MS SQL Study – Tools used: SQL Server 2019 [ 2 Core, 16 GB RAM, Server], Python, SMSS

Step 1: Import Python packages

**import time**

**import pandas as pd**

**import pyodbc**

**import sqlalchemy**

**from urllib.parse import quote\_plus**

Step 2: Read the dataset

**df = pd.read\_csv('D:/** **nics-checks-archive.csv')**

Step 3: Normalize the dataset for data integrity, ACID

**Task 1 # Group the dataframe by month and state**

**grouped\_df = df.groupby(['month', 'state'])**

**Task2 # Create tables for different transaction types**

selected\_columns = ['month', 'state', 'long\_gun', 'redemption\_long\_gun', 'returned\_long\_gun',

'rentals\_long\_gun', 'private\_sale\_long\_gun', 'return\_to\_seller\_long\_gun','prepawn\_long\_gun']

longgun\_df = df[selected\_columns].copy()

selected\_columns = ['month', 'state', 'other', 'multiple', 'admin', 'prepawn\_other', 'redemption\_other',

'returned\_other', 'private\_sale\_other','return\_to\_seller\_other']

other\_df = df[selected\_columns].copy()

selected\_columns = ['month', 'state', 'handgun', 'prepawn\_handgun','redemption\_handgun',

'returned\_handgun', 'rentals\_handgun', 'private\_sale\_handgun', 'return\_to\_seller\_handgun']

handgun\_df = df[selected\_columns].copy()

selected\_columns = ['month', 'state', 'permit', 'permit\_recheck']

permit\_df = df[selected\_columns].copy()

Task 3: # Confirm to remove duplicates if any

def g(df):

df['duplicate'] = df.astype(str).agg('|'.join, axis=1)

df = df.drop\_duplicates(subset='duplicate', keep='first')

df.drop(['duplicate'], axis=1, inplace=True)

return df

longgun\_df = g(longgun\_df.copy())

other\_df = g(other\_df.copy())

handgun\_df = g(handgun\_df.copy())

permit\_df = g(permit\_df.copy())

Step 4: Conver the dataframe ‘yyyy-dd’ column called month to datetime format

**longgun\_df['month'] = pd.to\_datetime(longgun\_df['month'], format='%Y-%m')**

**other\_df['month'] = pd.to\_datetime(other\_df['month'], format='%Y-%m')**

**handgun\_df['month'] = pd.to\_datetime(handgun\_df['month'], format='%Y-%m')**

**permit\_df['month'] = pd.to\_datetime(permit\_df['month'], format='%Y-%m')**

Step 5: # Create tables in MSSQL and set key to ensure data integrity.

**Task 1: Table 1 -** permit\_df

USE [fire\_arm]

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[permit\_df](

[month] [datetime] NOT NULL,

[state] [varchar](256) NOT NULL,

[permit] [float] NULL,

[permit\_recheck] [float] NULL,

CONSTRAINT [pk\_permit\_df] PRIMARY KEY CLUSTERED

(

[month] ASC,

[state] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = ON, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

**Task2: Table 2 -** other\_df

USE [fire\_arm]

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[other\_df](

[month] [datetime] NOT NULL,

[state] [varchar](256) NOT NULL,

[other] [float] NULL,

[multiple] [bigint] NULL,

[admin] [float] NULL,

[prepawn\_other] [float] NULL,

[redemption\_other] [float] NULL,

[returned\_other] [float] NULL,

[private\_sale\_other] [float] NULL,

[return\_to\_seller\_other] [float] NULL,

CONSTRAINT [pk\_other\_df] PRIMARY KEY CLUSTERED

(

[month] ASC,

[state] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = ON, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

**Task3- Table 3- longgun\_df**

USE [fire\_arm]

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[longgun\_df](

[month] [datetime] NOT NULL,

[state] [varchar](256) NOT NULL,

[long\_gun] [float] NULL,

[redemption\_long\_gun] [float] NULL,

[returned\_long\_gun] [float] NULL,

[rentals\_long\_gun] [float] NULL,

[private\_sale\_long\_gun] [float] NULL,

[return\_to\_seller\_long\_gun] [float] NULL,

[prepawn\_long\_gun] [float] NULL,

CONSTRAINT [pk\_longgun\_df] PRIMARY KEY CLUSTERED

(

[month] ASC,

[state] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = ON, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

**Task4- Table 4- handgun\_df**

USE [fire\_arm]

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[handgun\_df](

[month] [datetime] NOT NULL,

[state] [varchar](256) NOT NULL,

[handgun] [float] NULL,

[prepawn\_handgun] [float] NULL,

[redemption\_handgun] [float] NULL,

[returned\_handgun] [float] NULL,

[rentals\_handgun] [float] NULL,

[private\_sale\_handgun] [float] NULL,

[return\_to\_seller\_handgun] [float] NULL,

CONSTRAINT [pk\_handgun\_df] PRIMARY KEY CLUSTERED

(

[month] ASC,

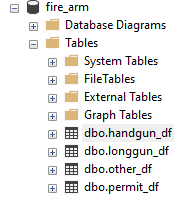
[state] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = ON, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

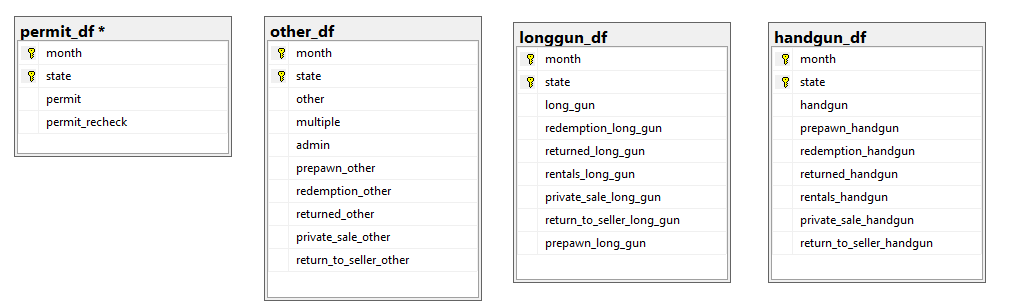
) ON [PRIMARY]

GO

Task4: Ensure Table created

****

**ERD**

****

Step 5: Convert the dataframe ‘yyyy-dd’ column called month to datetime format

**# Define the common server information**

**server, username, password, db\_name = ('xx.xx.xx.xx', 'xx', 'xxx', 'fire\_arm')**

**# Define a list of table names**

**table\_names = ['permit\_df', 'handgun\_df', 'other\_df', ‘longgun\_df’]**

**# Define the function to perform the database operations**

**def db\_connect(df, tb\_name):**

**odbc\_conn\_string = f"Driver={{ODBC Driver 17 for SQL Server}};SERVER={server},xxxx;" \**

**f"Database={db\_name};UID={username};PWD={password};"**

**conn = f"mssql+pyodbc:///?odbc\_connect={quote\_plus(odbc\_conn\_string)}"**

**dbEngine = sqlalchemy.create\_engine(conn, fast\_executemany=True, connect\_args={'connect\_timeout': 10},**

**echo=False)**

**df.to\_sql(con=dbEngine, schema="dbo", name=tb\_name, if\_exists="append", index=False)**

**# Loop through the table names and insert each DataFrame with the corresponding table name**

**for tb\_name in table\_names:**

**df = globals()[tb\_name].copy() # Assuming your DataFrames are in the global scope**

**db\_connect(df, tb\_name)**

Step 6: Query

**Query1: Country with maximum number of Background checks initiated for Hand gun including everything for Year 2022**

/\* Declare variables \*/

DECLARE @startTime datetime,

@endTime datetime;

/\* Set start time \*/

SET @startTime = GETDATE();

WITH PermitHandgunCounts AS (

SELECT

p.state,

SUM(p.permit\_recheck) AS total\_permits,

SUM(h.handgun + h.prepawn\_handgun + h.redemption\_handgun + h.returned\_handgun + h.rentals\_handgun + h.private\_sale\_handgun + h.return\_to\_seller\_handgun) AS total\_handguns

FROM

[fire\_arm].[dbo].[permit\_df] p

INNER JOIN

[fire\_arm].[dbo].[handgun\_df] h

ON

p.state = h.state

AND DATEPART(YEAR, p.month) = 2022

AND DATEPART(YEAR, h.month) = 2022

AND DATEPART(MONTH, p.month) = DATEPART(MONTH, h.month)

GROUP BY

p.state

)

SELECT TOP 1

state,

total\_handguns

FROM

PermitHandgunCounts

ORDER BY

total\_handguns DESC;

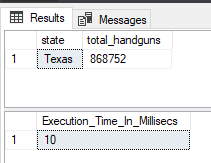
/\* Set end time \*/

SET @endTime = GETDATE();

/\* Find duration in milliseconds \*/

SELECT DATEDIFF(ms, @startTime, @endTime)

AS Execution\_Time\_In\_Millisecs

****

**Query2:** **Under redemption, Which category has requested for maximum number of background checks across 5 years**

/\* Declare variables \*/

DECLARE @startTime datetime,

@endTime datetime;

/\* Set start time \*/

SET @startTime = GETDATE();

WITH RedemptionTotals AS (

SELECT

'Handgun' AS category,

SUM(redemption\_handgun) AS total

FROM

handgun\_df

UNION ALL

SELECT

'Longgun' AS category,

SUM(redemption\_long\_gun) AS total

FROM

longgun\_df

)

SELECT TOP 1

category,

total

FROM

RedemptionTotals

ORDER BY

total DESC;

/\* Set end time \*/

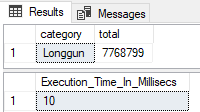
SET @endTime = GETDATE();

/\* Find duration in milliseconds \*/

SELECT DATEDIFF(ms, @startTime, @endTime)

AS Execution\_Time\_In\_Millisecs

AS Execution\_Time\_In\_Millisecs

****

**Query3: Out of totals which category of Background checks have majority percentage of requests in July 2022 which category and country**

/\* Declare variables \*/

DECLARE @startTime datetime,

@endTime datetime;

/\* Set start time \*/

SET @startTime = GETDATE();

WITH CombinedData AS (

SELECT

p.month,

p.state,

COALESCE(h.prepawn\_handgun, 0) + COALESCE(l.prepawn\_long\_gun, 0) + COALESCE(o.prepawn\_other, 0) AS prepawn\_total,

COALESCE(h.private\_sale\_handgun, 0) + COALESCE(l.private\_sale\_long\_gun, 0) + COALESCE(o.private\_sale\_other, 0) AS private\_sale\_total,

COALESCE(h.redemption\_handgun, 0) + COALESCE(l.redemption\_long\_gun, 0) + COALESCE(o.redemption\_other, 0) AS redemption\_total,

COALESCE(l.rentals\_long\_gun, 0) + COALESCE(h.rentals\_handgun, 0) AS rentals\_total,

COALESCE(h.return\_to\_seller\_handgun, 0) + COALESCE(l.return\_to\_seller\_long\_gun, 0) + COALESCE(o.return\_to\_seller\_other, 0) AS return\_to\_seller\_total,

COALESCE(h.returned\_handgun, 0) + COALESCE(l.returned\_long\_gun, 0) + COALESCE(o.returned\_other, 0) AS returned\_total

FROM

[fire\_arm].[dbo].[permit\_df] p

LEFT JOIN

[fire\_arm].[dbo].[handgun\_df] h

ON

p.month = h.month AND p.state = h.state

LEFT JOIN

[fire\_arm].[dbo].[longgun\_df] l

ON

p.month = l.month AND p.state = l.state

LEFT JOIN

[fire\_arm].[dbo].[other\_df] o

ON

p.month = o.month AND p.state = o.state

WHERE

p.month BETWEEN '2022-07-01 00:00:00.000' AND '2022-07-31 00:00:00.000'

)

SELECT

'Prepawn' AS category,

state,

prepawn\_total AS total

FROM

CombinedData

WHERE

prepawn\_total = (SELECT MAX(prepawn\_total) FROM CombinedData)

UNION ALL

SELECT

'PrivateSale' AS category,

state,

private\_sale\_total AS total

FROM

CombinedData

WHERE

private\_sale\_total = (SELECT MAX(private\_sale\_total) FROM CombinedData)

UNION ALL

SELECT

'Redemption' AS category,

state,

redemption\_total AS total

FROM

CombinedData

WHERE

redemption\_total = (SELECT MAX(redemption\_total) FROM CombinedData)

UNION ALL

SELECT

'Rentals' AS category,

state,

rentals\_total AS total

FROM

CombinedData

WHERE

rentals\_total = (SELECT MAX(rentals\_total) FROM CombinedData)

UNION ALL

SELECT

'ReturnToSeller' AS category,

state,

return\_to\_seller\_total AS total

FROM

CombinedData

WHERE

return\_to\_seller\_total = (SELECT MAX(return\_to\_seller\_total) FROM CombinedData)

UNION ALL

SELECT

'Returned' AS category,

state,

returned\_total AS total

FROM

CombinedData

WHERE

returned\_total = (SELECT MAX(returned\_total) FROM CombinedData);

/\* Set end time \*/

SET @endTime = GETDATE();

/\* Find duration in milliseconds \*/

SELECT DATEDIFF(ms, @startTime, @endTime)

AS Execution\_Time\_In\_Millisecs



|  |  |  |
| --- | --- | --- |
| Query 1 | Query 2 | Query3 |
| 10 ms | 10 ms | 10 ms |