MASSACHUSETTS GENERAL HOSPITAL

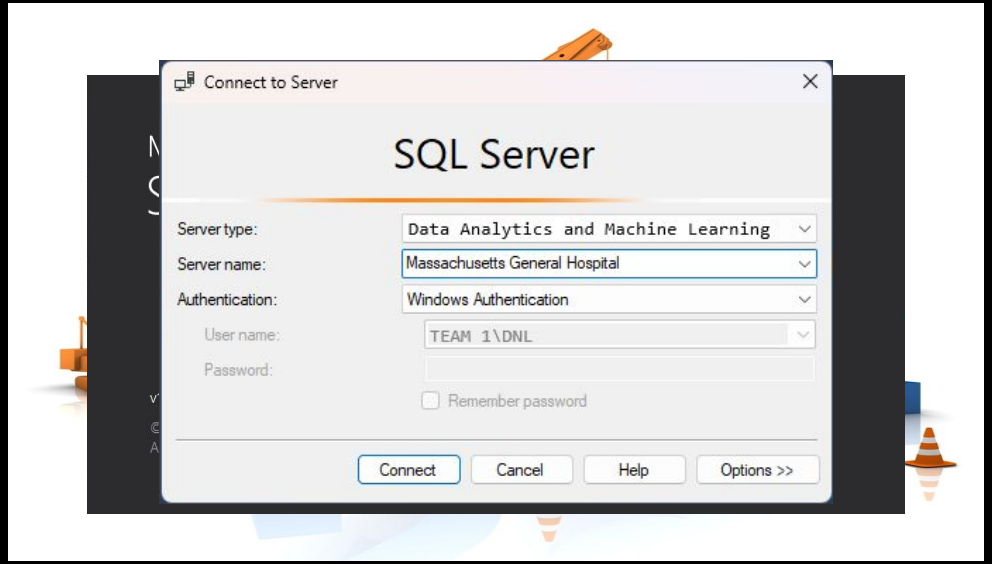
GROUP 1

Doris Waithaka

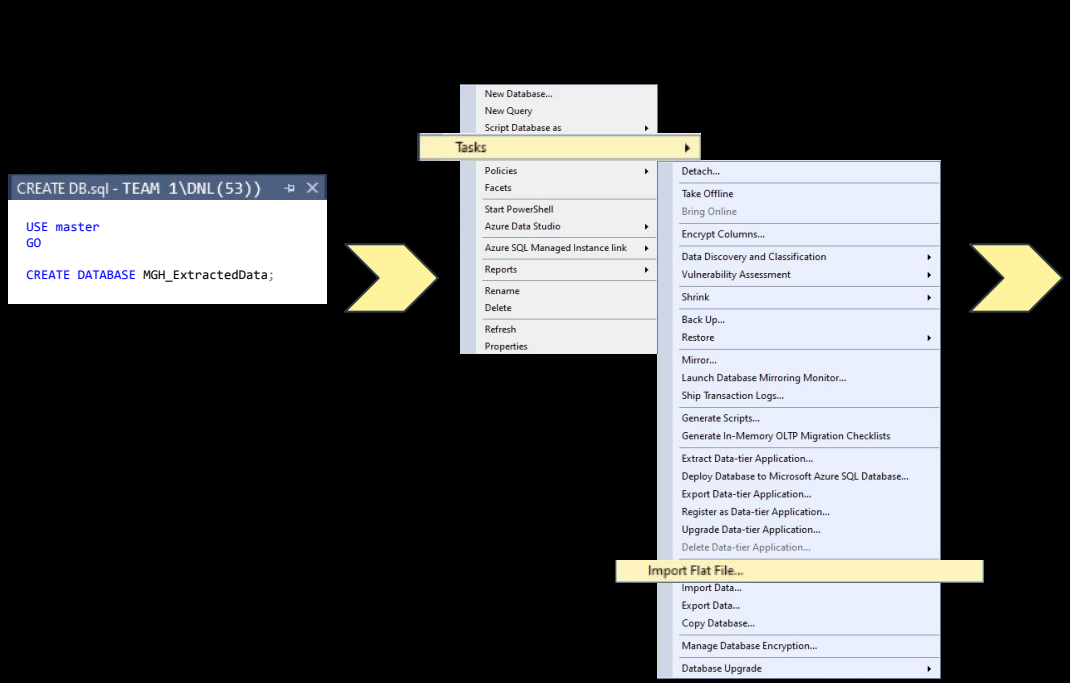
Vivian Njenga

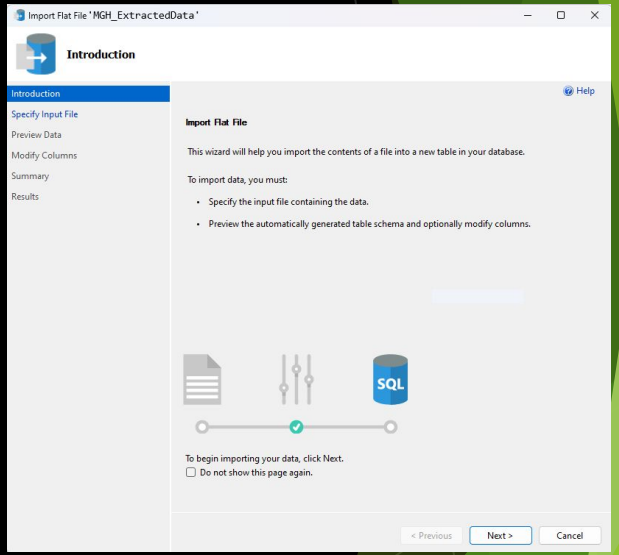
Michaela Jackson

Fidelis Makunike

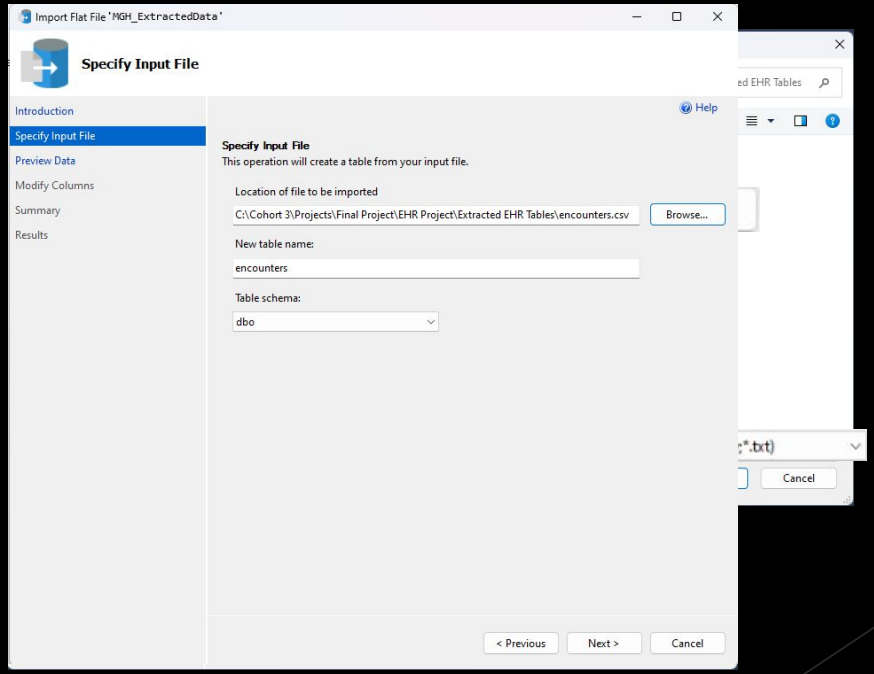

Connect to Server
SQL Server
Server type:
Data Analytics and Machine Learning
Server name:
Massachusetts General Hospital
Authentication:
Windows Authentication
User name:
TEAM 1\DNL
Password:
Remember password
Connect
Cancel
Help
Options >>

Import Start Up

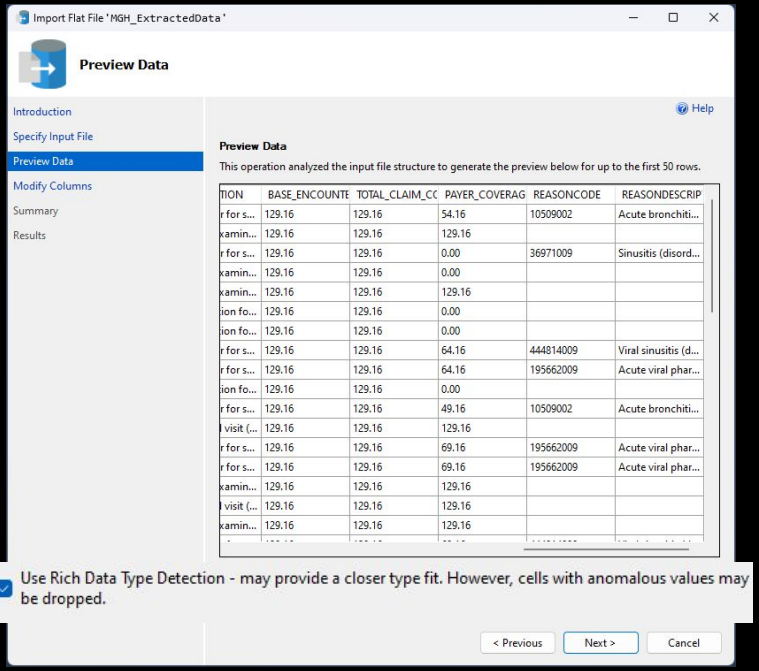




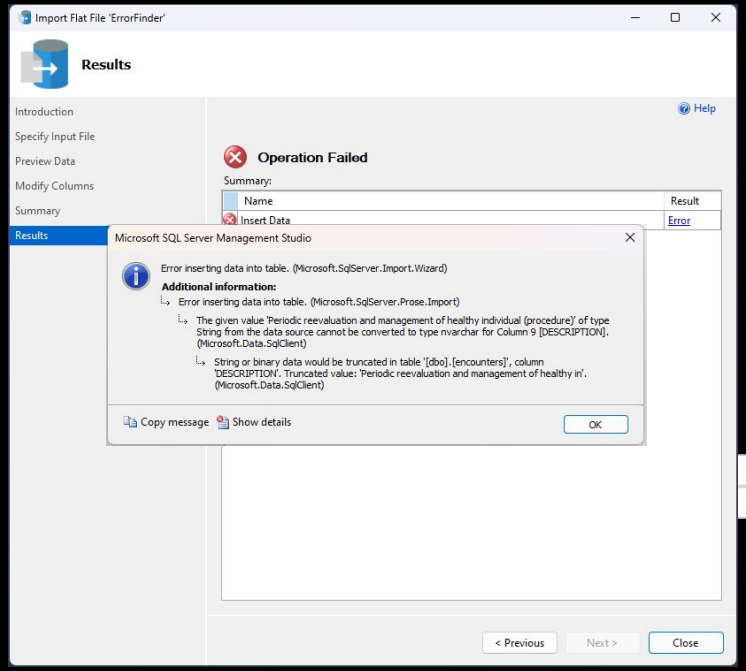
Importing EHR data



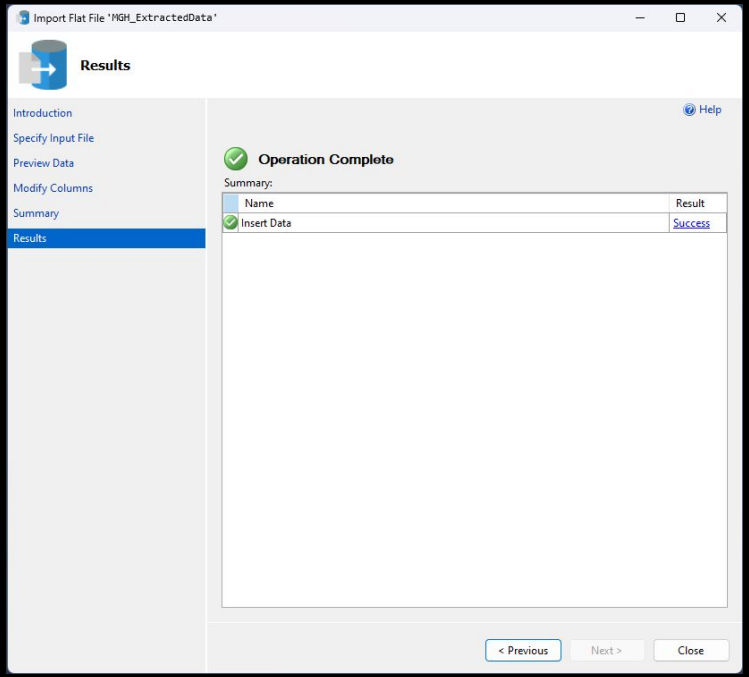
Importing EHR data



Importing EHR data



Importing EHR data



Discovering Relationships Between Tables

SQLQuery1.sql - TEAM 1\DNL(53))*

```
USE MGH_ExtractedData
GO

SELECT TOP ( ) *
FROM

SELECT *
'5259a506-b80b-3ed5-9c30-83188eabf33e'

SELECT *
FROM
WHERE    = '714b9c18-783d-4f52-aa64-cc3a05a286d9';

SELECT *
FROM
WHERE    = 'd47b3510-2895-3b70-9897-342d681c769d';

SELECT *
FROM
WHERE Id = '3a65bdb3-12b2-3247-8e8a-0e34d295c736';
```

ORGANIZATION

5259a506-b80b-3ed5-9c30-83188eabf33e

Id

5259a506-b80b-3ed5-9c30-83188eabf33e

Query executed successfully.

Massachusetts General Hospital | TEAM 1\DNL(53) | MGH_ExtractedData | 00:00:00 | 14 rows

Results | Messages

	Id	PROVIDER	PAYER	ENCOUNTERCLASS	CODE	DESCRIPTION	BASE_ENCOUNTER_COST	TOTAL_CLAIM_COST	PAYER_COVERAGE	REASONCODE	REASONDESCRIPTION
1	)0-83188eabf33e	3a65bdb3-12b2-3247-8e8a-0e34d295c736	d47b3510-2895-3b70-9897-342d681c769d	wellness	162673000	General examination of patient (procedure)	129.160003662109	129.160003662109	49.1599998474121	NULL	NULL
2	7-006a8e662e06	56c64467-dd8a-36ca-91c5-4c729475aa21	7c4411ce-02f1-39b5-b9ec-dfbea9ad3c1a	ambulatory	424619006	Prenatal visit	129.160003662109	129.160003662109	69.1600036621094	72892002	Normal pregnancy
3	)-ac43088b0c12	680f4af2-775d-34c4-b213-28791e0813e3	6e2f1a2d-27bd-3701-8d08-dae202c58632	ambulatory	371883000	Outpatient procedure	129.160003662109	129.160003662109	54.1599998474121	NULL	NULL
4	)b-5766fd987452	5735b7f6-9d26-3c4a-b6d2-c0a609678c83	7c4411ce-02f1-39b5-b9ec-dfbea9ad3c1a	wellness	162673000	General examination of patient (procedure)	129.160003662109	129.160003662109	69.1600036621094	NULL	NULL
5	)-b5a450b70fe4	f61a6a29-7081-3071-8d99-49bec2d4d22d	6e2f1a2d-27bd-3701-8d08-dae202c58632	outpatient	185349003	Encounter for check up (procedure)	129.160003662109	129.160003662109	54.1599998474121	NULL	NULL
6	15-53892bcf7ef1	793c18cd-8269-387d-b998-c135459e4248	b1c428d6-4f07-31e0-90f0-68ffa6ff8c76	ambulatory	185347001	Encounter for problem (procedure)	129.160003662109	129.160003662109	0	NULL	NULL
7	17-891a372d580f	0f365bed-2aa2-3340-974f-3fcf51fd477a	b1c428d6-4f07-31e0-90f0-68ffa6ff8c76	outpatient	698314001	Consultation for treatment	129.160003662109	129.160003662109	0	NULL	NULL
8	18-805e5ae45705	75895e9f-c056-3c5d-b845-a68d2a497f7b	047f6ec3-6215-35eb-9608f9dda363a44c	wellness	162673000	General examination of patient (procedure)	129.160003662109	129.160003662109	69.1600036621094	NULL	NULL

Discovering Relationships Between Tables

SQLQuery1.sql - TEAM 1\DNL(53))*

```
USE MGH_ExtractedData
GO

SELECT TOP (  )  *
FROM

SELECT  *
FROM
WHERE Id = '5259a506-b80b-3ed5-9c30-83188eabf33e';

SELECT  *
FROM
WHERE Id = '714b9c18-783d-4f52-aa64-cc3a05a286d9';

SELECT  *
FROM
WHERE     = 'd47b3510-2895-3b70-9897-342d681c769d';

SELECT  *
FROM
WHERE Id = '3a65bdb3-12b2-3247-8e8a-0e34d295c736';
```

PATIENT

714b9c18-783d-4f52-aa64-cc3a05a286d9

Id

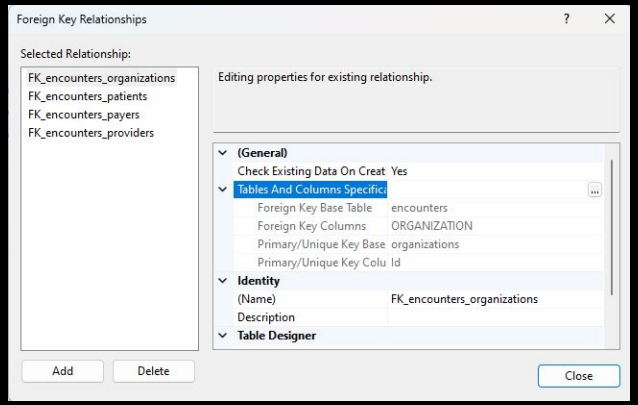
714b9c18-783d-4f52-aa64-cc3a05a286d9

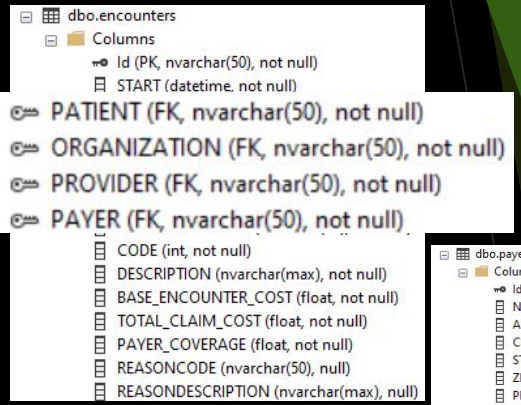
Query executed successfully.

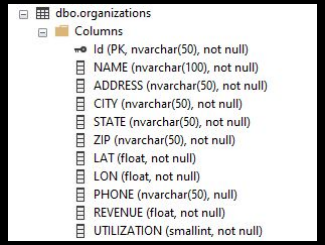
Results | Messages

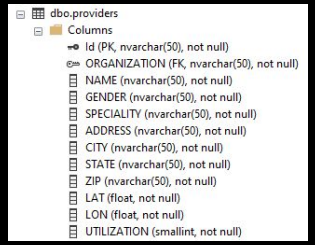
	PROVIDER	PAYER	ENCOUNTERCLASS	CODE	DESCRIPTION	BASE_ENCOUNTER_COST	TOTAL_CLAIM_COST	PAYER_COVERAGE	REASONCODE	REASONDESCRIPTION
1	3a65bdb3-12b2-3247-8e8a-0e34d295c736	d47b3510-2895-3b70-9897-342d681c769d	wellness	162673000	General examination of patient (procedure)	129.160003662109	129.160003662109	49.1599998474121	NULL	NULL
2	56c64467-dd8a-36ca-91c5-4c729475aa21	7c4411ce-02f1-39b5-b9ec-dfbea9ad3c1a	ambulatory	424619006	Prenatal visit	129.160003662109	129.160003662109	69.1600036621094	72892002	Normal pregnancy
3	680f4af2-775d-34c4-b213-28791e0813e3	6e2f1a2d-27bd-3701-8d08-dae202c58632	ambulatory	371883000	Outpatient procedure	129.160003662109	129.160003662109	54.1599998474121	NULL	NULL
4	5735b7f6-9d26-3c4a-b6d2-c0a609678c83	7c4411ce-02f1-39b5-b9ec-dfbea9ad3c1a	wellness	162673000	General examination of patient (procedure)	129.160003662109	129.160003662109	69.1600036621094	NULL	NULL
5	f61a6a29-7081-3071-8d99-49bec2d4d22d	6e2f1a2d-27bd-3701-8d08-dae202c58632	outpatient	185349003	Encounter for check up (procedure)	129.160003662109	129.160003662109	54.1599998474121	NULL	NULL
6	793c18cd-8269-387d-b998-c135459e4248	b1c428d6-4f07-31e0-90f0-68ffa6ff8c76	ambulatory	185347001	Encounter for problem (procedure)	129.160003662109	129.160003662109	0	NULL	NULL
7	0f365bed-2aa2-3340-974f-3fcf51fd477a	b1c428d6-4f07-31e0-90f0-68ffa6ff8c76	outpatient	698314001	Consultation for treatment	129.160003662109	129.160003662109	0	NULL	NULL
8	75895e9f-c056-3c5d-b845-a68d2a497f7b	047f6ec3-6215-35eb-9608f9dda363a44c	wellness	162673000	General examination of patient (procedure)	129.160003662109	129.160003662109	69.1600036621094	NULL	NULL

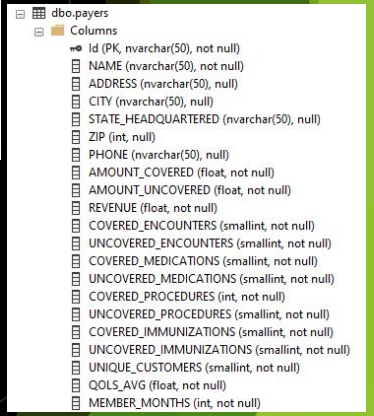
Setting Foreign Key Relationships



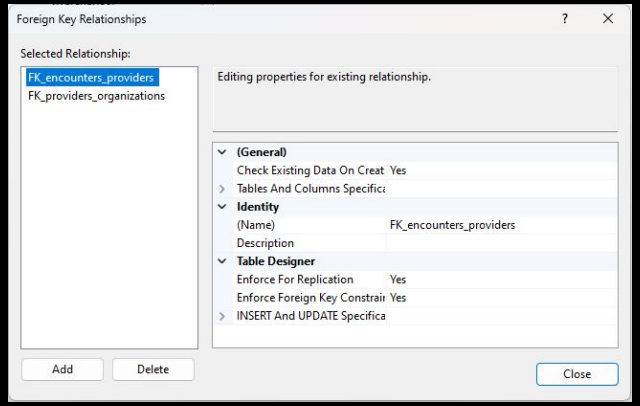


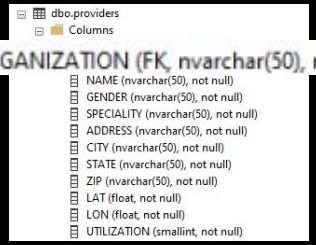


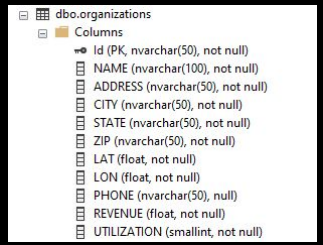


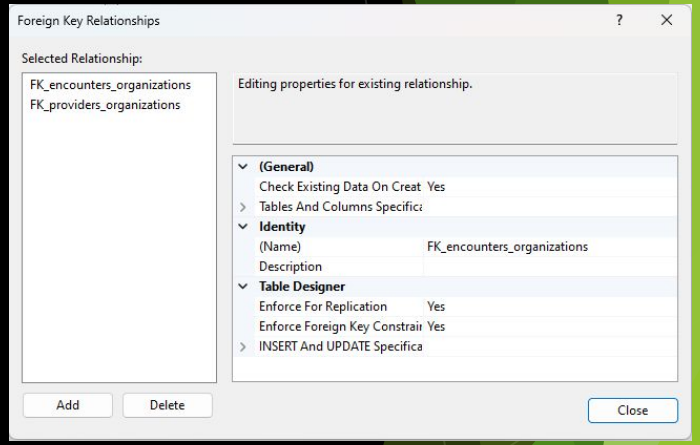


Setting Foreign Key Relationships

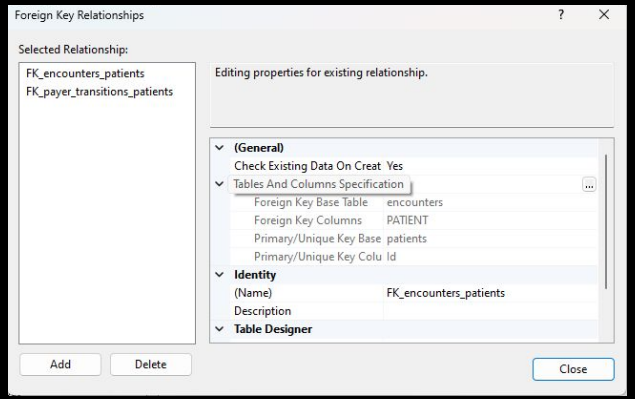


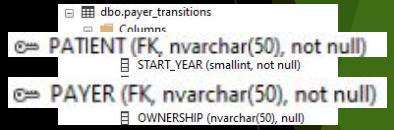


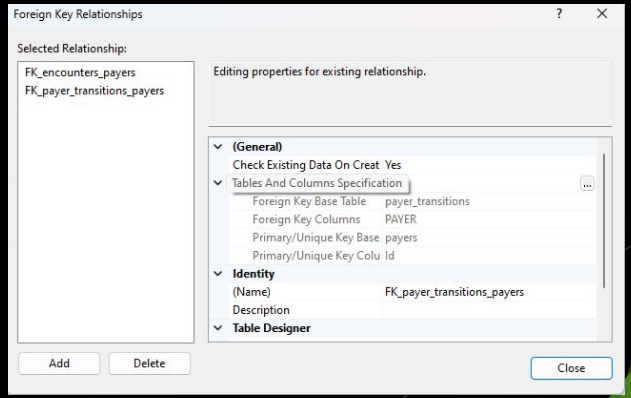


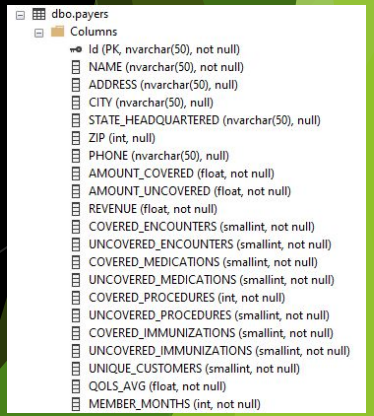


Setting Foreign Key Relationships

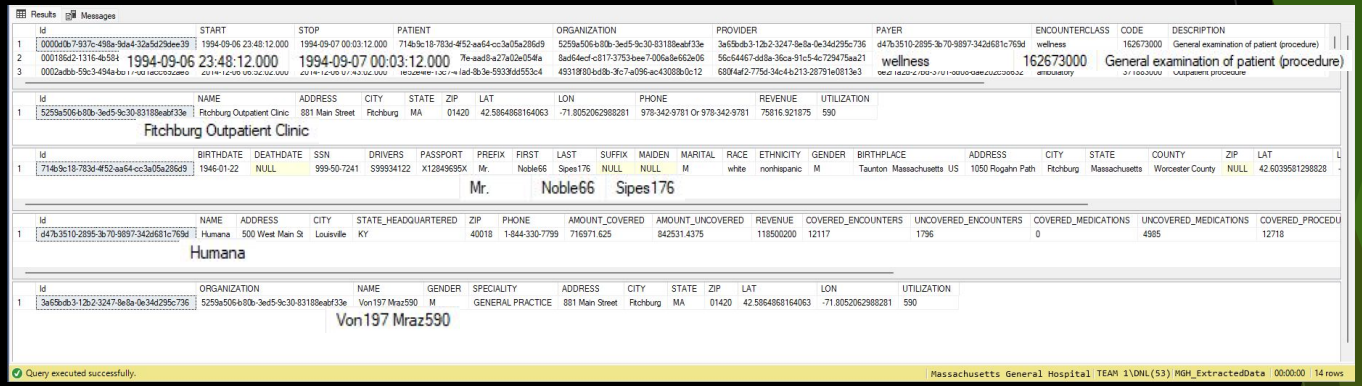








Examining Relationships Between Tables



Id	START	STOP	PATIENT	ORGANIZATION	PROVIDER	PAYER	ENCOUNTERCLASS	CODE	DESCRIPTION
1	1994-09-06 23:48:12.000	1994-09-07 00:03:12.000	714b9c18-783d-4f52-aa64-cc3a05a286d9	5259a506-b80b-3ed5-9c30-83188eabf33e	3a65bdb3-12b2-3247-8e8a-0e34d295c736	d47b3510-2895-3b70-9897-342d681c769d	wellness	162673000	General examination of patient (procedure)
2	000186d2-1316-4b58		7fe-aad8-a27a02e054fa	8ad64ecf-c817-3753-bee7-006a8e662e06	56c64467-dd8a-36ca-91c5-4c729475aa21				
3	0002adbb-59c3-494a-bb		lad-8b3e-5933fdd553c4	49318f80-bd8b-3fc7-a096-ac43088b0c12	680f4af2-775d-34c4-b213-28791e0813e3				

Id	NAME	ADDRESS	CITY	STATE	ZIP	LAT	LON	PHONE	REVENUE	UTILIZATION
1	Fitchburg Outpatient Clinic	881 Main Street	Fitchburg	MA	01420	42.5864868164063	-71.8052062988281	978-342-9781 Or 978-342-9781	75816.921875	590

Id	BIRTHDATE	DEATHDATE	SSN	DRIVERS	PASSPORT	PREFIX	FIRST	LAST	SUFFIX	MAIDEN	MARITAL	RACE	ETHNICITY	GENDER	BIRTHPLACE	ADDRESS	CITY	STATE	COUNTY	ZIP	LAT
1	1946-01-22	NULL	999-50-7241	S99934122	X12849695X	Mr.	Noble66	Sipes176	NULL	NULL	M	white	nonhispanic	M	Taunton Massachusetts US	1050 Rogahn Path	Fitchburg	Massachusetts	Worcester County	NULL	42.6039581298828

Id	NAME	ADDRESS	CITY	STATE_HEADQUARTERED	ZIP	PHONE	AMOUNT_COVERED	AMOUNT_UNCOVERED	REVENUE	COVERED_ENCOUNTERS	UNCOVERED_ENCOUNTERS	COVERED_MEDICATIONS	UNCOVERED_MEDICATIONS	COVERED_PROCEDU
1	Humana	500 West Main St	Louisville	KY	40018	1-844-330-7799	716971.625	842531.4375	118500200	12117	1796	0	4985	12718

Id	ORGANIZATION	NAME	GENDER	SPECIALITY	ADDRESS	CITY	STATE	ZIP	LAT	LON	UTILIZATION
1	5259a506-b80b-3ed5-9c30-83188eabf33e	Von197 Mraz590	M	GENERAL PRACTICE	881 Main Street	Fitchburg	MA	01420	42.5864868164063	-71.8052062988281	590

Query executed successfully. Massachusetts General Hospital | TEAM 1\DNL(53) | MGH_ExtractedData | 00:00:00 | 14 rows

	PROVIDER	PAYER	ENCOUNTERCLASS	CODE	DESCRIPTION	BASE_ENCOUNTER_COST	TOTAL_CLAIM_COST	PAYER_COVERAGE	REASONCODE	REASONDESCRIPTION
1	3a65bdb3-12b2-3247-8e8a-0e34d295c736	d47b3510-2895-3b70-9897-342d681c769d	wellness	162673000	General examination of patient (proced	129.160003662109	129.160003662109	49.1599998474121		
2	56c64467-dd8a-36ca-91c5-4c729475aa21	7c4411ce-02f1-39b5-b9ec-dfbea9ad3c1a	ambulatory	424619006	Prenatal visit					Normal pregnancy
3	680f4af2-775d-34c4-b213-28791e0813e3	6e2f1a2d-27bd-3701-8d08-dae202c58632	outpatient	371883000	Outpatient procedure	129.160003662109	129.160003662109	54.1599998474121	NULL	NULL
4	5735b7f6-9d26-3c4a-b6d2-c0a609678c83	7c4411ce-02f1-39b5-b9ec-dfbea9ad3c1a	wellness	162673000	General examination of patient (procedure)	129.160003662109	129.160003662109	69.1600036621094	NULL	NULL
5	f61a6a29-7081-3071-8d99-49bec2d4d22d	6e2f1a2d-27bd-3701-8d08-dae202c58632	outpatient	185349003	Encounter for check up (procedure)	129.160003662109	129.160003662109	54.1599998474121	NULL	NULL
6	793c18cd-8269-387d-b998-c135459e4248	b1c428d6-4f07-31e0-90f0-68ffa6ff8c76	ambulatory	185347001	Encounter for problem (procedure)	129.160003662109	129.160003662109	0	NULL	NULL
7	0f369bed-2aa2-3340-974f-3fcf51fd477a	b1c428d6-4f07-31e0-90f0-68ffa6ff8c76	outpatient	698314001	Consultation for treatment	129.160003662109	129.160003662109	0	NULL	NULL
8	75895e9f-c056-3c5d-b845-a68d2a497f7b	047f6ec3-6215-35eb-9608-f9dda363a44c	wellness	162673000	General examination of patient (procedure)	129.160003662109	129.160003662109	69.1600036621094	NULL	NULL

1994-09-06 23:48:12.000 | 1994-09-07 00:03:12.000 | Fitchburg Outpatient Clinic | wellness | 162673000 | General examination of patient (procedure) | 129.160003662109 | 129.160003662109 | 49.1599998474121

Von197 Mraz590 | Mr. | Noble66 | Sipes176 | Humana

Examining Relationships Between Tables

SQLQuery1.sql - TEAM 1\DNL(53))*

```
USE MGH_ExtractedData
GO

SELECT  FORMAT(E.START, 'D') [Encounter Date],
        FORMAT(E.START, 'hh:mm tt') [Encounter Time],
        DATEDIFF(YEAR, Pt.BIRTHDATE, E.START) [Patient Age],
        CONCAT(Pt.PREFIX, ' ', LEFT(Pt.FIRST, 5),' ', LEFT(Pt.LAST, 5)) [Patient],
        O.NAME [Location],
        E.ENCOUNTERCLASS,
        E.DESCRIPTION,
        P.NAME [Provider],
        Pay.NAME [Insurance],
        CONCAT('$', CAST(E.TOTAL_CLAIM_COST - E.PAYER_COVERAGE AS INT)) [Copay]
FROM encounters E
INNER JOIN organizations O ON O.Id = E.ORGANIZATION
INNER JOIN providers P ON P.Id = E.PROVIDER
INNER JOIN patients Pt ON Pt.Id = E.PATIENT
INNER JOIN payers Pay ON Pay.Id = E.PAYER
WHERE E.Id = '0000d0b7-937c-498a-9da4-32a5d29dee39';
```

Results | Messages

	Encounter Date	Encounter Time	Patient Age	Patient	Location	ENCOUNTERCLASS	DESCRIPTION	Provider	Insurance	Copay
1	Tuesday, September 6, 1994	11:48 PM	48	Mr. Noble Sipes	Fitchburg Outpatient Clinic	wellness	General examination of patient (procedure)	Von197 Mraz590	Humana	$80

Query executed successfully.

More Queries

SQLQuery1.sql - TEAM 1\DNL(53))*

```
USE MGH_ExtractedData
GO

SELECT  COUNT (DISTINCT (Id)) [Number of Patients],
        COUNT (DISTINCT (STATE)) [Number of States],
        COUNT (DISTINCT (CITY)) [Number of Cities],
        COUNT (DISTINCT (COUNTY)) [Number of Counties]
FROM patients;

SELECT  DISTINCT STATE,
        COUNT (Id) [Number of Patients]
FROM patients
GROUP BY STATE;

SELECT  DISTINCT CITY,
        COUNT (Id) [Number of Patients]
FROM patients
GROUP BY CITY
ORDER BY [Number of Patients];

SELECT  DISTINCT COUNTY,
        COUNT (Id) [Number of Patients]
FROM patients
GROUP BY COUNTY;
```

SQLQuery1.sql - TEAM 1\DNL(53))*

```
USE MGH_ExtractedData
GO

- Amount of patients and average Healthcare expense by race
SELECT  Pt.RACE,
        COUNT(DISTINCT Pt.Id) [Number of Patients]
FROM encounters E
INNER JOIN patients Pt ON Pt.Id = E.PATIENT
WHERE E.PAYER_COVERAGE = 0
GROUP BY Pt.RACE
ORDER BY Pt.RACE;

SELECT  Pt.RACE,
        AVG(E.TOTAL_CLAIM_COST) [Average Healthcare Expense]
FROM encounters E
INNER JOIN patients Pt ON Pt.Id = E.PATIENT
WHERE E.PAYER_COVERAGE = 0
GROUP BY Pt.RACE
ORDER BY Pt.RACE;
```

SQLQuery1.sql - TEAM 1\DNL(53))*

```
USE MGH_ExtractedData
GO

- Average Healthcare expense by gender
SELECT  Pt.GENDER,
        AVG(E.TOTAL_CLAIM_COST) [Avg_Healthcare_Expense]
FROM encounters E
INNER JOIN patients Pt ON Pt.Id = E.PATIENT
GROUP BY Pt.GENDER;
```

Results | Messages

	GENDER	Avg_Healthcare_Expense
1	F	128.760339789213
2	M	128.740081105064

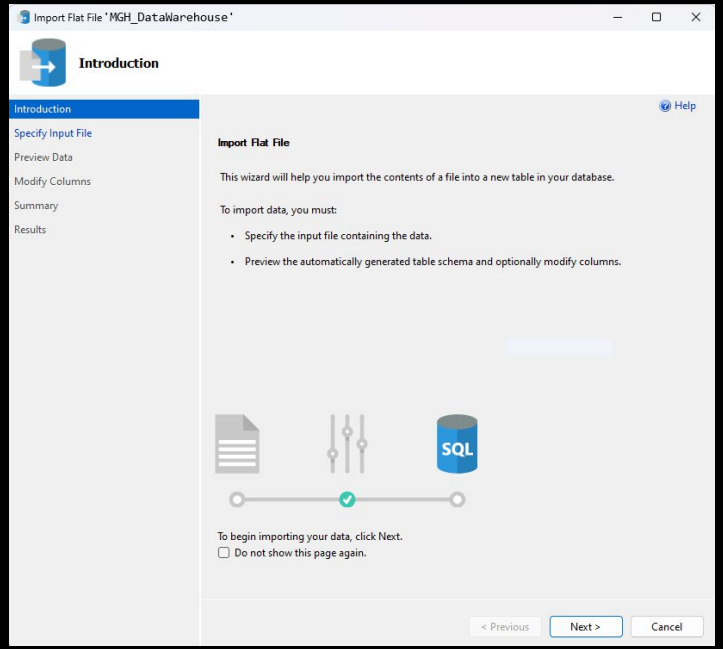
Results | Messages

	RACE	avg_healthcare_expense
1	asian	128.926598779903
2	black	129.050234302295
3	white	128.76438702203
4	native	126.282879006672
5	other	129.160003662109

Duplicating the Original Tables

Empty Duplicate.sql - TEAM 1\DNL(53))*

```
USE master
GO

CREATE DATABASE MGH_DataWarehouse;
```



Empty Duplicate.sql - TEAM 1\DNL(53))*

```
USE MGH_DataWarehouse
GO

DELETE FROM encounters;

DELETE FROM organizations;

DELETE FROM patients;

DELETE FROM payer_transitions;

DELETE FROM payers;

DELETE FROM providers;
```

Data Analysis

SQLQuery1.sql - TEAM 1\DNL(53))*

```
USE MGH_ExtractedData
GO

SELECT  COUNT (Id) [Number of Patients],
        COUNT (DISTINCT (STATE)) [Number of States],
        COUNT (DISTINCT (CITY)) [Number of Cities],
        COUNT (DISTINCT (COUNTY)) [Number of Counties]
FROM patients;

SELECT  DISTINCT STATE,
        COUNT (Id) [Number of Patients]
FROM patients
GROUP BY STATE;

SELECT  DISTINCT CITY,
        COUNT (Id) [Number of Patients]
FROM patients
GROUP BY CITY
ORDER BY [Number of Patients];

SELECT  DISTINCT COUNTY,
        COUNT (Id) [Number of Patients]
FROM patients
GROUP BY COUNTY;
```

Results | Messages

	Number of Patients	Number of States	Number of Cities	Number of Counties
1	1171	1	237	14

Query executed successfully.

Results | Messages

	CITY	Number of Patients
1	Acushnet	1
2	Adams	1
3	Andover	1
4	Ashburnham	1
5	Athol	1
6	Avon	1
7	Belmont	1
8	Bourne	1
9	Carver	1
10	Cheshire	1
11	Chester	1
12	Cohasset	1
13	Colrain	1
14	Dalton	1
15	Deerfield	1
16	Douglas	1
17	Dudley	1
18	East Bridgewater	1
19	East Brookfield	1
20	Easton	1
21	Essex	1
22	Falmouth	1
23	Freetown	1
24	Gill	1
25	Great Barrington	1
26	Groveland	1
27	Halifax	1
28	Hanson	1
29	Harwich	1
30	Holbrook	1
31	Holden	1
32	Holland	1
33	Lancaster	1
34	Lenox	1
35	Leyden	1
36	Littleton	1
37	Leverett	1
38	Manchester-by-...	1
39	Mashpee	1
40	Mattapoisett	1
41	Lynnfield	1

Query executed successfully.

Results | Messages

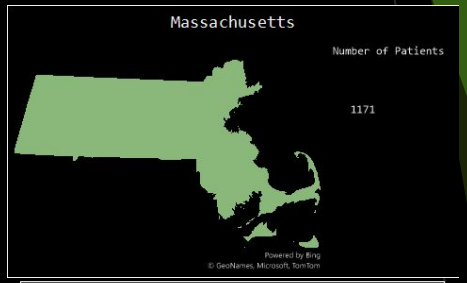
	STATE	Number of Patients
1	Massachusetts	1171

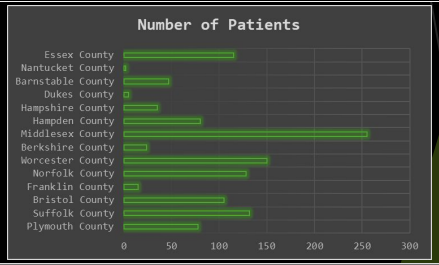
Query executed successfully.

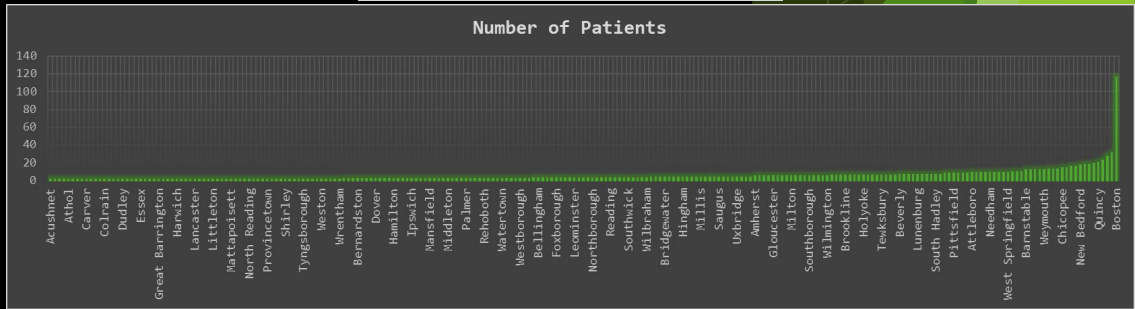
Results | Messages

	COUNTY	Number of Patients
1	Plymouth County	78
2	Suffolk County	132
3	Bristol County	105
4	Franklin County	15
5	Norfolk County	128
6	Worcester County	150
7	Berkshire County	24
8	Middlesex County	255
9	Hampden County	80
10	Hampshire County	35
11	Dukes County	5
12	Barnstable County	47
13	Nantucket County	2
14	Essex County	115

Query executed successfully.







Data Analysis

SQLQuery1.sql - TEAM 1\DNL(53))*

```
USE MGH_ExtractedData
GO

SELECT  COUNT(Id) [Providers in Providers Table],
        COUNT(DISTINCT Speciality) [Specialities in Providers Table]
FROM providers;

SELECT  COUNT(DISTINCT E.PROVIDER) [Providers in Encounters Table],
        COUNT(DISTINCT Pro.SPECIALITY) [Specialities in Encounters Table]
FROM encounters E
INNER JOIN providers Pro ON Pro.Id = E.PROVIDER

SELECT  COUNT(DISTINCT ORGANIZATIONS) [Organizations in Encounters Table]
FROM encounters;

SELECT  REASONDESCRIPTION,
        COUNT (Id) [Number of Encounters]
FROM encounters
GROUP BY REASONDESCRIPTION
ORDER BY REASONDESCRIPTION;
```

Results | Messages

	Providers in Providers Table	Specialities in Providers Table
1	5855	69

	Providers in Encounters Table	Specialities in Encounters Table
1	1104	1

Query executed successfully.

Results | Messages

	Organizations in Encounters Table
1	1103

✔ Which county has the highest healthcare expenses?

SQLQuery1.sql - TEAM 1\DNL(53))*

```
USE MGH_ExtractedData
GO

SELECT  Pt.COUNTY,
        SUM(E.TOTAL_CLAIM_COST) [Total Claims]
FROM encounters E
INNER JOIN patients Pt ON Pt.Id = E.PATIENT
GROUP BY Pt.COUNTY
ORDER BY [Total Claims] DESC;


SELECT  COUNTY,
        SUM(HEALTHCARE_EXPENSES) [Total Healthcare Expenses]
FROM patients
GROUP BY COUNTY
ORDER BY [Total Healthcare Expenses] DESC;
```

Results | Messages

	COUNTY	Total Claims
1	Middlesex County	1287027.58635712
2	Worcester County	996649.958045959
3	Suffolk County	882059.214805603
4	Norfolk County	872449.914680481
5	Essex County	614517.117172241
6	Bristol County	531234.614715576
7	Plymouth County	414035.241691589
8	Hampden County	408352.141487122
9	Barnstable County	324759.348800659
10	Hampshire County	218771.166168213
11	Franklin County	132673.113731384
12	Berkshire County	95500.8826904297
13	Dukes County	75558.602142334
14	Nantucket County	14724.2404174805

Query executed successfully.

Results | Messages

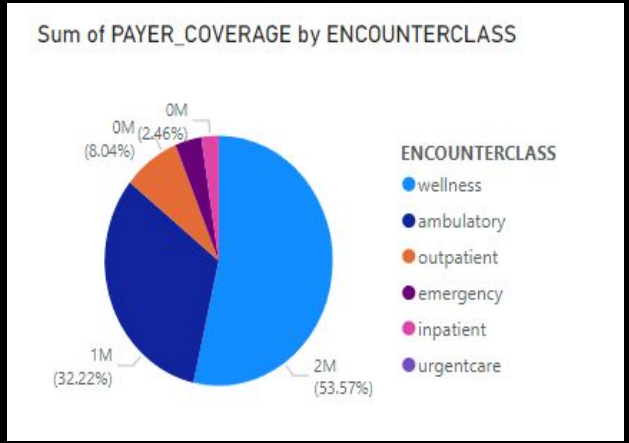
	COUNTY	Total Healthcare Expenses
1	Middlesex County	187484541.057129
2	Worcester County	131865518.449219
3	Norfolk County	105687814.407227
4	Bristol County	87750366.7563477
5	Essex County	86420286.260498
6	Suffolk County	81792734.3118896
7	Plymouth County	61923879.0559082
8	Hampden County	48784674.9926758
9	Barnstable County	37698127.2490234
10	Hampshire County	32963319.796875
11	Berkshire County	14051454.1708984
12	Franklin County	9131916.296875
13	Dukes County	8224328.5
14	Nantucket County	1966835.25

Query executed successfully.

✔ What is the total coverage per encounter class?

SQLQuery1.sql - TEAM 1\DNL(53))*

```
USE MGH_ExtractedData
GO

--Which encounter classes are covered the least by payers?
SELECT  E.ENCOUNTERCLASS,
        AVG(E.PAYER_COVERAGE) [avg_payer_coverage]
FROM encounters E
GROUP BY E.ENCOUNTERCLASS
ORDER BY avg_payer_coverage;
```



✔ Which Provider has the most encounters recorded?

SQLQuery1.sql - TEAM 1\DNL(53))*

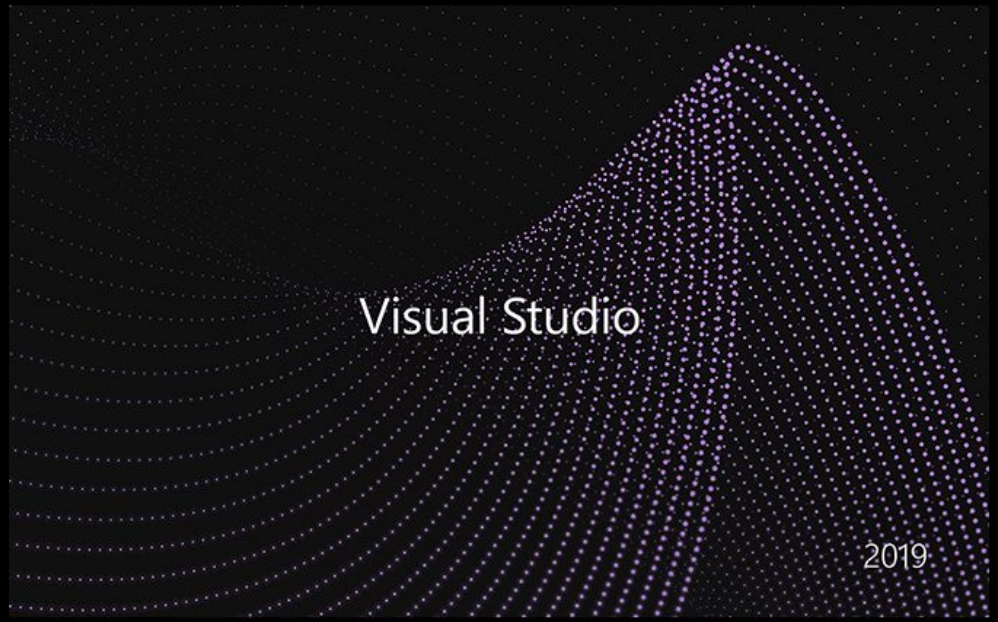
```
USE MGH_ExtractedData
GO

SELECT  Pro.NAME [Name of Provider],
        O.NAME [Organization],
        COUNT(E.Id) [Number of Encounters]
FROM providers Pro
INNER JOIN encounters E ON E.PROVIDER = Pro.Id
INNER JOIN organizations O ON O.Id = Pro.ORGANIZATION
GROUP BY Pro.NAME, O.NAME
ORDER BY [Number of Encounters] DESC;
```

Results | Messages

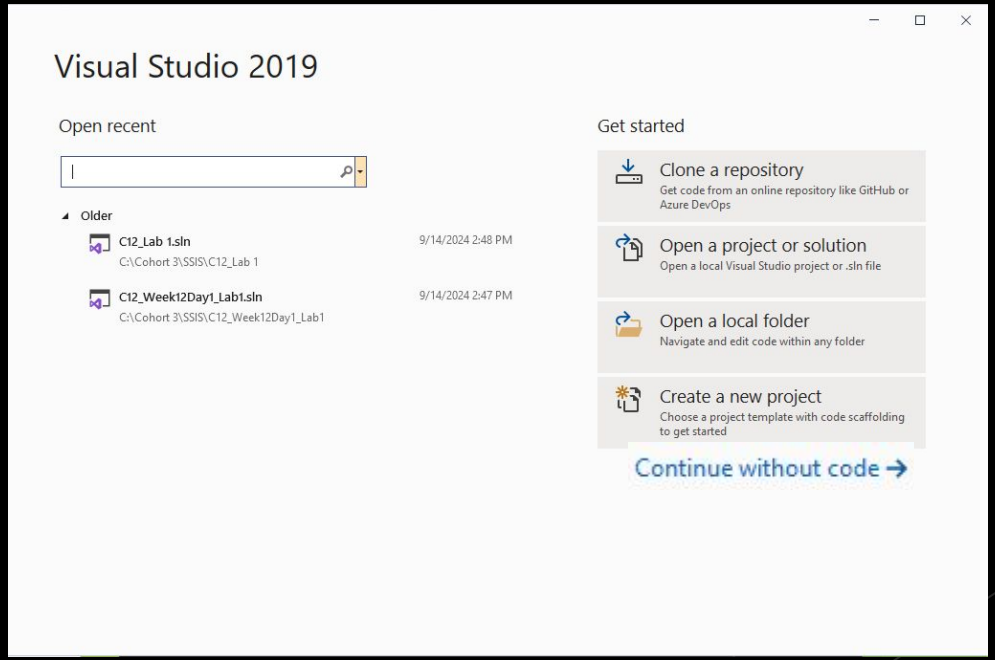
	ENCOUNTERCLASS	avg_payer_coverage
1	urgentcare	0
2	outpatient	29.5304730491719
3	inpatient	44.2587445630602
4	ambulatory	56.2987746350313
5	emergency	58.8934183294123
6	wellness	92.6944630895499

	provider_name	organization_name	encounter_count
1	Gaynell126 Streich926	VA Boston Healthcare System Jamaica Plain Campus	3217
2	Gertrudis163 Schaden604	Worcester Outpatient Clinic	1972
3	Vern731 Powlowski563	HALLMARK HEALTH SYSTEM	1658
4	Jeanmarie510 Beatty507	CAMBRIDGE HEALTH ALLIANCE	1028
5	Maile198 Frami345	NEWTON-WELLESLEY HOSPITAL	955
6	Luke971 Rath779	NORWOOD HOSPITAL	859
7	Ken25 Schmidt332	METROWEST MEDICAL CENTER	857
8	Lonna614 Dietrich576	MILFORD REGIONAL MEDICAL CENTER	824
9	Garth972 Wyman904	HOLYOKE MEDICAL CENTER	783
10	Gaynell126 Streich926	STURDY MEMORIAL HOSPITAL	763
11	Bryant814 Murazik203	NANTUCKET COTTAGE HOSPITAL	666
12	Zana914 Considine820	LAHEY HOSPITAL & MEDICAL CENTER BURLIN...	662
13	Suzette512 Monahan736	MOUNT AUBURN HOSPITAL	646
14	Guadalupe206 Valenzue...	COOLEY DICKINSON HOSPITAL INC THE	625
15	Gertrudis163 Schaden604	SIGNATURE HEALTHCARE BROCKTON HOSPIT...	620
16	Sanford861 Gottlieb798	BETH ISRAEL DEACONESS HOSPITAL-MILTON I...	609
17	Houston994 Osinski784	NORTH SHORE MEDICAL CENTER -	604
18	Myong12 Heidenreich818	SOUTHCOAST HOSPITAL GROUP INC	581
19	Phillip440 McCullough561	CAPE COD HOSPITAL	561
20	Veda284 Pfeffer420	LOWELL GENERAL HOSPITAL	554

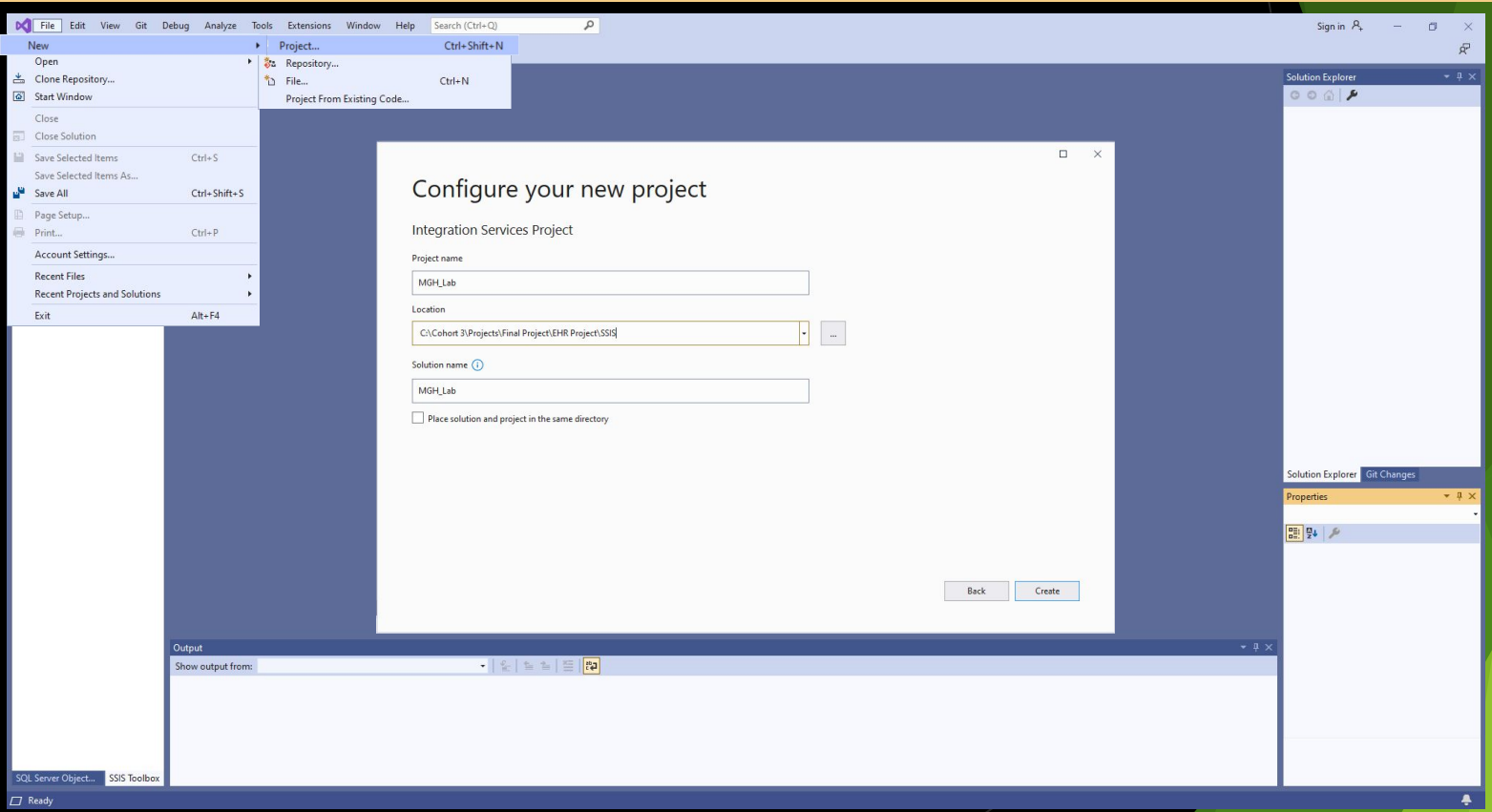
SSIS


Visual Studio
2019

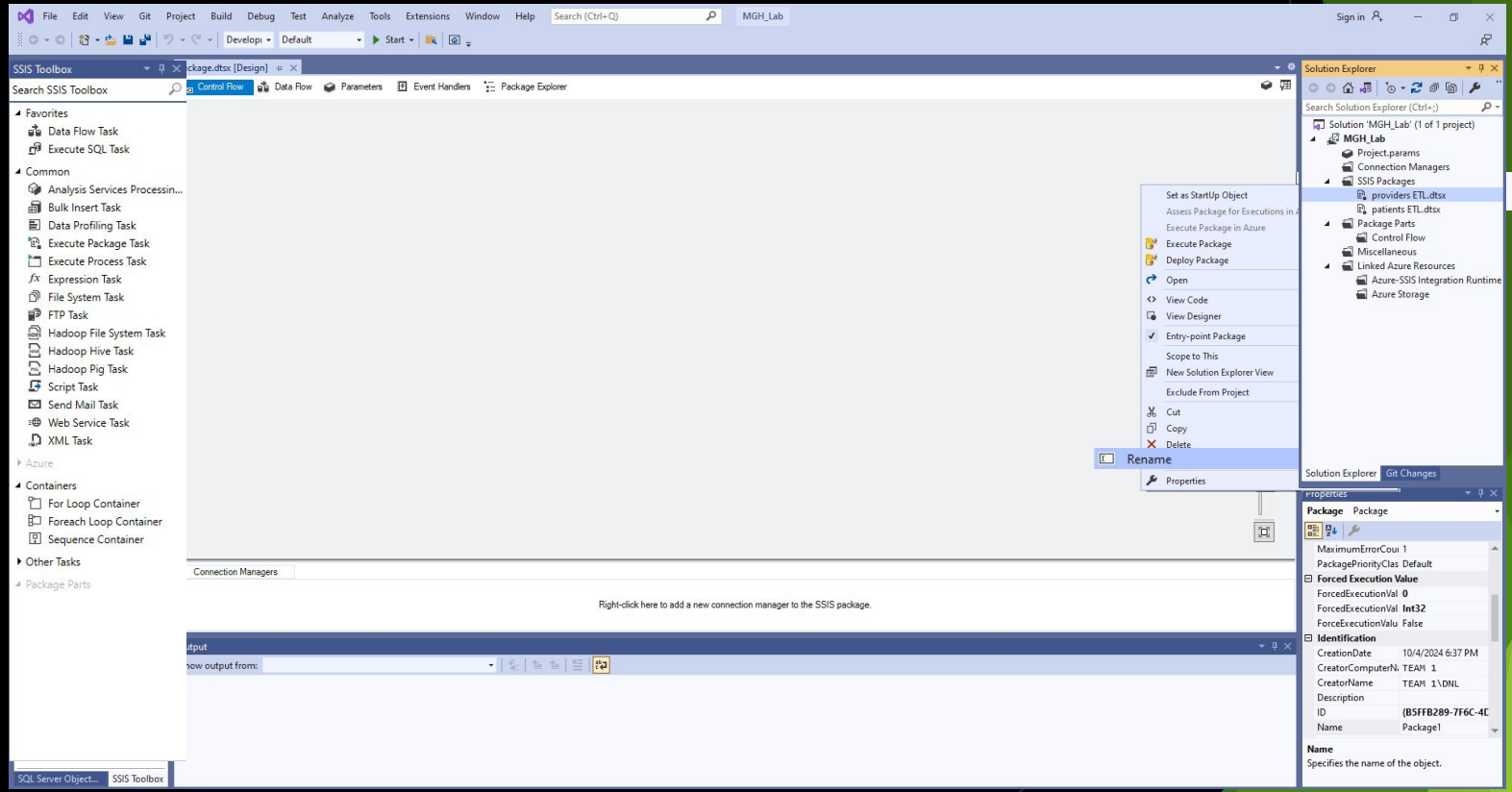
Start New SSIS Project



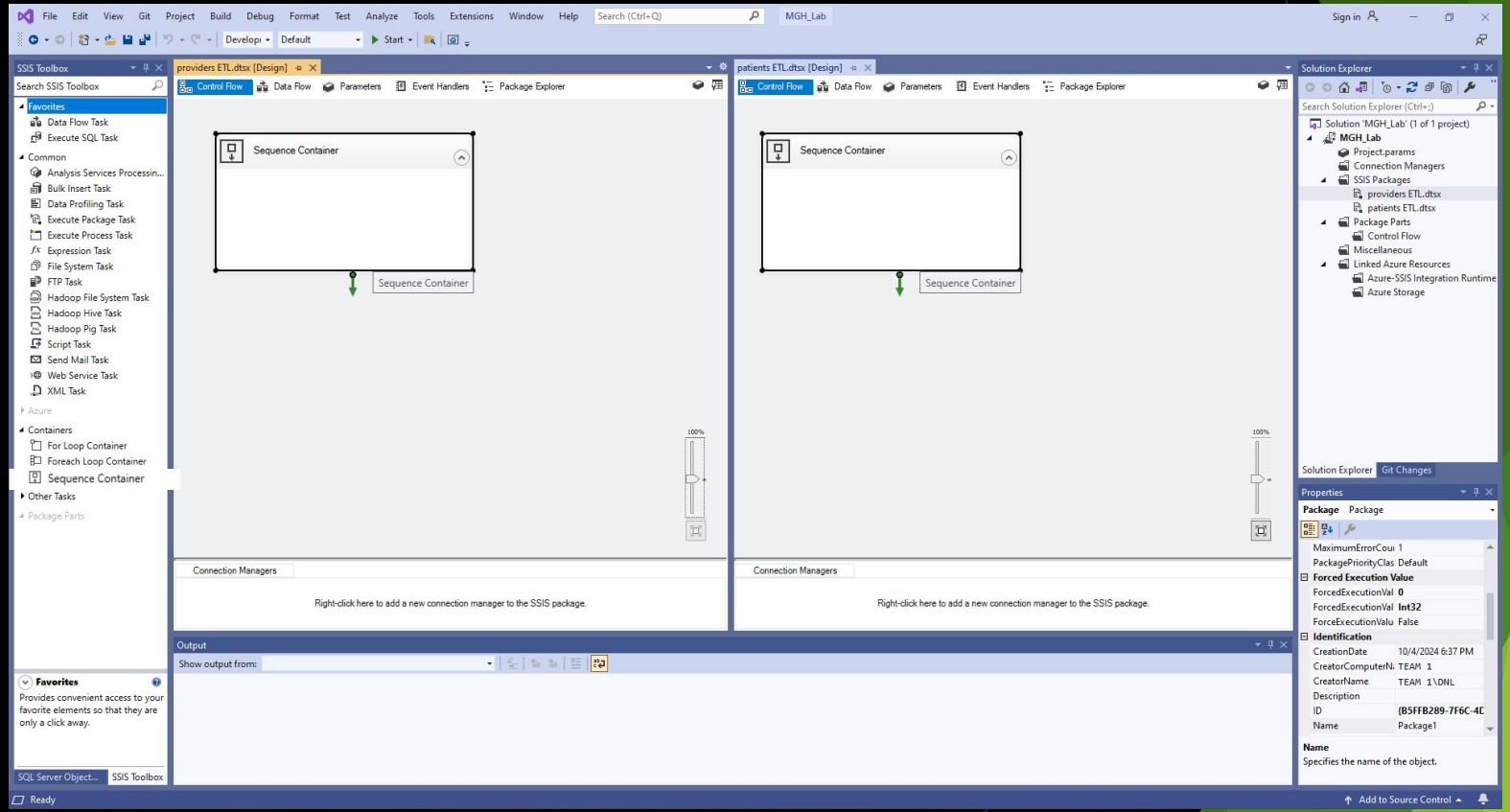
Start New SSIS Project



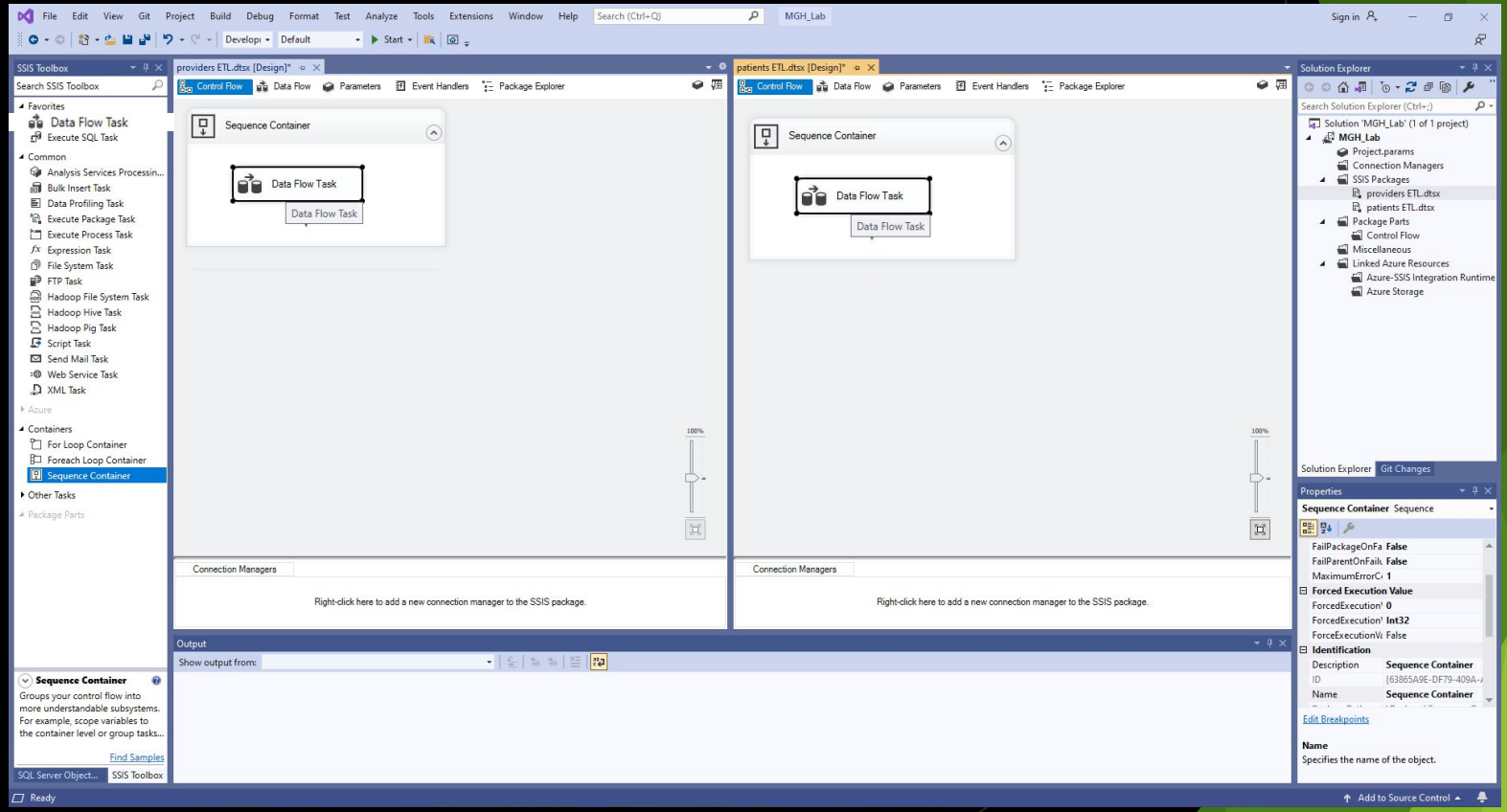
Rename Packages



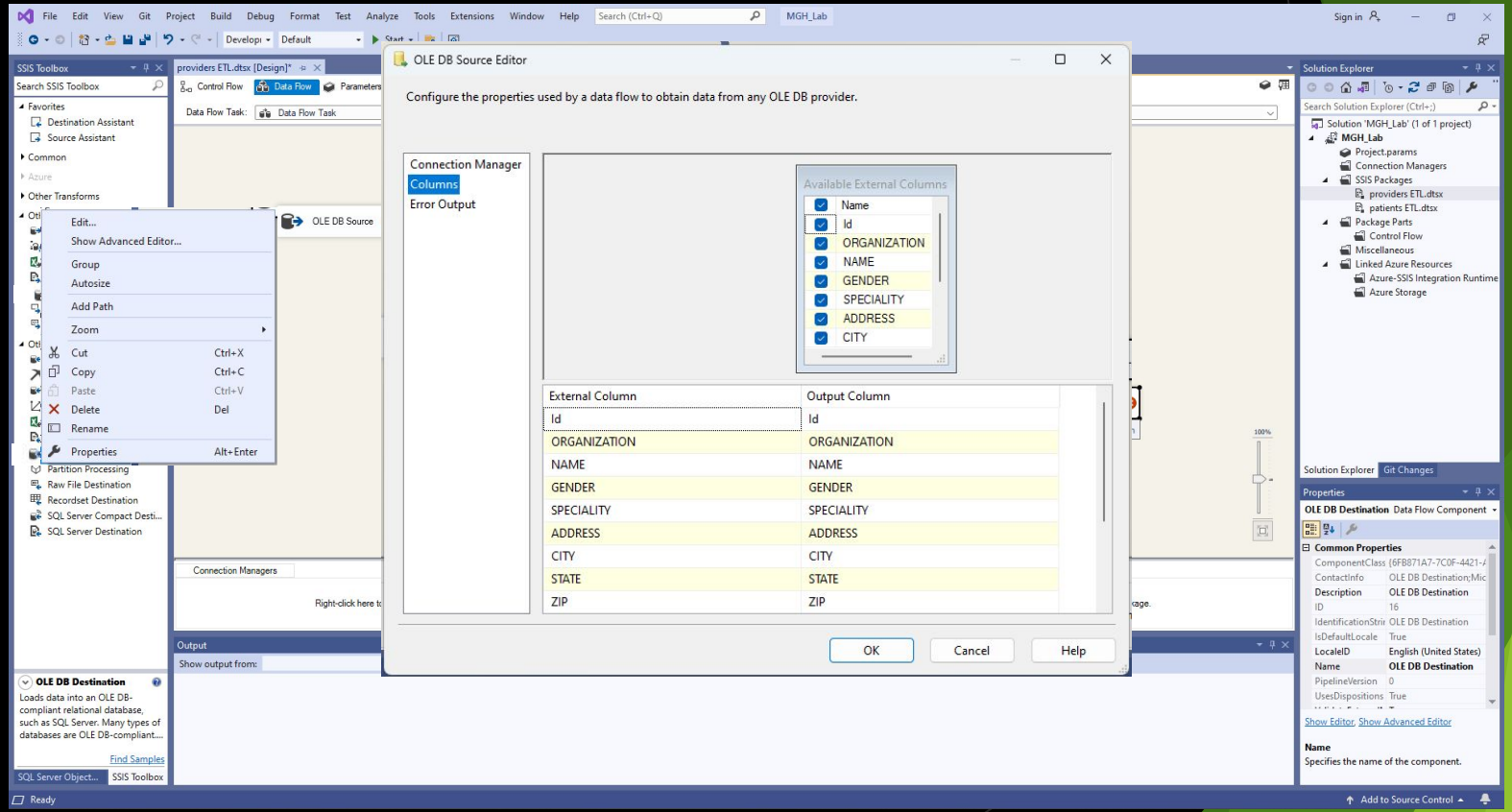
Create Sequence Containers



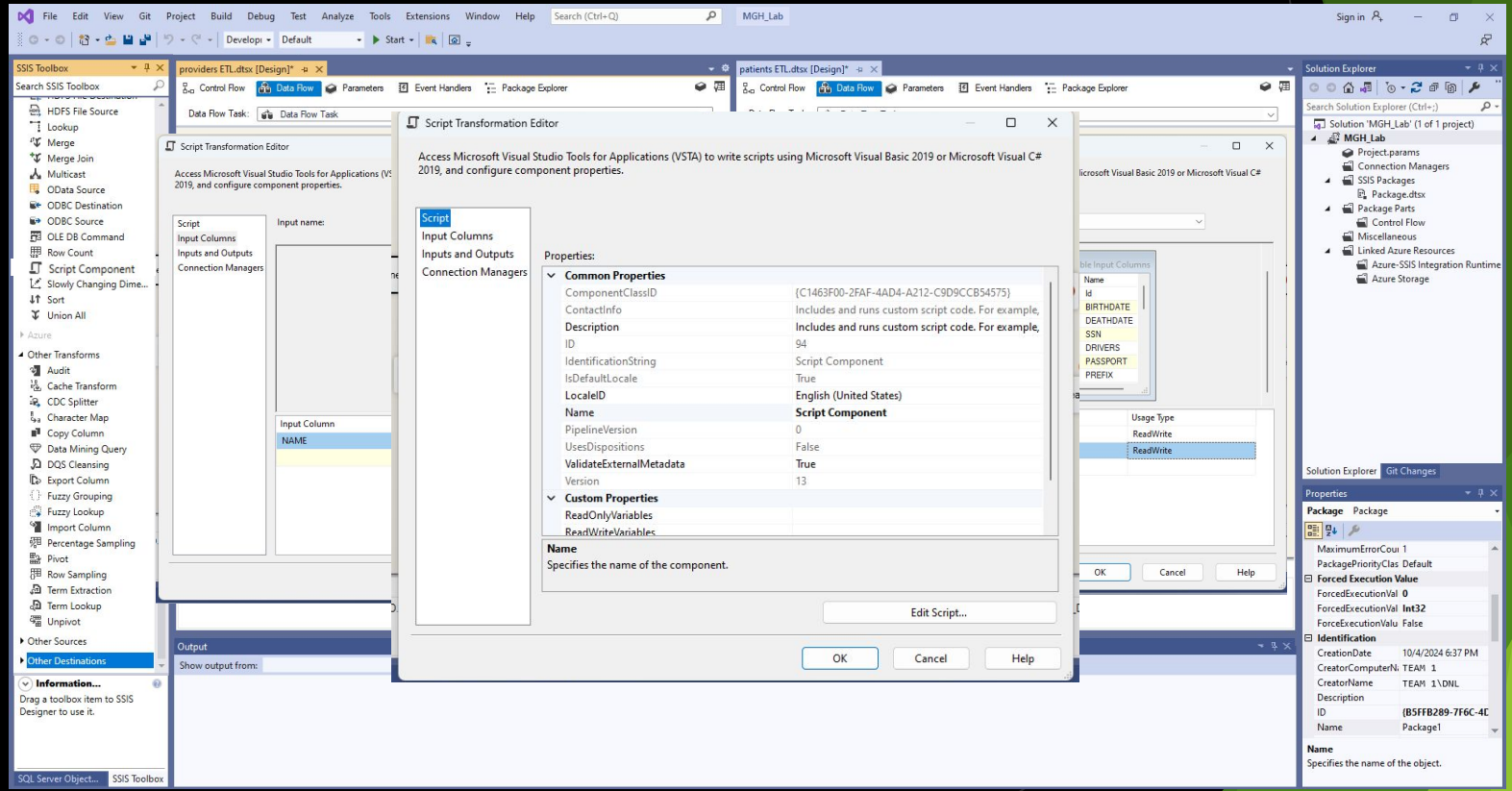
Create Data Flows



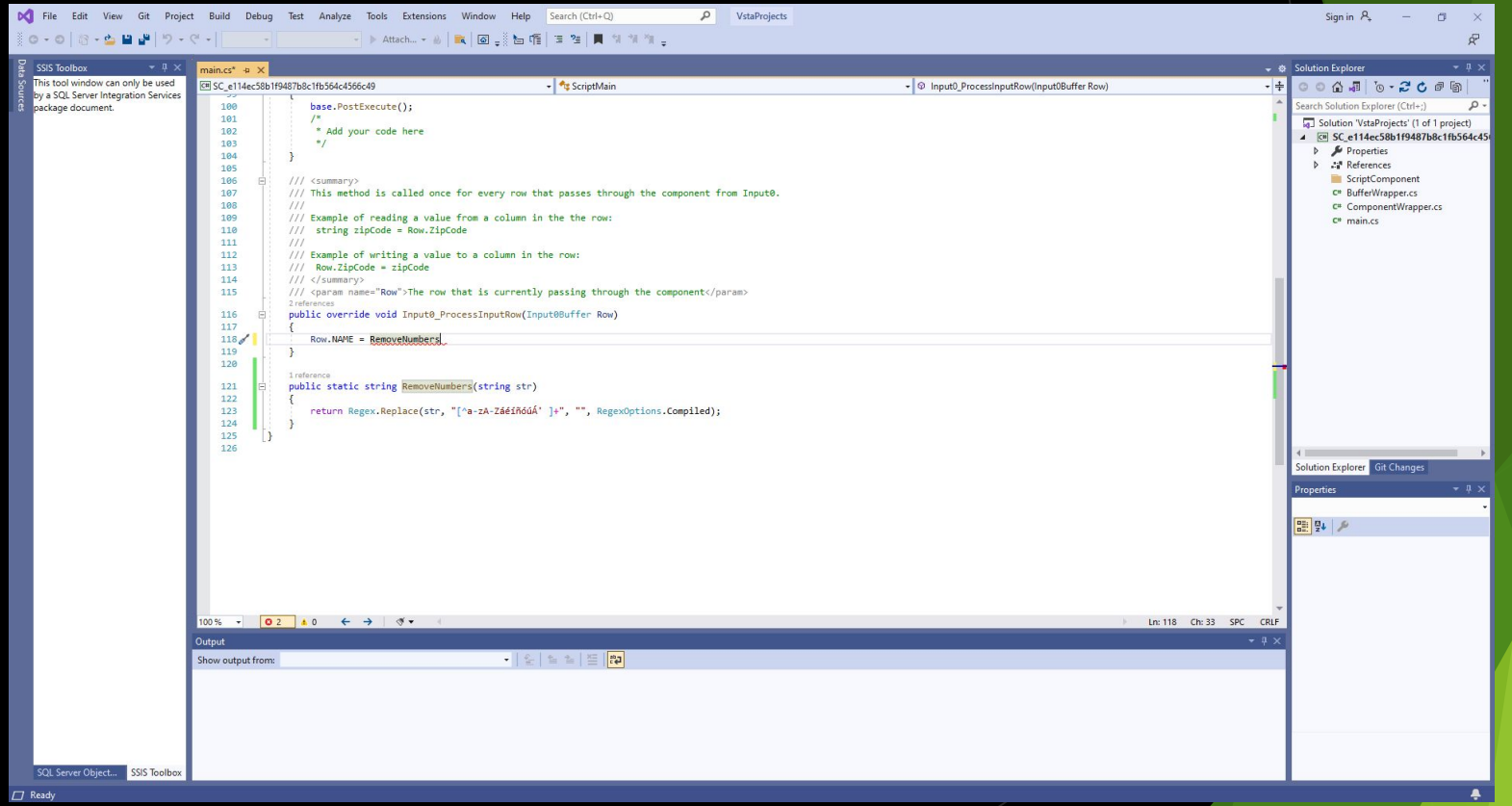
Set up Connections

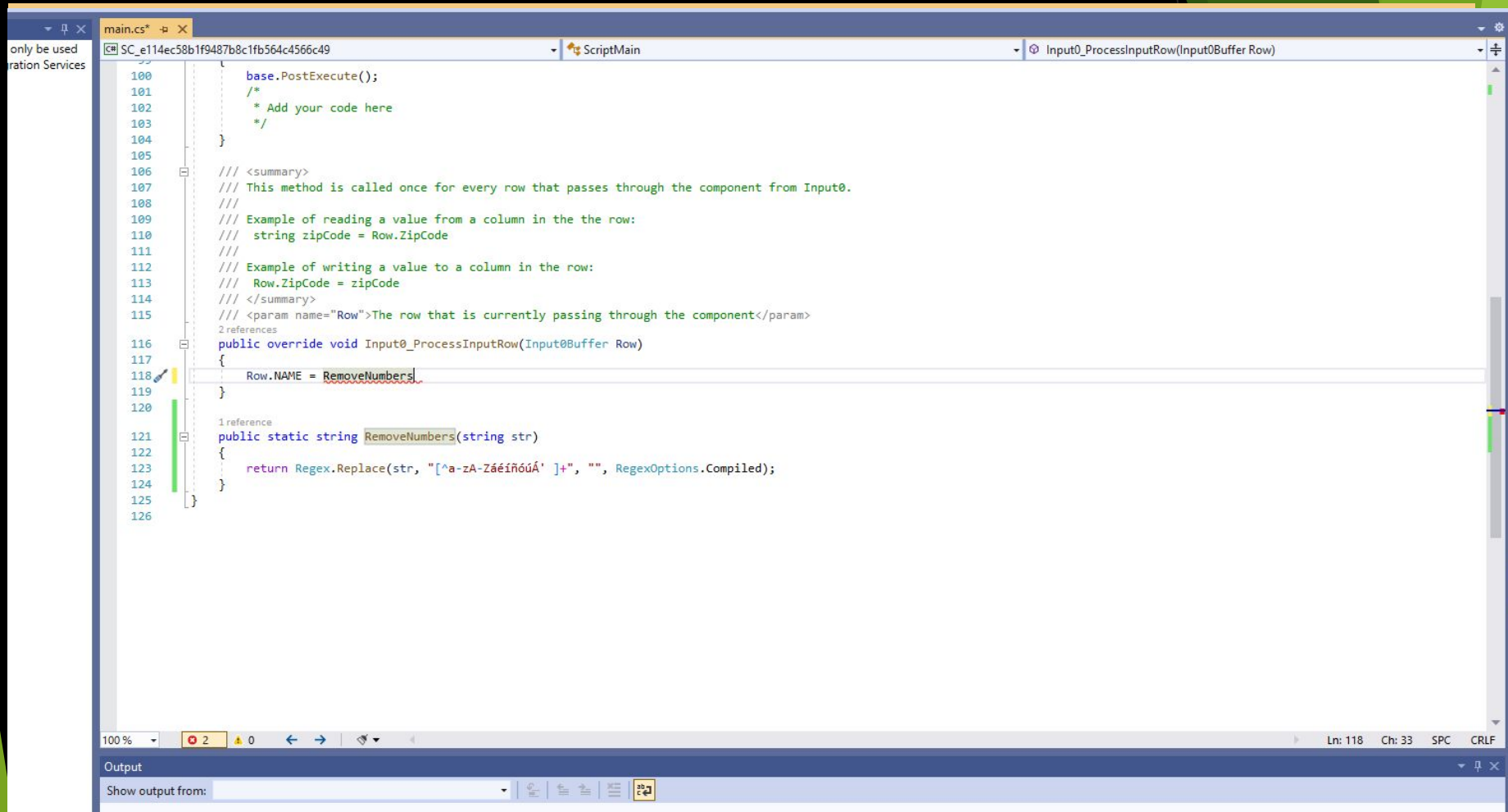


Create Script Components

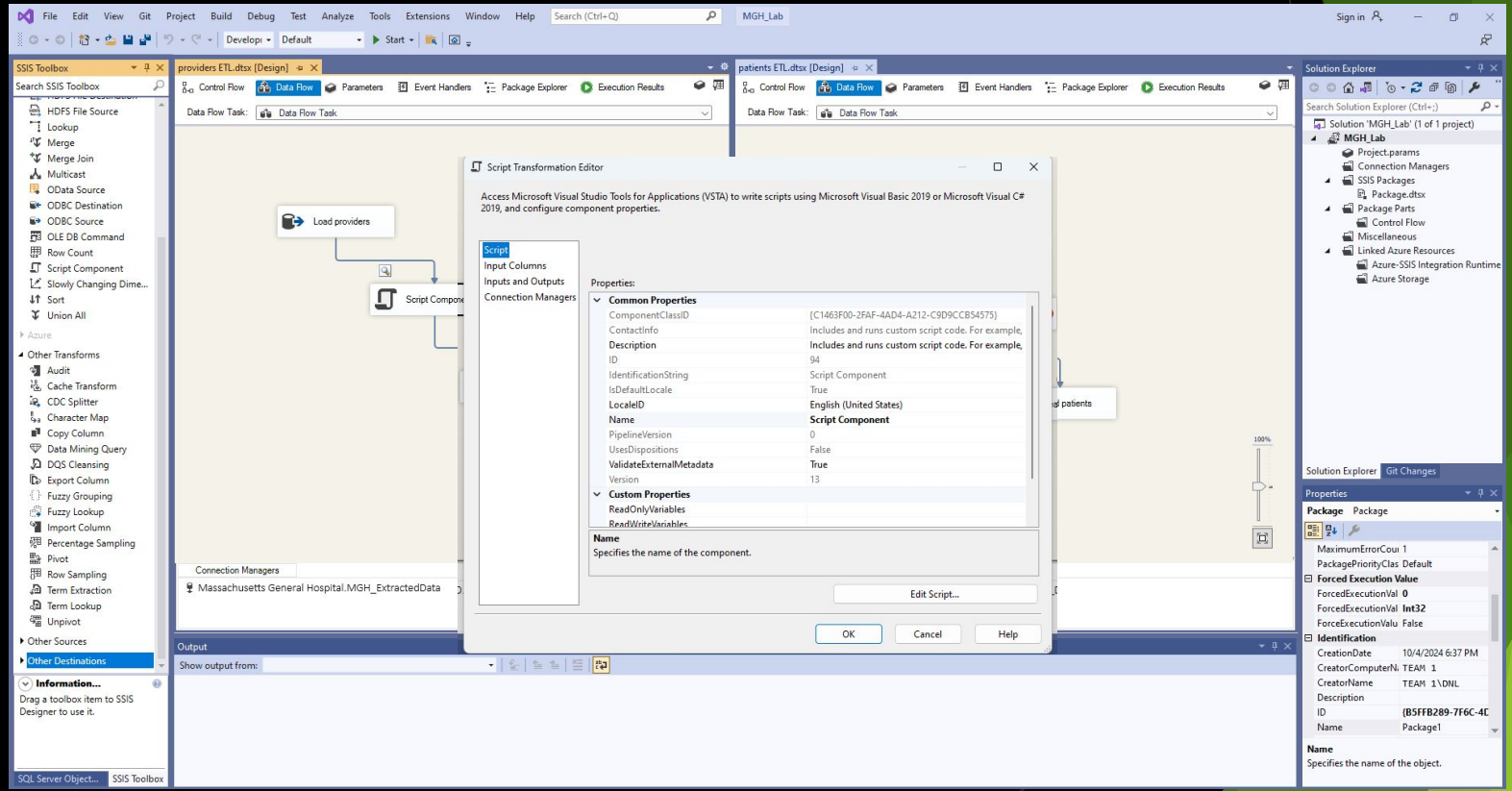


Enter Code & Objects

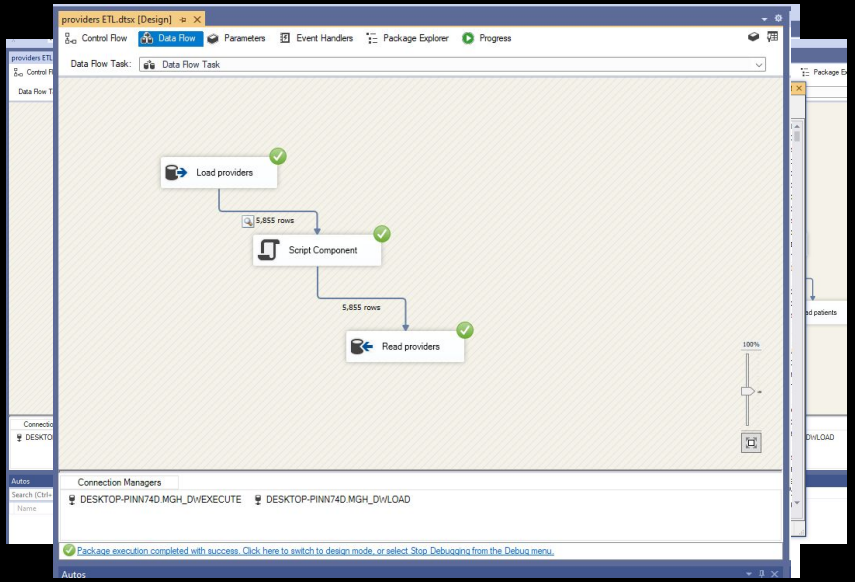


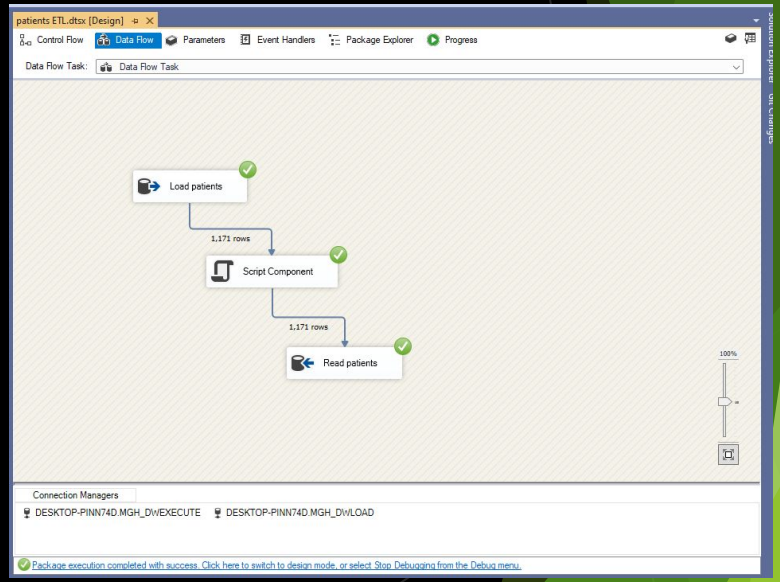

main.cs*
SC_e114ec58b1f9487b8c1fb564c4566c49
ScriptMain
Input0_ProcessInputRow(Input0Buffer Row)
base.PostExecute();
/*
* Add your code here
*/
}
/// <summary>
/// This method is called once for every row that passes through the component from Input0.
///
/// Example of reading a value from a column in the the row:
/// string zipCode = Row.ZipCode
///
/// Example of writing a value to a column in the row:
/// Row.ZipCode = zipCode
/// </summary>
/// <param name="Row">The row that is currently passing through the component</param>
2 references
public override void Input0_ProcessInputRow(Input0Buffer Row)
{
Row.NAME = RemoveNumbers
}
1 reference
public static string RemoveNumbers(string str)
{
return Regex.Replace(str, "[^a-zA-ZáéíñóúÁ']+", "", RegexOptions.Compiled);
}
}
100 %
2
0
Ln: 118
Ch: 33
SPC
CRLF
Output
Show output from:

Verify Effects



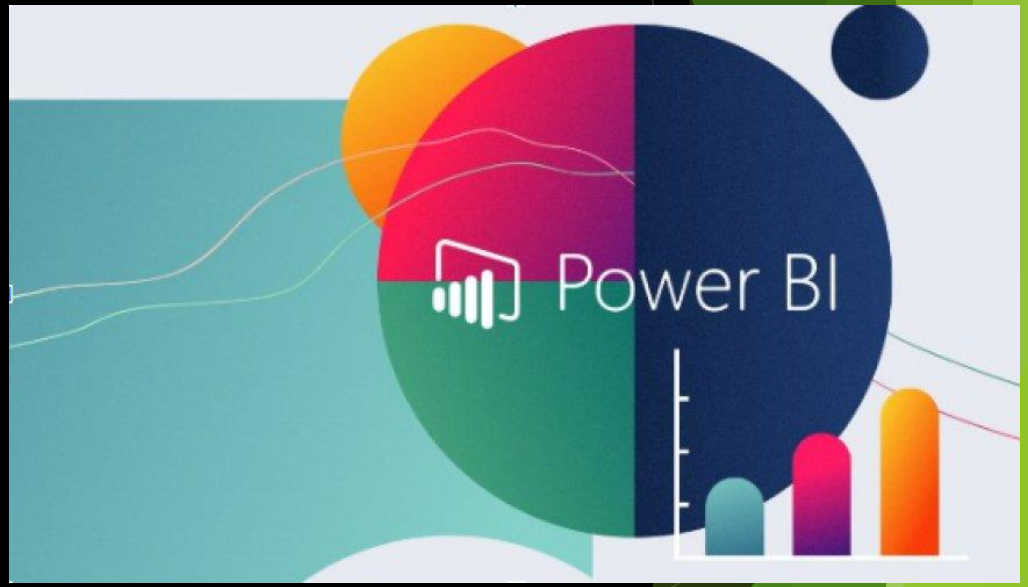
Execute Packages





POWER BI


SQL

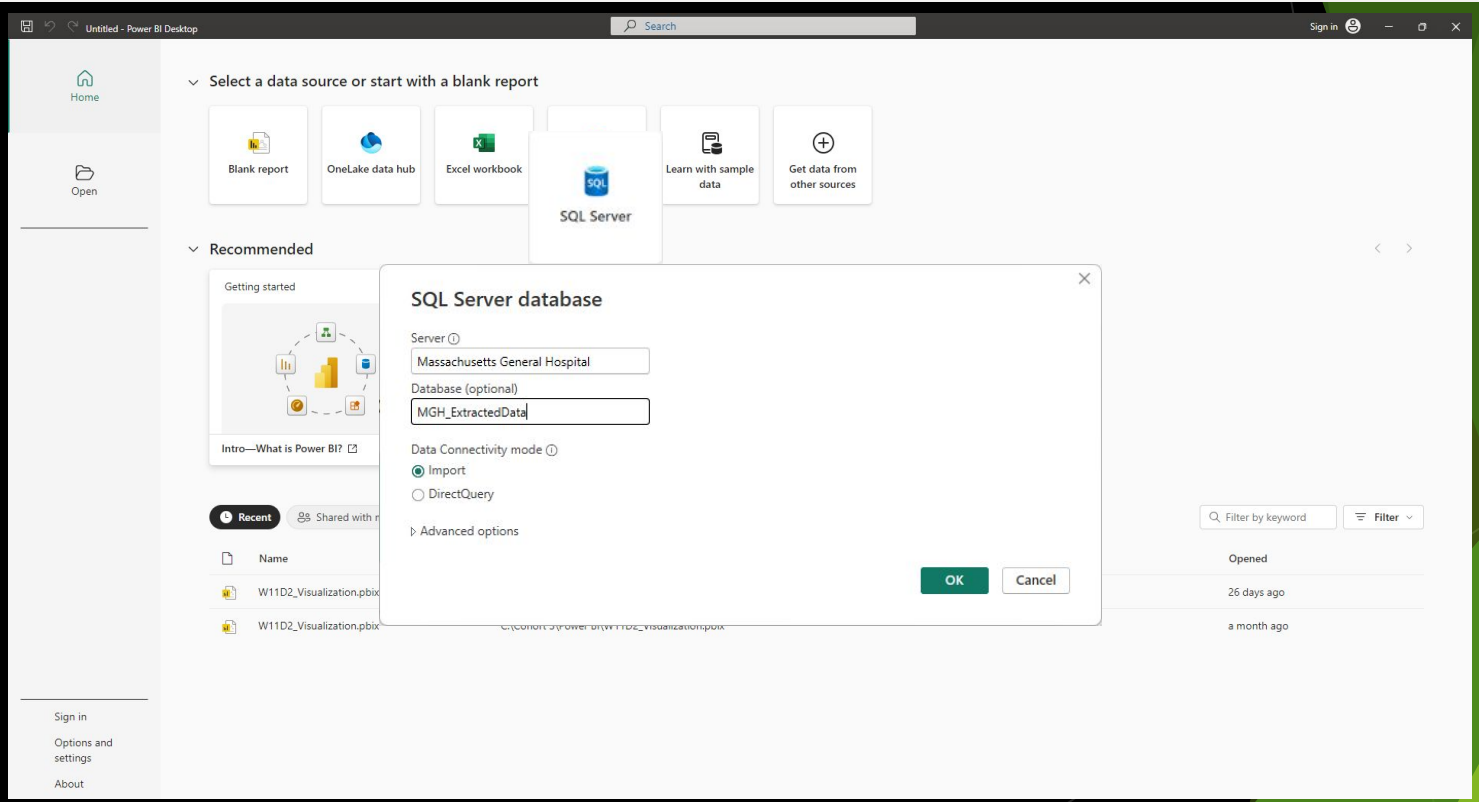

Power BI

Moving data from SQL to power BI

1. Connect Power BI to SQL database
2. Choose data to import
3. Transform data (optional)- power BI will open the power query editor –(filter, remove/ add columns, change data types)
4. Create reports and visualizations

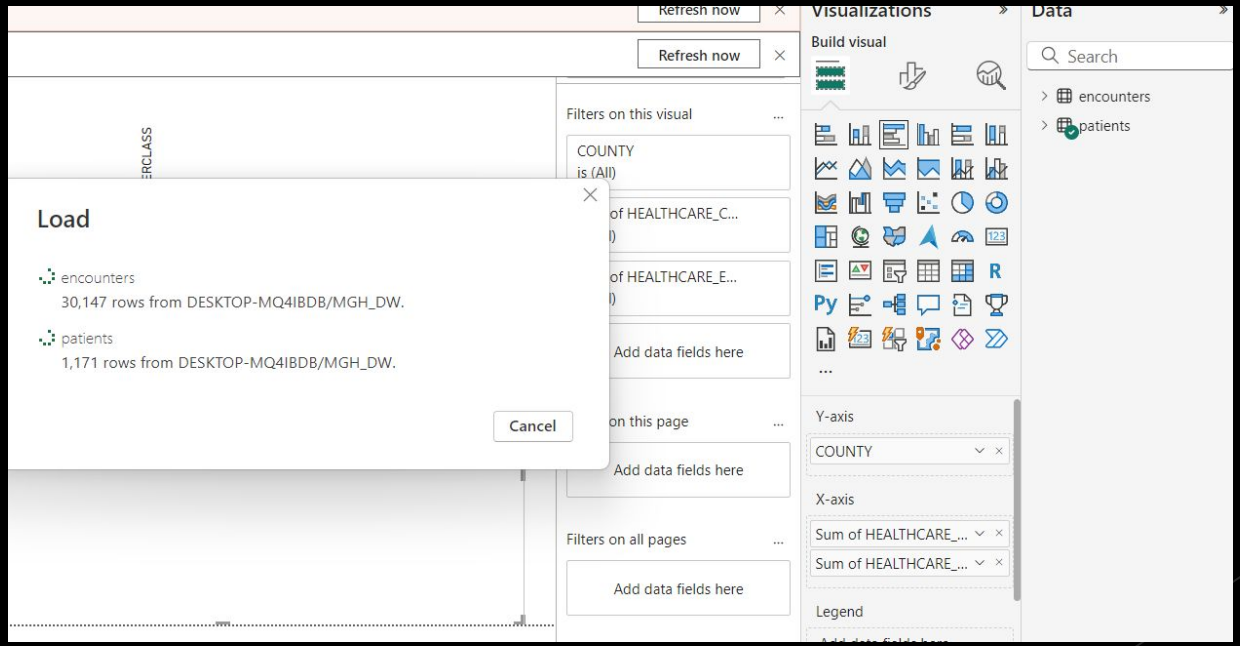

Microsoft

Start New Power Bi Report

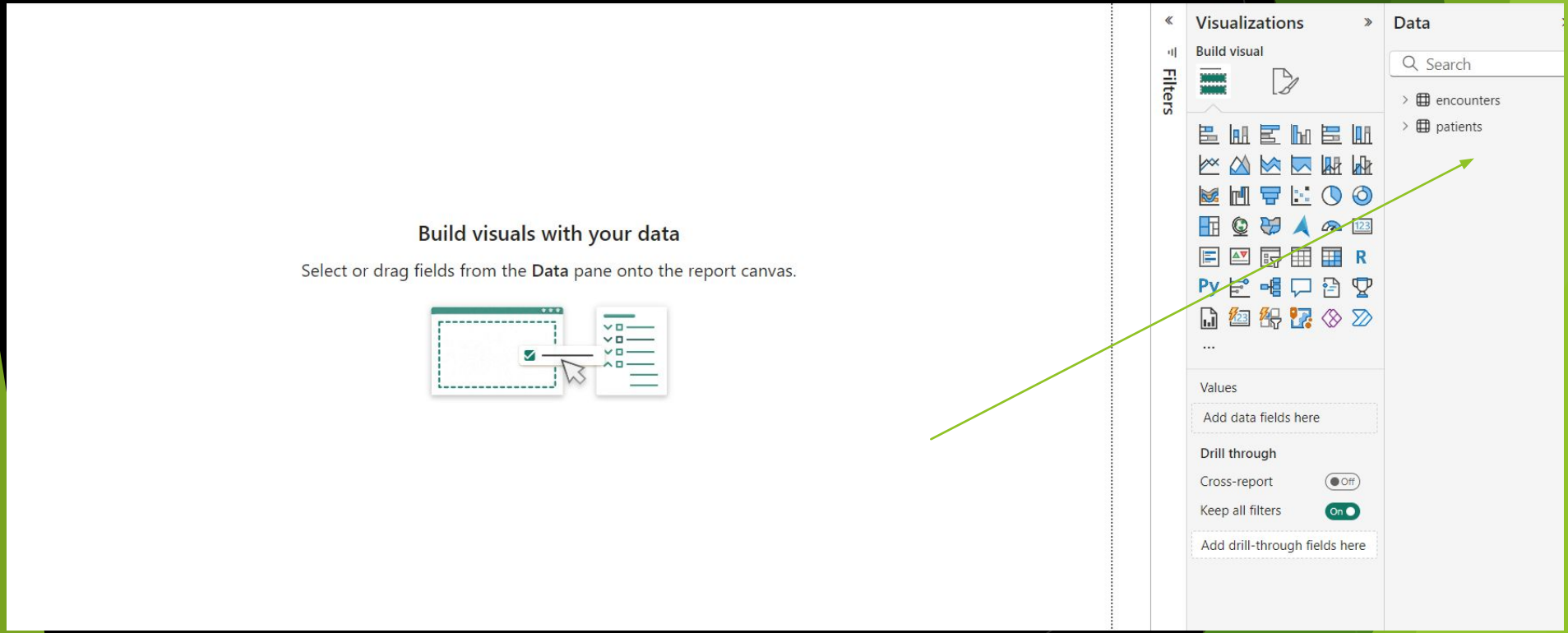


Import data from from SQL

- Sync data on SQL and power BI



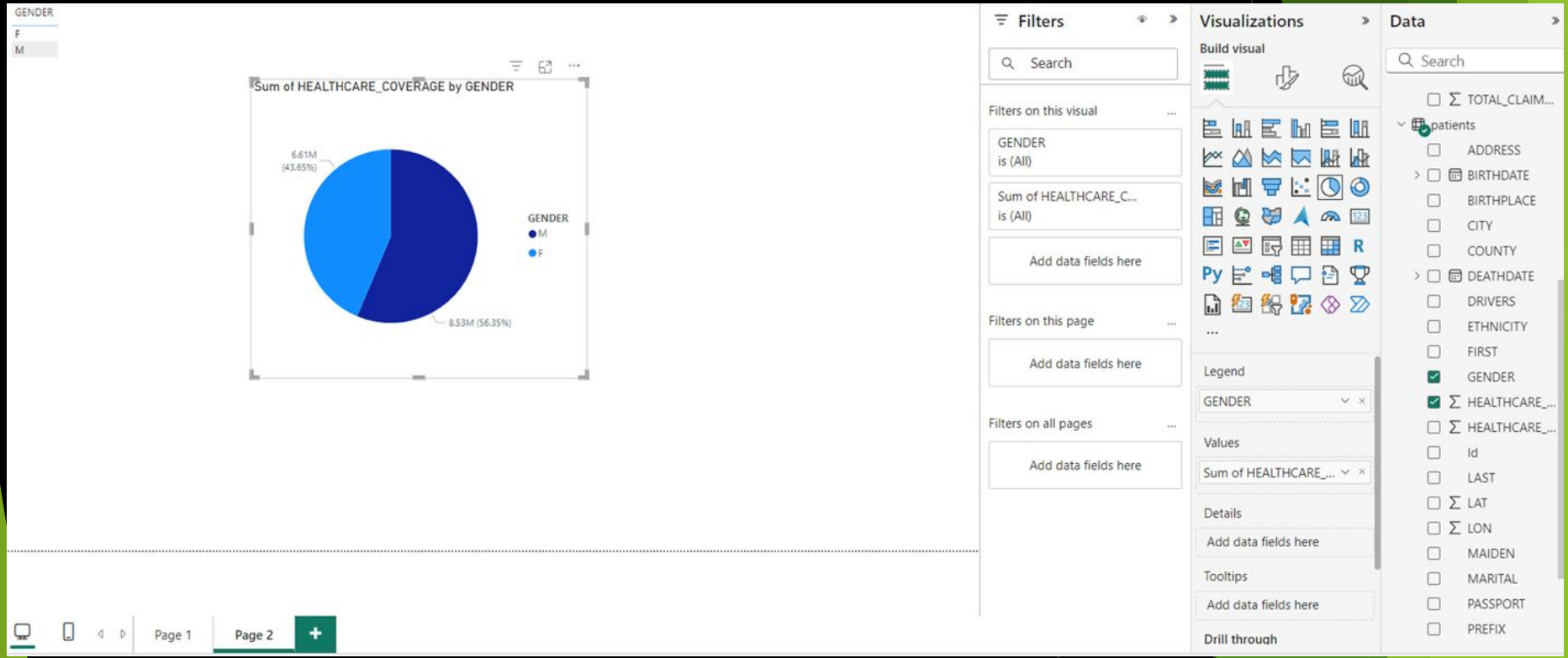
Import data from from SQL



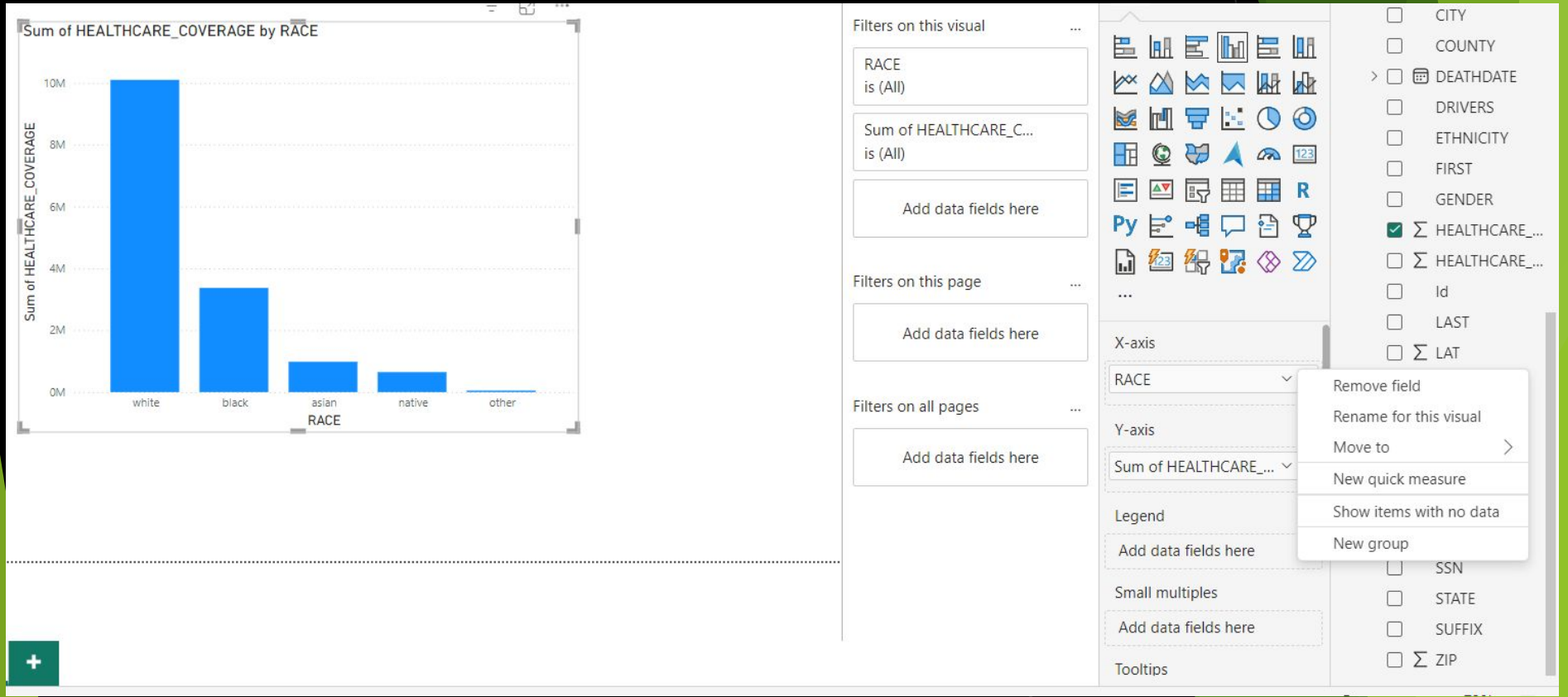
Summary questions

1. What's the average healthcare expense per patient by gender?
2. What's the average healthcare expense per patient by race?
3. Which geographic areas (by state/county) have the highest healthcare expenses?
4. What is the total coverage per encounter class?

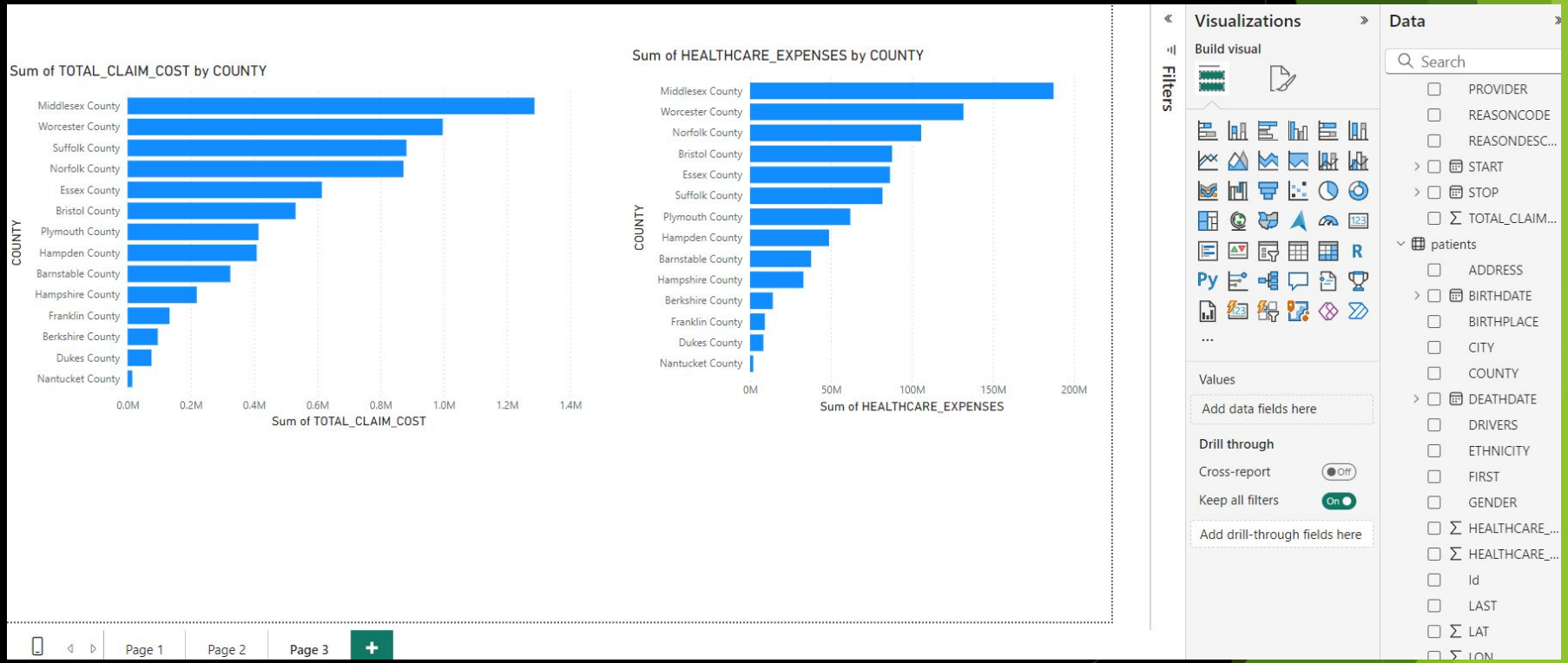
Q1. What is the health coverage per gender



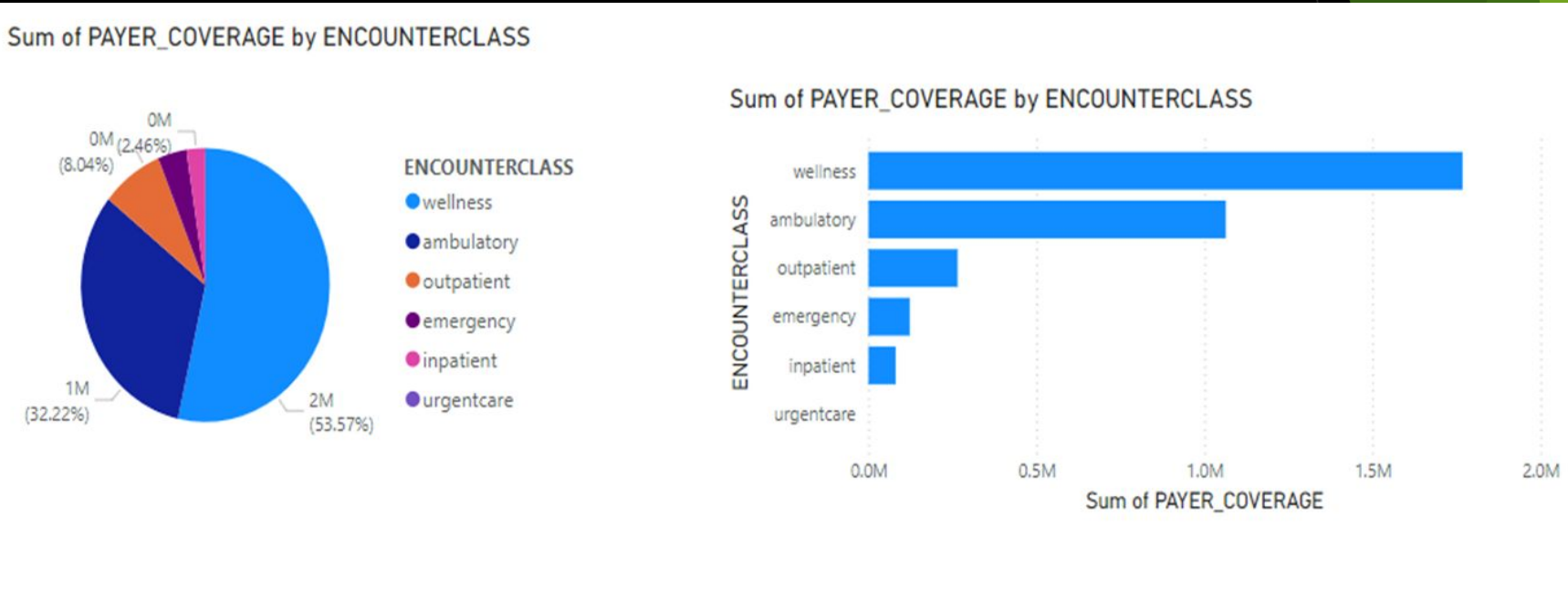
Show the health coverage for the various races



Visualize the healthcare clam and expense per county

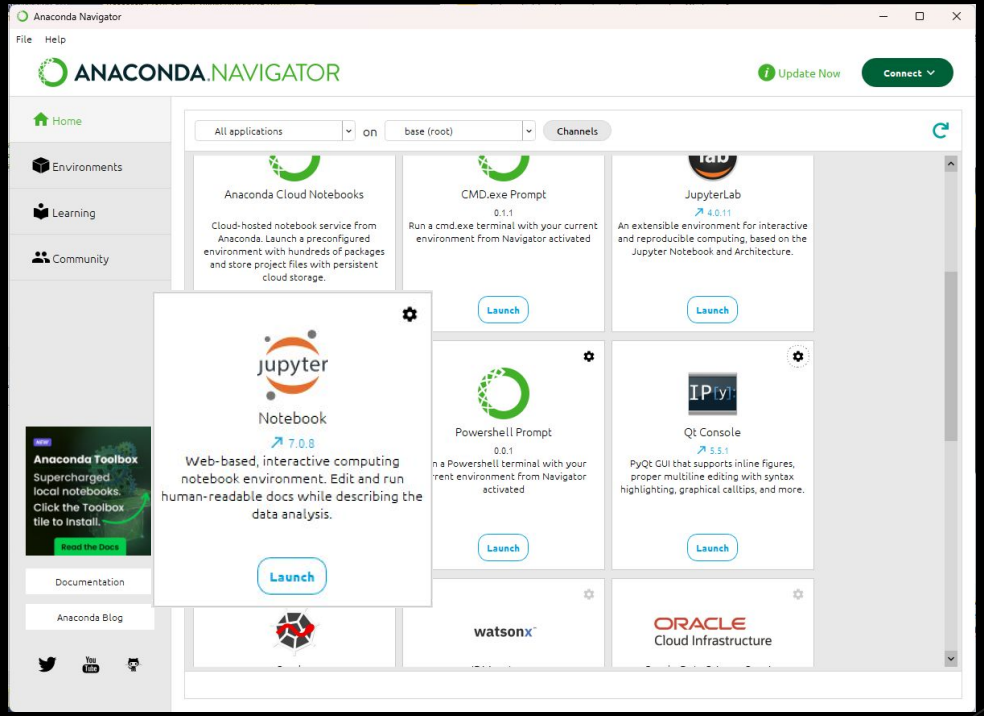


Show the coverage per encounter class



PYTHON


Initializing...


Anaconda Navigator
File Help
ANACONDA.NAVIGATOR
Update Now
Connect
Home
Environments
Learning
Community
All applications
on
base (root)
Channels
Anaconda Cloud Notebooks
Cloud-hosted notebook service from Anaconda. Launch a preconfigured environment with hundreds of packages and store project files with persistent cloud storage.
CMD.exe Prompt
0.1.1
Run a cmd.exe terminal with your current environment from Navigator activated
Launch
JupyterLab
4.0.11
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.
Launch
jupyter
Notebook
7.0.8
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.
Launch
Powershell Prompt
0.0.1
Launch
IPy
Qt Console
5.5.1
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.
Launch
watsonx
ORACLE
Cloud Infrastructure
Anaconda Toolbox
Supercharged local notebooks.
Click the Toolbox tile to install.
Read the Docs
Documentation
Anaconda Blog

Data Processing

```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.impute import SimpleImputer
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report, confusion_matrix, roc_auc_score, roc_curve
import xgboost as xgb
import seaborn as sns
import matplotlib.pyplot as plt
```

```python
def load_and_examine_data(data_str):
    # Create DataFrame from the data
    df = pd.read_csv(data_str)

    print("\nDataset Info:")
    print(df.info())

    print("\nMissing Values:")
    print(df.isnull().sum())

    print("\nBasic Statistics:")
    print(df.describe())

    return df
```

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import scipy as sc
import sklearn as sk
import seaborn as sns
```

[2]:
```
!pip install pyreadstat
```

```
Requirement already satisfied: pyreadstat in c:\users\owner\anaconda3\lib\site-packages (1.2.7)
Requirement already satisfied: pandas>=1.2.0 in c:\users\owner\anaconda3\lib\site-packages (from pyreadstat) (2.0.3)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\owner\anaconda3\lib\site-packages (from pandas>=1.2.0->pyreadstat) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\owner\anaconda3\lib\site-packages (from pandas>=1.2.0->pyreadstat) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in c:\users\owner\anaconda3\lib\site-packages (from pandas>=1.2.0->pyreadstat) (2023.3)
Requirement already satisfied: numpy>=1.21.0 in c:\users\owner\anaconda3\lib\site-packages (from pandas>=1.2.0->pyreadstat) (1.24.3)
Requirement already satisfied: six>=1.5 in c:\users\owner\anaconda3\lib\site-packages (from python-dateutil>=2.8.2->pandas>=1.2.0->pyreadstat) (1.16.0)
```

[3]:
```
df = pd.read_csv('MGH_PredictionDataSet.csv')
```

[4]:
```
df.head()
```

[4]:

	sex	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	totChol	sysBP	diaBP	BMI	heartRate	glucose	TenYearCHD
0	1	39	4.0	0	0.0	0.0	0	0	0	195.0	106.0	70.0	26.97	80.0	77.0	0
1	0	46	2.0	0	0.0	0.0	0	0	0	250.0	121.0	81.0	28.73	95.0	76.0	0
2	1	48	1.0	1	20.0	0.0	0	0	0	245.0	127.5	80.0	25.34	75.0	70.0	0
3	0	61	3.0	1	30.0	0.0	0	1	0	225.0	150.0	95.0	28.58	65.0	103.0	1
4	0	46	3.0	1	23.0	0.0	0	0	0	285.0	130.0	84.0	23.10	85.0	85.0	0

```
##dimension
df.shape
```

```
(4240, 16)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4240 entries, 0 to 4239
Data columns (total 16 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   sex              4240 non-null   int64
 1   age              4240 non-null   int64
 2   education        4135 non-null   float64
 3   currentSmoker    4240 non-null   int64
 4   cigsPerDay       4211 non-null   float64
 5   BPMeds           4187 non-null   float64
 6   prevalentStroke  4240 non-null   int64
 7   prevalentHyp     4240 non-null   int64
 8   diabetes         4240 non-null   int64
 9   totChol          4190 non-null   float64
 10  sysBP            4240 non-null   float64
 11  diaBP            4240 non-null   float64
 12  BMI              4221 non-null   float64
 13  heartRate        4239 non-null   float64
 14  glucose          3852 non-null   float64
 15  TenYearCHD       4240 non-null   int64
dtypes: float64(9), int64(7)
memory usage: 530.1 KB
```

[7]:
```
##basic descriptice
df.describe().transpose()
```

[7]:

	count	mean	std	min	25%	50%	75%	max
sex	4240.0	0.429245	0.495027	0.00	0.00	0.0	1.00	1.0
age	4240.0	49.580189	8.572942	32.00	42.00	49.0	56.00	70.0
education	4135.0	1.979444	1.019791	1.00	1.00	2.0	3.00	4.0
currentSmoker	4240.0	0.494104	0.500024	0.00	0.00	0.0	1.00	1.0
cigsPerDay	4211.0	9.005937	11.922462	0.00	0.00	0.0	20.00	70.0
BPMeds	4187.0	0.029615	0.169544	0.00	0.00	0.0	0.00	1.0
prevalentStroke	4240.0	0.005896	0.076569	0.00	0.00	0.0	0.00	1.0
prevalentHyp	4240.0	0.310613	0.462799	0.00	0.00	0.0	1.00	1.0
diabetes	4240.0	0.025708	0.158280	0.00	0.00	0.0	0.00	1.0
totChol	4190.0	236.699523	44.591284	107.00	206.00	234.0	263.00	696.0
sysBP	4240.0	132.354599	22.033300	83.50	117.00	128.0	144.00	295.0
diaBP	4240.0	82.897759	11.910394	48.00	75.00	82.0	90.00	142.5
BMI	4221.0	25.800801	4.079640	15.54	23.07	25.4	28.04	56.8
heartRate	4239.0	75.878981	12.025348	44.00	68.00	75.0	83.00	143.0
glucose	3852.0	81.963655	23.954335	40.00	71.00	78.0	87.00	394.0
TenYearCHD	4240.0	0.151887	0.358953	0.00	0.00	0.0	0.00	1.0

Normalizing/Standardizing Numerical Features

```
[14]: df.duplicated().sum()

[14]: 0

[15]: df.info()

<class 'pandas.core.frame.DataFrame'>
Index: 3658 entries, 0 to 4239
Data columns (total 16 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   sex              3658 non-null   int64
 1   age              3658 non-null   int64
 2   education        3658 non-null   float64
 3   currentSmoker    3658 non-null   int64
 4   cigsPerDay       3658 non-null   float64
 5   BPMeds           3658 non-null   float64
 6   prevalentStroke  3658 non-null   int64
 7   prevalentHyp     3658 non-null   int64
 8   diabetes         3658 non-null   int64
 9   totChol          3658 non-null   float64
 10  sysBP            3658 non-null   float64
 11  diaBP            3658 non-null   float64
 12  BMI              3658 non-null   float64
 13  heartRate        3658 non-null   float64
 14  glucose          3658 non-null   float64
 15  TenYearCHD       3658 non-null   int64
dtypes: float64(9), int64(7)
memory usage: 485.8 KB
```

```
[33]: #Normalising the data

[34]: df ['TenYearCHD'].value_counts(normalize=True)

[34]: TenYearCHD
      0    0.847731
      1    0.152269
      Name: proportion, dtype: float64
```

Handling Missing Data

```python
def preprocess_data(data_str):
    # Create a copy of the data
    df = data_str.copy()

    # 1. Handle Missing Values
    # First, ensure all numeric columns are properly typed
    numeric_features = ['cigsPerDay', 'totChol', 'sysBP', 'diaBP',
                        'BMI', 'heartRate','age',  'glucose']

    # Convert numeric columns to float, keeping NA values as NA
    for col in numeric_features:
        df[col] = pd.to_numeric(df[col], errors='coerce')

    # Impute missing values for numeric features
    numerical_imputer = SimpleImputer(strategy='mean')
    df[numeric_features] = numerical_imputer.fit_transform(df[numeric_features])
```

```
[8]: ##filtering the data

[9]: df.isnull()
```

[9]:

	sex	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	totChol	sysBP	diaBP	BMI	heartRate	glucose	TenYearCHD
0	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4235	False	False	False	False	False	True	False	False	False	False	False	False	False	False	False	False
4236	False	False	False	False	False	False	False	False	False	False	False	False	False	False	True	False
4237	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
4238	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
4239	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False

4240 rows × 16 columns

```
[10]: df.isna()
```

[10]:

	sex	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	totChol	sysBP	diaBP	BMI	heartRate	glucose	TenYearCHD
0	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4235	False	False	False	False	False	True	False	False	False	False	False	False	False	False	False	False
4236	False	False	False	False	False	False	False	False	False	False	False	False	False	False	True	False
4237	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
4238	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
4239	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False

4240 rows × 16 columns

```
[11]: (df.isna().sum())*100/len(df)
```

[11]:

```
sex              0.000000
age              0.000000
education        2.476415
currentSmoker    0.000000
cigsPerDay       0.683962
BPMeds           1.250000
prevalentStroke  0.000000
prevalentHyp     0.000000
diabetes         0.000000
totChol          1.179245
sysBP            0.000000
diaBP            0.000000
BMI              0.448113
heartRate        0.023585
glucose          9.150943
TenYearCHD       0.000000
dtype: float64
```

```
[12]: df = df.dropna()

[13]: (df.isna().sum())*100/len(df)
```

[13]:

```
sex              0.0
age              0.0
education        0.0
currentSmoker    0.0
cigsPerDay       0.0
BPMeds           0.0
prevalentStroke  0.0
prevalentHyp     0.0
diabetes         0.0
totChol          0.0
sysBP            0.0
diaBP            0.0
BMI              0.0
heartRate        0.0
glucose          0.0
TenYearCHD       0.0
dtype: float64
```

```python
# def handle_missing_values(df):

#     numeric_imputer = SimpleImputer(strategy='median')

#     # Get numeric columns
#     numeric_columns = df.select_dtypes(include=['float64', 'int64']).columns

#     # Impute missing values
#     df[numeric_columns] = numeric_imputer.fit_transform(df[numeric_columns])

#     return df

def prepare_features(df):
    # Standardize numeric features
    scaler = StandardScaler()
    numeric_features = ['age', 'cigsPerDay', 'totChol', 'sysBP', 'diaBP', 'BMI',
                        'heartRate', 'glucose']
    df[numeric_features] = scaler.fit_transform(df[numeric_features])

    return df

```

Feature Scaling/Encoding Variables

```
scaler = StandardScaler()
df[numeric_features] = scaler.fit_transform(df[numeric_features])
```

```
    # One-hot encode categorical features
    categorical_features = ['age_group', 'bmi_category', 'bp_category']
    df = pd.get_dummies(df, columns=categorical_features,
prefix=categorical features)
```

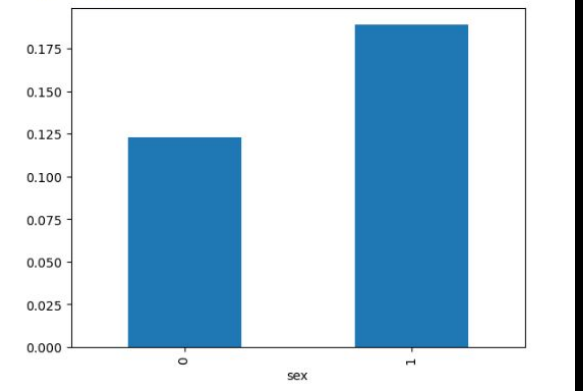
Exploratory Data Analysis

```
[16]: ##Exploratory Data Analysis

[17]: df.groupby('sex').mean()
```

[17]:

sex	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	totChol	sysBP	diaBP	BMI	heartRate	glucose	TenYea
0	49.738575	1.964128	0.397052	5.509091	0.038329	0.005897	0.311057	0.025061	239.616708	133.265111	82.362408	25.517386	76.964128	81.790172	0.1
1	49.317930	2.000616	0.604436	13.434381	0.020333	0.005545	0.312384	0.029575	233.375847	131.248922	83.612446	26.115595	74.184227	81.931608	0.1

```
[18]: ##df['sex'] == df['sex'].replace([1.0, 0.0], ['male', 'female'])

[19]: df.groupby('sex')['TenYearCHD'].mean()

[19]: sex
      0    0.122850
      1    0.189156
      Name: TenYearCHD, dtype: float64

[20]: df.groupby('sex')['TenYearCHD'].mean().plot(kind='bar')

[20]: <Axes: xlabel='sex'>
```




```
49  # 2. Handle categorical variables
50  # Ensure education is numeric, handle NA values
51  df['education'] = pd.to_numeric(df['education'], errors='coerce')
52  df['education'] = df['education'].fillna(df['education'].median())
53  # print(df['education'].unique())
54
55  # 3. Handle binary variables
56  binary_features = ['sex', 'currentSmoker', 'BPMeds', 'prevalentStroke',
57                     'prevalentHyp', 'diabetes', 'TenYearCHD']
58
59  # Convert binary features to int, filling NA with 0
60  for col in binary_features:
61      df[col] = df[col].fillna(0).astype(int)
```

```
[21]: df.corr()
```

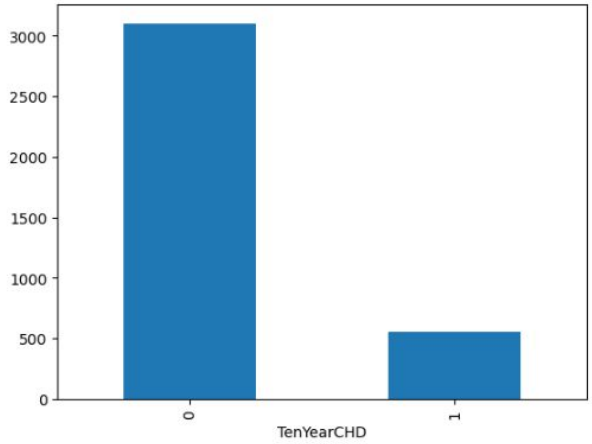
[21]:

	sex	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	totChol	sysBP	diaBP	BMI	heartRate	glucose	TenYearCHD
sex	1.000000	-0.024412	0.017729	0.206114	0.330322	-0.052124	-0.002312	0.001424	0.013819	-0.070321	-0.045358	0.051872	0.073111	-0.115285	0.002940	0.091688
age	-0.024412	1.000000	-0.159499	-0.210771	-0.189295	0.134732	0.050893	0.306239	0.109092	0.268252	0.388267	0.208283	0.137511	-0.002722	0.118349	0.233983
education	0.017729	-0.159499	1.000000	0.025251	0.013790	-0.013739	-0.030386	-0.078828	-0.039627	-0.013522	-0.124360	-0.058059	-0.137555	-0.064214	-0.031998	-0.063279
currentSmoker	0.206114	-0.210771	0.025251	1.000000	0.773913	-0.051923	-0.038150	-0.108078	-0.041849	-0.051034	-0.134428	-0.115955	-0.159821	0.050841	-0.053242	0.019165
cigsPerDay	0.330322	-0.189295	0.013790	0.773913	1.000000	-0.046504	-0.036286	-0.070460	-0.036961	-0.030400	-0.094781	-0.056746	-0.087395	0.064030	-0.053726	0.052014
BPMeds	-0.052124	0.134732	-0.013739	-0.051923	-0.046504	1.000000	0.113125	0.262910	0.049066	0.094083	0.271263	0.199630	0.105642	0.012889	0.054232	0.089152
prevalentStroke	0.002312	0.050893	-0.030386	-0.038150	-0.036286	0.113125	1.000000	0.066057	0.009625	0.012736	0.061070	0.055834	0.036496	-0.017019	0.016061	0.048366
prevalentHyp	0.001424	0.306239	-0.078828	-0.108078	-0.070460	0.262910	0.066057	1.000000	0.080556	0.166655	0.697675	0.617734	0.302949	0.146818	0.086942	0.181387
diabetes	0.013819	0.109092	-0.039627	-0.041849	-0.036961	0.049066	0.009625	0.080556	1.000000	0.048451	0.102552	0.050686	0.089009	0.060984	0.614820	0.093431
totChol	-0.070321	0.268252	-0.013522	-0.051034	-0.030400	0.094083	0.012736	0.166655	0.048451	1.000000	0.219925	0.174422	0.121056	0.093053	0.049884	0.091338
sysBP	-0.045358	0.388267	-0.124360	-0.134428	-0.094781	0.271263	0.061070	0.697675	0.102552	0.219925	1.000000	0.786669	0.330917	0.184797	0.134651	0.222821
diaBP	0.051872	0.208283	-0.058059	-0.115955	-0.056746	0.199630	0.055834	0.617734	0.050686	0.174422	0.786669	1.000000	0.385348	0.178744	0.063540	0.150124
BMI	0.073111	0.137511	-0.137555	-0.159821	-0.087395	0.105642	0.036496	0.302949	0.089009	0.121056	0.330917	0.385348	1.000000	0.074131	0.083683	0.082055
heartRate	-0.115285	-0.002722	-0.064214	0.050841	0.064030	0.012889	-0.017019	0.146818	0.060984	0.093053	0.184797	0.178744	0.074131	1.000000	0.097074	0.020514
glucose	0.002940	0.118349	-0.031998	-0.053242	-0.053726	0.054232	0.016061	0.086942	0.614820	0.049884	0.134651	0.063540	0.083683	0.097074	1.000000	0.121990
TenYearCHD	0.091688	0.233983	-0.063279	0.019165	0.052014	0.089152	0.048366	0.181387	0.093431	0.091338	0.222821	0.150124	0.082055	0.020514	0.121990	1.000000

Visualizing the Data Distribution

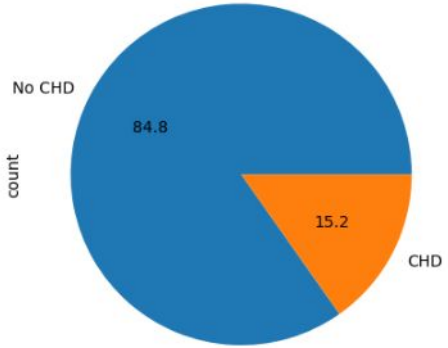
```
[35]: #Visualizing the data Distribution
      df ['TenYearCHD'].value_counts().plot(kind='bar')
```

```
[35]: <Axes: xlabel='TenYearCHD'>
```

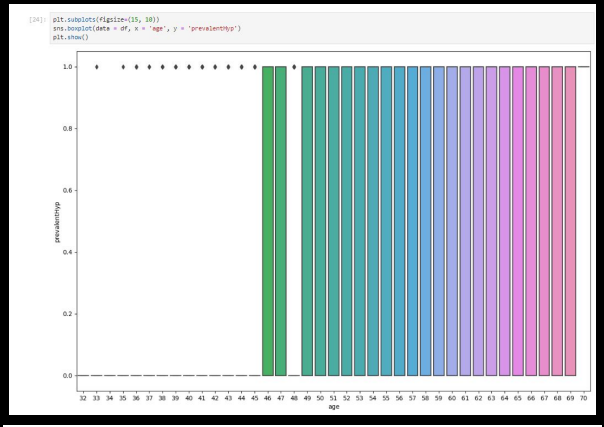


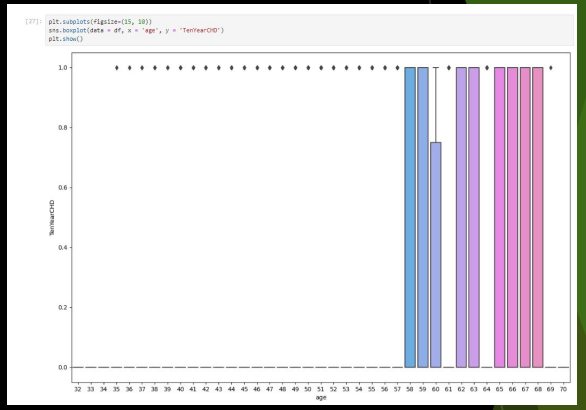

```
[36]: df ['TenYearCHD'].value_counts().plot(kind='pie', autopct='%1.1f', labels=['No CHD', 'CHD'])
```

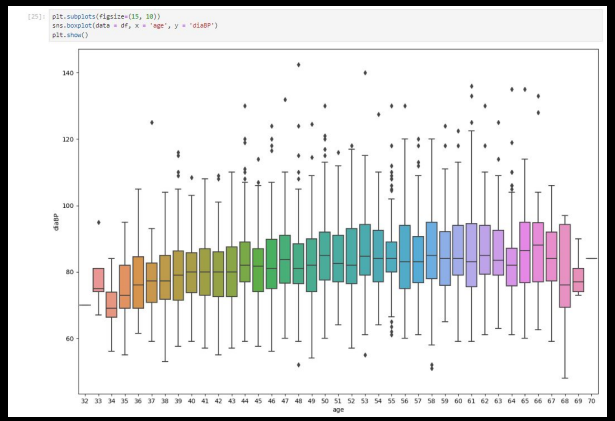
```
[36]: <Axes: ylabel='count'>
```

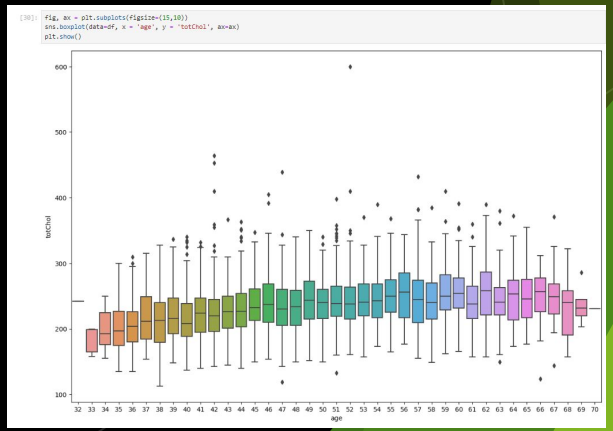


Exploratory Data Analysis

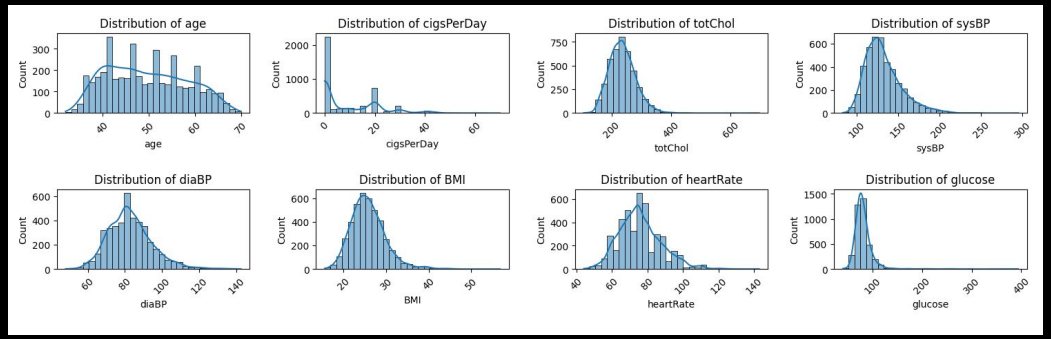


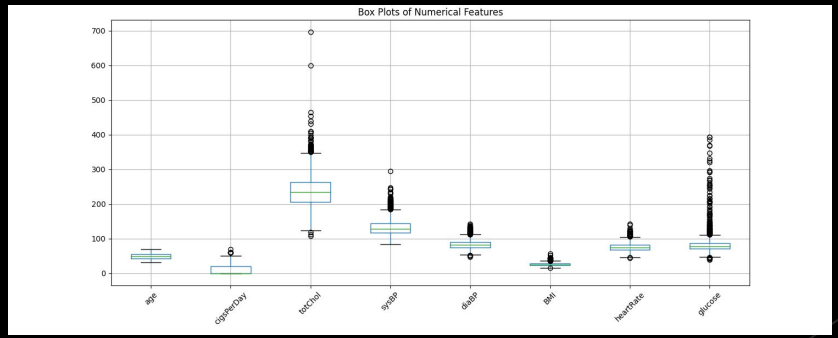






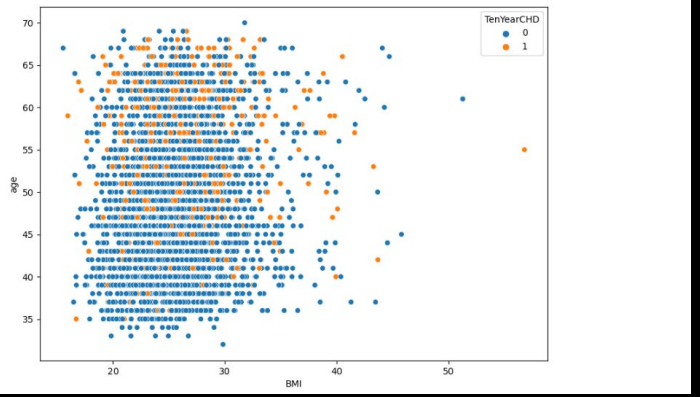
Numerical Distributions of Features

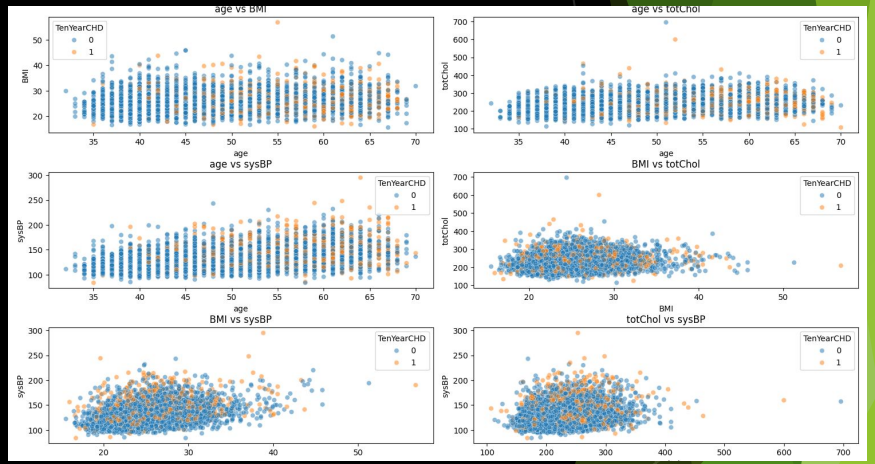




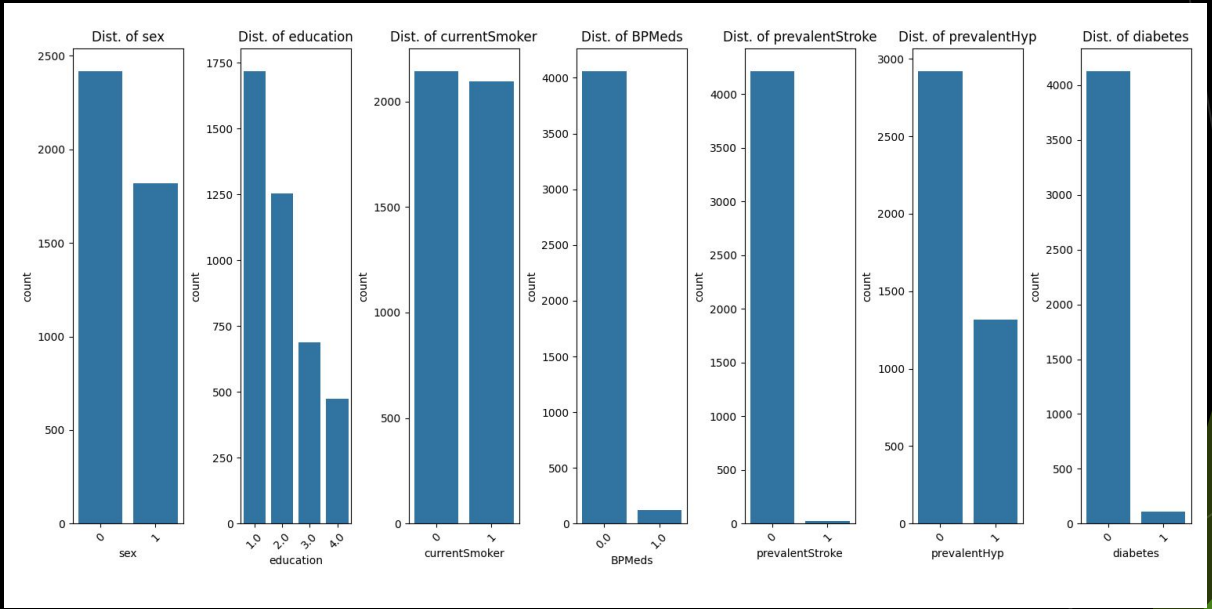
Scatter Relationships

```
[37]: plt.figure(figsize=(10,7))
      sns.scatterplot(data =df, x = 'BMI', y = 'age', hue = 'TenYearCHD')
      plt.show()
```

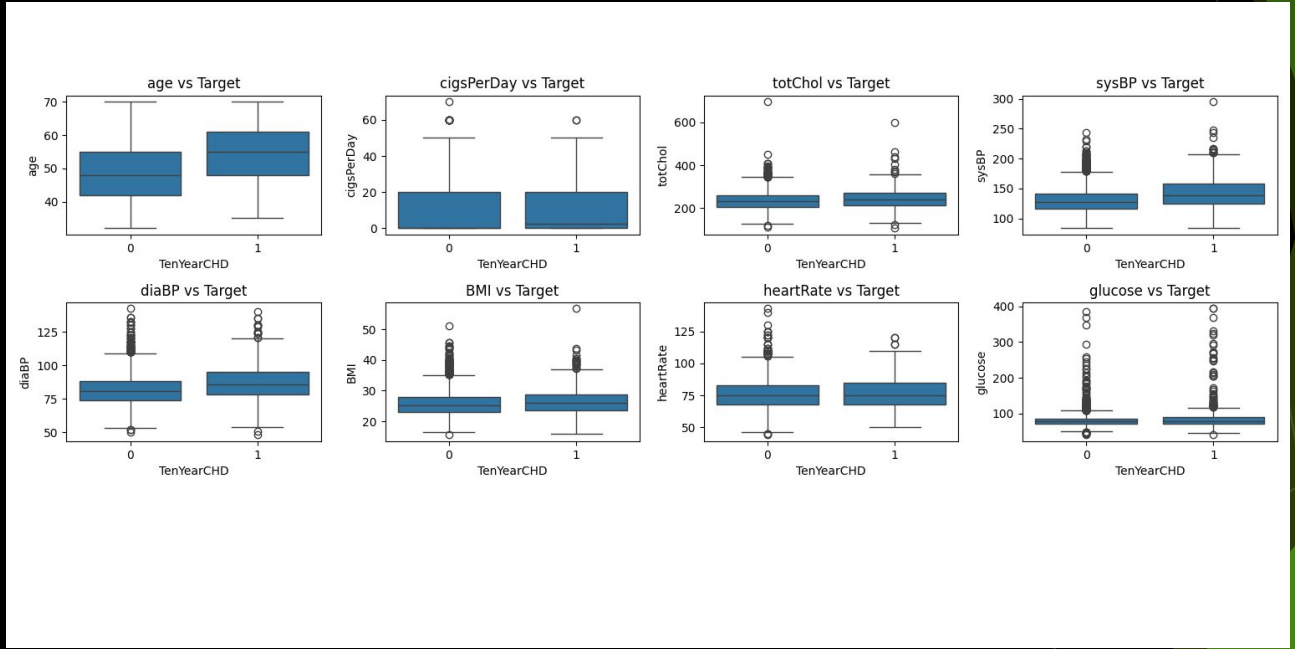




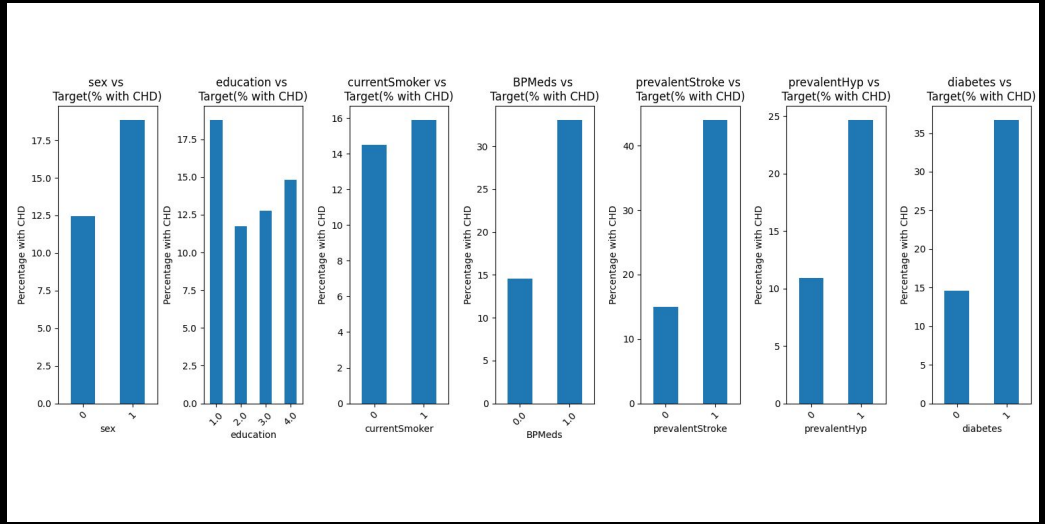
Categorical Distributions

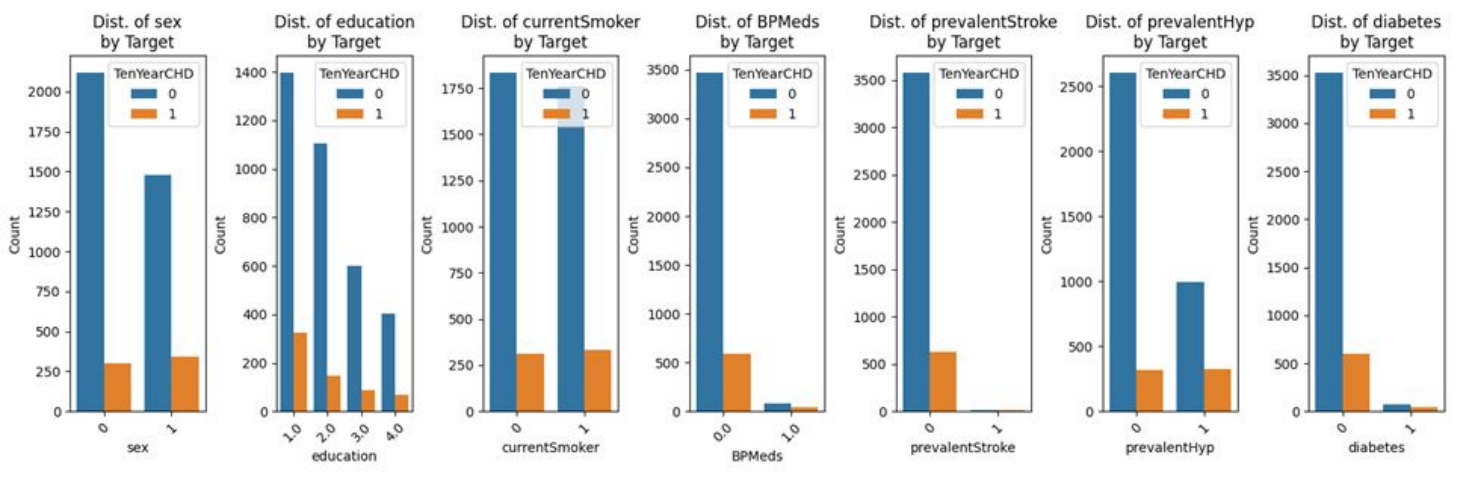


Analyzing Relationships with Target Variable



Categorical Relationships vs Target




Dist. of sex by Target
Dist. of education by Target
Dist. of currentSmoker by Target
Dist. of BPMeds by Target
Dist. of prevalentStroke by Target
Dist. of prevalentHyp by Target
Dist. of diabetes by Target
TenYearCHD
0
1
Count
sex
education
currentSmoker
BPMeds
prevalentStroke
prevalentHyp
diabetes

Splitting the Data

```
[38]: #Splitting the data
      from sklearn.model_selection import train_test_split

[39]: x = df[['cigsPerDay', 'BPMeds']]

[40]: y = df['TenYearCHD']

[41]: y

[41]: 0       0
      1       0
      2       0
      3       1
      4       0
             ..
      4233    1
      4234    0
      4237    0
      4238    0
      4239    0
      Name: TenYearCHD, Length: 3658, dtype: int64

[42]: x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2)

[43]: x

[43]:
```

	cigsPerDay	BPMeds
0	0.0	0.0
1	0.0	0.0
2	20.0	0.0
3	30.0	0.0
4	23.0	0.0
...	...	...
4233	1.0	0.0
4234	43.0	0.0
4237	0.0	0.0
4238	0.0	0.0
4239	30.0	0.0

3658 rows × 2 columns

```
[44]: y

[44]: 0       0
      1       0
      2       0
      3       1
      4       0
             ..
      4233    1
      4234    0
      4237    0
      4238    0
      4239    0
      Name: TenYearCHD, Length: 3658, dtype: int64
```

```
[47]: train_test_split(x,y,test_size=0.2)

[47]: [      cigsPerDay  BPMeds
       2454         0.0     0.0
       608         35.0     0.0
       1795        35.0     0.0
       2697         0.0     0.0
       702          1.0     0.0
       ...          ...     ...
       1426        23.0     0.0
       3523        10.0     0.0
       3254        30.0     0.0
       2884        20.0     0.0
       2705        15.0     0.0

       [2926 rows x 2 columns],
             cigsPerDay  BPMeds
       138         30.0     0.0
       4036         0.0     0.0
       1804         0.0     0.0
       1309         0.0     0.0
       1343        25.0     0.0
       ...          ...     ...
       3414         0.0     0.0
       848          0.0     0.0
       3914         0.0     0.0
       3642         0.0     0.0
       2950         0.0     0.0

       [732 rows x 2 columns],
       2454    0
       608     0
       1795    0
       2697    0
       702     0
              ..
       1426    0
       3523    0
       3254    1
       2884    0
       2705    0
       Name: TenYearCHD, Length: 2926, dtype: int64,
       138     1
       4036    0
       1804    0
       1309    0
       1343    0
              ..
       3414    0
       848     0
       3914    0
       3642    0
       2950    0
       Name: TenYearCHD, Length: 732, dtype: int64]

[48]: x_train, x_test, y_train,y_test = train_test_split(x,y,test_size=0.2)
```

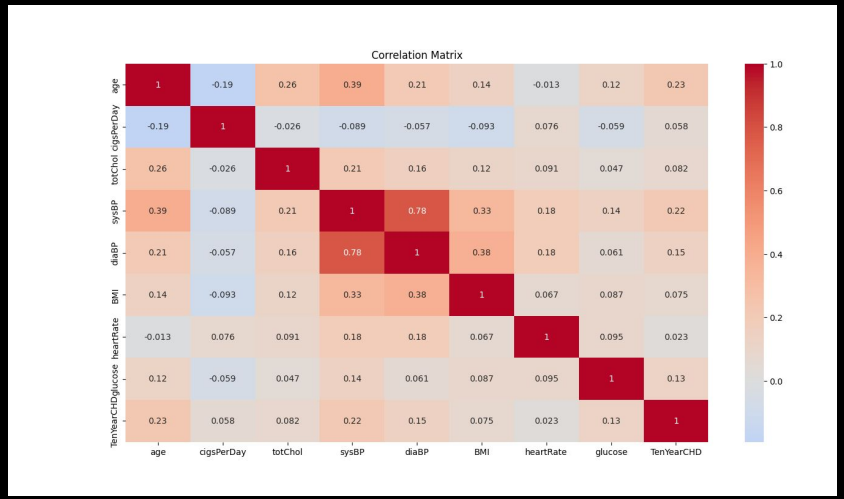
```
[49]: len(y_train)

[49]: 2926

[50]: len(x_train)

[50]: 2926

[51]: len(x_test)

[51]: 732

[52]: len(y_test)

[52]: 732

[53]: x.shape

[53]: (3658, 2)

[54]: y.shape

[54]: (3658,)
```

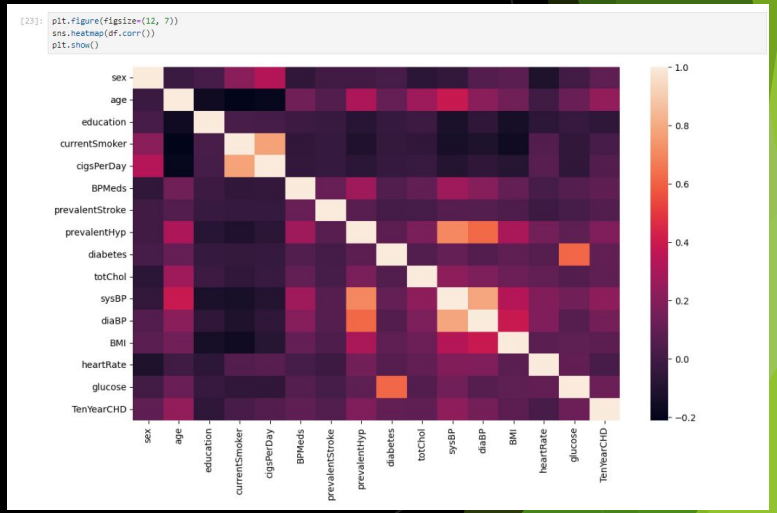
Model Training and Validation

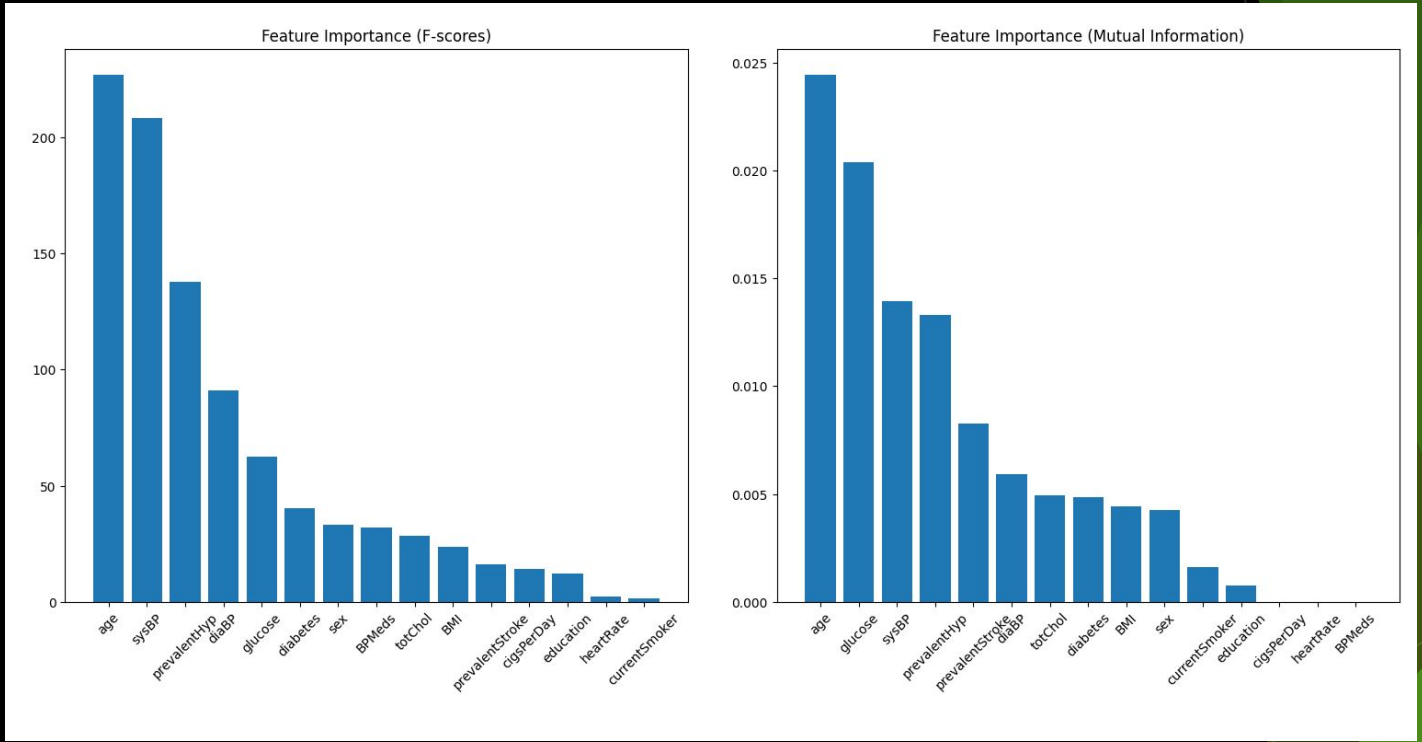
```python

# models training and evaluation functions
def train_multiple_models(X_train, X_test, y_train, y_test):
    # Initialize models
    models = {
        'Logistic Regression': LogisticRegression(random_state=42, max_iter=1000),
        'Decision Tree': DecisionTreeClassifier(random_state=42),
        'KNeighbors Classifier': KNeighborsClassifier(n_neighbors=5),
        'Random Forest': RandomForestClassifier(n_estimators=100, random_state=42),
        'XGBoost': xgb.XGBClassifier(random_state=42, use_label_encoder=False, eval_metric='logloss')
    }

    #store results
    results = {}

    # Train and evaluate each model
    for name, model in models.items():
        # Train model
        model.fit(X_train, y_train)

        # Make predictions
        y_pred = model.predict(X_test)
        y_pred_proba = model.predict_proba(X_test)[:, 1]

        # Calculate metrics
        results[name] = {
            'model': model,
            'predictions': y_pred,
            'probabilities': y_pred_proba,
            'accuracy': model.score(X_test, y_test),
            'roc_auc': roc_auc_score(y_test, y_pred_proba),
            'classification_report': classification_report(y_test, y_pred),
            'confusion_matrix': confusion_matrix(y_test, y_pred)
        }

        # Add feature importance if available
        if hasattr(model, 'feature_importances_'):
            results[name]['feature_importance'] = pd.DataFrame({
                'feature': X_train.columns,
                'importance': model.feature_importances_
            }).sort_values('importance', ascending=False)
        elif hasattr(model, 'coef_'):
            results[name]['feature_importance'] = pd.DataFrame({
                'feature': X_train.columns,
                'importance': abs(model.coef_[0])
            }).sort_values('importance', ascending=False)

    return results
```

```python

def compare_models(results):
    # Comparison DataFrame
    comparison = pd.DataFrame({
        name: {
            'Accuracy': results[name]['accuracy'],
            'ROC AUC': results[name]['roc_auc']
        } for name in results.keys()
    }).T

    return comparison

def perform_cross_validation(X, y, models, cv=5):
    cv_results = {}
    for name, model in models.items():
        scores = cross_val_score(model, X, y, cv=cv, scoring='roc_auc')
        cv_results[name] = {
            'mean_score': scores.mean(),
            'std_score': scores.std(),
            'all_scores': scores
        }
    return cv_results

```

Correlation Analysis






Feature Importance (F-scores)
0
50
100
150
200
age
sysBP
prevalentHyp
diaBP
glucose
diabetes
sex
BPMeds
totChol
BMI
prevalentStroke
cigsPerDay
education
heartRate
currentSmoker
Feature Importance (Mutual Information)
0.000
0.005
0.010
0.015
0.020
0.025
age
glucose
sysBP
prevalentHyp
prevalentStroke
diaBP
totChol
diabetes
BMI
sex
currentSmoker
education
cigsPerDay
heartRate
BPMeds

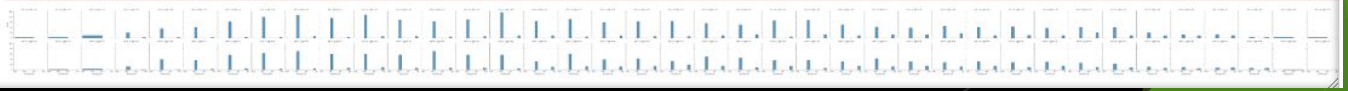
Feature Selection

```python
    # 4. Feature Scaling
    scaler = StandardScaler()
    df[numeric_features] = scaler.fit_transform(df[numeric_features])

    # 5. Feature Engineering
    # Create age groups
    df['age_group'] = pd.qcut(df['age'], q=5, labels=['young_adults', 'adults', 'middle_aged', 'old', 'youth'])

    # Create BMI categories
    df['bmi_category'] = pd.cut(df['BMI'],
                                bins=[0, 18.5, 25, 30, float('inf')],
                                labels=['Underweight', 'Normal', 'Overweight', 'Obese'])

    # Calculate blood pressure category
    def bp_category(row):
        if row['sysBP'] < 120 and row['diaBP'] < 80:
            return 'Normal'
        elif row['sysBP'] < 130 and row['diaBP'] < 80:
            return 'Elevated'
        else:
            return 'High'

    df['bp_category'] = df.apply(bp_category, axis=1)

    # One-hot encode categorical features
    categorical_features = ['age_group', 'bmi_category', 'bp_category']
    df = pd.get_dummies(df, columns=categorical_features, prefix=categorical_features)

    print("\nProcessed Features:")
    print(df.columns.tolist())

    print("\nMissing Values After Processing:")
    print(df.isnull().sum().sum())

    return df
```

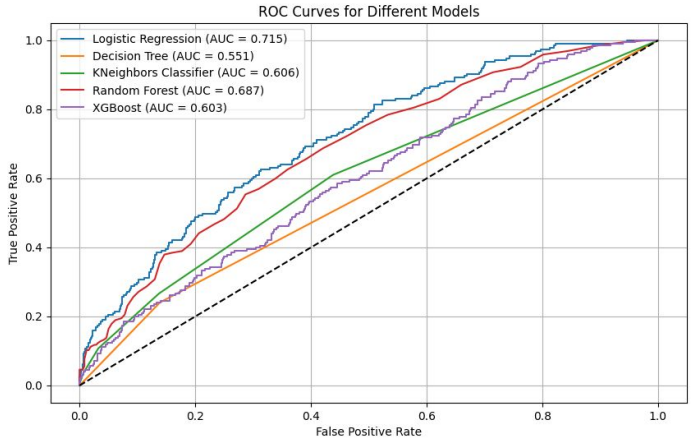
```python
plt.subplots(figsize=(15, 10))
sns.boxplot(data = df, x = 'sex', y = 'TenYearCHD')
plt.show()
```

[28]:
```python
g = sns.FacetGrid(df, row = 'sex', col = 'age')
g.map(sns.histplot, 'TenYearCHD')
plt.show()
```

```
C:\Users\Owner\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```



Model Selection



Logistic Regression Results:

	precision	recall	f1-score	support
0	0.86	0.99	0.92	725
1	0.62	0.07	0.12	123
accuracy			0.86	848
macro avg	0.74	0.53	0.52	848
weighted avg	0.83	0.86	0.81	848

Decision Tree Results:

	precision	recall	f1-score	support
0	0.86	0.84	0.85	725
1	0.15	0.16	0.16	123
accuracy			0.75	848
macro avg	0.50	0.50	0.50	848
weighted avg	0.75	0.75	0.75	848

KNeighbors Classifier Results:

	precision	recall	f1-score	support
0	0.86	0.95	0.90	725
1	0.21	0.07	0.11	123
accuracy			0.83	848
macro avg	0.53	0.51	0.51	848
weighted avg	0.76	0.83	0.79	848

Random Forest Results:

	precision	recall	f1-score	support
0	0.86	0.99	0.92	725
1	0.46	0.05	0.09	123
accuracy			0.85	848
macro avg	0.66	0.52	0.50	848
weighted avg	0.80	0.85	0.80	848

XGBoost Results:

	precision	recall	f1-score	support
0	0.86	0.95	0.90	725
1	0.26	0.10	0.14	123
accuracy			0.83	848
macro avg	0.56	0.52	0.52	848
weighted avg	0.77	0.83	0.79	848

Questions