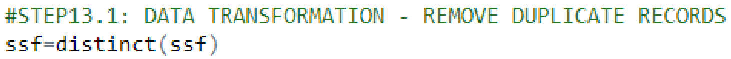
```
# STEP-13: DATA VALIDITY - DUPLICATE RECORDS – CHECK
print('DATA VALIDITY - CHECK DUPLICATE RECORDS')
paste0("TOTAL RECORDS IN STOP SEARCH DATAFRAME: ",nrow(ssf))
paste0("DUPLICATE RECORDS IN STOP SEARCH DATAFRAME: ",(nrow(ssf)-nrow(collect(distinct(ssf)))))
print("--------------------------------------------------------------------------------")
```

Output →

```
“______________________________”
“DATA VALIDITY - CHECK DUPLICATE RECORDS”
“TOTAL RECORDS IN STOP SEARCH DATAFRAME: 7368”
“DUPLICATE RECORDS IN STOP SEARCH DATAFRAME: 0”
“______________________________”
```

13.1 Data Transformation - Remove Duplicate Records

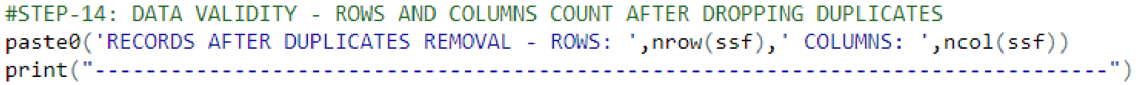
Code →



```
#STEP13.1: DATA TRANSFORMATION - REMOVE DUPLICATE RECORDS
ssf=distinct(ssf)
```

14. Data Validity - Rows and Columns Count After dropping duplicates

Code →



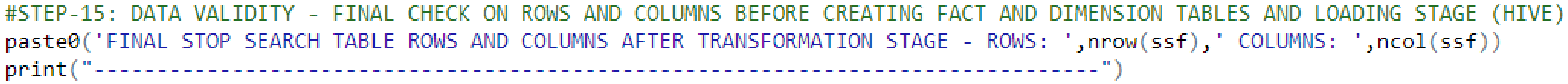
```
#STEP-14: DATA VALIDITY - ROWS AND COLUMNS COUNT AFTER DROPPING DUPLICATES
paste0('RECORDS AFTER DUPLICATES REMOVAL - ROWS: ',nrow(ssf),' COLUMNS: ',ncol(ssf))
print("--------------------------------------------------------------------------------")
```

Output →

```
“________________________________”
“RECORDS AFTER DUPLICATES REMOVAL - ROWS: 7368 COLUMNS: 6”
“________________________________”
```

15. Data Validity - Final Check on Rows and Columns before creating Fact and Dimension tables and Loading Stage (Hive)

Code →



```
#STEP-15: DATA VALIDITY - FINAL CHECK ON ROWS AND COLUMNS BEFORE CREATING FACT AND DIMENSION TABLES AND LOADING STAGE (HIVE)
paste0('FINAL STOP SEARCH TABLE ROWS AND COLUMNS AFTER TRANSFORMATION STAGE - ROWS: ',nrow(ssf),' COLUMNS: ',ncol(ssf))
print("--------------------------------------------------------------------------------")
```

Output →

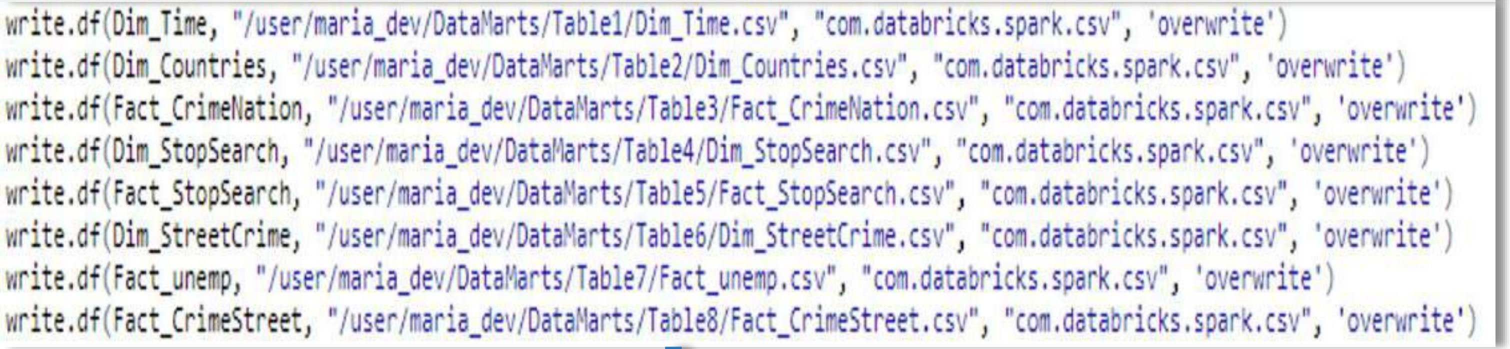
```
“________________________________”
“FINAL STOP SEARCH TABLE ROWS AND COLUMNS AFTER TRANSFORMATION STAGE - ROWS: 7368 COLUMNS: 6”
“________________________________”
```

DATA LOADING

Enter into HDFS in Putty: su – hdfs

1) hadoop fs -mkdir /user/maria_dev/DataMarts/Table1 – This command creates a directory in HDFS location to hold the data tables. Create separate directories for each fact and dimension table.

2) Load the SparkR **transformed** managed tables in their respective folders



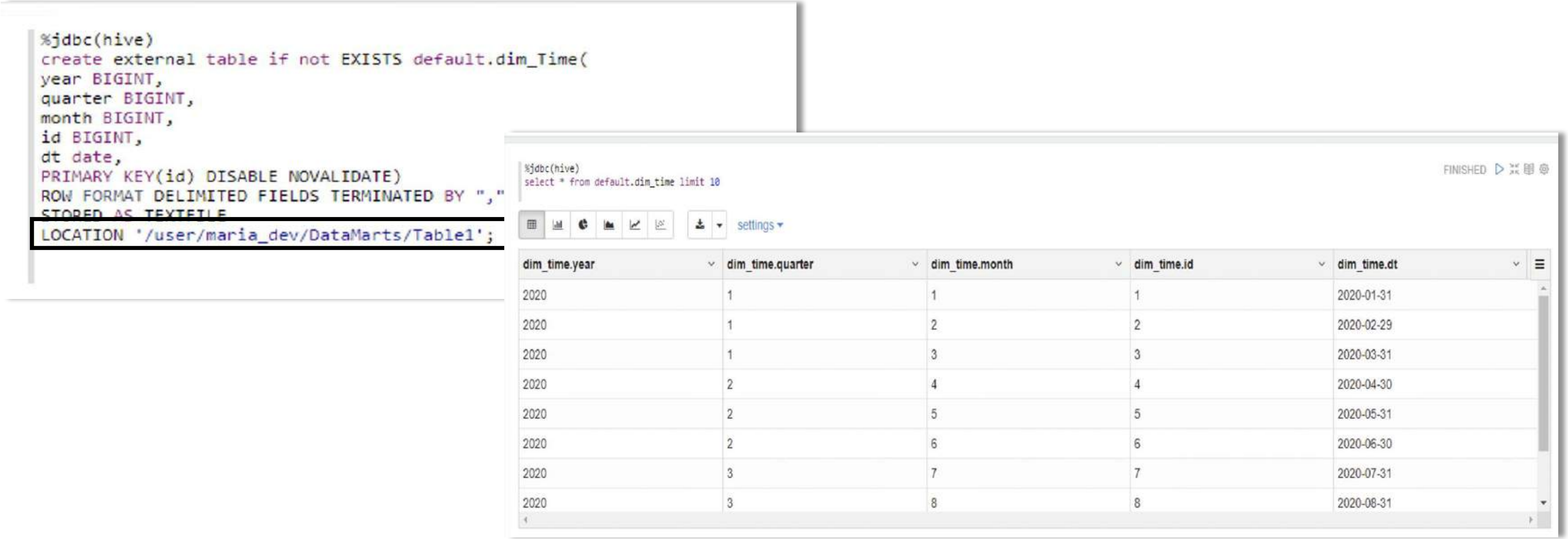
```
write.df(Dim_Time, "/user/maria_dev/DataMarts/Table1/Dim_Time.csv", "com.databricks.spark.csv", 'overwrite')
write.df(Dim_Countries, "/user/maria_dev/DataMarts/Table2/Dim_Countries.csv", "com.databricks.spark.csv", 'overwrite')
write.df(Fact_CrimeNation, "/user/maria_dev/DataMarts/Table3/Fact_CrimeNation.csv", "com.databricks.spark.csv", 'overwrite')
write.df(Dim_StopSearch, "/user/maria_dev/DataMarts/Table4/Dim_StopSearch.csv", "com.databricks.spark.csv", 'overwrite')
write.df(Fact_StopSearch, "/user/maria_dev/DataMarts/Table5/Fact_StopSearch.csv", "com.databricks.spark.csv", 'overwrite')
write.df(Dim_StreetCrime, "/user/maria_dev/DataMarts/Table6/Dim_StreetCrime.csv", "com.databricks.spark.csv", 'overwrite')
write.df(Fact_unemp, "/user/maria_dev/DataMarts/Table7/Fact_unemp.csv", "com.databricks.spark.csv", 'overwrite')
write.df(Fact_CrimeStreet, "/user/maria_dev/DataMarts/Table8/Fact_CrimeStreet.csv", "com.databricks.spark.csv", 'overwrite')
```

3) hadoop fs -chmod -R 777 /user/maria_dev/DataMarts/* - Run the command to give permission for creating the tables in the DataMarts/Table[1-8] - folders

4) Create fact and dimension tables in the respective data frame locations using HiveQL

DATA LOADING

- ➢ The data frame present in the location mentioned gets loaded into the hive table.



```
%jdbc(hive)
create external table if not EXISTS default.dim_Time(
year BIGINT,
quarter BIGINT,
month BIGINT,
id BIGINT,
dt date,
PRIMARY KEY(id) DISABLE NOVALIDATE)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ","
STORED AS TEXTFILE
LOCATION '/user/maria_dev/DataMarts/Table1';
```

```
%jdbc(hive)
select * from default.dim_time limit 10
```

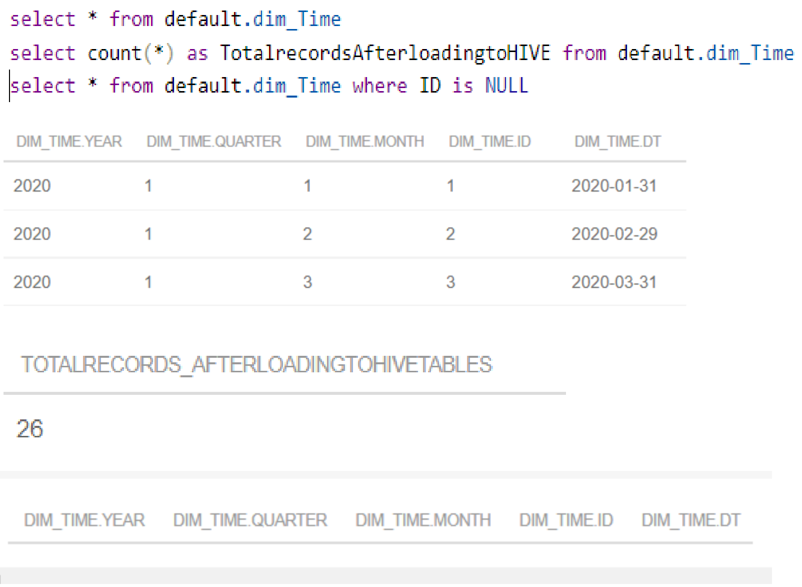
dim_time.year	dim_time.quarter	dim_time.month	dim_time.id	dim_time.dt
2020	1	1	1	2020-01-31
2020	1	2	2	2020-02-29
2020	1	3	3	2020-03-31
2020	2	4	4	2020-04-30
2020	2	5	5	2020-05-31
2020	2	6	6	2020-06-30
2020	3	7	7	2020-07-31
2020	3	8	8	2020-08-31

DATA LOADING – COMPLETED

HIVE – Data Quality

HIVE Fact and Dimensions

dim_Time



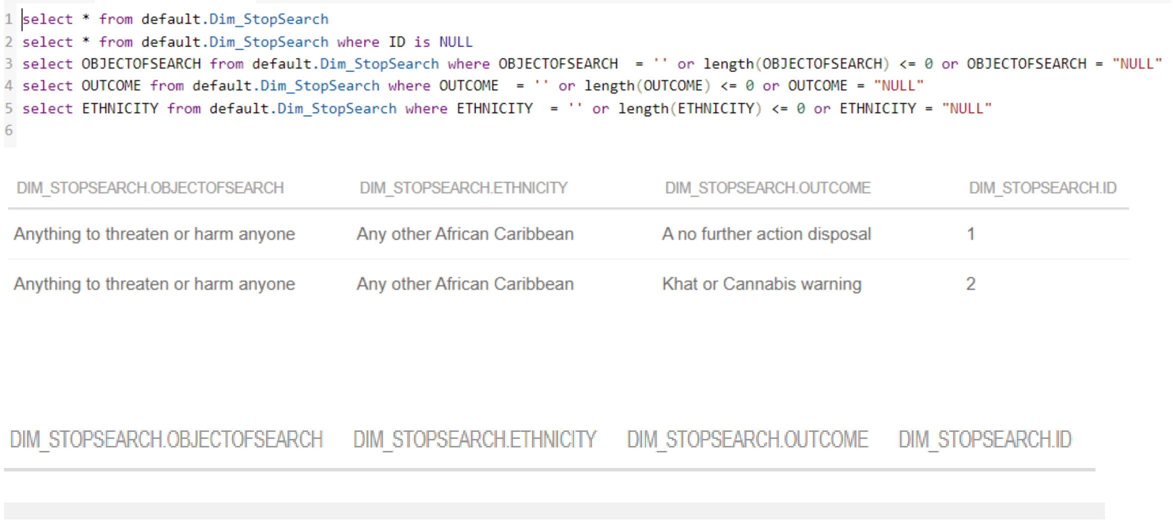
```
select * from default.dim_Time
select count(*) as TotalrecordsAfterloadingtoHIVE from default.dim_Time
select * from default.dim_Time where ID is NULL
```

DIM_TIME.YEAR	DIM_TIME.QUARTER	DIM_TIME.MONTH	DIM_TIME.ID	DIM_TIME.DT
2020	1	1	1	2020-01-31
2020	1	2	2	2020-02-29
2020	1	3	3	2020-03-31

TOTALRECORDS_AFTERLOADINGTOHIVETABLES
26

DIM_TIME.YEAR	DIM_TIME.QUARTER	DIM_TIME.MONTH	DIM_TIME.ID	DIM_TIME.DT

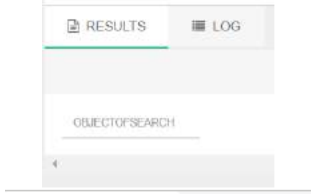
Dim_StopSearch



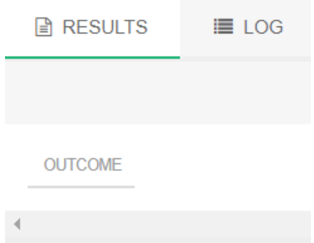
```
select * from default.Dim_StopSearch
select * from default.Dim_StopSearch where ID is NULL
select OBJECTOFSEARCH from default.Dim_StopSearch where OBJECTOFSEARCH  = '' or length(OBJECTOFSEARCH) <= 0 or OBJECTOFSEARCH = "NULL"
select OUTCOME from default.Dim_StopSearch where OUTCOME  = '' or length(OUTCOME) <= 0 or OUTCOME = "NULL"
select ETHNICITY from default.Dim_StopSearch where ETHNICITY  = '' or length(ETHNICITY) <= 0 or ETHNICITY = "NULL"
```

DIM_STOPSEARCH.OBJECTOFSEARCH	DIM_STOPSEARCH.ETHNICITY	DIM_STOPSEARCH.OUTCOME	DIM_STOPSEARCH.ID
Anything to threaten or harm anyone	Any other African Caribbean	A no further action disposal	1
Anything to threaten or harm anyone	Any other African Caribbean	Khat or Cannabis warning	2

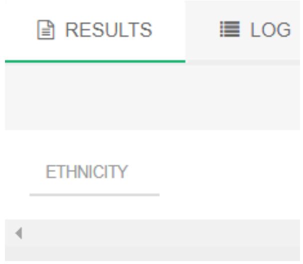
DIM_STOPSEARCH.OBJECTOFSEARCH	DIM_STOPSEARCH.ETHNICITY	DIM_STOPSEARCH.OUTCOME	DIM_STOPSEARCH.ID



OBJECTOFSEARCH



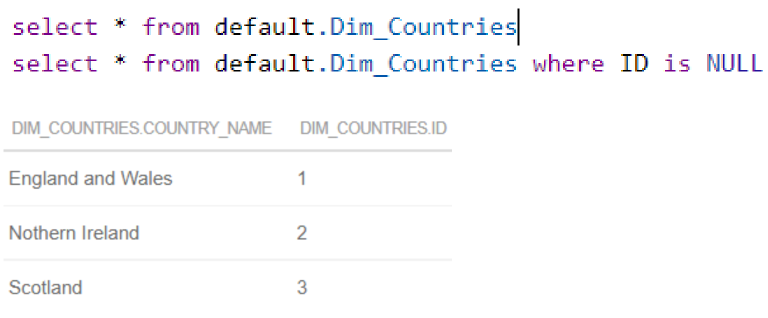
OUTCOME



ETHNICITY

No Invalid / NULL / Blank Records found

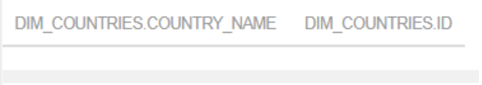
Dim_Countries



```
select * from default.Dim_Countries
select * from default.Dim_Countries where ID is NULL
```

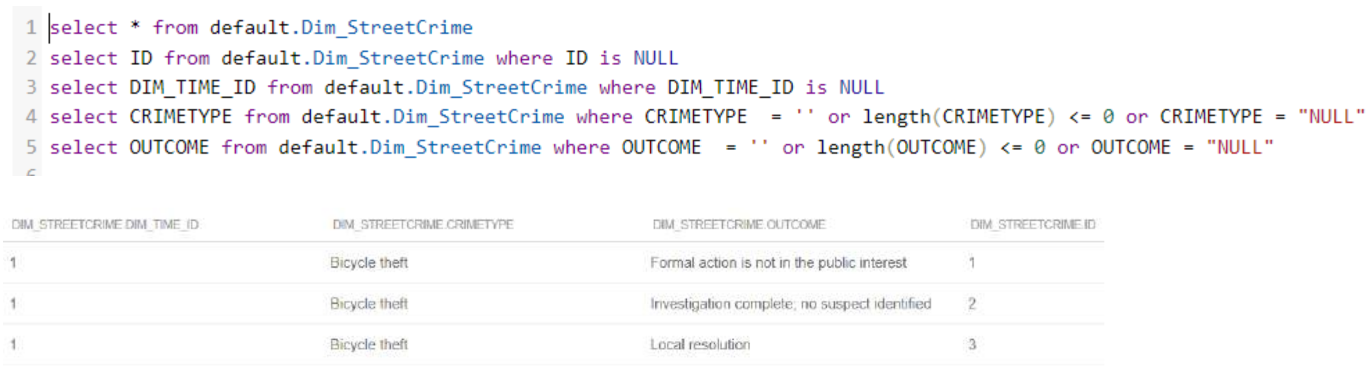
DIM_COUNTRIES.COUNTRY_NAME	DIM_COUNTRIES.ID
England and Wales	1
Nothern Ireland	2
Scotland	3

Total Records → Only 3 records visually confirmed



DIM_COUNTRIES.COUNTRY_NAME	DIM_COUNTRIES.ID

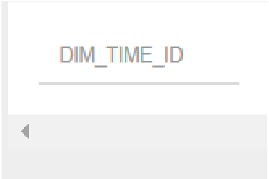
Dim_StreetCrime



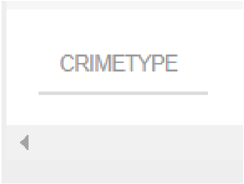
```
select * from default.Dim_StreetCrime
select ID from default.Dim_StreetCrime where ID is NULL
select DIM_TIME_ID from default.Dim_StreetCrime where DIM_TIME_ID is NULL
select CRIMETYPE from default.Dim_StreetCrime where CRIMETYPE  = '' or length(CRIMETYPE) <= 0 or CRIMETYPE = "NULL"
select OUTCOME from default.Dim_StreetCrime where OUTCOME  = '' or length(OUTCOME) <= 0 or OUTCOME = "NULL"
```

DIM_STREETCRIME.DIM_TIME_ID	DIM_STREETCRIME.CRIMETYPE	DIM_STREETCRIME.OUTCOME	DIM_STREETCRIME.ID
1	Bicycle theft	Formal action is not in the public interest	1
1	Bicycle theft	Investigation complete; no suspect identified	2
1	Bicycle theft	Local resolution	3

ID



DIM_TIME_ID

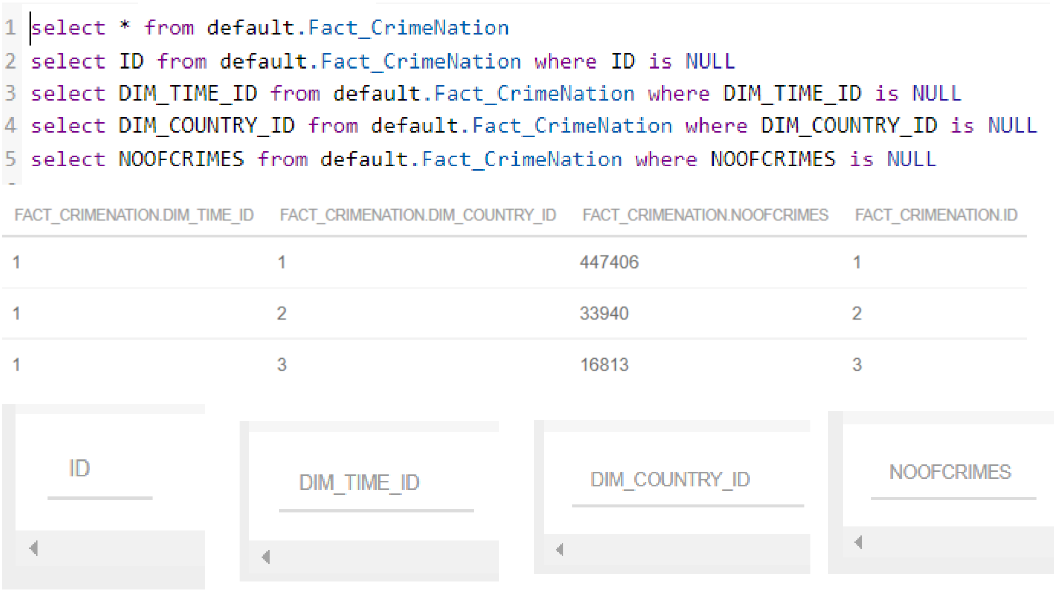


CRIMETYPE

OUTCOME

HIVE – Data Quality

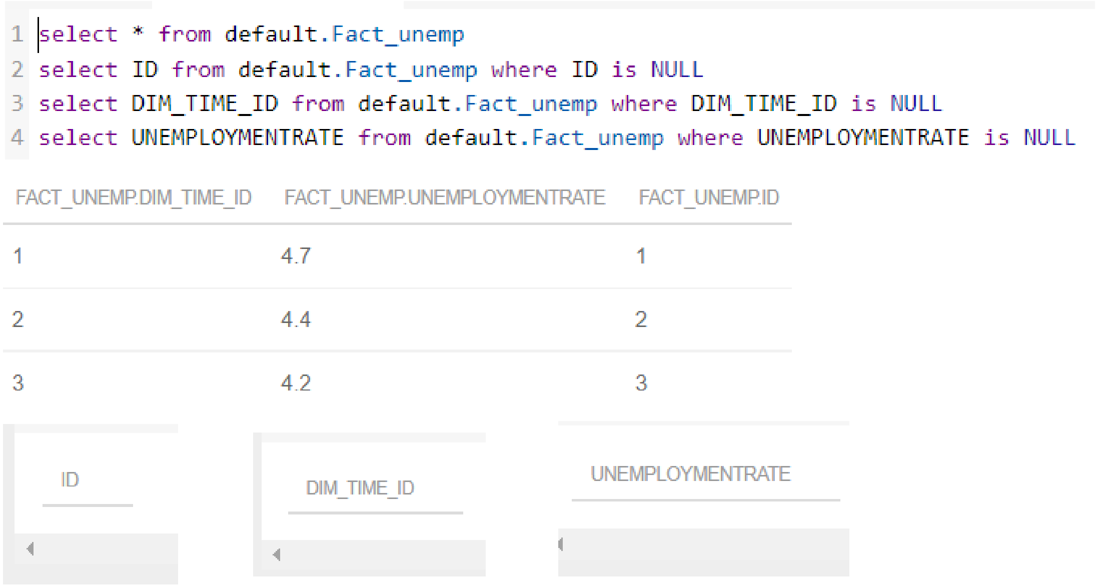
Fact_CrimeNation



```
select * from default.Fact_CrimeNation
select ID from default.Fact_CrimeNation where ID is NULL
select DIM_TIME_ID from default.Fact_CrimeNation where DIM_TIME_ID is NULL
select DIM_COUNTRY_ID from default.Fact_CrimeNation where DIM_COUNTRY_ID is NULL
select NOOFCRIMES from default.Fact_CrimeNation where NOOFCRIMES is NULL
```

FACT_CRIMENATION.DIM_TIME_ID	FACT_CRIMENATION.DIM_COUNTRY_ID	FACT_CRIMENATION.NOOFCRIMES	FACT_CRIMENATION.ID
1	1	447406	1
1	2	33940	2
1	3	16813	3

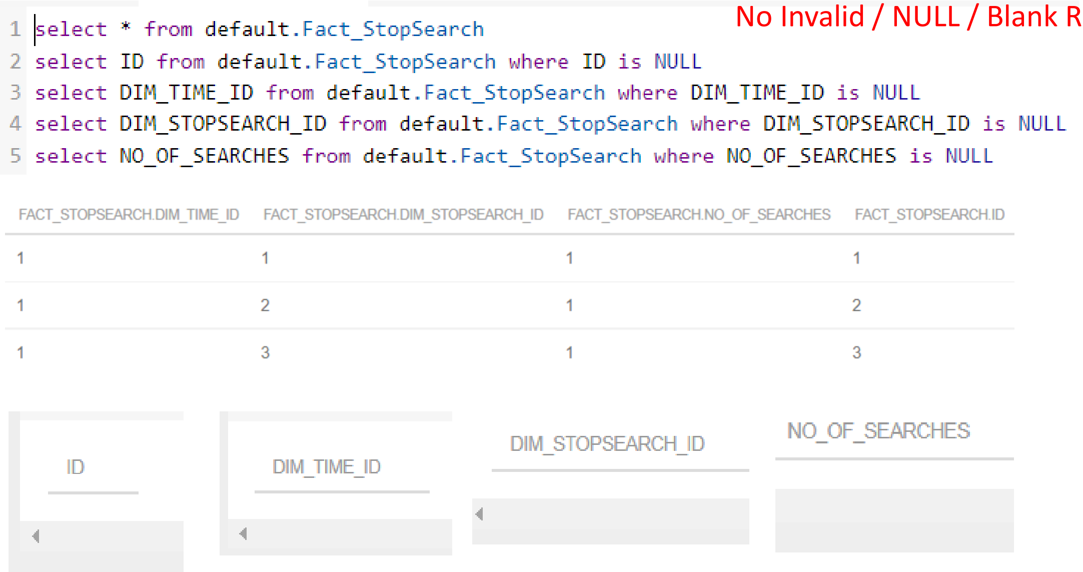
Fact_unemp



```
select * from default.Fact_unemp
select ID from default.Fact_unemp where ID is NULL
select DIM_TIME_ID from default.Fact_unemp where DIM_TIME_ID is NULL
select UNEMPLOYMENTRATE from default.Fact_unemp where UNEMPLOYMENTRATE is NULL
```

FACT_UNEMP.DIM_TIME_ID	FACT_UNEMP.UNEMPLOYMENTRATE	FACT_UNEMP.ID
1	4.7	1
2	4.4	2
3	4.2	3

Fact_StopSearch

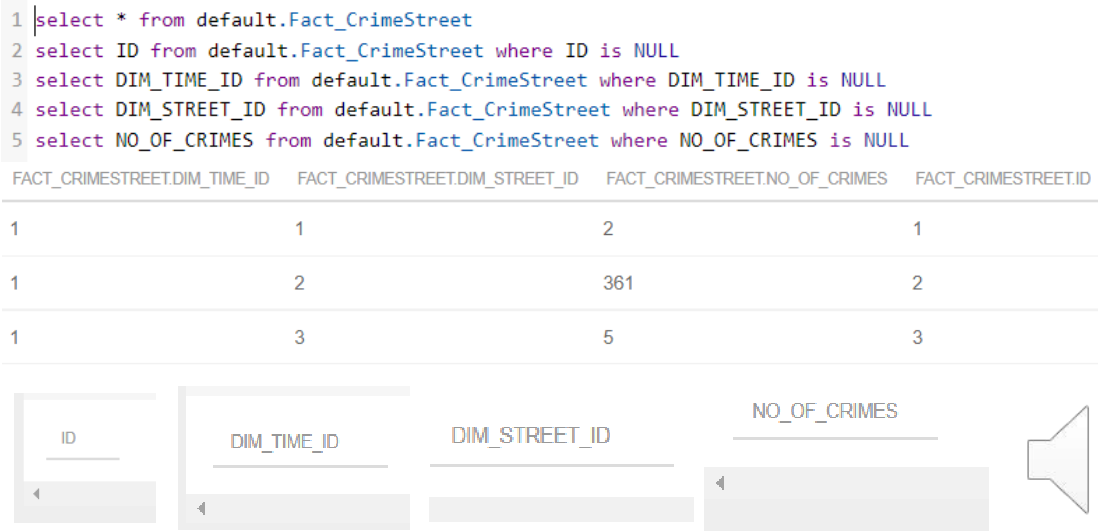


```
select * from default.Fact_StopSearch
select ID from default.Fact_StopSearch where ID is NULL
select DIM_TIME_ID from default.Fact_StopSearch where DIM_TIME_ID is NULL
select DIM_STOPSEARCH_ID from default.Fact_StopSearch where DIM_STOPSEARCH_ID is NULL
select NO_OF_SEARCHES from default.Fact_StopSearch where NO_OF_SEARCHES is NULL
```

FACT_STOPSEARCH.DIM_TIME_ID	FACT_STOPSEARCH.DIM_STOPSEARCH_ID	FACT_STOPSEARCH.NO_OF_SEARCHES	FACT_STOPSEARCH.ID
1	1	1	1
1	2	1	2
1	3	1	3

No Invalid / NULL / Blank Records found

Fact_CrimeStreet



```
select * from default.Fact_CrimeStreet
select ID from default.Fact_CrimeStreet where ID is NULL
select DIM_TIME_ID from default.Fact_CrimeStreet where DIM_TIME_ID is NULL
select DIM_STREET_ID from default.Fact_CrimeStreet where DIM_STREET_ID is NULL
select NO_OF_CRIMES from default.Fact_CrimeStreet where NO_OF_CRIMES is NULL
```

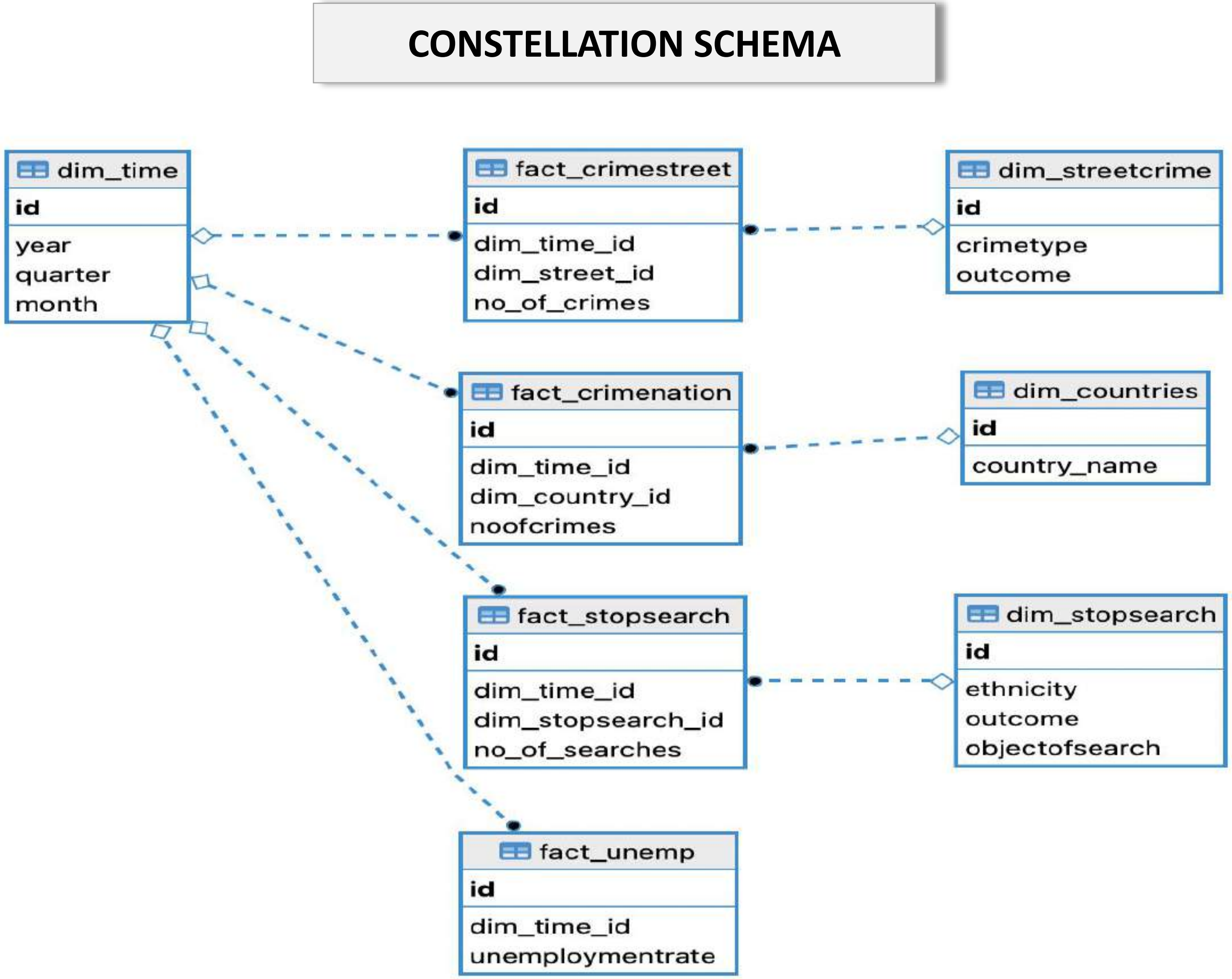
FACT_CRIMESTREET.DIM_TIME_ID	FACT_CRIMESTREET.DIM_STREET_ID	FACT_CRIMESTREET.NO_OF_CRIMES	FACT_CRIMESTREET.ID
1	1	2	1
1	2	361	2
1	3	5	3

ENTITY RELATIONSHIP DIAGRAM

- Totally four fact and four dimension tables are used to address the business questions

- The fact and dimension tables connections are designed in the form of a **constellation schema**

- The fact tables have **many-to-one** relationship with the dimension tables



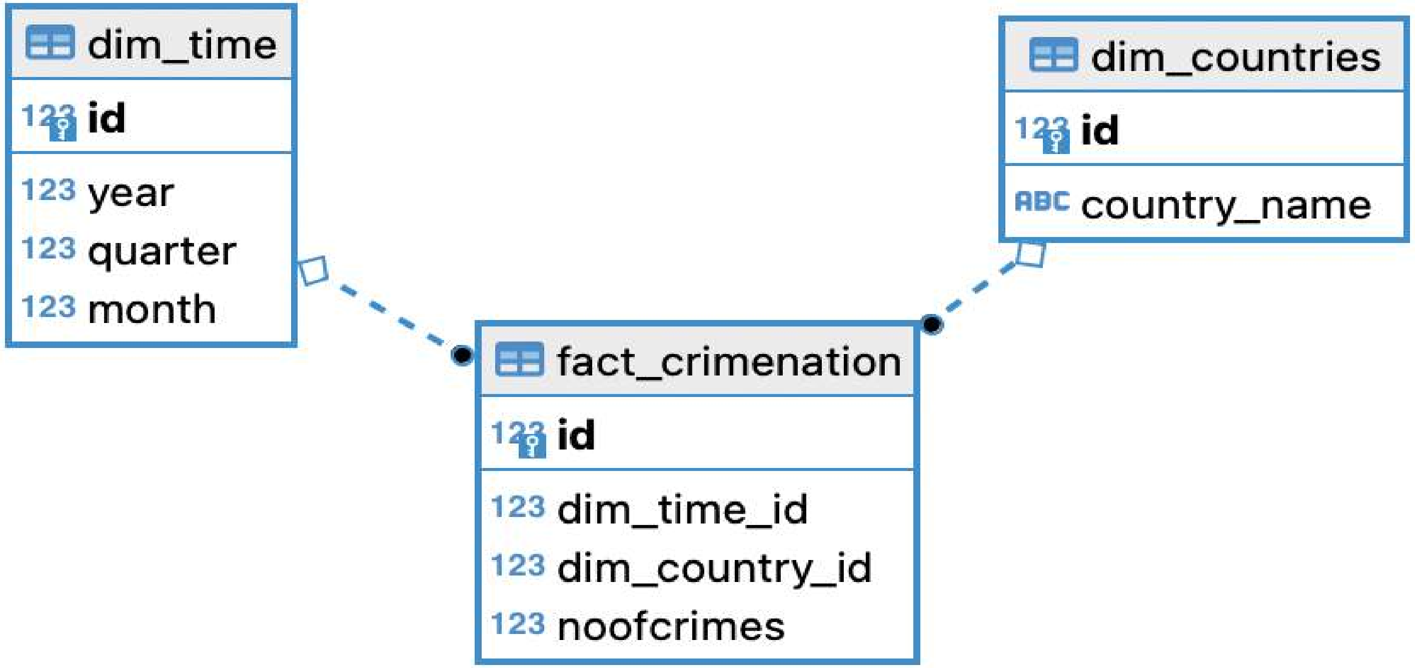
BUSINESS QUESTIONS – DATA MARTS

1

Provide a monthly breakdown of the overall the United Kingdom crime rate by country from 2020 till 2022

- Dimensions
  - By time
  - By country
- Dimesion Tables
  - Dim_Time
  - Dim_Countries
- Fact Table
  - Fact_Crimenation
- Lowest level of granularity
  - Month
  - Country
- Metrics
  - Crime rate

STAR SCHEMA



dim_time
- id
- year
- quarter
- month

dim_countries
- id
- country_name

fact_crimenation
- id
- dim_time_id
- dim_country_id
- noofcrimes

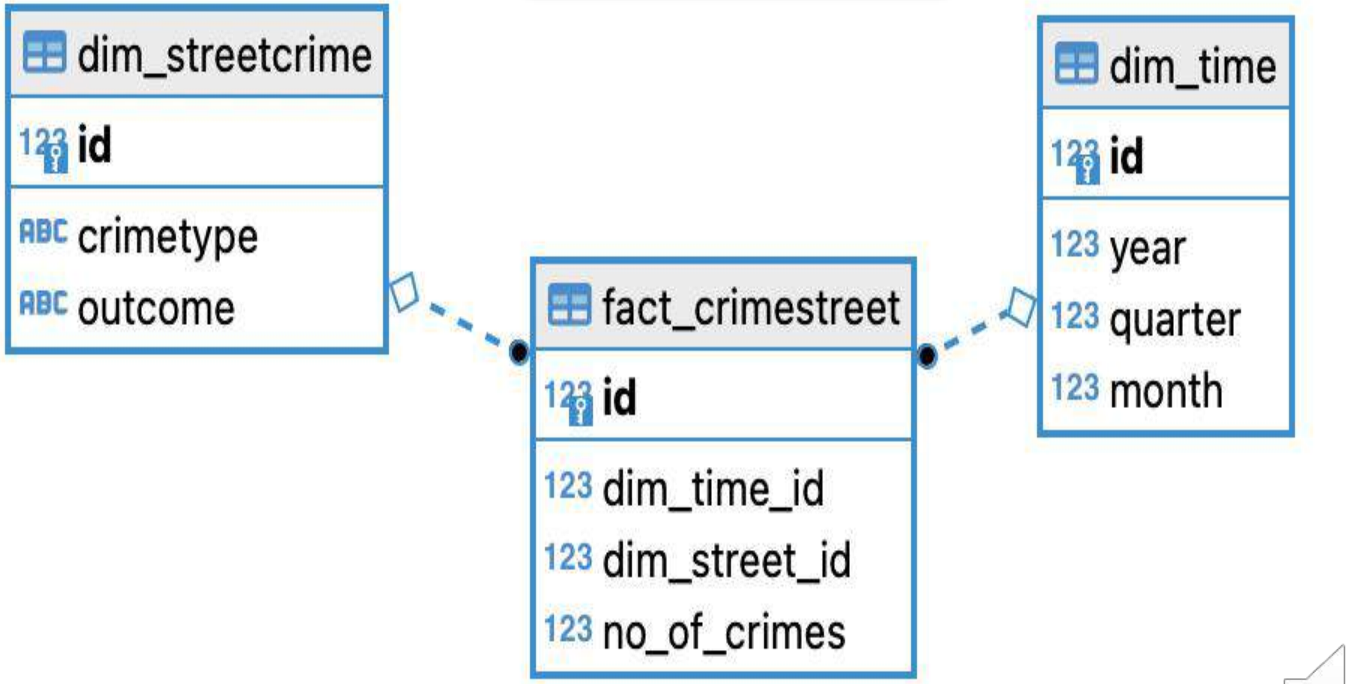
BUSINESS QUESTIONS – DATA MARTS

2

Provide a monthly breakdown of crimes rates in Yorkshire & Humber by crime-type from from 2020 till 2022

- Dimensions
  - By time
  - By crime type
- Dimesion Tables
  - Dim_Time
  - Dim_Streetcrime
- Fact Table
  - Fact_Crimestreet
- Lowest level of granularity
  - Month
  - Crime Type
- Metrics
  - Crime rate

STAR SCHEMA



dim_streetcrime
- id
- crimetype
- outcome

fact_crimestreet
- id
- dim_time_id
- dim_street_id
- no_of_crimes

dim_time
- id
- year
- quarter
- month

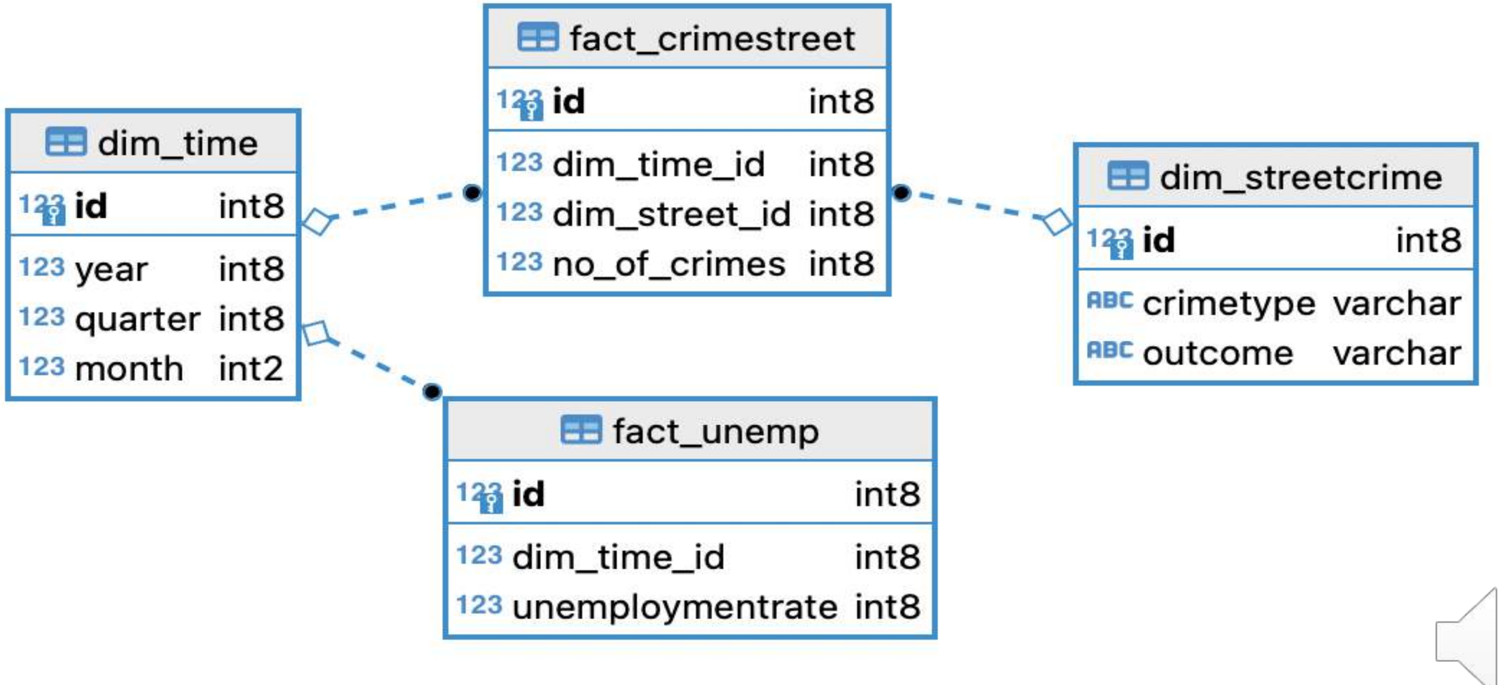
BUSINESS QUESTIONS – DATA MARTS

3

Provide the monthly statistics of crimes rates and unemployment rate in Yorkshire & Humber by crime-type from 2020 to 2022

- Dimensions
  - By time
  - By crime type
- Dimesion Tables
  - Dim_Time
  - Dim_Streetcrime
- Fact Table
  - Fact_Unemp
  - Fact_crimestreet
- Lowest level of granularity
  - Month
- Metrics
  - Crime rate

CONSTELLATION SCHEMA



dim_time	
id	int8
year	int8
quarter	int8
month	int2

fact_crimestreet	
id	int8
dim_time_id	int8
dim_street_id	int8
no_of_crimes	int8

dim_streetcrime	
id	int8
crimetype	varchar
outcome	varchar

fact_unemp	
id	int8
dim_time_id	int8
unemploymentrate	int8

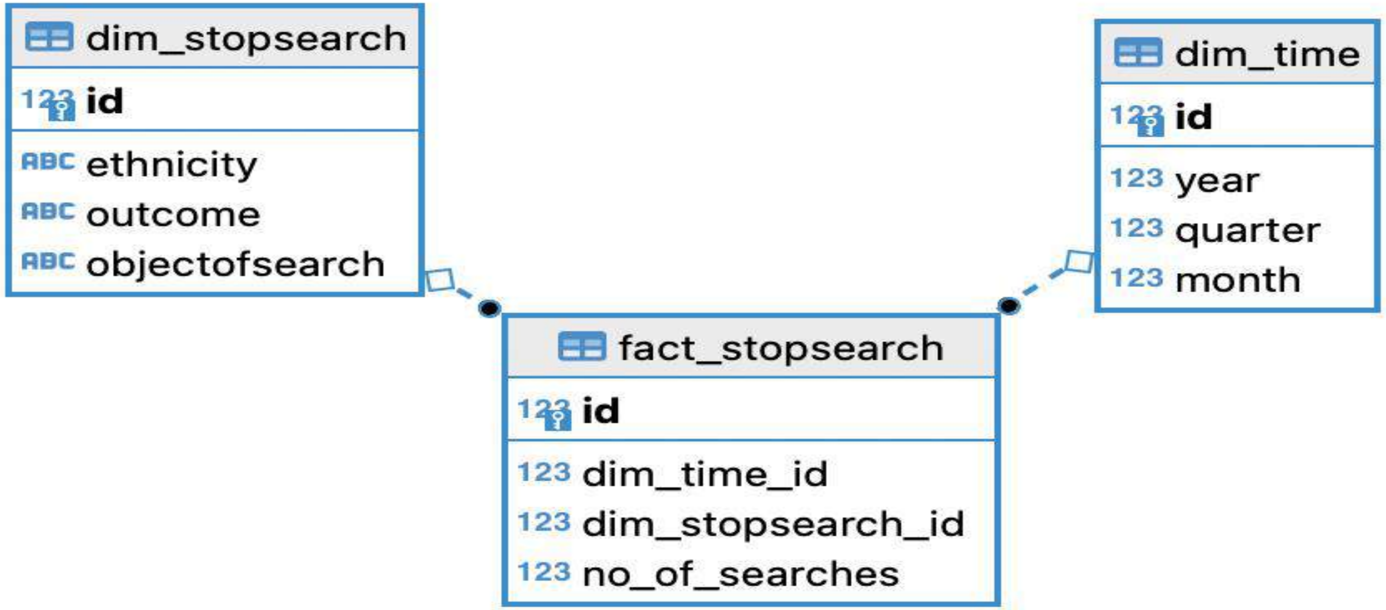
BUSINESS QUESTIONS – DATA MARTS

4

Provide a monthly breakdown of stop and search crime of Yorkshire & Humber, by ethnicity from 2020 to 2022

- Dimensions
  - By time
  - By ethnicity
- Dimesion Tables
  - Dim_Time
  - Dim_Stopsearch
- Fact Table
  - Fact_Stopsearch
- Lowest level of granularity
  - Month
- Metrics
  - Stop search rate

STAR SCHEMA



dim_stopsearch
- id
- ethnicity
- outcome
- objectofsearch

dim_time
- id
- year
- quarter
- month

fact_stopsearch
- id
- dim_time_id
- dim_stopsearch_id
- no_of_searches

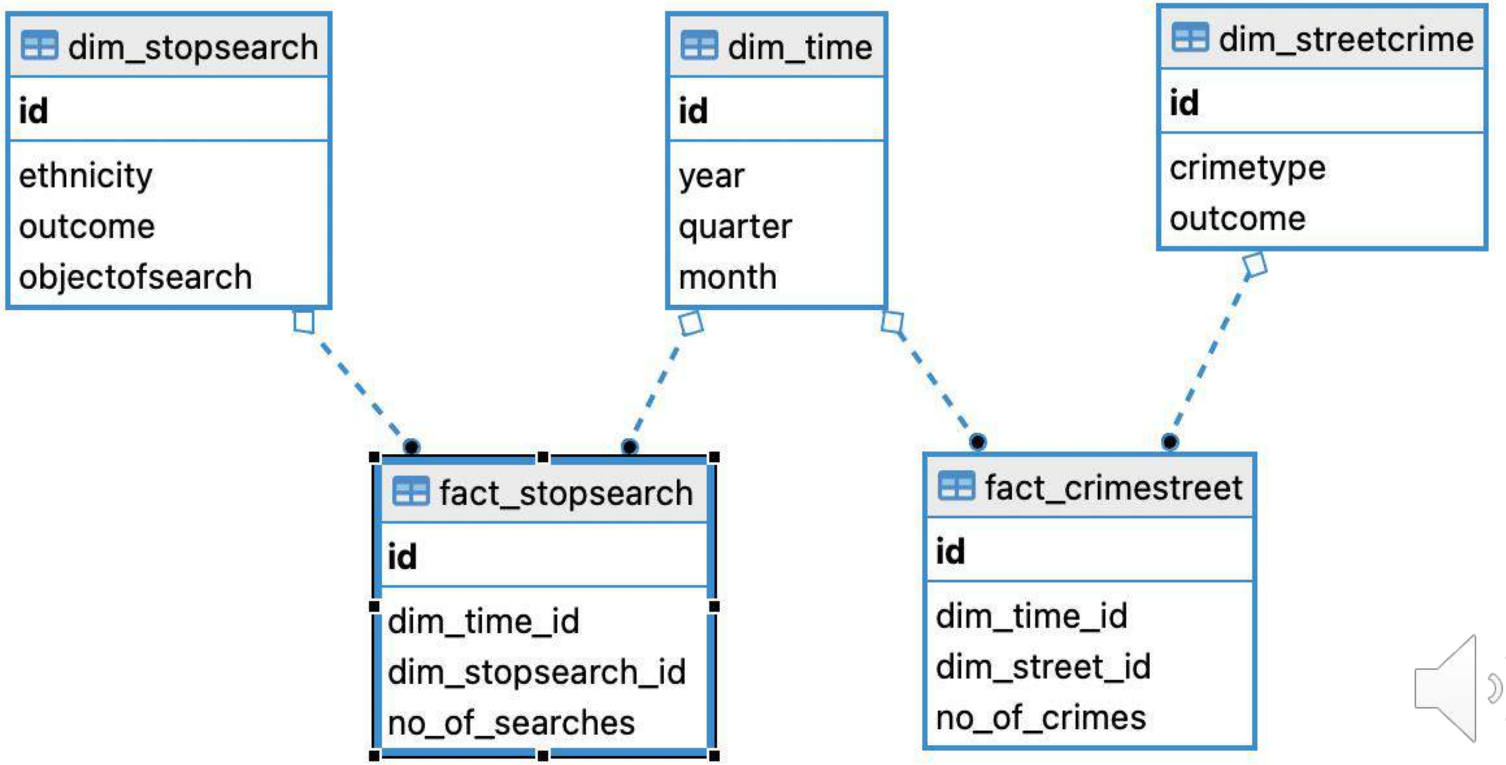
BUSINESS QUESTIONS – DATA MARTS

5

Provide a breakdown of latest crime outcome rates for street crime and stop & search crime by outcome type, by month from 2020 to 2022

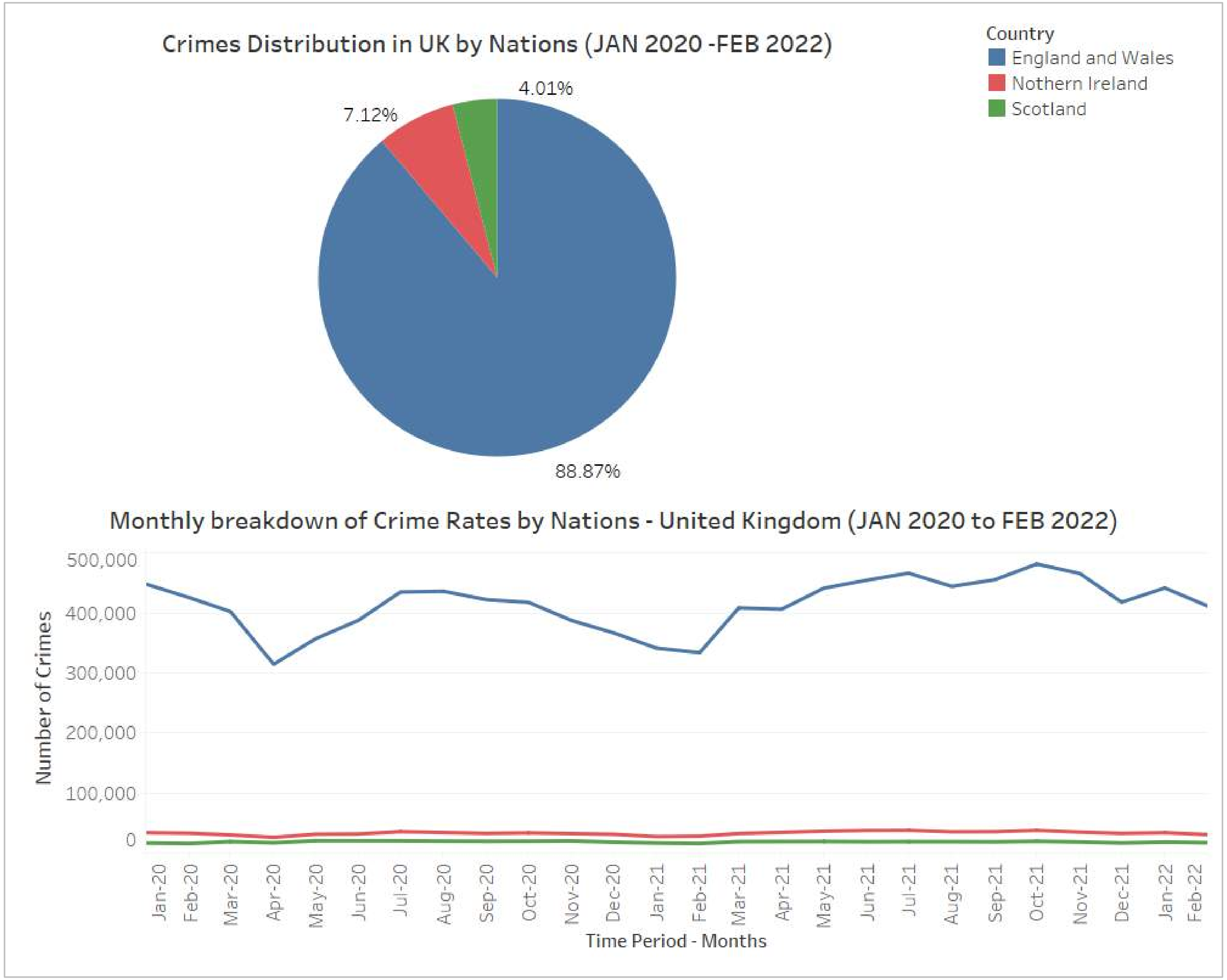
- Dimensions
  - By time
  - By outcome type
- Dimesion Tables
  - Dim_Time
  - Dim_Stopsearch
  - Dim_Streetcrime
- Fact Table
  - Fact_Stopsearch
  - Fact_crimestreet
- Lowest level of granularity
  - Month
- Metrics
  - Outcome rate

CONSTELLATION SCHEMA



CRIME BUSINESS INTELLIGENCE REPORT

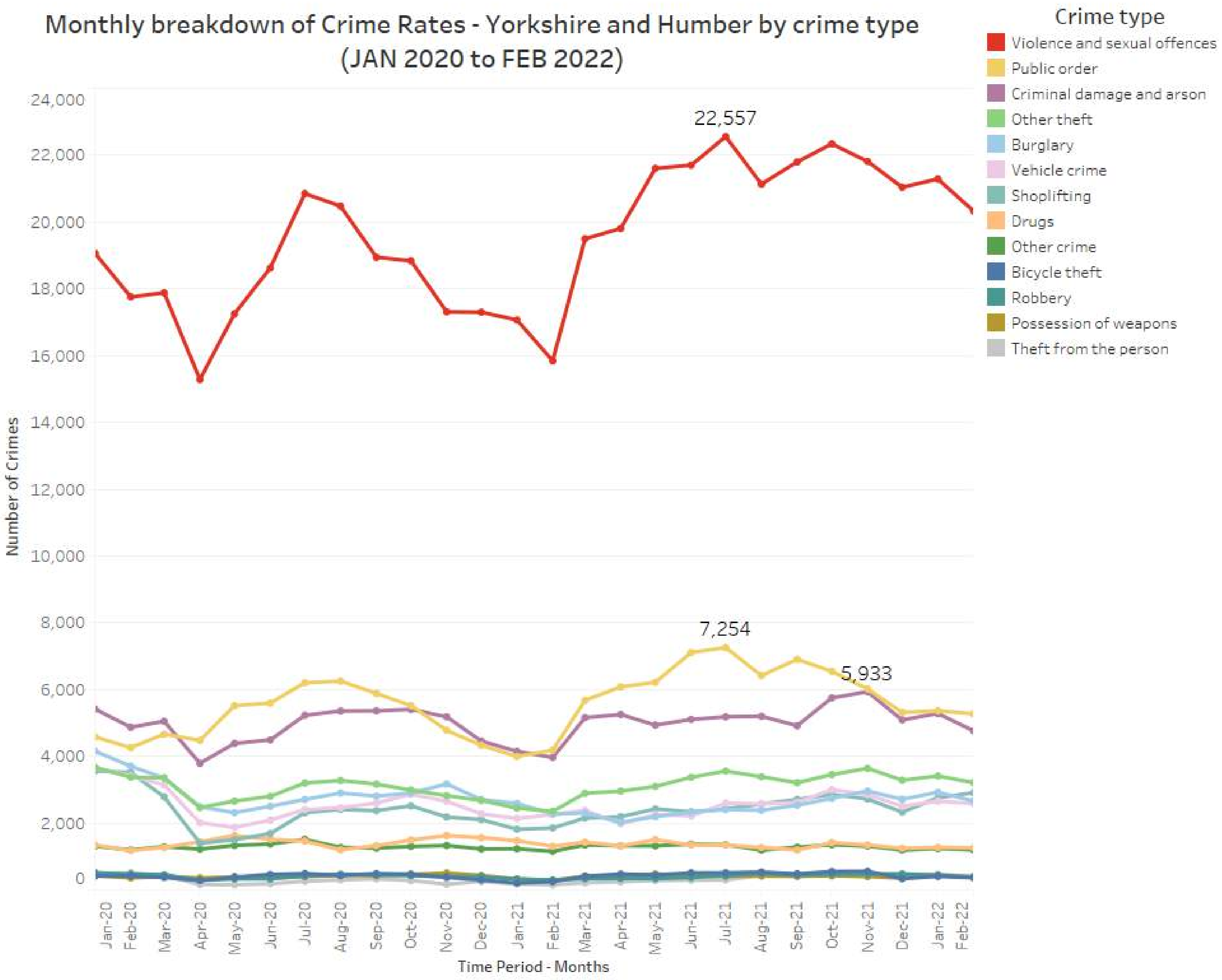
Provide the monthly breakdown of the overall the United Kingdom crime rate by country from 2020 till 2022



INSIGHTS:

- For the reporting time period Jan 2020 to Feb 2022, England and Wales has a high crime rate of 88.87%
- Within reporting time window, there exist a seasonal drop in crimes during 1st quarter of every year.
- Crime rate disparities exist between different nations of united kingdom.

Provide a monthly breakdown of crimes rates in Yorkshire & Humber by crime-type from from 2020 till 2022

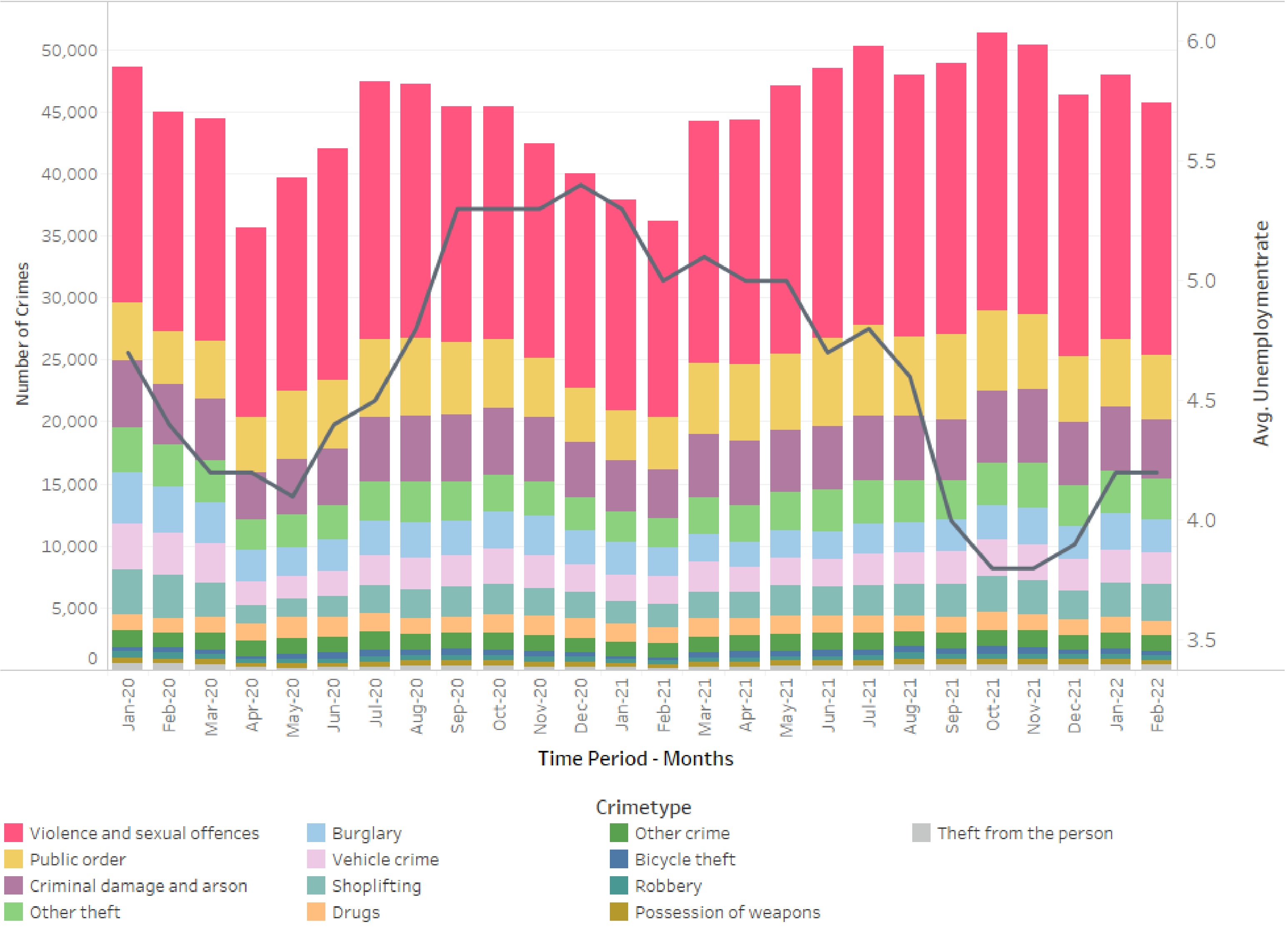


INSIGHTS:

- During the reporting time period (Jan 2020 to Feb 2022), Violence and Sexual Offences, Public order, Criminal damage, and arson crimes are top 3 crime types recorded.
- During the reporting time period (Jan 2020 to Feb 2022), Violence and Sexual Offences are more prevalent in Yorkshire and Humber region. However, there exist a seasonal drop in crimes frequencies in 1st quarter of every year.
- Public order and Criminal damage and arson were also showing same seasonal drop in frequency like Violence and Sexual Offences.
- During the reporting time period (Jan 2020 to Feb 2022), 2nd and 3rd quarter of every year has the highest crimes recorded.
- The legend of the graph is sorted in a descending order of crime rate by crime type. Only around 500 cases are recorded throughout the time period for the last four crime types.

Provide the monthly statistics of crimes rates and unemployment rate in Yorkshire & Humber by crime-type from 2020 to 2022

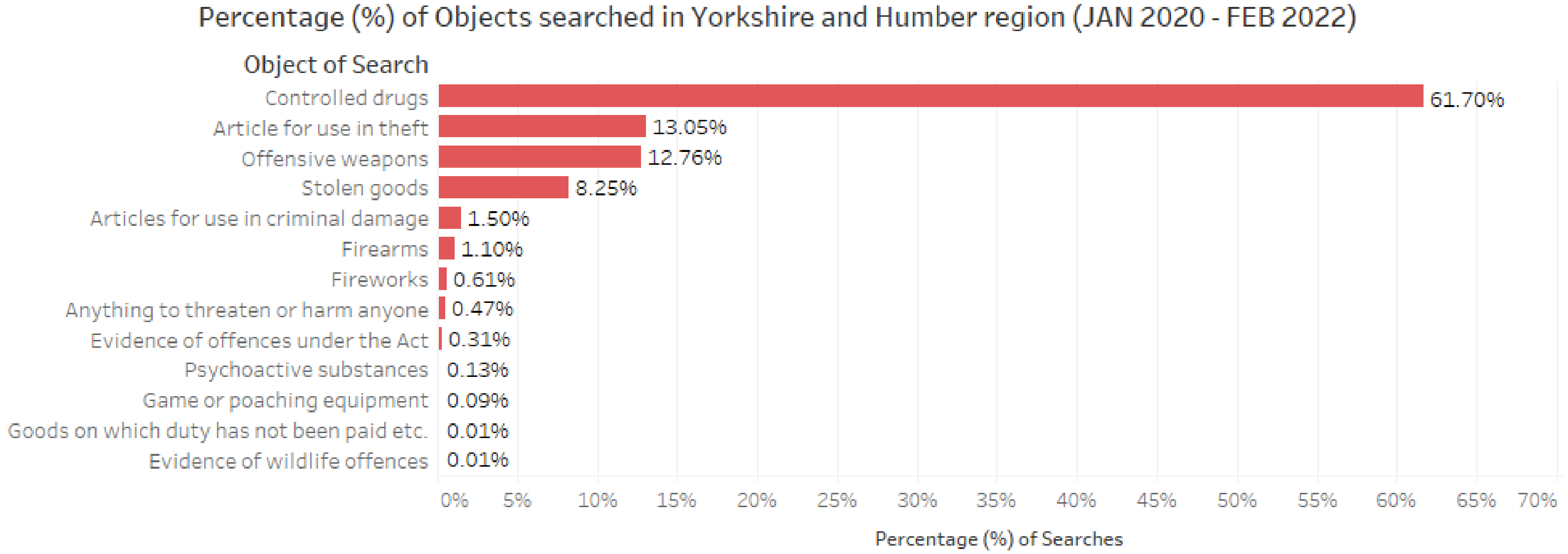
Monthly breakdown of crime rates by crime type vs unemployment rate for Yorkshire and Humber region (JAN 2020 to FEB 2022)



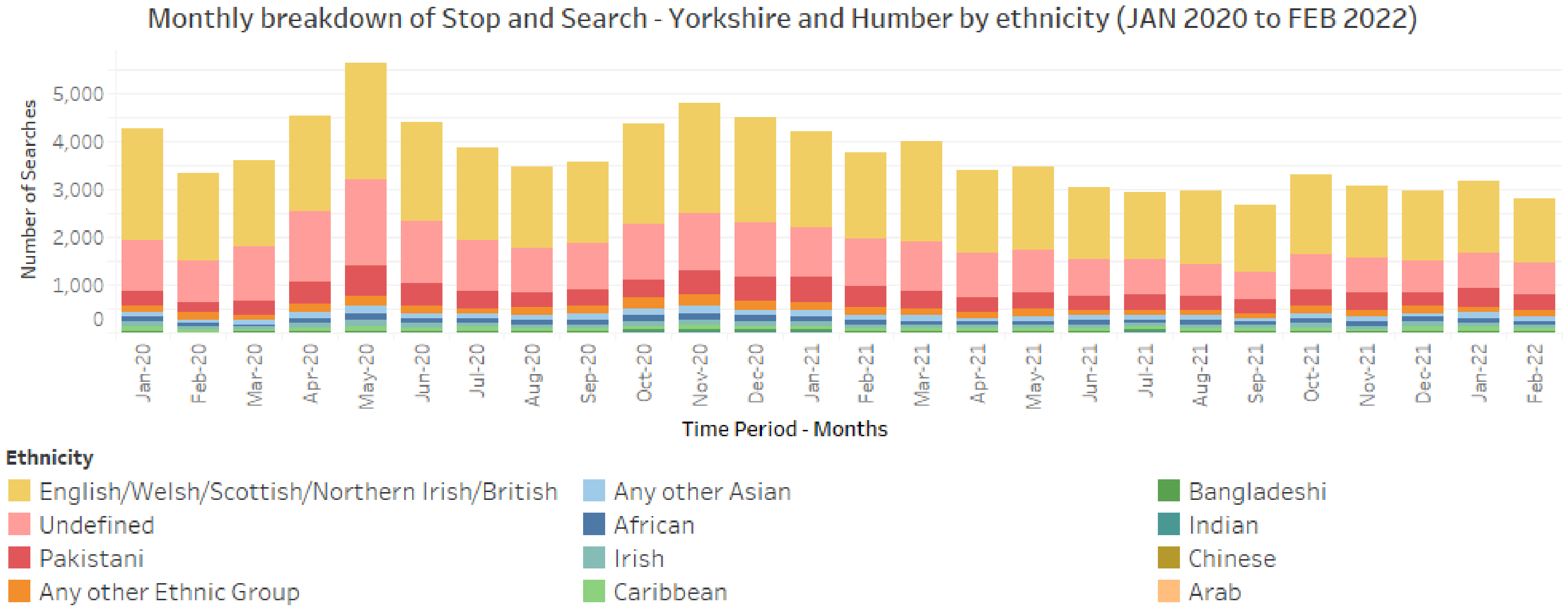
INSIGHTS:

- One of the key economic factors (unemployment rate) of Yorkshire and Humber during Jan 2020 - Feb 2022 is considered to find the impact in crime rates.
- In the Year 2020, as the unemployment rates increases, the crimes rates also increases. Similarly, crime rates decreases as the unemployment rate decreases. This pattern is prevailing even in Crime type level.
- In the Year 2021, the effect of unemployment rate is exactly opposite to the number of crimes recorded even by crime type, which contradicts the previous year (2020) trend.
- Based on above facts, for the recent years (2021, 2022) the unemployment rate doesn’t have an impact on crimes recorded for Yorkshire and Humber region.

Provide a monthly breakdown of stop and search crime of Yorkshire & Humber, by ethnicity from 2020 to 2022



Percentage (%) of Objects searched in Yorkshire and Humber region (JAN 2020 - FEB 2022)

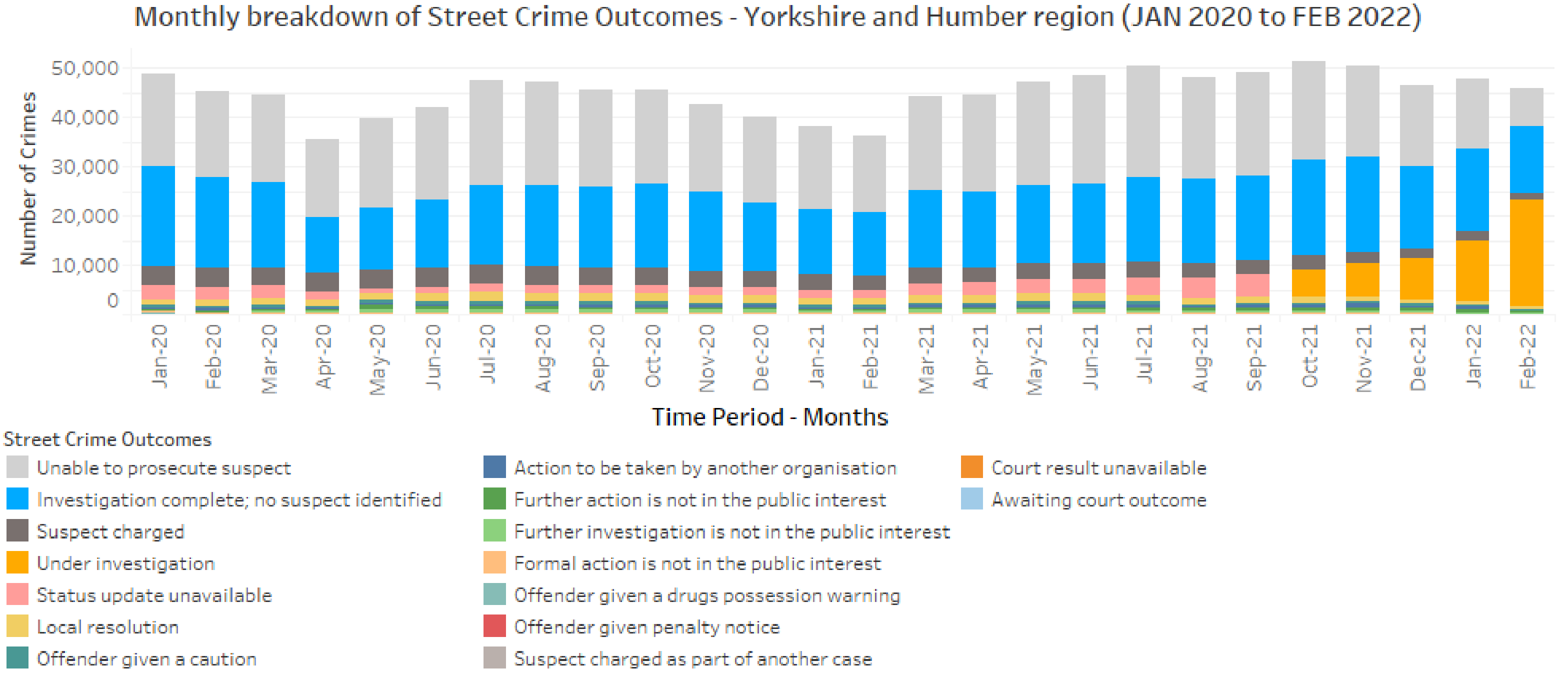


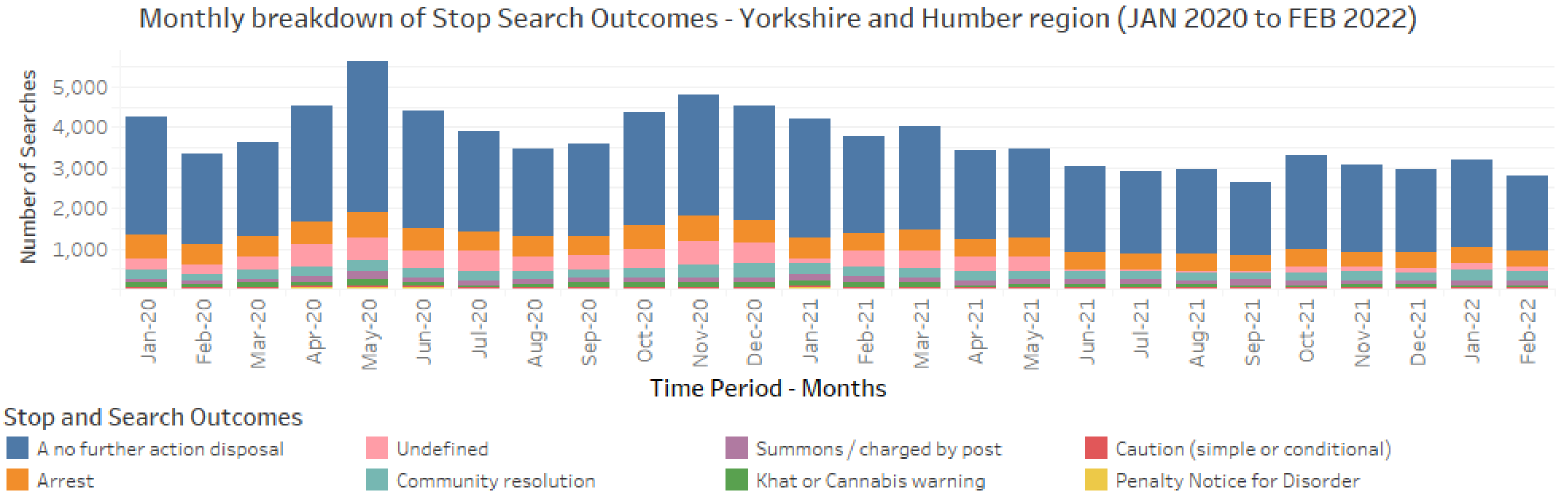
Monthly breakdown of Stop and Search - Yorkshire and Humber by ethnicity (JAN 2020 to FEB 2022)

INSIGHTS:

- During the reporting time period (Jan 2020 to Feb 2022), the Yorkshire and Humber region's most searched object is "Controlled Drugs" with 61.7%.
- The top 4 objects searched (Controlled Drugs, Article for use in theft, Offensive weapons and Stolen Goods) constitutes 95.76% of entire searches.
- Number of stop and searches has reduced gradually from 2020 to Feb 2022.
- Among the various ethnic people in Yorkshire and Humber, British people are stopped the most in entire analysis time window.
- The stop and search data contains considerable proportion of nulls, which are later treated as "Undefined". There exist a gap in recording the ethnic background of person searched.
- Pakistanis are searched second most during the entire reporting time window.
- Indian, Chinese and Arabs are among the least searched ethnicities in the reporting time window.

Provide a breakdown of latest crime outcome rates for street crime and stop & search crime by outcome type, by month from 2020 to 2022





INSIGHTS (Street Crime Outcomes):

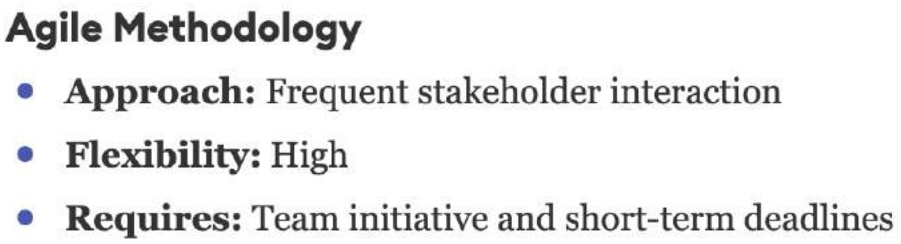
- During the reporting time period (Jan 2020 to Feb 2022), the police were not able to prosecute the suspects in most of the crimes. however, in the past quarter, there is a significant reduction in crimes on which the suspects weren't prosecuted.
- From Sep 2021, "Status update unavailable" outcome status reduced drastically and same holds for the consecutive months till end of reporting time frame.
- In the recent 2 quarters (Q4 2021 and Q1 2022), Number of crimes on which the suspects got charged reduced from 2,743 (Oct 2021) to 1,174 (Feb 2022).
- From Oct 2021, number of crimes which are under investigation is increasing gradually and as of Feb 2022, number of crimes under investigation is 21,684.

INSIGHTS (Stop and Search Outcomes):

- During the specific time period (Apr 2021 to Feb 2022), number of searches is slowly getting reduced.
- During reporting time window (Jan 2020 to Feb 2022), most of the people searched are left to walkaway without any action.
- Number of search outcomes classified as "Undefined" have reduced drastically during Jun 2021 – Sep 2021. However, a slight increase in the recent 2 quarters.
- The proportion of people getting arrested, community resolutions, people summoned or charged by post are almost same across the reporting time period.
- Since the total number of stop searches reduced in recent 2 quarters, People getting cannabis warnings have reduced to an average of 56 per month. During reporting time period excluding recent 2 quarters (Oct 2021 to Feb 2022), the average people getting cannabis warning is approximately 90.

FINAL RECOMMENDATIONS

- Currently, the waterfall approach is being used for the development. For the future business needs, the agile methodology can be adopted for improving the reports, end-products in an iterative way



Agile Methodology

- **Approach:** Frequent stakeholder interaction
- **Flexibility:** High
- **Requires:** Team initiative and short-term deadlines

  - Hoory, L. (2021, October 27). *Agile vs. Waterfall: Which Project Management Methodology Should I Use?* Forbes Advisor. https://www.forbes.com/advisor/business/agile-vs-waterfall-methodology/

- Increasing the clusters and appropriate configurations in Hadoop ensure high performance in processing the data (Data management)
  - *What is Hadoop cluster? - Definition from WhatIs.com*. (n.d.). SearchBusinessAnalytics. Retrieved April 26, 2022, from https://www.techtarget.com/searchbusinessanalytics/definition/Hadoop-cluster

- To handle the data variety in future, more facts or dimensions can be added to the current data warehouse using the bottom-up Kimball’s method.

DECISIONS TAKEN DURING THE DEVELOPMENT PROCESS

- **Data service**: The UK Police websites provides API support & CSV file download for fetching the crime data. There were a limited set of parameters in the API request call namely, latlon, date, and location ID(Street ID). The API is only capable of returning 10000 records per each GET request. Any location with records higher than 10000 will not be returned. Therefore, CSV file download option was finalized.

- **ACL (Access control list issue)**: Multiple users have access to the HDFS location (Zeppelin, maria_dev, hive). But the users are restricted to perform any write operation on tasks created by other users. Superuser group access should be given for reading or writing data in directories owned by other users.

SUMMARY AND CONCLUSION

- An adequate amount of analysis on the crime problems should be performed for the police team to make any decisions on resource allocation and funding.

- Based on SHU Consultancy Group analysis, England & Wales have the highest crime rate with 88.87%. And within the country of England & Wales, according to a latest article Yorkshire and Humber region have the highest number of crimes recorded.

- In Yorkshire & Humber, the Violence and Sexual offences are the most recorded crimes. Comparatively, a less number of crimes are recorded for other crime types. Therefore, the police department needs to concentrate on deploying the appropriate force for the crimes that have high number of records.

- The socio-economic factor – unemployment rate, doesn’t have any impact on the crime rates for any crime-type in the 2021 and 2022 years. The police department need not have to concentrate much on this factor.

- In the stop and search police operation, the English ethnic group were stopped and searched the most among all the groups. The team needs to cross monitor the records to verify the fairness of the operation.

- In the final analysis, the number of crimes under investigation have eventually increased. There can be multiple reasons for this increase, but the department should consider this fact and deploy some additional force to investigate and clear the cases.

THANK YOU!!