

MASSACHUSETTS GENERAL HOSPITAL

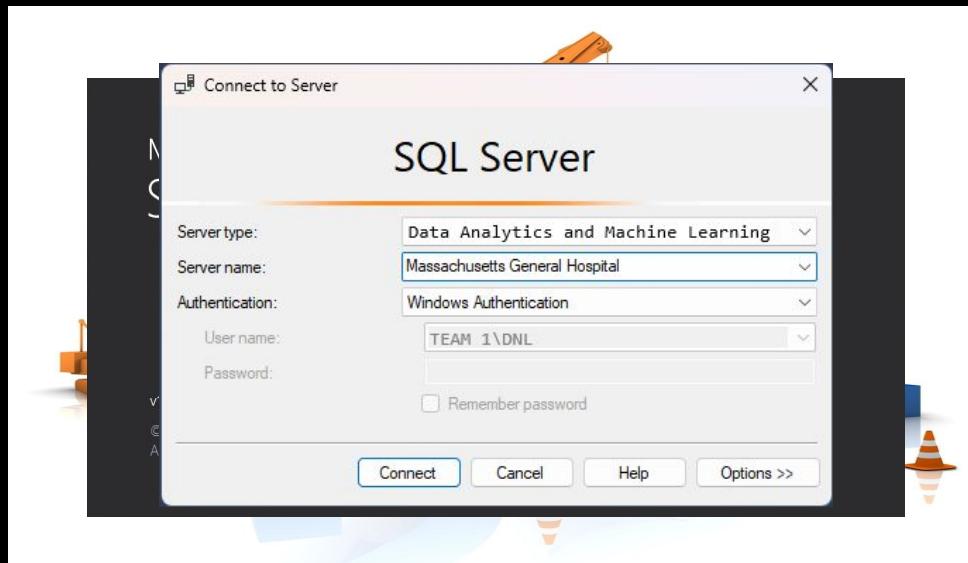
GROUP 1

Doris Waithaka

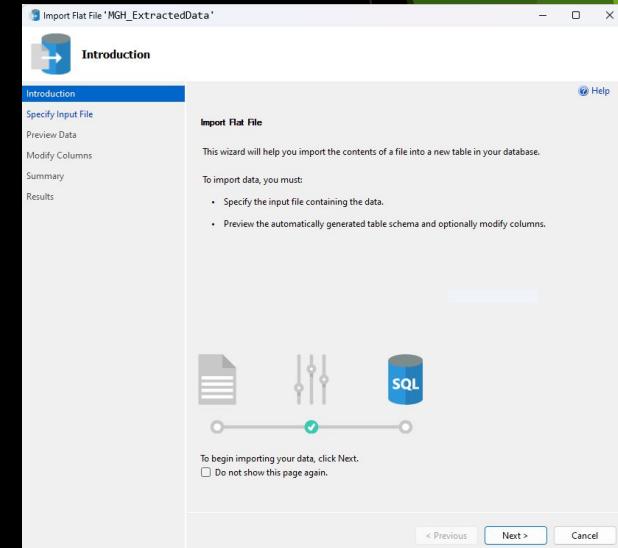
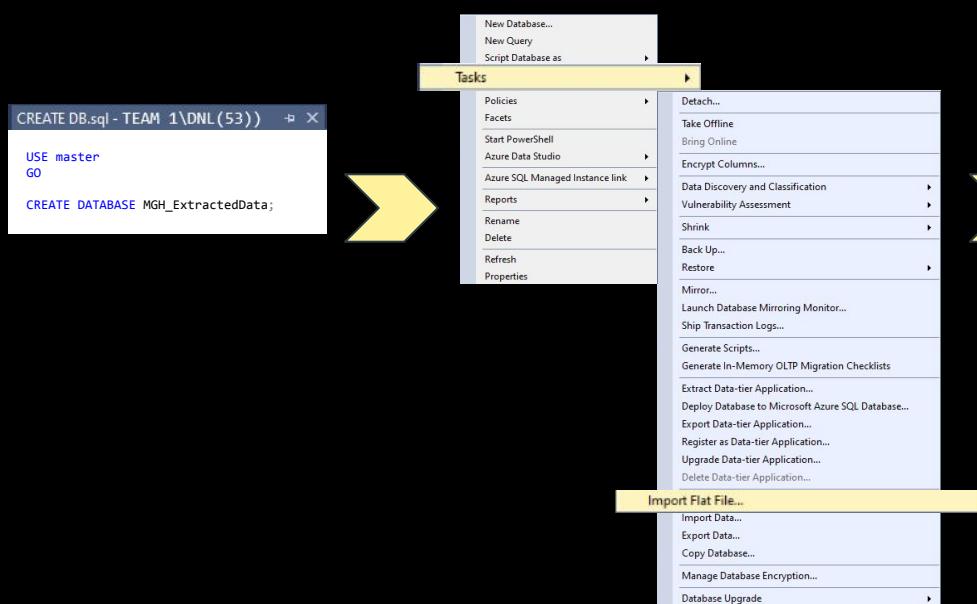
Vivian Njenga

Michaela Jackson

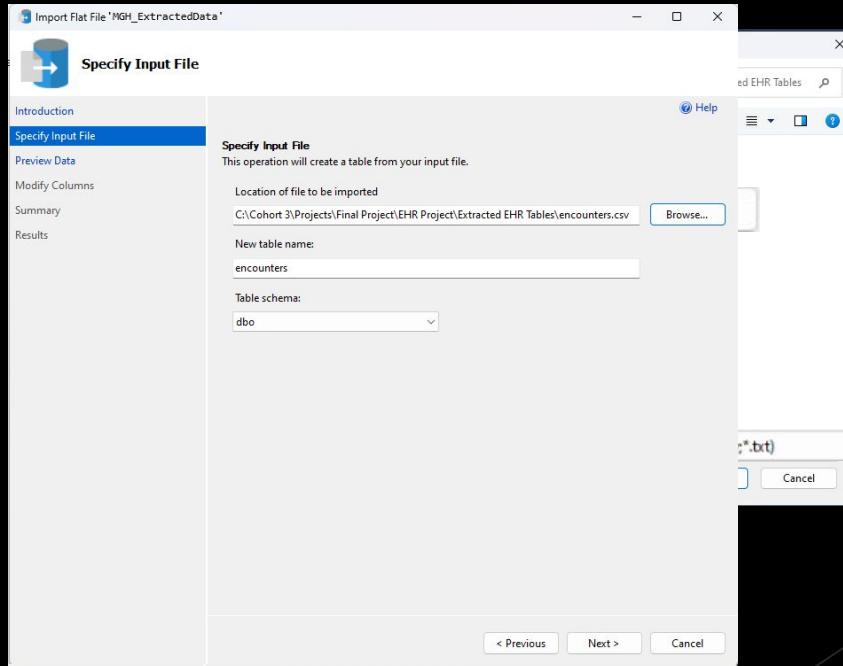
Fidelis Makunike



Import Start Up



Importing EHR data



Importing EHR data

Import Flat File 'MGH_ExtractedData'

Preview Data

Introduction
Specify Input File
Preview Data
Modify Columns
Summary
Results

Help

Preview Data

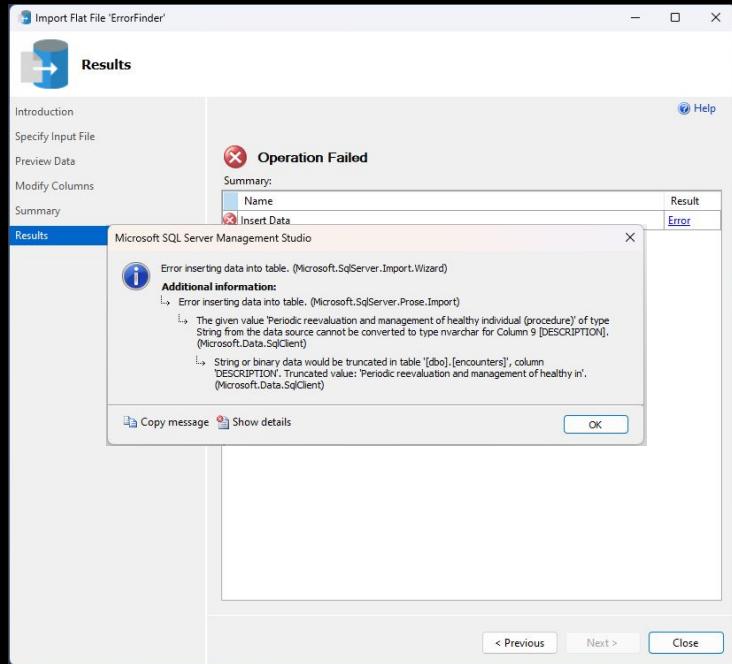
This operation analyzed the input file structure to generate the preview below for up to the first 50 rows.

ITION	BASE_ENCOUNTER	TOTAL_CLAIM_CC	PAYER_COVERAG	REASONCODE	REASONDESCRIP
rfer s...	129.16	129.16	54.16	10509002	Acute bronchiti...
kamin...	129.16	129.16	129.16		
r for s...	129.16	129.16	0.00	36971009	Sinusitis (disord...
kamin...	129.16	129.16	0.00		
kamin...	129.16	129.16	129.16		
ion fo...	129.16	129.16	0.00		
ion fo...	129.16	129.16	0.00		
r for s...	129.16	129.16	64.16	444814009	Viral sinusitis (d...
r for s...	129.16	129.16	64.16	195662009	Acute viral phar...
ion fo...	129.16	129.16	0.00		
r for s...	129.16	129.16	49.16	10509002	Acute bronchiti...
I visit (...)	129.16	129.16	129.16		
r for s...	129.16	129.16	69.16	195662009	Acute viral phar...
r for s...	129.16	129.16	69.16	195662009	Acute viral phar...
kamin...	129.16	129.16	129.16		
I visit (...)	129.16	129.16	129.16		
kamin...	129.16	129.16	129.16		
...

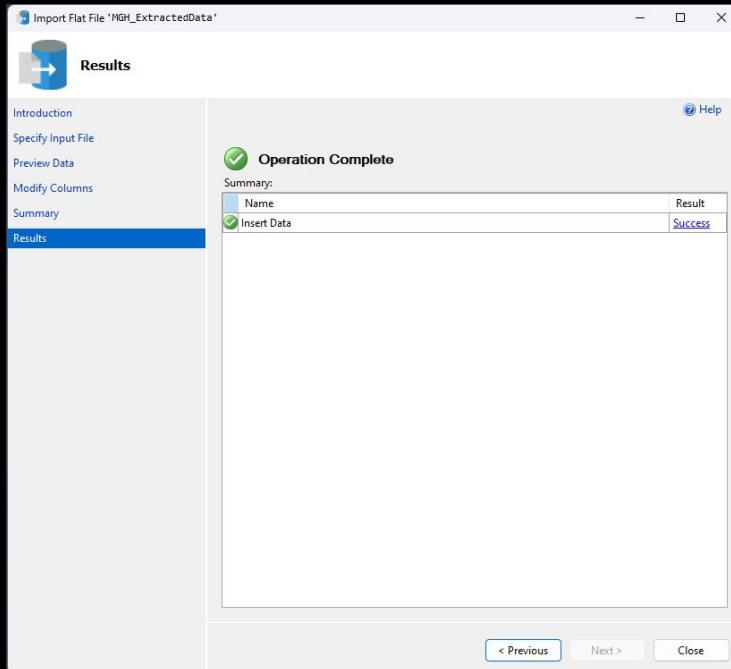
Use Rich Data Type Detection - may provide a closer type fit. However, cells with anomalous values may be dropped.

< Previous Next > Cancel

Importing EHR data



Importing EHR data



Discovering Relationships Between Tables

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

```
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

Id	BIRTHDATE	DEATHDATE	SSN	DRIVERS	PASSPORT	PREFIX	FIRST	LAST	SUFFIX	MAIDEN	MARITAL	RACE	ETHNICITY	GENDER	BIRTHPLACE	ADDRESS	CITY	STATE	COUNTY	ZIP	LAT	PAYER	ENCOUNTERCLASS	CODE	DESCRIPTION
1	1994-09-26	23:48:12.000	00000000-0000-0000-0000-000000000000	1994-09-07 00:03:12.000	35c736	d47b3510-2895-3b70-9897-342d681c769d	wellness	162673000														General examination of patient (procedure)			
2	2014-12-03 07:11:00.000	2014-12-03 08:42:11.000	000186d2-1316-4b58-be65-2722339363b	75ea21	7c441ce-0213b95b9ec-dbea3d3c1a	ambulatory	424619006															Prenatal visit			
3	2014-12-06 05:02:00.000	2014-12-06 07:43:02.000	0002adbb-5916-3c94b-bb17-accc692e8	313e3	6e2fa12d-27bd-3701-8d08-dae202c58632	ambulatory	371883000															Outpatient procedure			

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

Id	IS	CITY	STATE	ZIP	LAT	LONG	PHONE	REVENUE	UTILIZATION
1	Street	Fitchburg	MA	01420	42.5864981614063	-71.8052062368281	978-342-9781 Or 978-342-9781	75816.921875	590

Id	NAME	ADDRESS	CITY	STATE	HEADQUARTERED	ZIP	PHONE	AMOUNT_COVERED	AMOUNT_UNCOVERED	REVENUE	COVERED_ENCOUNTERS	UNCOVERED_ENCOUNTERS	COVERED_MEDICATIONS	UNCOVERED_MEDICATIONS	COVERED_PROCEDURE
1	Humans	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	SPECIALTY	ADDRESS	CITY	STATE	ZIP	LAT	LONG	UTILIZATION
1	5259a506-b80b-3ed5-9c30-83188eabf33e	Von197Maz590	M	GENERAL PRACTICE	881 Main Street	Fitchburg	MA	01420	42.5864981614063	-71.8052062368281	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	REASONDDESCRIPTION
1 30-83188eabf33e	3a5b5bd3-12b2-3247-8e8a-0e34d295c736	d47b3510-2895-3b70-9897-342d681c769d	wellness	162673000	General examination of patient (procedure)	129.160003662109	49.159998474121	NULL	NULL
2 7006a8e6b206	56c64467-dd9a-36ca-91c5-4c729475aa21	7c441ce-0213b95b9ec-dbea3d3c1a	ambulatory	424619006	Prenatal visit	129.160003662109	69.1600036621094	72892002	Normal pregnancy
3 9ac430880c12	6804a42-7754-34c4-213-28791e0813e	6e2fa12d-27bd-3701-8d08-dae202c58632	ambulatory	371883000	Outpatient procedure	129.160003662109	54.159998474121	NULL	NULL
4 b-5765d987452	573b76-9d26-3c4-b62c-0a609578b3	7c441ce-0213b95b9ec-dbea3d3c1a	wellness	162673000	General examination of patient (procedure)	129.160003662109	69.160003662109	NULL	NULL
5 i-5a45b070e4	f161a6a29-7081-8d071-8d99-49bec2d4d22d	6e2fa12d-27bd-3701-8d08-dae202c58632	outpatient	185349003	Encounter for check up (procedure)	129.160003662109	54.159998474121	NULL	NULL
6 15-33932d74f1	793c18d-8269-387a-9998-c13549e4248	b-1c428de-4073-1e90-9f91-6ff8c76	ambulatory	185347001	Encounter for problem (procedure)	129.160003662109	0	NULL	NULL
7 17-9191372d580f	0365bed-2aa2-3340-9745-3df5fd1477a	b-1c428de-4073-1e90-9f91-6ff8c76	outpatient	698314001	Consultation for treatment	129.160003662109	0	NULL	NULL
8 10-805e5ae45705	75895e9f-c056-3c5d-b845-e68d2a4971b	047fe63-6215-35eb-9608-f9da36344e	wellness	162673000	General examination of patient (procedure)	129.160003662109	69.1600036621094	NULL	NULL

Discovering Relationships Between Tables

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

```
USE MGH_ExtractedData
GO

SELECT TOP ( ) *
FROM

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

SELECT *
714b9c18-783d-4f52-aa64-cc3a05a286d9*
FROM
WHERE = 'd47b3510-2895-3b70-9897-342d681c769d';

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

PATIENT

Id	NAME	ADDRESS	CITY	STATE	ZIP	LAT	LON	PHONE	REVENUE	UTILIZATION										
1	5259a506-b80b-3ed5-9c30-83188eabf33e	Fitchburg Outpatient Clinic	881 Main Street	Fitchburg	MA	01420	42.586498164063	-71.8052062388281	978-342-9781 Or 978-342-9781	75816.921875										
2	714b9c18-783d-4f52-aa64-cc3a05a286d9																			
Id	DRIVERS	PASSPORT	PREFIX	FIRST	LAST	SUFFIX	MAIDEN	MARITAL	RACE	ETHNICITY	GENDER	BIRTHPLACE	ADDRESS	CITY	STATE	COUNTY	ZIP	LAT	LON	
1	50-7241	S99934122	X12849695X	Mr.	Noble66	Spes176	NULL	NULL	M	white	nonhispanic	M	Taunton Massachusetts US	1050 Rogahn Path	Fitchburg	Massachusetts	Worcester County	NULL	42.6039591298828	
2	714b9c18-783d-4f52-aa64-cc3a05a286d9																			
Id	NAME	ADDRESS	CITY	STATE	HEADQUARTERED	ZIP	PHONE	AMOUNT_COVERED	AMOUNT_UNCOVERED	REVENUE	COVERED_ENCOUNTERS	UNCOVERED_ENCOUNTERS	COVERED_MEDICATIONS	UNCOVERED_MEDICATIONS	COVERED_PROCEDURE					
1	d47b3510-2895-3b70-9897-342d681c769d	Humans	500 West Main St	Louisville	KY	40018	1-844-330-7799	716971.625	842531.4375	11850200	12117	1796	0	4985	12718					
Id	ORGANIZATION	NAME	GENDER	SPECIALTY	ADDRESS	CITY	STATE	ZIP	LAT	LON	UTILIZATION									
1	3a65bdb3-12b2-3247-8e8a-0e34d295c736	5259a506-b80b-3ed5-9c30-83188eabf33e	Von197Maz90	M	GENERAL PRACTICE	881 Main Street	Fitchburg	MA	01420	42.586498164063	-71.8052062388281	590								

Query executed successfully.

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

RESULTS

Provider	Payer	EncounterClass	Code	Description	Base_Encounter_Cost	Total_Claim_Cost	Payer_Coverage	ReasonCode	ReasonDescription
1 30-83188eabf33e	3a65bdb3-12b2-3247-8e8a-0e34d295c736	d47b3510-2895-3b70-9897-342d681c769d	wellness	162673000	General examination of patient (procedure)	129.160003662109	49.159998474121	NULL	NULL
2 70-006a8e62b06	56c64467-dd3a-91c-5-4c729475aa21	7c441ce-021-3b5-b9ec-dbea9ad3c1a	ambulatory	424619006	Prenatal visit	129.160003662109	69.1600036621094	72892002	Normal pregnancy
3 5-ac43088-0-12	68042-2-754-34c-4213-28791e0819e3	6e21a2d-27bd-3d7-8d08-dae202c5832	ambulatory	371883000	Outpatient procedure	129.160003662109	54.159998474121	NULL	NULL
4 b-5765fd87452	573b-76-9d2-6-3-a-b-62c-0a60957863	7c441ce-021-3b5-b9ec-dbea9ad3c1a	wellness	162673000	General examination of patient (procedure)	129.160003662109	69.160003662109	NULL	NULL
5 i-5a45b0-70e4	f161a6a-29-7081-8d071-8d99-49bec2d4d22d	6e21a2d-27bd-3d7-8d08-dae202c5832	outpatient	185349003	Encounter for check up (procedure)	129.160003662109	54.159998474121	NULL	NULL
6 15-33932b-74f1	793c-18d-8269-387a-9598-13549e-4248	b-1c428d-4073-1e-900-6f9-6f8c-76	ambulatory	185347001	Encounter for problem (procedure)	129.160003662109	0	NULL	NULL
7 17-9191372d580f	0365bed-2aa-2-3340-974-3df5fd1477a	b-1c428d-4073-1e-900-6f9-6f8c-76	outpatient	698314001	Consultation for treatment	129.160003662109	0	NULL	NULL
8 10-805e5ae45705	75895e-9f-c-056-3-5d-b-845-68d2a4971b	047fe-3-6215-35eb-9608-f9da-63344e	wellness	162673000	General examination of patient (procedure)	129.160003662109	69.1600036621094	NULL	NULL

Setting Foreign Key Relationships

The screenshot shows the 'Foreign Key Relationships' dialog box open in SQL Server Management Studio. The 'Selected Relationship:' dropdown contains 'FK_encounters_organizations'. The main area displays the properties for this relationship:

- (General)**: Check Existing Data On Create: Yes
- Tables And Columns Specific**:
 - Foreign Key Base Table: encounters
 - Foreign Key Columns: ORGANIZATION
 - Primary/Unique Key Base: organizations
 - Primary/Unique Key Colu Id: organization_id
- Identity**:
 - Name: FK_encounters_organizations
 - Description:
- Table Designer**

Below the dialog, the object explorer shows the database structure:

- dbo.patients
 - Id (PK, nvarchar(50), not null)
 - BIRTHDATE (date, not null)
 - DEATHDATE (date, null)
 - SSN (nvarchar(50), not null)
 - DRIVERS (nvarchar(50), null)
 - PASSPORT (nvarchar(50), null)
 - PREFIX (nvarchar(50), null)
 - FIRST (nvarchar(50), not null)
 - LAST (nvarchar(50), not null)
 - SUFFIX (nvarchar(50), null)
 - MAIDEN (nvarchar(50), null)
 - MARITAL (nvarchar(50), null)
 - RACE (nvarchar(50), not null)
 - ETHNICITY (nvarchar(50), not null)
 - GENDER (nvarchar(50), not null)
 - BIRTHPLACE (nvarchar(100), not null)
 - ADDRESS (nvarchar(50), not null)
 - CITY (nvarchar(50), not null)
 - STATE (nvarchar(50), not null)
 - COUNTY (nvarchar(50), not null)
 - ZIP (smallint, null)
 - LAT (float, not null)
 - LON (float, not null)
 - HEALTHCARE_EXPENSES (float, not null)
 - HEALTHCARE_COVERAGE (float, not null)
- dbo.organizations
 - Id (PK, nvarchar(50), not null)
 - NAME (nvarchar(100), not null)
 - ADDRESS (nvarchar(50), not null)
 - CITY (nvarchar(50), not null)
 - STATE (nvarchar(50), not null)
 - ZIP (nvarchar(50), not null)
 - LAT (float, not null)
 - LON (float, not null)
 - PHONE (nvarchar(50), null)
 - REVENUE (float, not null)
 - UTILIZATION (smallint, not null)
- dbo.encounters
 - Id (PK, nvarchar(50), not null)
 - START (datetime, not null)
- dbo.providers
 - Id (PK, nvarchar(50), not null)
 - ORGANIZATION (FK, nvarchar(50), not null)
 - NAME (nvarchar(50), not null)
 - ADDRESS (nvarchar(50), not null)
 - CITY (nvarchar(50), null)
 - STATE_HEADQUARTERED (nvarchar(50), null)
 - ZIP (int, null)
 - PHONE (nvarchar(50), null)
 - AMOUNT_COVERED (float, not null)
 - AMOUNT_UNCOVERED (float, not null)
 - REVENUE (float, not null)
 - COVERED_ENCOUNTERS (smallint, not null)
 - UNCOVERED_ENCOUNTERS (smallint, not null)
 - COVERED_MEDICATIONS (smallint, not null)
 - UNCOVERED_MEDICATIONS (smallint, not null)
 - COVERED_PROCEDURES (int, not null)
 - UNCOVERED_PROCEDURES (smallint, not null)
 - COVERED_IMMUNIZATIONS (smallint, not null)
 - UNCOVERED_IMMUNIZATIONS (smallint, not null)
 - UNIQUE_CUSTOMERS (smallint, not null)
 - QOLS_AVG (float, not null)
 - MEMBER_MONTHS (int, not null)
- dbo.payers
 - Id (PK, nvarchar(50), not null)
 - PATIENT (FK, nvarchar(50), not null)
 - ORGANIZATION (FK, nvarchar(50), not null)
 - PROVIDER (FK, nvarchar(50), not null)
 - PAYER (FK, nvarchar(50), not null)
 - CODE (int, not null)
 - DESCRIPTION (nvarchar(max), not null)
 - BASE_ENCOUNTER_COST (float, not null)
 - TOTAL_CLAIM_COST (float, not null)
 - PAYER_COVERAGE (float, not null)
 - REASONCODE (nvarchar(50), null)
 - REASONDDESCRIPTION (nvarchar(max), null)

Setting Foreign Key Relationships

The screenshot shows the SQL Server Management Studio interface with several windows open, illustrating the process of setting up foreign key relationships.

Left Window: Foreign Key Relationships

- Selected Relationship: **FK_encounters_providers** (FK_providers_organizations)
- Editing properties for existing relationship.
- General:** Check Existing Data On Create Yes
- Identity:** (Name) FK_encounters_providers
- Table Designer:** Enforce For Replication Yes, Enforce Foreign Key Constraint Yes
- Buttons:** Add, Delete, Close

Middle Windows: Organization (FK, nvarchar(50), not null)

- Shows the **dbo.providers** table structure with columns: NAME, GENDER, SPECIALTY, ADDRESS, CITY, STATE, ZIP, LAT, LON, and UTILIZATION.
- Shows the **dbo.organizations** table structure with columns: Id (PK, nvarchar(50), not null), NAME, ADDRESS, CITY, STATE, ZIP, LAT, LON, PHONE, REVENUE, and UTILIZATION.

Right Window: Foreign Key Relationships

- Selected Relationship: **FK_encounters_organizations** (**FK_providers_organizations**)
- Editing properties for existing relationship.
- General:** Check Existing Data On Create Yes
- Identity:** (Name) FK_encounters_organizations
- Table Designer:** Enforce For Replication Yes, Enforce Foreign Key Constraint Yes
- Buttons:** Add, Delete, Close

Setting Foreign Key Relationships

The image shows a screenshot of a database management system interface, likely Microsoft SQL Server Management Studio (SSMS), demonstrating the process of setting up foreign key relationships between tables.

Left Panel: Shows the structure of the `dbo.patients` table. It contains the following columns:

- Id (PK, nvarchar(50), not null)
- BIRTHDATE (date, not null)
- DEATHDATE (date, null)
- SSN (nvarchar(50), not null)
- DRIVERS (nvarchar(50), null)
- PASSPORT (nvarchar(50), null)
- PREFIX (nvarchar(50), null)
- FIRST (nvarchar(50), not null)
- LAST (nvarchar(50), not null)
- SUFFIX (nvarchar(50), null)
- MAIDEN (nvarchar(50), null)
- MARITAL (nvarchar(50), null)
- RACE (nvarchar(50), not null)
- ETHNICITY (nvarchar(50), not null)
- GENDER (nvarchar(50), not null)
- BIRTHPLACE (nvarchar(100), not null)
- ADDRESS (nvarchar(50), not null)
- CITY (nvarchar(50), not null)
- STATE (nvarchar(50), not null)
- COUNTY (nvarchar(50), not null)
- ZIP (smallint, null)
- LAT (float, not null)
- LON (float, not null)
- HEALTHCARE_EXPENSES (float, not null)
- HEALTHCARE_COVERAGE (float, not null)

Central Window (Top): Shows the "Foreign Key Relationships" dialog for the relationship `FK_encounters_patients`. The "Tables And Columns Specification" section is expanded, showing:

- Foreign Key Base Table: `encounters`
- Foreign Key Columns: `PATIENT`
- Primary/Unique Key Base: `patients`
- Primary/Unique Key Colu: `Id`

Central Window (Bottom): Shows the "Foreign Key Relationships" dialog for the relationship `FK_payer_transitions_payers`. The "Tables And Columns Specification" section is expanded, showing:

- Foreign Key Base Table: `payer_transitions`
- Foreign Key Columns: `PAYER`
- Primary/Unique Key Base: `payers`
- Primary/Unique Key Colu: `Id`

Right Panel: Shows the structure of the `dbo.payers` table. It contains the following columns:

- Id (PK, nvarchar(50), not null)
- NAME (nvarchar(50), not null)
- ADDRESS (nvarchar(50), null)
- CITY (nvarchar(50), null)
- STATE_HEADQUARTERED (nvarchar(50), null)
- ZIP (int, null)
- PHONE (nvarchar(50), null)
- AMOUNT_COVERED (float, not null)
- AMOUNT_UNCOVERED (float, not null)
- REVENUE (float, not null)
- COVERED_ENCOUNTERS (smallint, not null)
- UNCOVERED_ENCOUNTERS (smallint, not null)
- COVERED_MEDICATIONS (smallint, not null)
- UNCOVERED_MEDICATIONS (smallint, not null)
- COVERED_PROCEDURES (int, not null)
- UNCOVERED_PROCEDURES (smallint, not null)
- COVERED_IMMUNIZATIONS (smallint, not null)
- UNCOVERED_IMMUNIZATIONS (smallint, not null)
- UNIQUE_CUSTOMERS (smallint, not null)
- QOLS_AVG (float, not null)
- MEMBER_MONTHS (int, not null)

Diagram Labels:

- `PATIENT (FK, nvarchar(50), not null)`
- `PAYER (FK, nvarchar(50), not null)`
- `dbo.payer_transitions`
- `dbo.patients`
- `dbo.payers`

Examining Relationships Between Tables

Results Messages														
	START	STOP	PATIENT	ORGANIZATION	PROVIDER	PAYER	ENCOUNTERCLASS	CODE	DESCRIPTION					
1	0000007-937c-498e-9d4e-32a5d29de39	1994-09-06 23:48:12.000	1994-09-07 00:03:12.000	71469c18-783d-4f52-aa54-cc3a05a286d9	5259a506-603b-3ed5-9c30-83180ebf33e	d47b3510-2895-3b70-9897-342d681c769d	wellness	162673000	General examination of patient (procedure)					
2	0001862-1316-46584	1994-09-06 23:48:12.000	1994-09-07 00:03:12.000	7fe-aed8-a27a02e054a	8ed64ef-c817-3753-bee7-0068a662e06	56c64467-d03a-36ca-91c5-4c729475ea21	wellness	162673000	General examination of patient (procedure)					
3	00020bb-59c-3494-ba1-10a4ec0520e0	2014-12-09 07:32:00.000	2014-12-09 07:43:00.000	7e-ae2e-e1c2-1ad-8b3e-5933bd553c4	493180a6-60b-3c7a-0596-ac4308b0c12	6894af2-775d-3a4c-4b213-2879fe0813e3	ambulatory	162673000	Outpatient procedure					

	NAME	ADDRESS	CITY	STATE	ZIP	LAT	LON	PHONE	REVENUE	UTILIZATION
1	5259a506-603b-3ed5-9c30-83180ebf33e	Fitchburg Outpatient Clinic	881 Main Street	Fitchburg	MA	01420	42.5964088164063	-71.8052062988281	978-342-9781 Or 978-342-9781	590

	BIRTHDATE	DEATHDATE	SSN	DRIVERS	PASSPORT	PREFIX	FIRST	LAST	SUFFIX	MAIDEN	MARITAL	RACE	ETHNICITY	GENDER	BIRTHPLACE	ADDRESS	CITY	STATE	COUNTY	ZIP	LAT	LON
1	71469c18-783d-4f52-aa54-cc3a05a286d9	1946-01-22	NULL	999-50-7241	599934122	X12849635X	Mr.	Noble66	Sipes176	Spies176	NULL	white	nonhispanic	M	Taunton Massachusetts US	1050 Rogahn Path	Fitchburg	Massachusetts	Worcester County	NULL	42.6039581298828	-71.8052062988281

	NAME	ADDRESS	CITY	STATE_HEADQUARTERED	ZIP	PHONE	AMOUNT_COVERED	AMOUNT_UNCOVERED	REVENUE	COVERED_ENCOUNTERS	UNCOVERED_ENCOUNTERS	COVERED_MEDICATIONS	UNCOVERED_MEDICATIONS	COVERED_PROCEDURE	
1	d47b3510-2895-3b70-9897-342d681c769d	Humana	500 West Main St	Louisville	KY	40018	1-844-330-7799	716971625	8425314375	118500200	12117	1796	0	4985	12718

	ORGANIZATION	NAME	GENDER	SPECIALTY	ADDRESS	CITY	STATE	ZIP	LAT	LON	UTILIZATION
1	3a69db-312b-2-3247-8e8a-0e342d95c73e	5259a506-603b-3ed5-9c30-83180ebf33e	Von197 Mraz590	M	GENERAL PRACTICE	881 Main Street	Fitchburg	MA	01420	42.5964088164063	-71.8052062988281

Query executed successfully.

Massachusetts General Hospital TEAM 1\ONL\53\ MGH_ExtractedData | 00:00:00 | 14 rows

	PROVIDER	PAYER	ENCOUNTERCLASS	CODE	DESCRIPTION	BASE_ENCOUNTER_COST	TOTAL_CLAIM_COST	PAYER_COVERAGE	REASONCODE	REASONDDESCRIPTION
1	3a69db-312b-2-3247-8e8a-0e342d95c73e	d47b3510-2895-3b70-9897-342d681c769d	wellness	162673000	General examination of patient (procedure)	129.160003662109	129.160003662109	49.1599998474121		
2	7-006a06-62e06	56c64467-d03a-36ca-91c5-4c729475ea21	7c441ce-0213-9b5-b9ec-dbea93d31a	424619006	Prenatal visit	129.160003662109	129.160003662109	69.159998474121		
3	iac4308b9-0c12	6004f2-775d-34c4-2123-28791e0813e3	6e31a2d-27bd-3701-8d08-doe202a59632	371803000	Outpatient procedure	129.160003662109	129.160003662109	54.159998474121		normal pregnancy
4	ab-5769d987452	573b9-76-9d26-3c4-b6d2-c0a609678c83	7c441ce-0213-9b5-b9ec-dbea93d31a	162673000	General examination of patient (procedure)	129.160003662109	129.160003662109	69.1600036621094		
5	b-545a07-7064	f161a6a-29-7061-3071-8d59-9b8ec264d22d	6e2fa12d-27bd-3701-8d08-doe202a59632	185349003	Encounter for check up (procedure)	129.160003662109	129.160003662109	54.159998474121		
6	15-53892bd7e11	793c18cd-3269-3d7b-6998-c135459e4248	b1c423d6-4f07-31e0-9070-69fa6fb6c76	185347001	Encounter for problem (procedure)	129.160003662109	129.160003662109	0		
7	17-81a13725050	0365bed-2a334-374-3cf51fd477a	b1c423d6-4f07-31e0-9070-69fa6fb6c76	698314001	Consultation for treatment	129.160003662109	129.160003662109	0		
8	18-805e5ae4705	75895e9-c05-3c5d-b845-a682da4977b	wellness	162673000	General examination of patient (procedure)	129.160003662109	129.160003662109	69.1600036621094		

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 49.159998474121

Von197 Mraz590

Mr. Noble66 Sipes176

Humana

Examining Relationships Between Tables

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

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)* ↗ X

```
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)* ↗ X

```
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)* ↗ X

```
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;
```

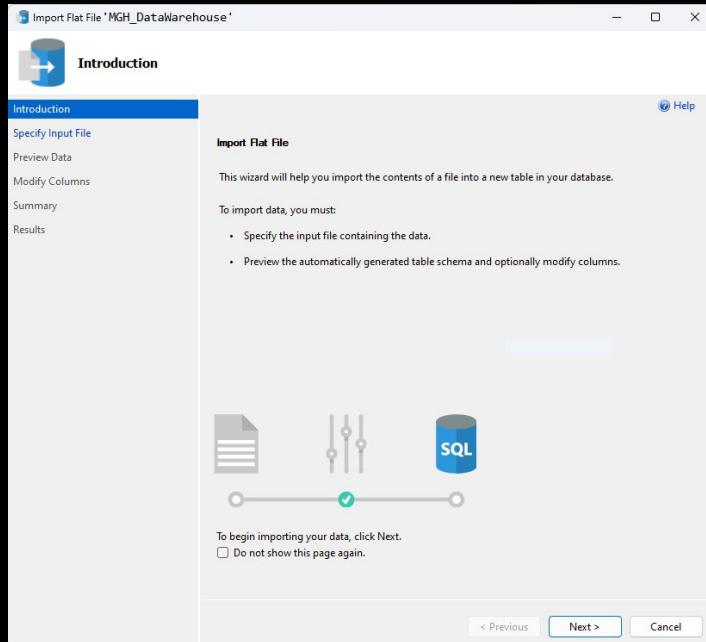
Results		Messages
	GENDER	Avg_Healthcare_Expense
1	F	128.760339789213
2	M	128.740081105064

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))* p X
```

```
USE master  
GO  
  
CREATE DATABASE MGH_DataWarehouse;
```



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

```
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;
```

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

Results Messages

	CITY	Number of Patients
1	Acushnet	1
2	Adams	1
3	Anover	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	Colan	1
14	Dalton	1
15	Deefield	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	Grindvalley	1
27	Halifax	1
28	Hanson	1
29	Hanwich	1
30	Holbrook	1
31	Holden	1
32	Holland	1
33	Lancaster	1
34	Lenox	1
35	Leyden	1
36	Littleton	1
37	Levett	1
38	Manchester-by-the-Sea	1
39	Mashpee	1
40	Mattapoisett	1
41	Ironfield	1

Results Messages

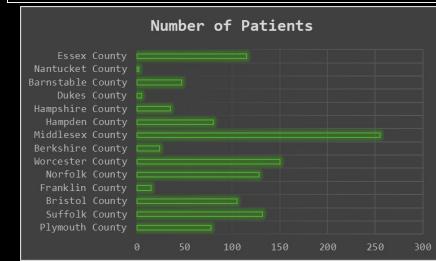
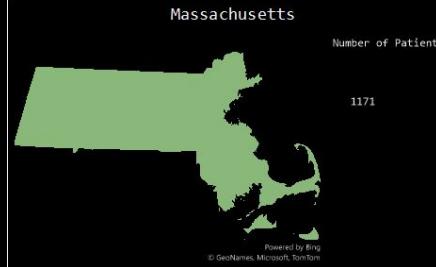
	STATE	Number of Pat
1	Massachusetts	1171

 Query executed successfully

Results Messages

	COUNTY	Number of Firms
1	Plymouth County	
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	Easx County	115

 Query executed successfully.



Data Analysis

SQLQuery1.sql - TEAM 1\DNL(53)* ↻ X

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

Results		Messages	
Providers in Providers Table	5855	Specialties in Providers Table	69
Providers in Encounters Table	1104	Specialties in Encounters Table	1

Query executed successfully.

Results		Messages	
Organizations in Encounters Table	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;
```

COUNTY	Total Claims
1 Middlesex County	12870275.8635712
2 Worcester County	99649.958045959
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	32479.348900659
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.

COUNTY	Total Healthcare Expenses
1 Middlesex County	187494541.057129
2 Worcester County	131865518.449219
3 Norfolk County	105687814.407227
4 Bristol County	8775036.7634777
5 Essex County	8642026.260498
6 Suffolk County	81792734.3118896
7 Plymouth County	61923879.0559082
8 Hampden County	48784674.9926758
9 Barnstable County	3769127.2490234
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

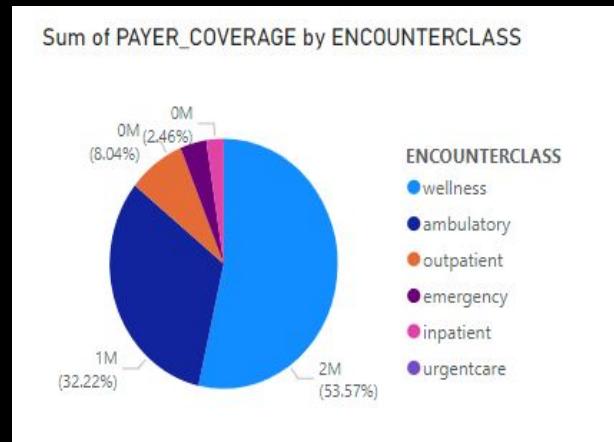
✓ Query executed successfully.

✓ What is the total coverage per encounter class?

SQLQuery1.sql - TEAM 1\DNL(53)* ↻ X

```
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;
```

	ENCOUNTERCLASS	avg_payer_coverage
1	urgentcare	0
2	outpatient	29.5304730491719
3	inpatient	44.2587445630602
4	ambulatory	56.2987749350313
5	emergency	58.8934183294123
6	wellness	92.6944630895499

	provider_name	organization_name	encounter_count
1	Gaynell125 Streich926	VA Boston Healthcare System Jamaica Plain Campus	3217
2	Gertruds163 Schaden604	Worcester Outpatient Clinic	1972
3	Vern731 Powlowki563	HALLMARK HEALTH SYSTEM	1658
4	Jeanmarie510 Beatty507	CAMBRIDGE HEALTH ALLIANCE	1028
5	Male198 Fram345	NEWTON-WELLESLEY HOSPITAL	955
6	Luke971 Rath777	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	Gaynell125 Streich926	STURDY MEMORIAL HOSPITAL	783
11	Bryant814 Murazki203	NANTUCKET COTTAGE HOSPITAL	666
12	Zara914 Considine820	LAHEY HOSPITAL & MEDICAL CENTER BURLIN...	662
13	Suzette512 Monahan736	MOUNT AUBURN HOSPITAL	646
14	Guadalupe206 Valenzue...	COOLEY DICKINSON HOSPITAL INC THE	625
15	Gertruds163 Schaden604	SIGNATURE HEALTHCARE BROCKTON HOSPIT...	620
16	Sanford861 Gottlieb798	BETH ISRAEL DEACONESS HOSPITAL-MILTON ...	609
17	Houston994 Osinski784	NORTH SHORE MEDICAL CENTER -	604
18	Myong12 Heidenereich818	SOUTHCOAST HOSPITAL GROUP INC	581
19	Philip440 McCullough561	CAPE COD HOSPITAL	561
20	Veda284 Pfeffer420	LOWELL GENERAL HOSPITAL	554

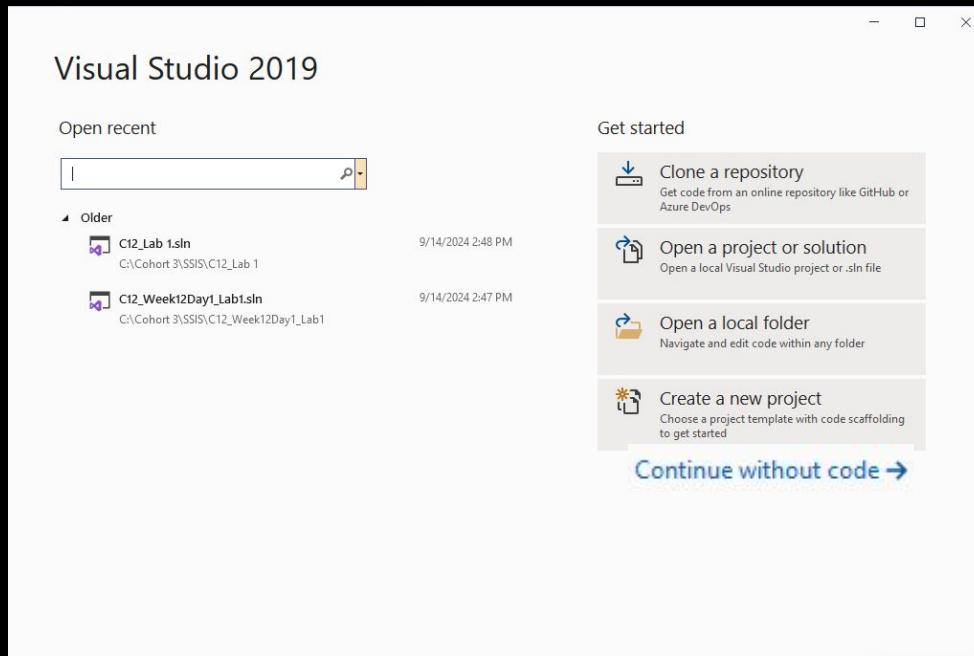
SSIS



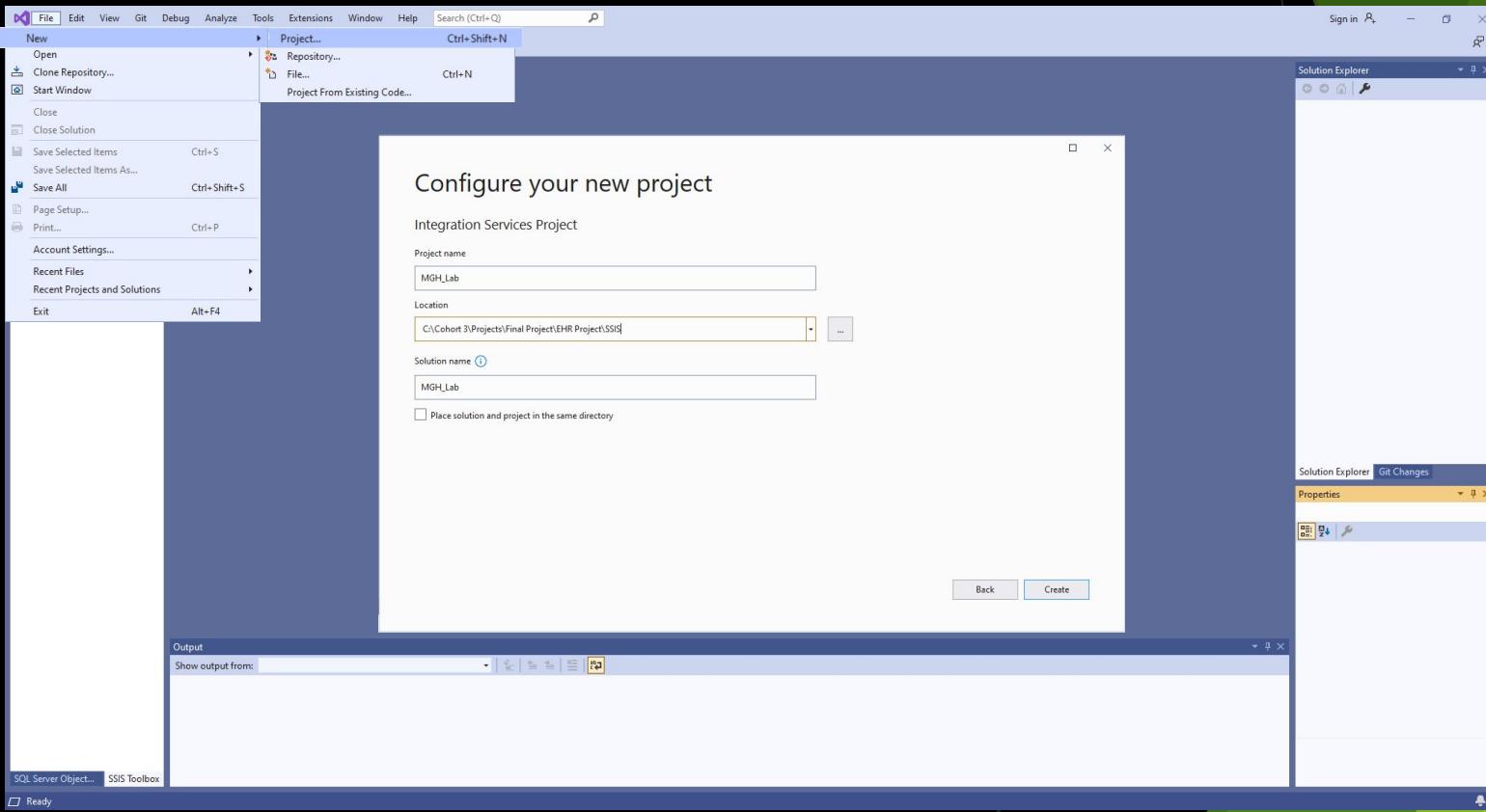
Visual Studio

2019

Start New SSIS Project



Start New SSIS Project



Rename Packages

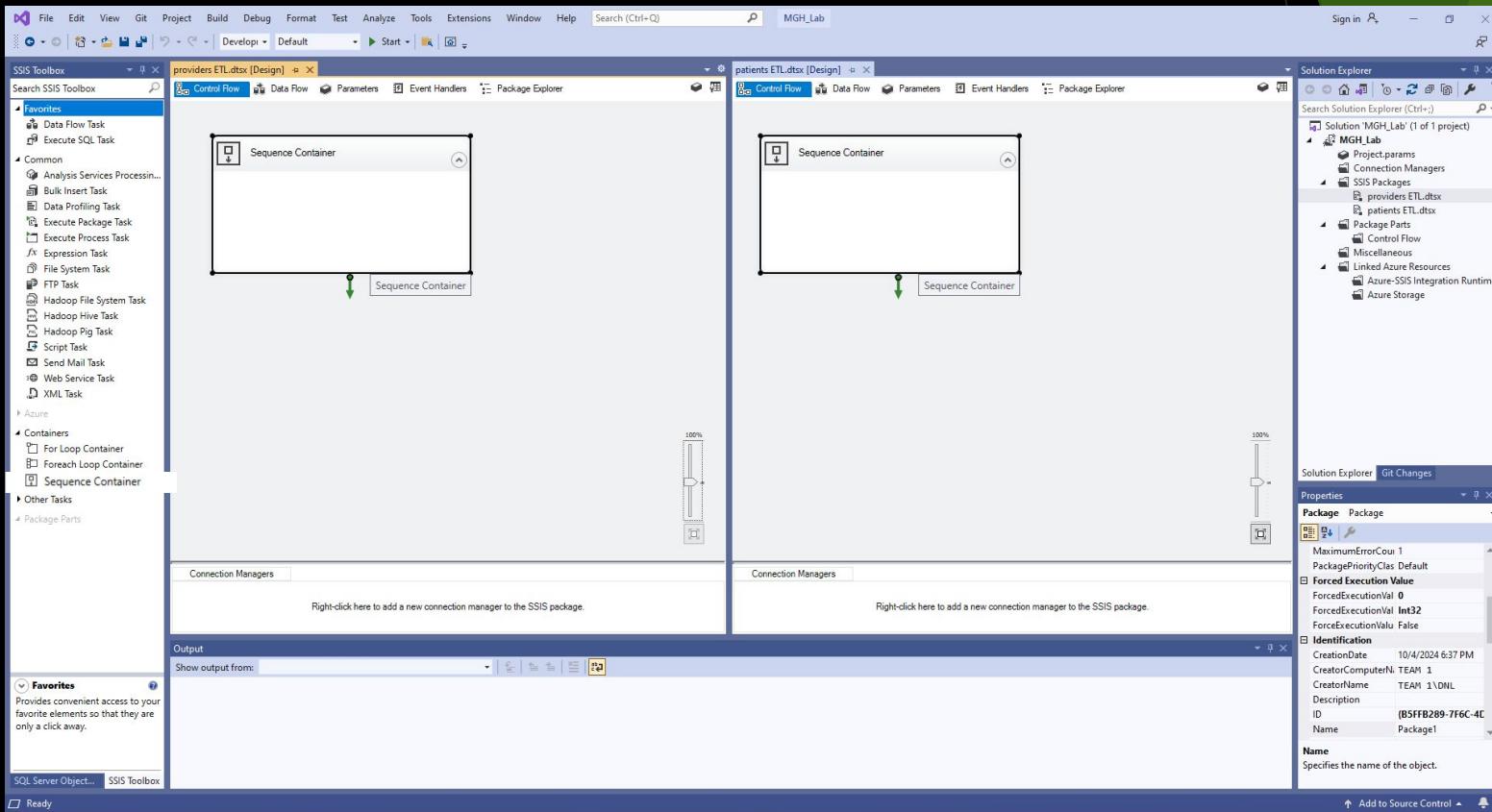
The screenshot shows the Microsoft SQL Server Data Tools (SSIS) interface. In the center, there is a blank SSIS package design surface with a toolbar at the top. On the left, the SSIS Toolbox is open, displaying various task categories like Common, Azure, Containers, and Other Tasks. On the right, the Solution Explorer shows a project named 'MGH_Lab' containing a package named 'ETL.dtsx'. A context menu is open over the package node in the Solution Explorer, with 'Rename' highlighted. The 'Properties' window is also visible on the right, showing the package's properties such as Name, Description, and ID.

Right-click here to add a new connection manager to the SSIS package.

Properties

Package	Value
MaximumErrorCount	1
PackageName	Default
ForceExecutionValue	0
ForceExecutionVal	Int32
ForceExecutionValue	False
Identification	
CreationDate	10/4/2024 6:37 PM
CreatorComputerName	TEAM 1\DNL
CreatorName	TEAM 1\DNL
Description	
ID	(B5FFB289-7F6C-4C
Name	Package1

Create Sequence Containers



Create Data Flows

The screenshot displays the Microsoft SQL Server Integration Services (SSIS) Designer interface, showing two SSIS packages: "providers ETL.dtsx [Design]" and "patients ETL.dtsx [Design]".

Solution Explorer: Shows the project structure for "MGH_Lab" which includes "Project.params", "Connection Managers", "SSIS Packages" (containing "providers ETL.dtsx" and "patients ETL.dtsx"), "Package Parts", "Miscellaneous", and "Linked Azure Resources" (including "Azure-SSIS Integration Runtime" and "Azure Storage").

Properties Window: The properties for the "Sequence Container Sequence" are displayed, including:

- General:** FailPackageOnFailure: False, FailParentOnFailure: False, MaximumErrorCount: 1.
- Forced Execution Value:** ForcedExecution: 0, ForcedExecutionType: Int32, ForcedExecutionValue: False.
- Identification:** Description: Sequence Container, ID: (63865A9E-DF79-409A-
Name: Sequence Container.

Output Window: Shows the output configuration for the Sequence Container.

SSIS Toolbox: The "Sequence Container" task is selected in the toolbox.

Control Flow and Data Flow: Both packages have a "Sequence Container" which contains a "Data Flow Task". The "Control Flow" tab is selected in both package windows.

Set up Connections

The screenshot shows the Microsoft SQL Server Integration Services (SSIS) environment. The main window is the "OLE DB Source Editor" for the "providers ETL.dtsx [Design]" package. The "Connection Manager" dropdown is set to "OLE DB Source". The "Columns" tab is selected, displaying a list of available external columns from the source: Name, Id, ORGANIZATION, NAME, GENDER, SPECIALITY, ADDRESS, CITY, and ZIP. Below this, the "External Column" and "Output Column" mapping table is shown:

External Column	Output Column
Id	Id
ORGANIZATION	ORGANIZATION
NAME	NAME
GENDER	GENDER
SPECIALITY	SPECIALITY
ADDRESS	ADDRESS
CITY	CITY
STATE	STATE
ZIP	ZIP

At the bottom right of the editor are buttons for "OK", "Cancel", and "Help". To the right of the editor, the "Solution Explorer" pane shows the project structure for "MGH_Lab" with files like "providers ETL.dtsx", "OLE DB Destination", and "OLE DB Destination Data Flow Component". The "Properties" pane is open for the "OLE DB Destination Data Flow Component", showing its component class as "OLE DB Destination" and other properties like "Name" and "ConnectionString".

Create Script Components

The screenshot shows the Microsoft Visual Studio interface for creating a Script Component in an SSIS package.

Solution Explorer: Shows the solution 'MGH_Lab' with one project 'MGH_Lab'. The project contains 'Project.params', 'SSIS Packages', 'Package.dtsx', 'Package Parts', 'Control Flow', 'Miscellaneous', and 'Linked Azure Resources'.

Script Transformation Editor: The main window displays the properties of a 'Script' component. The 'Properties' pane is open, showing the following details:

Property	Value
ComponentClassID	{C1463F00-2FAF-4AD4-A212-C9D9CCB54575}
ContactInfo	Includes and runs custom script code. For example,
Description	Includes and runs custom script code. For example,
ID	94
IdentificationString	Script Component
IsDefaultLocale	True
LocaleID	English (United States)
Name	Script Component
PipelineVersion	0
UsesDispositions	False
ValidateExternalMetadata	True
Version	13

Input Columns: The 'Input Column' list contains 'NAME'.

Output: The 'Output' section is currently empty.

Toolbox: The SSIS Toolbox is visible on the left, showing various transformation components like ODBC Source, ODBC Destination, Script Component, and Union All.

Enter Code & Objects

The screenshot shows the Microsoft Visual Studio interface with the following components visible:

- File, Edit, View, Git, Project, Build, Debug, Test, Analyze, Tools, Extensions, Window, Help** menu bar.
- Search (Ctrl+Q)** search bar.
- VstaProjects** tab in the title bar.
- SSIS Toolbox** window on the left.
- Code Editor** window titled `main.cs*` containing C# script code for an SSIS component. The code includes comments explaining methods like `PostExecute` and `Input0_ProcessInputRow`, and a utility method `RemoveNumbers`.
- Solution Explorer** window on the right showing the project structure for "VstaProjects".
- Properties** window below Solution Explorer.
- Output** window at the bottom.
- Ready** status bar at the bottom.

```
SC_e114ec58b1f9487b8c1fb564c456cc49
100    base.PostExecute();
101    /* Add your code here
102    */
103
104
105
106    /// <summary>
107    /// This method is called once for every row that passes through the component from Input0.
108    ///
109    /// Example of reading a value from a column in the the row:
110    /// string zipCode = Row.ZipCode;
111    ///
112    /// Example of writing a value to a column in the row:
113    /// Row.ZipCode = zipCode;
114    /// </summary>
115    /// <param name="Row">The row that is currently passing through the component</param>
116    public override void Input0_ProcessInputRow(Input0Buffer Row)
117    {
118        Row.NAME = RemoveNumbers();
119
120
121        public static string RemoveNumbers(string str)
122        {
123            return Regex.Replace(str, "[^a-zA-ZáéíőűÁÉÍŐŰ]", "", RegexOptions.Compiled);
124        }
125    }
126
```

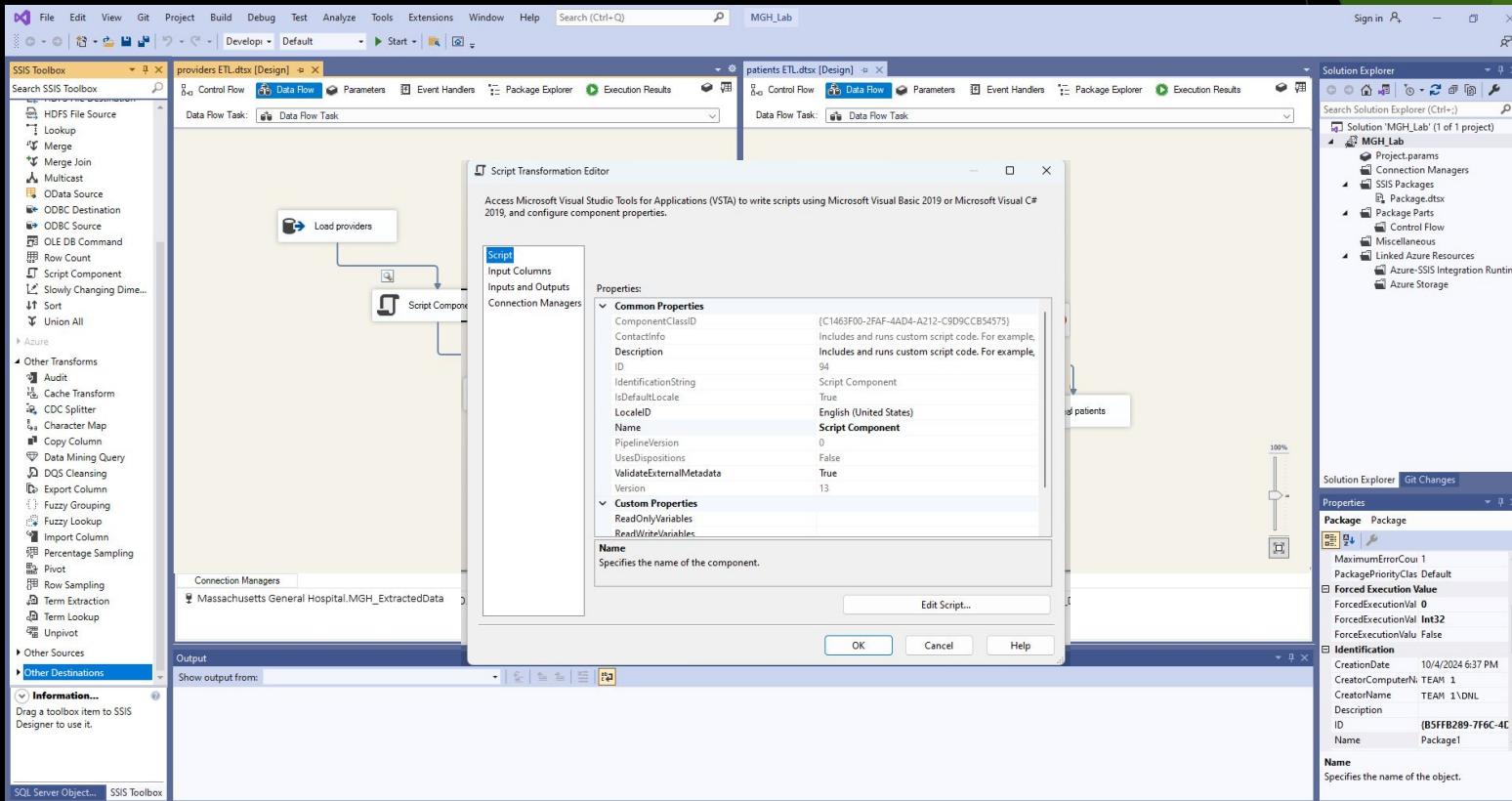
The screenshot shows a software development environment with a code editor and an output window.

Code Editor:

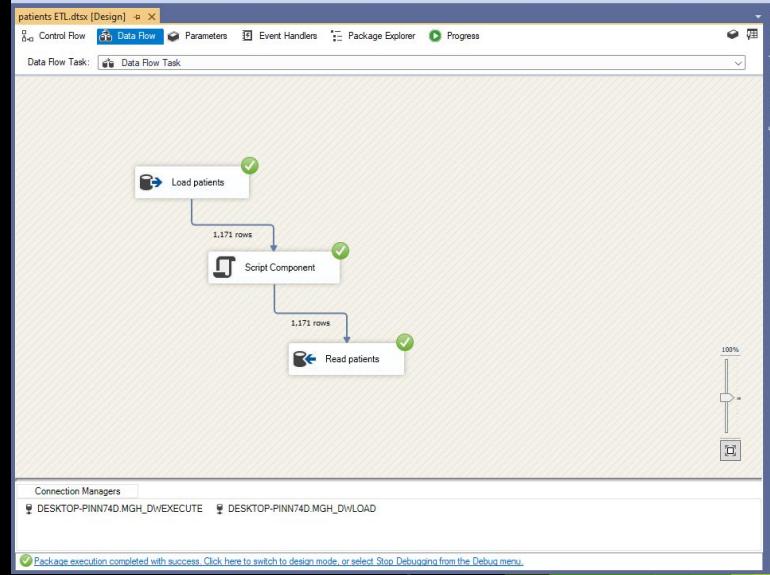
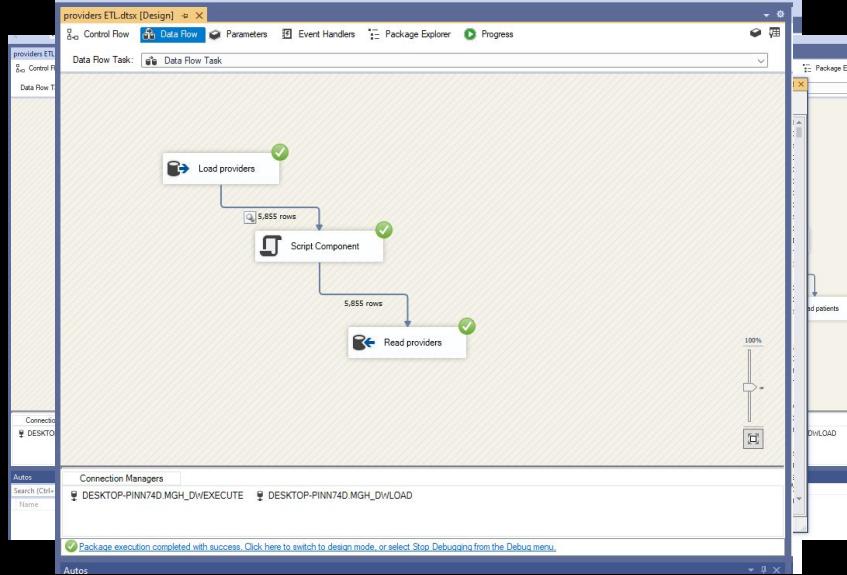
- Title Bar:** main.cs* SC_e114ec58b1f9487b8c1fb564c4566c49
- Code Content:**

```
100     base.PostExecute();
101     /*
102      * Add your code here
103      */
104
105
106     /// <summary>
107     /// This method is called once for every row that passes through the component from Input0.
108     ///
109     /// Example of reading a value from a column in the the row:
110     /// string zipCode = Row.ZipCode
111     ///
112     /// Example of writing a value to a column in the row:
113     /// Row.ZipCode = zipCode
114     /// </summary>
115     /// <param name="Row">The row that is currently passing through the component</param>
2 references
116     public override void Input0_ProcessInputRow(Input0Buffer Row)
117     {
118         Row.NAME = RemoveNumbers();
119     }
120
1 reference
121     public static string RemoveNumbers(string str)
122     {
123         return Regex.Replace(str, "[^a-zA-Záéíñóú ]+", "", RegexOptions.Compiled);
124     }
125
126 }
```
- Output Window:** Shows the message "Ln: 118 Ch: 33 SPC CRLF".

Verify Effects

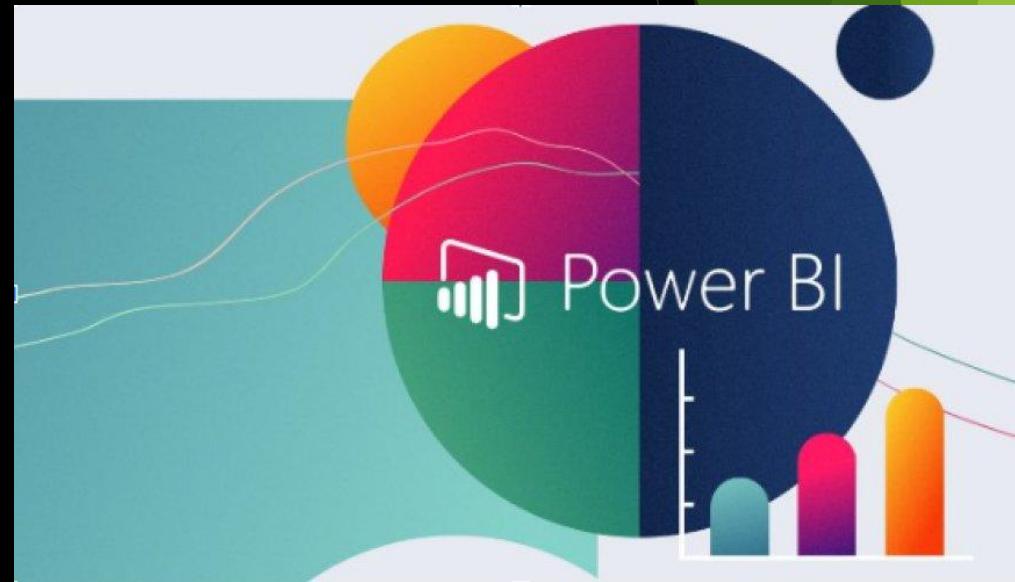


Execute Packages





POWER BI

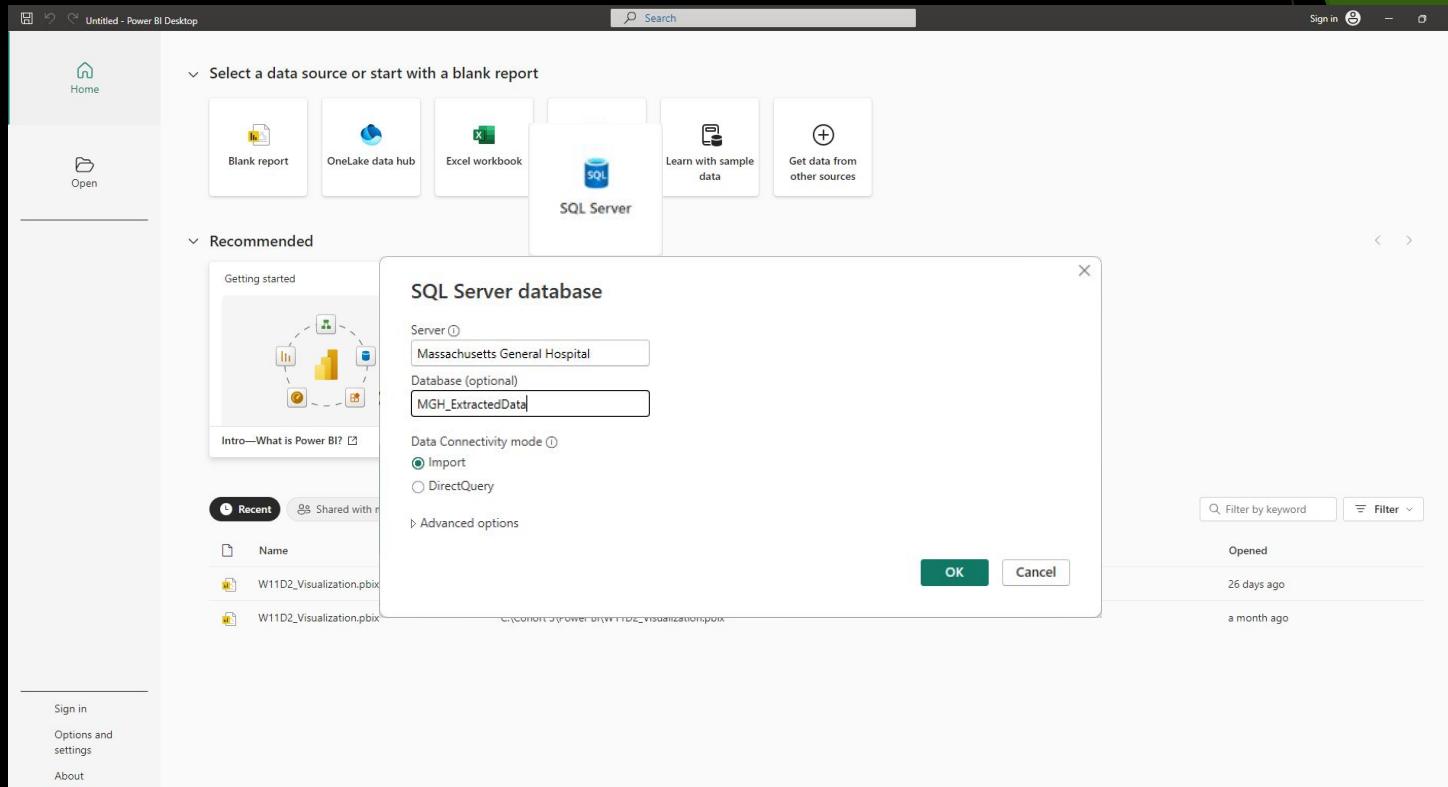


Moving data from SQL to power BI

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

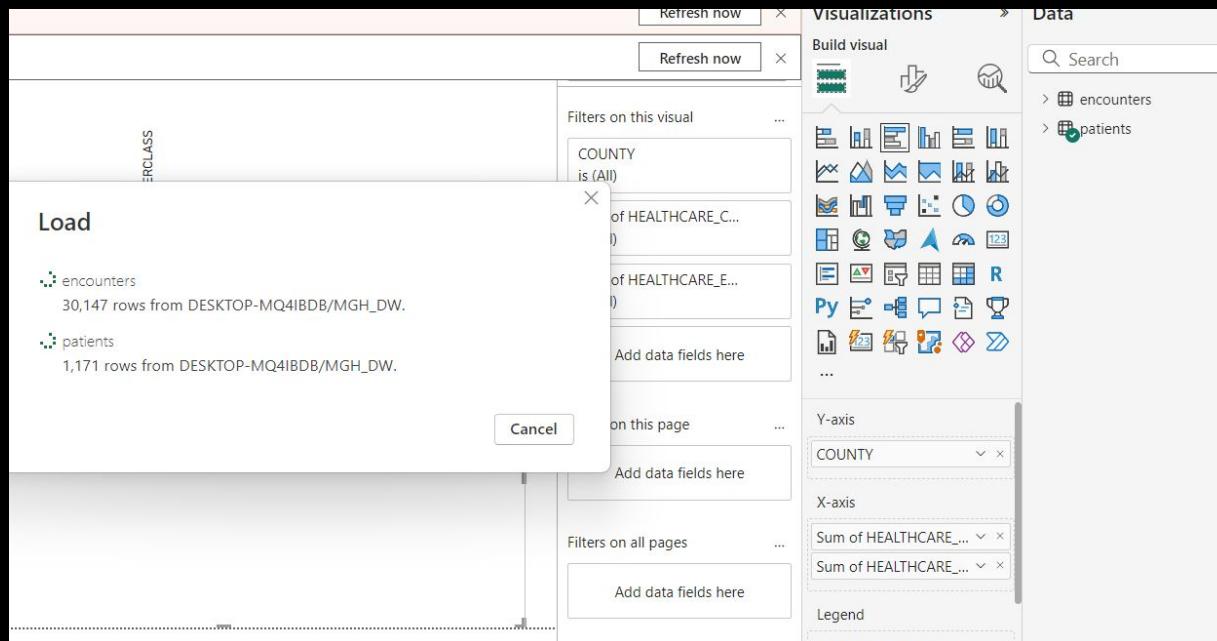


Start New Power Bi Report



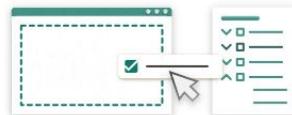
Import data from SQL

- Sync data on SQL and power BI



Import data from SQL

Build visuals with your data
Select or drag fields from the Data pane onto the report canvas.



« **Visualizations** »

Filters

Data

Build visual

encounters patients

Values

Add data fields here

Drill through

Cross-report off

Keep all filters On

Add drill-through fields here

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

GENDER
F
M

Sum of HEALTHCARE_COVERAGE by GENDER

6.61M (43.65%)
8.53M (56.35%)

GENDER
• M
• F

Filters

Search:

Filters on this visual

- GENDER is (All)
- Sum of HEALTHCARE_C... is (All)
- Add data fields here

Filters on this page

- Add data fields here

Filters on all pages

- Add data fields here

Visualizations

Build visual

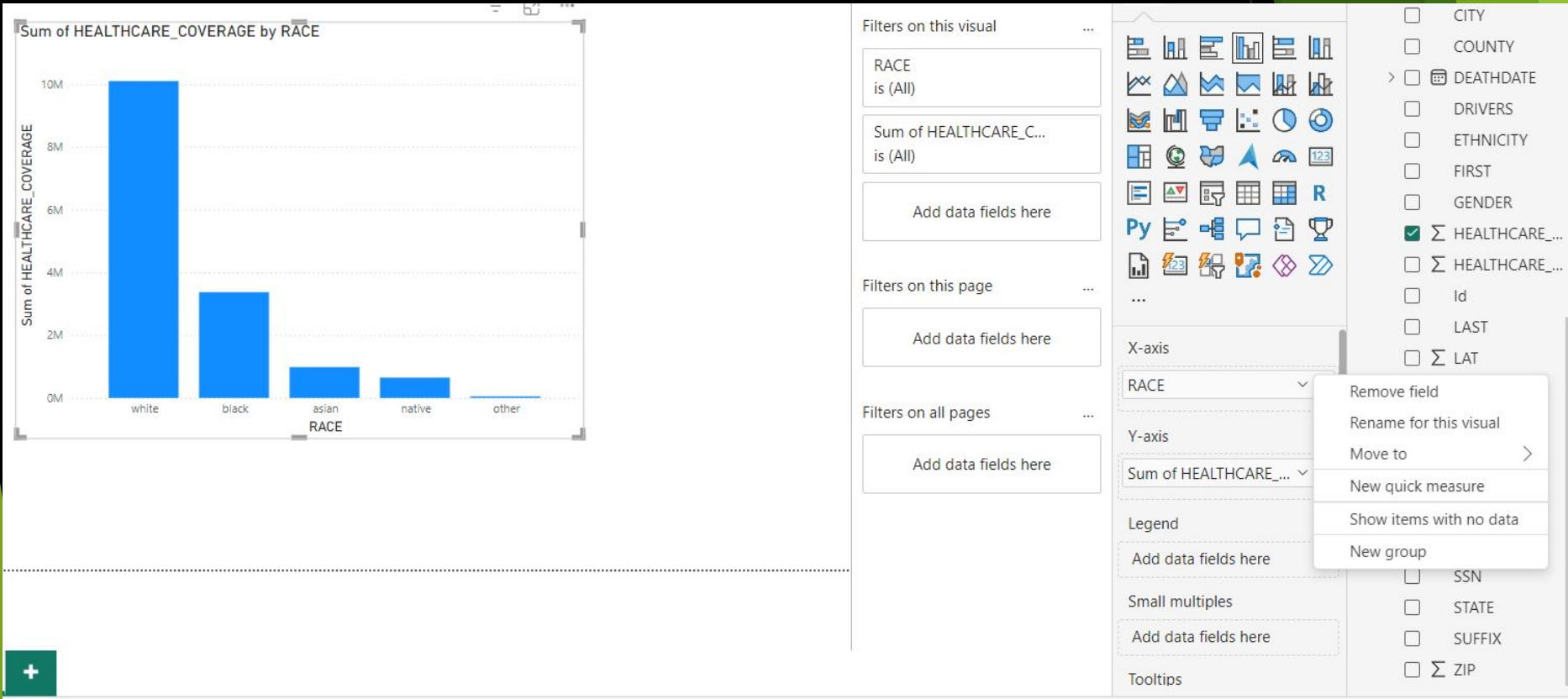
Data

Search:

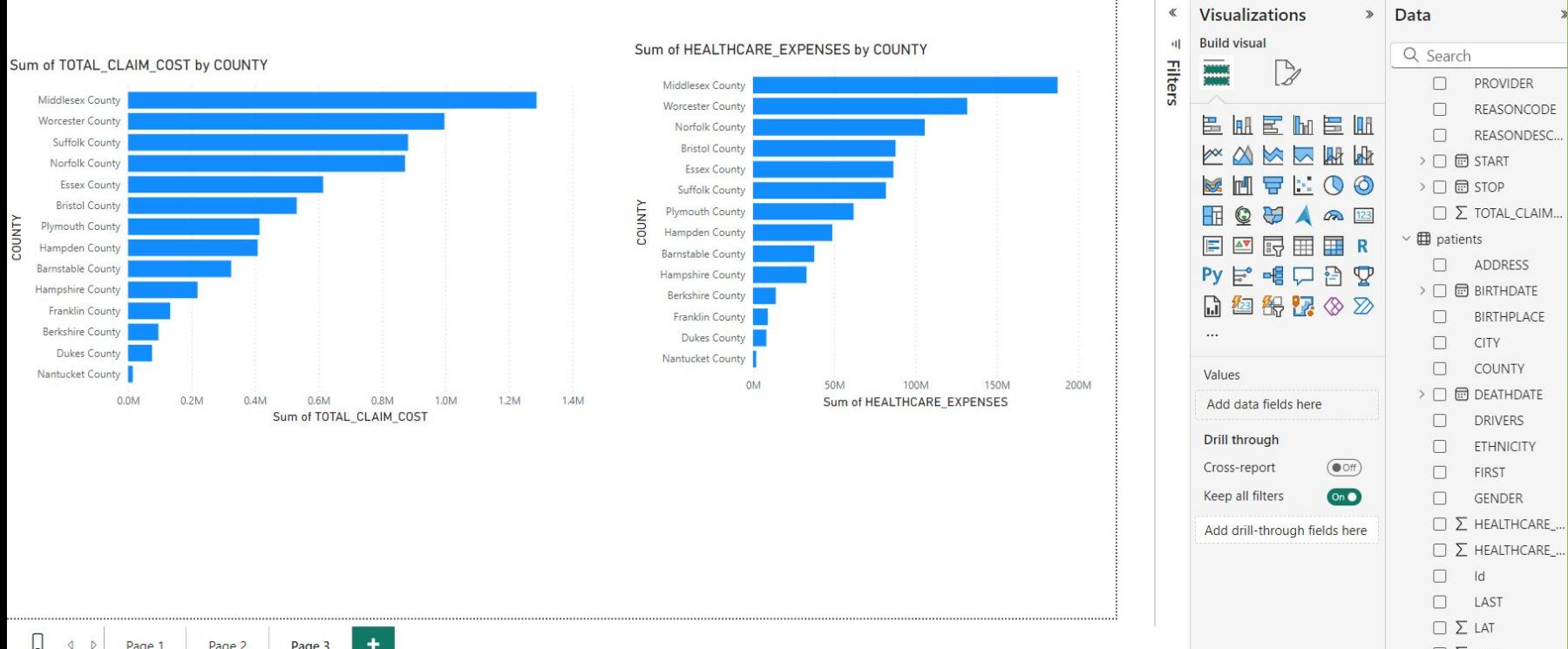
patients

- \sum TOTAL_CLAIM...
- patients
- ADDRESS
- BIRTHDATE
- BIRTHPLACE
- CITY
- COUNTY
- DEATHDATE
- DRIVERS
- ETHNICITY
- FIRST
- GENDER
- \sum HEALTHCARE_...
- \sum HEALTHCARE_...
- Id
- LAST
- \sum LAT
- \sum LON
- MAIDEN
- MARITAL
- PASSPORT
- PREFIX

Show the health coverage for the various races

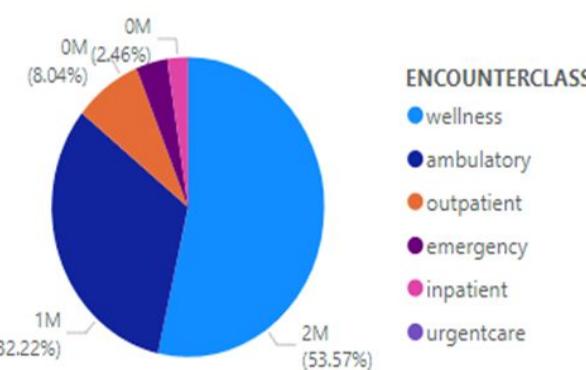


Visualize the healthcare claim and expense per county

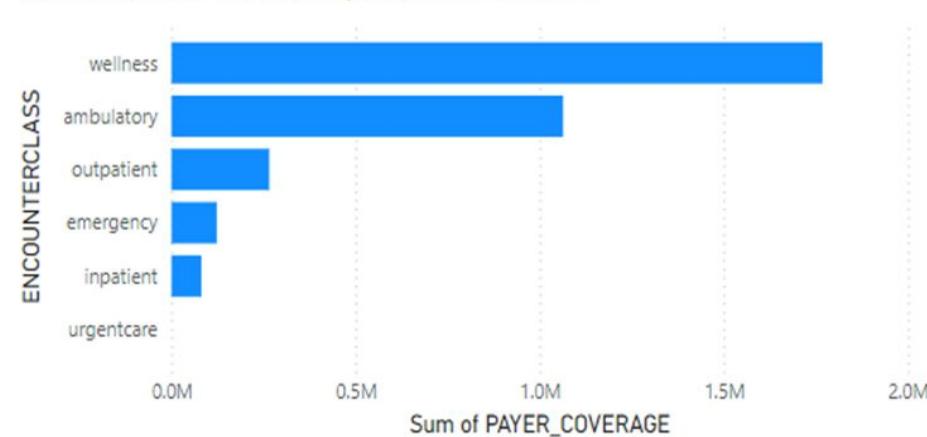


Show the coverage per encounter class

Sum of PAYER_COVERAGE by ENOUNTERCLASS

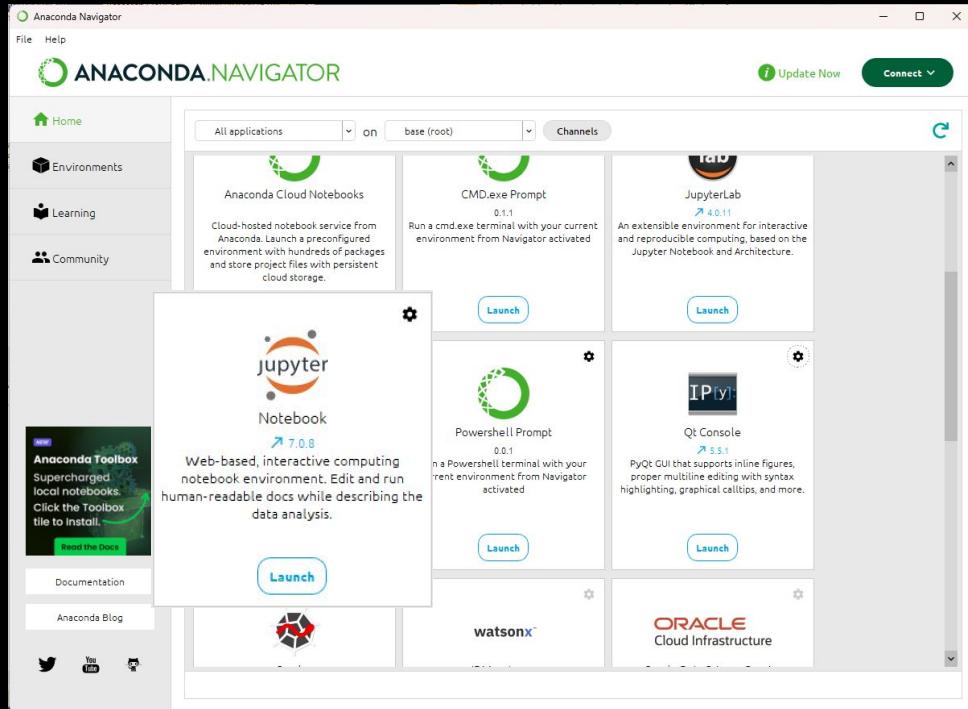


Sum of PAYER_COVERAGE by ENOUNTERCLASS



PYTHON





Data Processing

```
1 import pandas as pd
2 import numpy as np
3 from sklearn.model_selection import train_test_split, cross_val_score
4 from sklearn.preprocessing import StandardScaler, LabelEncoder
5 from sklearn.impute import SimpleImputer
6 from sklearn.neighbors import KNeighborsClassifier
7 from sklearn.ensemble import RandomForestClassifier
8 from sklearn.linear_model import LogisticRegression
9 from sklearn.tree import DecisionTreeClassifier
10 from sklearn.metrics import classification_report, confusion_matrix, roc_auc_score, roc_curve
11 import xgboost as xgb
12 import seaborn as sns
13 import matplotlib.pyplot as plt
14
```

```
[1]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import scikit_learn as sk
import seaborn 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<3.8,!=3.8.* in c:\users\owner\anaconda3\lib\site-packages (from pyreadstat) (2.0.2)
Requirement already satisfied: pytz==2021.1 in c:\users\owner\anaconda3\lib\site-packages (from pyreadstat) (2021.3-post1)
Requirement already satisfied: tzdata>=2021.1 in c:\users\owner\anaconda3\lib\site-packages (from pyreadstat) (2023.3)
Requirement already satisfied: numpy<1.23.0 in c:\users\owner\anaconda3\lib\site-packages (from pyreadstat) (1.24.3)
Requirement already satisfied: six>=1.5 in c:\users\owner\anaconda3\lib\site-packages (from pyreadstat) (1.16.0)

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

```
[4]: df.head()
[4]:   sex age education currentSmoker cigsPerDay BPMeds prevalentStroke prevalentHyp diabetes totChol sysBP diaBP BMI heartRate glucose TenYearCHD
0  1  39.0        4.0      0.00  0.00      0  0.0  195.0 106.0  70.0 26.97  80.0    77.0     0
1  0  46.0        2.0      0.00  0.00      0  0.0  250.0 121.0  81.0 28.73  95.0    76.0     0
2  1  48.0        1.0      1.00  200.0     0.0  0.0  245.0 127.5  80.0 25.34  75.0    70.0     0
3  0  61.0        3.0      1.00  300.0     0.0  0.0  225.0 150.0  95.0 28.58  65.0   103.0    103.0
4  0  46.0        3.0      1.00  230.0     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          3652 non-null   float64
 15  TenYearCHD       4240 non-null   int64  
dtypes: float64(9), int64(7)
memory usage: 530.1 KB
```

```
17 def load_and_examine_data(data_str):
18     # Create DataFrame from the data
19     df = pd.read_csv(data_str)
20
21     print("\nDataset Info:")
22     print(df.info())
23
24     print("\nMissing Values:")
25     print(df.isnull().sum())
26
27     print("\nBasic Statistics:")
28     print(df.describe())
29
30
30 return df
```

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

[7]:   count  mean  std  min  25%  50%  75%  max
sex      4240.0  0.429245  0.495027  0.00  0.00  0.00  1.00  1.00
age      4240.0  49.580189  8.572942  32.00  42.00  49.00  56.00  70.0
education  4135.0  1.979444  1.019791  1.00  1.00  2.00  4.00  4.00
currentSmoker  4240.0  0.494104  0.500024  0.00  0.00  0.00  1.00  1.00
cigsPerDay   4211.0  9.005937  11.922462  0.00  0.00  0.00  20.00  70.0
BPMeds      4187.0  0.026615  0.169544  0.00  0.00  0.00  0.00  1.00
prevalentStroke  4240.0  0.005896  0.076569  0.00  0.00  0.00  0.00  1.00
prevalentHyp    4240.0  0.310613  0.462799  0.00  0.00  0.00  1.00  1.00
diabetes      4240.0  0.257078  0.156280  0.00  0.00  0.00  0.00  1.00
totChol       4190.0  236.699523  44.591284  107.00  206.00  234.00  263.00  696.0
sysBP        4240.0  132.354599  22.033300  83.50  117.00  128.00  144.00  295.0
diaBP        4240.0  82.697759  11.913984  48.00  75.00  82.00  90.00  142.5
BMI          4221.0  25.800801  4.079840  15.54  23.07  25.44  28.04  56.8
heartRate     4239.0  75.678981  12.023340  44.00  68.00  75.00  83.00  143.0
glucose       3652.0  81.963655  23.954335  40.00  71.00  78.00  87.00  394.0
TenYearCHD    4240.0  0.151687  0.358953  0.00  0.00  0.00  0.00  1.00
```

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

```
32 def preprocess_data(data_str):
33     # Create a copy of the data
34     df = data_str.copy()
35
36     # 1. Handle Missing Values
37     # First, ensure all numeric columns are properly typed
38     numeric_features = ['cigsPerDay', 'totChol', 'sysBP', 'diaBP',
39                         'BMI', 'heartRate', 'age', 'glucose']
40
41     # Convert numeric columns to float, keeping NA values as NA
42     for col in numeric_features:
43         df[col] = pd.to_numeric(df[col], errors='coerce')
44
45     # Impute missing values for numeric features
46     numerical_imputer = SimpleImputer(strategy='mean')
47     df[numeric_features] = numerical_imputer.fit_transform(df[numeric_features])
48
```

```
[8]: #Filtering the data
df.isnull()
```

```
[9]:
```

	sex	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	totChol	sysBP	diaBP	BMI	heartRate	glucose	TenYearCHD
0	Female	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
1	Female	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	True	False	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	Female	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	True	False	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	

```
[100]: # def handle_missing_values(df):
101    # numeric_imputer = SimpleImputer(strategy='median')
102    # numeric_imputer = StandardScaler()
103    # Get numeric columns
104    numeric_columns = df.select_dtypes(include=['float64', 'int64']).columns
105    # Impute missing values
106    df[numeric_columns] = numeric_imputer.fit_transform(df[numeric_columns])
107    # return df
108
109    # # Standardize numeric features
110    # scaler = StandardScaler()
111    # numeric_features = ['age', 'cigsPerDay', 'totChol', 'sysBP', 'diaBP', 'BMI',
112    #                      'heartRate', 'glucose']
113    # df[numeric_features] = scaler.fit_transform(df[numeric_features])
114
115    # return df
```

Feature Scaling/Encoding Variables

```
scaler = StandardScaler()  
df[numerical_features] = scaler.fit_transform(df[numerical_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]:  
age education currentSmoker cigsPerDay BPMeds prevalentStroke prevalentHyp diabetes totChol sysBP diaBP BMI heartRate glucose TenYearCHD  
sex  
0 49.738575 1.964128 0.397052 5.509091 0.036329 0.005897 0.311057 0.025061 239.616708 133.265111 82.362405 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
```

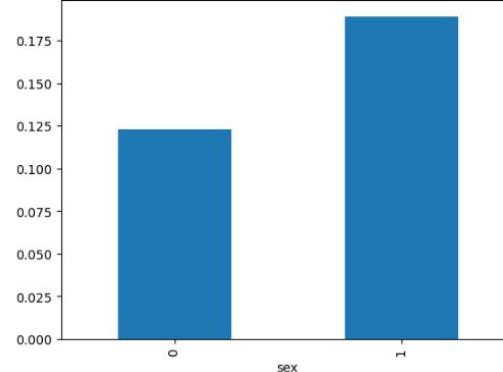
```
[18]: ##df['sex'] == df['sex'].replace([1.0, 0.0], ['male', 'female'])
```

```
[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'>
```



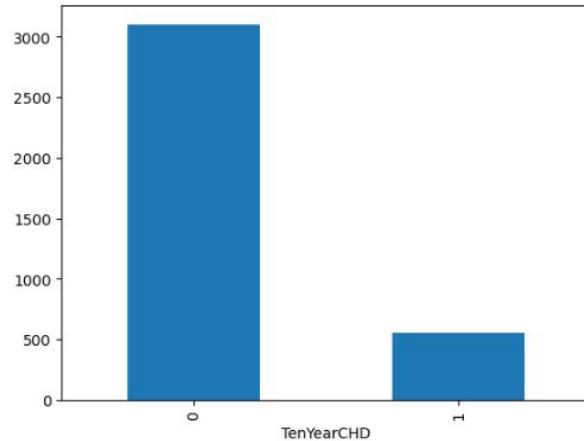
```
[49]: # 2. Handle categorical variables  
[50]: # Ensure education is numeric, handle NA values  
[51]: df['education'] = pd.to_numeric(df['education'], errors='coerce')  
[52]: df['education'] = df['education'].fillna(df['education'].median())  
[53]: # print(df['education'].unique())  
[54]:  
[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)  
[62]:
```

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

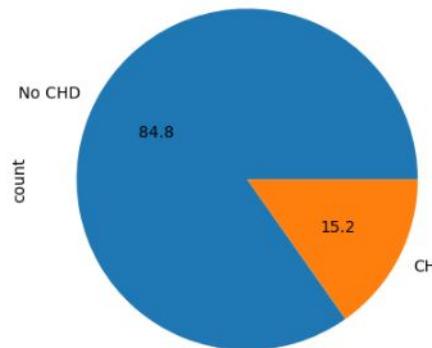
	sex	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	totChol	sysBP	diaBP	BMI	heartRate	glucos	TenYearCHD
sex	1.000000	-0.024412	0.017729	0.206114	0.330322	-0.052124	-0.002312	0.001424	0.013819	-0.070321	-0.045358	0.051872	0.073111	-0.115285	0.002940	0.091688
age	-0.024412	1.000000	-0.159499	-0.210771	-0.168295	0.134732	0.050893	0.036239	0.019092	0.268252	0.388267	0.208283	0.137511	0.118349	0.233983	-0.002722
education	0.017729	-0.159499	1.000000	0.025251	0.013790	-0.013739	-0.030386	0.078828	0.019627	-0.015352	-0.124360	-0.058059	0.137555	-0.064214	-0.031998	-0.003279
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	-0.045054	-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	-0.046594	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	0.060657	1.000000	0.080556	0.166655	0.697675	0.177734	0.302949	0.146818	0.066942	0.181387
diabetes	0.018019	0.109492	-0.034627	-0.041849	-0.036961	0.046066	0.096925	0.080556	1.000000	0.048451	0.102552	0.050684	0.089009	0.060984	0.074820	0.094341
totChol	-0.070321	0.268252	-0.013522	-0.051034	-0.030400	0.094083	0.012736	0.166655	0.048451	1.000000	0.219922	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	0.219922	1.000000	0.786669	0.330917	0.184797	0.114651	0.222821
diaBP	0.051872	0.208283	-0.058059	-0.115955	-0.057476	0.199630	0.055834	0.177734	0.056666	0.174422	0.786669	1.000000	0.385348	0.178744	0.063540	0.159124
BMI	0.073111	0.137511	-0.137555	-0.159821	-0.087395	0.105642	0.036496	0.302949	0.080099	0.121056	0.330917	0.385348	1.000000	0.074131	0.036363	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
glucos	0.002940	0.118349	-0.031998	-0.051324	-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	0.02055	0.020514	0.121990	1.000000

Visualizing the Data Distribution

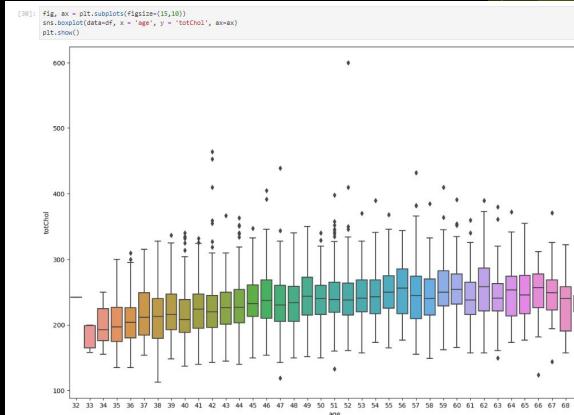
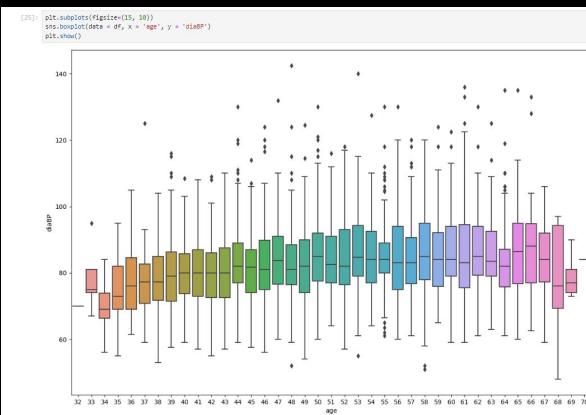
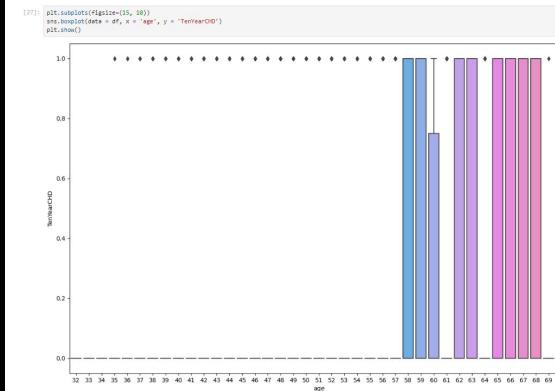
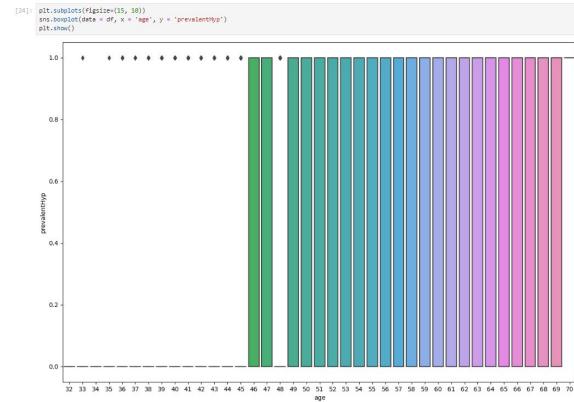
```
[35]: #Visualizing the data Distribution  
df ['TenYearCHD'].value_counts().plot(kind='bar')  
  
[35]: <Axes: xlabel='TenYearCHD'>
```



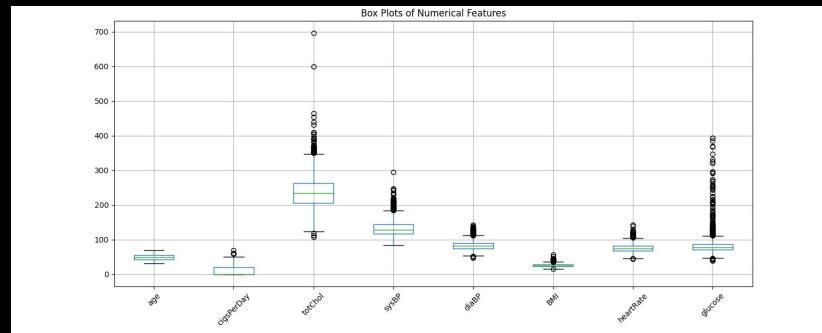
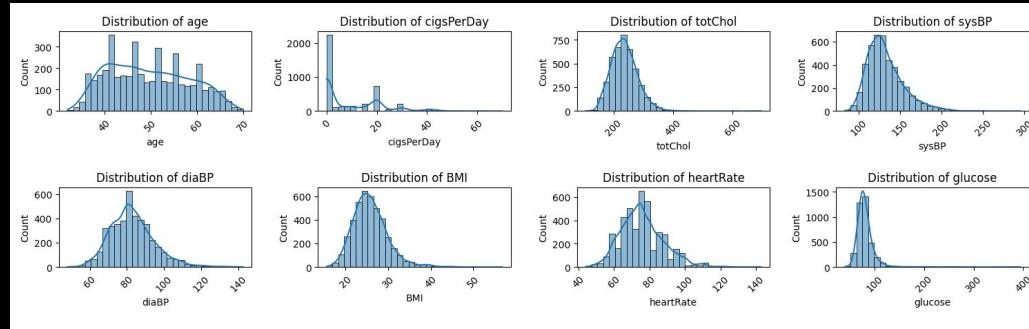
```
[36]: df ['TenYearCHD'].value_counts().plot(kind='pie', autopct='%1.1f', labels=['No CHD', 'CHD'])  
  
[36]: <Axes: ylabel='count'>
```



Exploratory Data Analysis

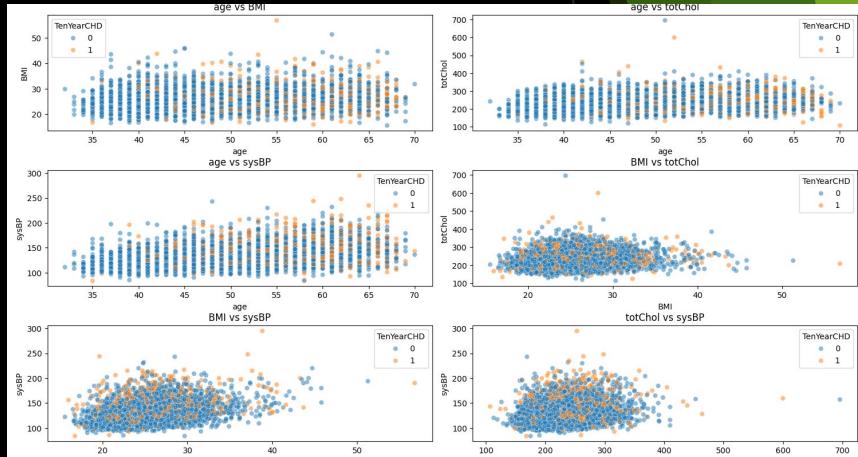
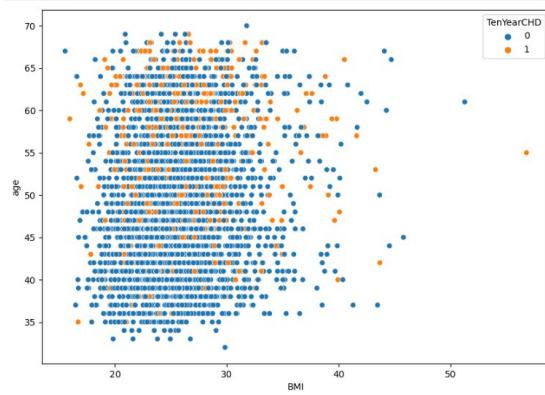


Numerical Distributions of Features

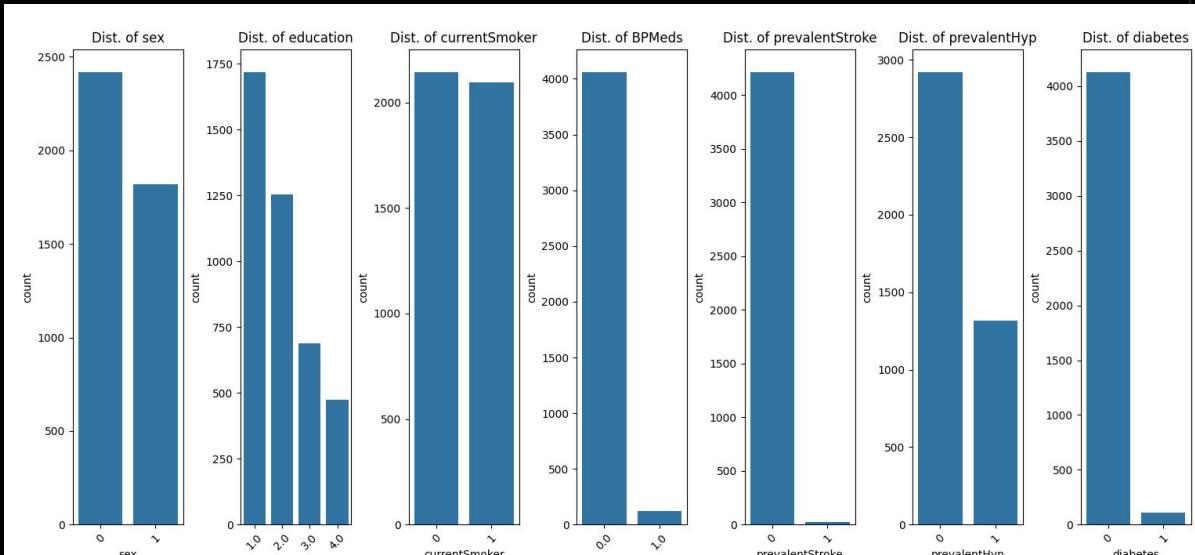


Scatter Relationships

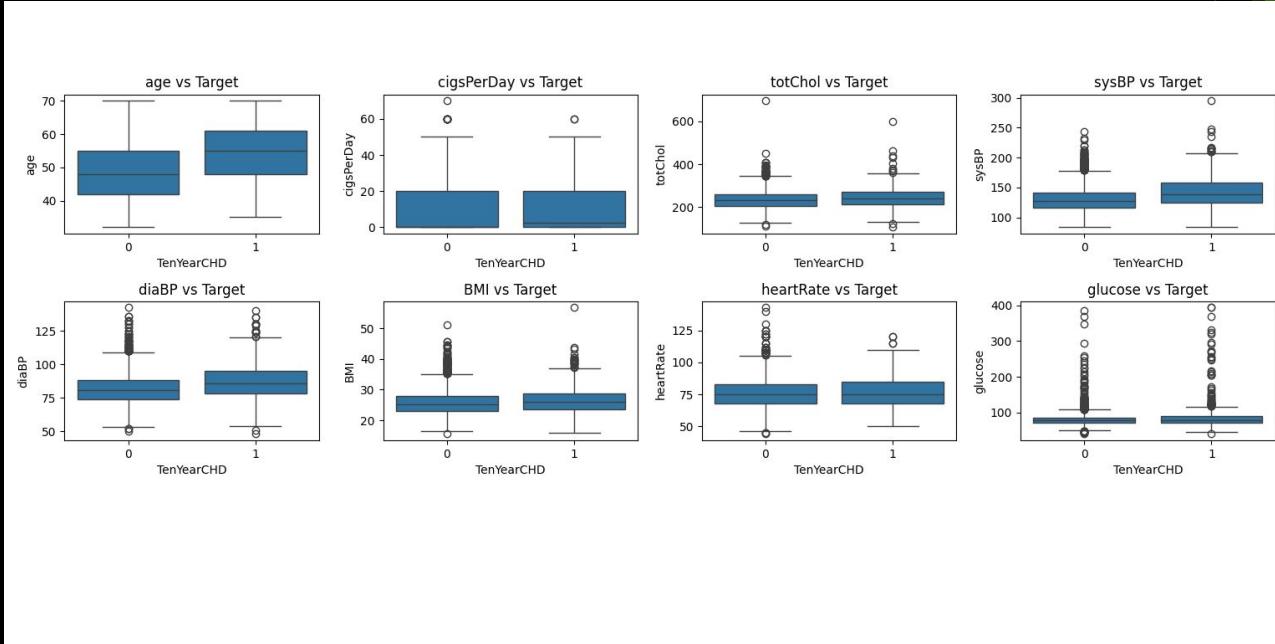
```
[37]: plt.figure(figsize=(10,7))
sns.scatterplot(data=df, x = 'BMI', y = 'age', hue = 'TenYearCHD')
plt.show()
```



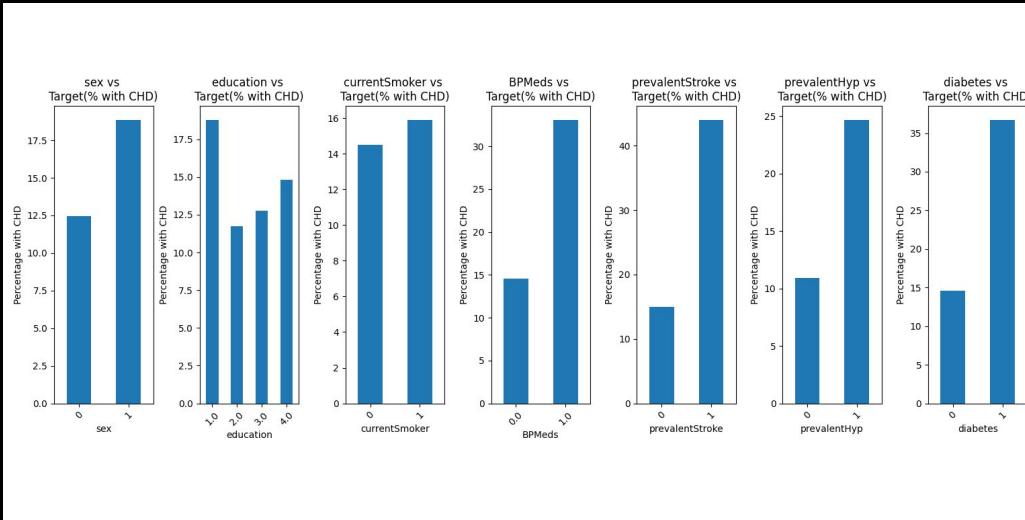
Categorical Distributions

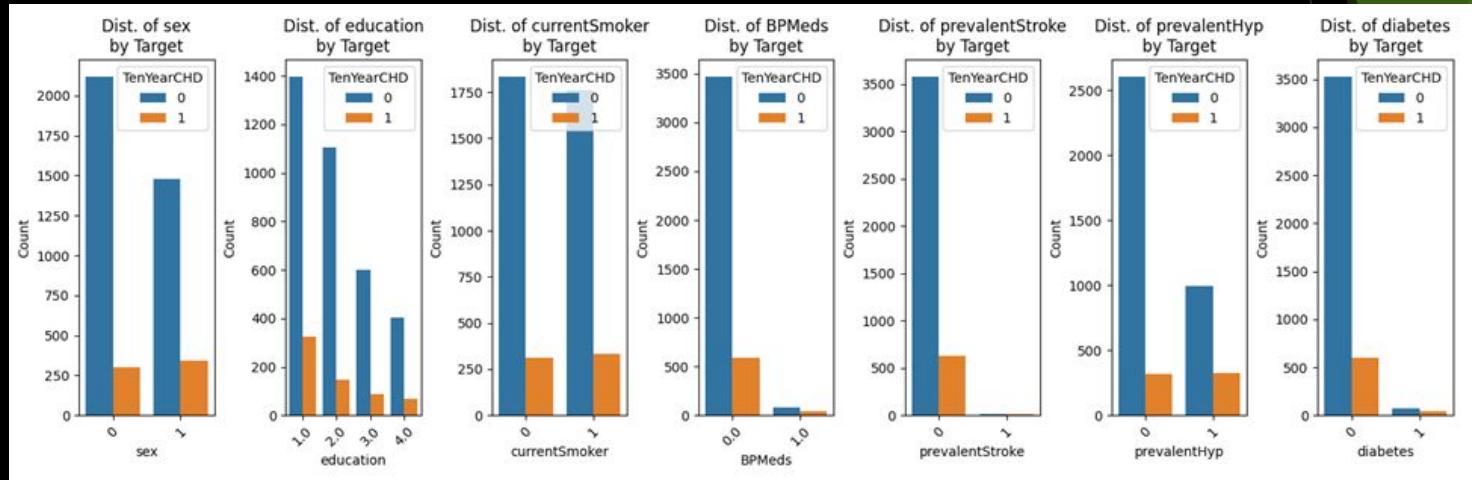


Analyzing Relationships with Target Variable



Categorical Relationships vs Target





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
```

```
[47]: train_test_split(x,y,test_size=0.2)
[47]: [   cigsPerDay  BPMeds
       2454      0.0     0.0
       688       35.0     0.0
       1795      35.0     0.0
       2697      0.0     0.0
       762       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
       ...
       138       30.0     0.0
       4836      0.0     0.0
       1804      0.0     0.0
       1389      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
       2958      0.0     0.0
       ...
       2454      0
       688       0
       1795      0
       2697      0
       762       0
       ...
       1426      0
       3523      0
       3254      1
       2884      0
       2705      0
Name: TenYearCHD, Length: 2926, dtype: int64,
138      1
4816      0
1804      0
1389      0
1343      0
...
3414      0
848       0
3914      0
3642      0
2958      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)
```

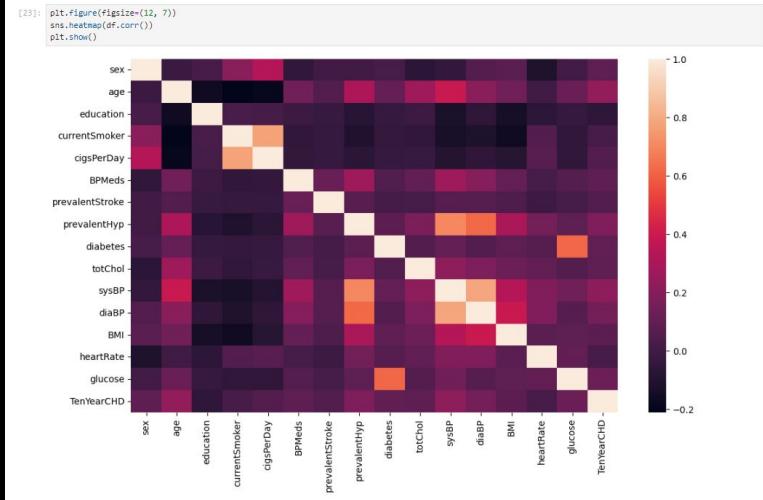
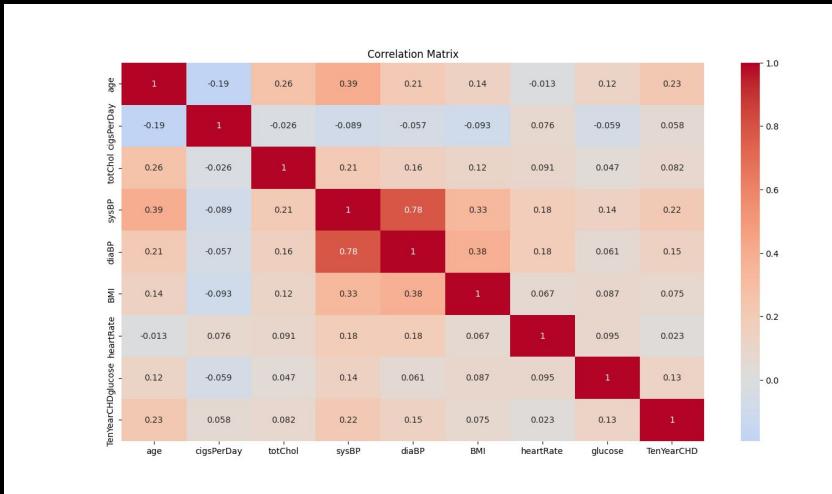
```
[49]: len(y_train)
[49]: 2926
[50]: len(x_train)
[50]: 2926
[51]: len(x_test)
[51]: 732
[52]: len(y_test)
[52]: 732
[53]: x.shape
[53]: (3658, 2)
[54]: y.shape
[54]: (3658,)
```

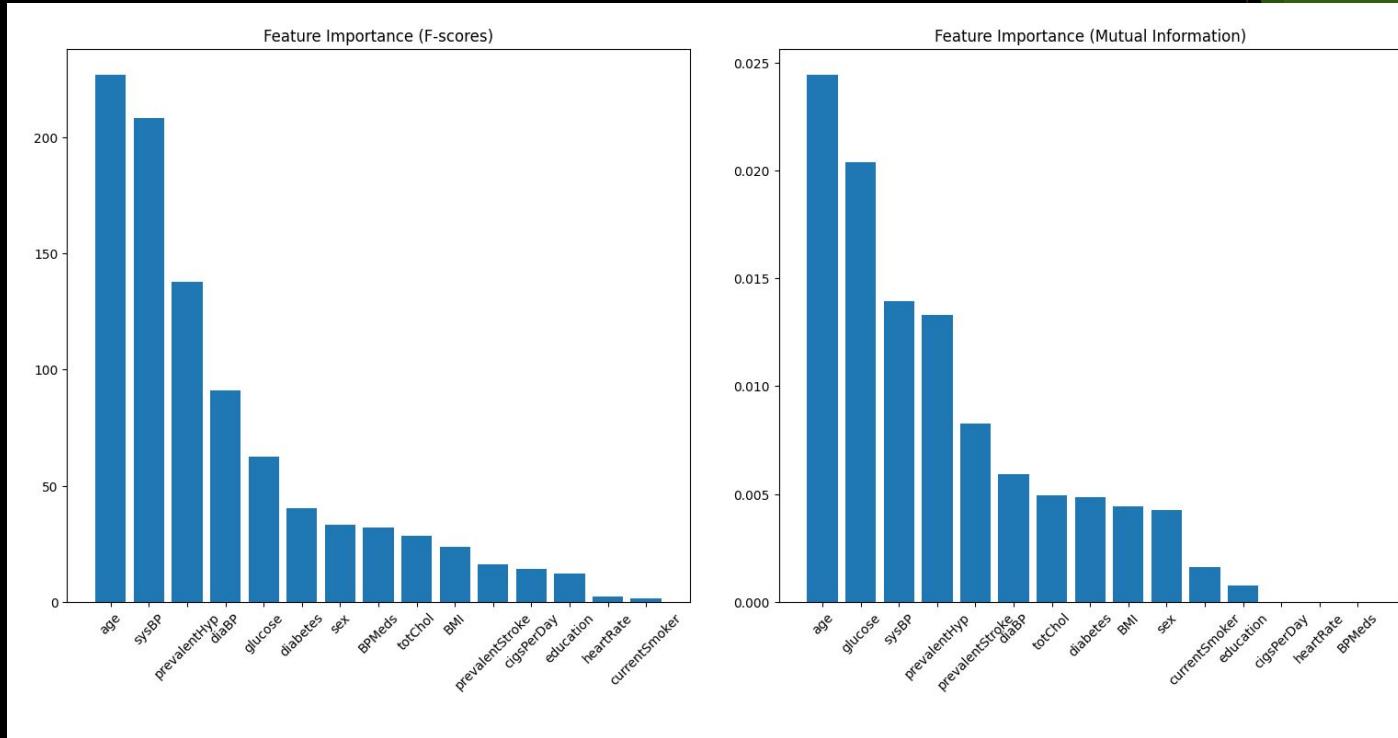
Model Training and Validation

```
121
122 # models training and evaluation functions
123 def train_multiple_models(X_train, X_test, y_train, y_test):
124     # Initialize models
125     models = [
126         'Logistic Regression': LogisticRegression(random_state=42, max_iter=1000),
127         'Decision Tree': DecisionTreeClassifier(random_state=42),
128         'KNeighbors Classifier': KNeighborsClassifier(n_neighbors=5),
129         'Random Forest': RandomForestClassifier(n_estimators=100, random_state=42),
130         'XGBoost': xgb.XGBClassifier(random_state=42, use_label_encoder=False, eval_metric='logloss')
131     ]
132
133     # Store results
134     results = {}
135
136     # Train and evaluate each model
137     for name, model in models.items():
138         # Train model
139         model.fit(X_train, y_train)
140
141         # Make predictions
142         y_pred = model.predict(X_test)
143         y_pred_proba = model.predict_proba(X_test)[:, 1]
144
145         # Calculate metrics
146         results[name] = {
147             'model': model,
148             'predictions': y_pred,
149             'probabilities': y_pred_proba,
150             'accuracy': model.score(X_test, y_test),
151             'roc_auc': roc_auc_score(y_test, y_pred_proba),
152             'classification_report': classification_report(y_test, y_pred),
153             'confusion_matrix': confusion_matrix(y_test, y_pred)
154         }
155
156         # Add feature importance if available
157         if hasattr(model, 'feature_importances_'):
158             results[name]['feature_importance'] = pd.DataFrame({
159                 'feature': X_train.columns,
160                 'importance': model.feature_importances_
161             }).sort_values('importance', ascending=False)
162         elif hasattr(model, 'coef_'):
163             results[name]['feature_importance'] = pd.DataFrame({
164                 'feature': X_train.columns,
165                 'importance': abs(model.coef_[0])
166             }).sort_values('importance', ascending=False)
167
168     return results
```

```
169
170 def compare_models(results):
171     # Comparison DataFrame
172     comparison = pd.DataFrame({
173         name: [
174             'Accuracy': results[name]['accuracy'],
175             'ROC AUC': results[name]['roc_auc']
176         ] for name in results.keys()
177     }).T
178
179     return comparison
180
181 def perform_cross_validation(X, y, models, cv=5):
182     cv_results = {}
183     for name, model in models.items():
184         scores = cross_val_score(model, X, y, cv=cv, scoring='roc_auc')
185         cv_results[name] = {
186             'mean_score': scores.mean(),
187             'std_score': scores.std(),
188             'all_scores': scores
189         }
190
191     return cv_results
```

Correlation Analysis



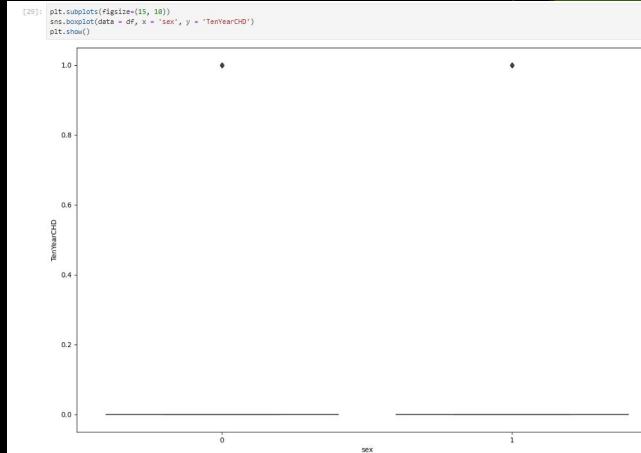


Feature Selection

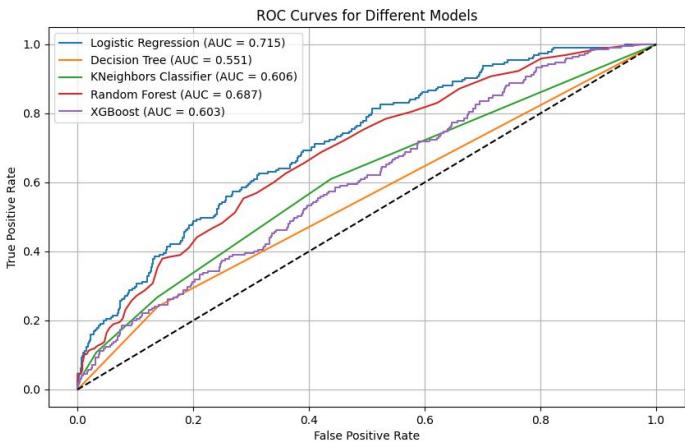
```
63 # 4. Feature Scaling
64 scaler = StandardScaler()
65 df[numeric_features] = scaler.fit_transform(df[numeric_features])
66
67 # 5. Feature Engineering
68 # Create age groups
69 df['age_group'] = pd.qcut(df['age'], q=5, labels=['young_adults', 'adults', 'middle_aged', 'old', 'youth'])
70
71 # Create BMI categories
72 df['bmi_category'] = pd.cut(df['BMI'],
73                             bins=[0, 18.5, 25, 30, float('inf')],
74                             labels=['Underweight', 'Normal', 'Overweight', 'Obese'])
75
76 # Calculate blood pressure category
77 def bp_category(row):
78     if row['sysBP'] < 120 and row['diaBP'] < 80:
79         return 'Normal'
80     elif row['sysBP'] < 130 and row['diaBP'] < 80:
81         return 'Elevated'
82     else:
83         return 'High'
84
85 df['bp_category'] = df.apply(bp_category, axis=1)
86
87 # One-hot encode categorical features
88 categorical_features = ['age_group', 'bmi_category', 'bp_category']
89 df = pd.get_dummies(df, columns=categorical_features, prefix=categorical_features)
90
91 print("\nProcessed Features:")
92 print(df.columns.tolist())
93
94 print("\nMissing Values After Processing:")
95 print(df.isnull().sum().sum())
96
97 return df
98
```

```
[28]: 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	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	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	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	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	0.83	848
macro avg	0.56	0.52	0.52	848
weighted avg	0.77	0.83	0.79	848

Questions

