From ChatAI to AIR·MS:
Leveraging Large Language Models

Andrew Deonarine
Ashwin Sawant

Credits: Eugenia Alleva

Apr 14th, 2026

HPI·MS

Mount Sinai

Hasso Plattner Institute for Digital Health at Mount Sinai

Agenda

1. Recap of training session I: OMOP Tables and Mapping
2. Basics of SQL for OMOP and HANA querying
3. LLMs to the rescue: vibe-querying and vibe-coding
4. ChatAI
5. Ollama and AIR·MS

https://labs.icahn.mssm.edu/minervalab/air-ms-artificial-intelligence-ready-mount-sinai/

Recap of Training Session I

What is AIR·MS?

Artificial Intelligence-Ready Mount Sinai (AIR·MS) is a research platform that is composed of:

1) A (very fast) integrated database including Mount Sinai Data Warehouse (MSDW), Pathology and Radiology metadata; and the included data is growing.

2) A Research Environment that allows interactions with the AIR·MS database from Python or R.

3) An Application Tier to host a growing number of applications – including cohort building tools and annotation apps.

https://labs.icahn.mssm.edu/minervalab/air-ms-artificial-intelligence-ready-mount-sinai/

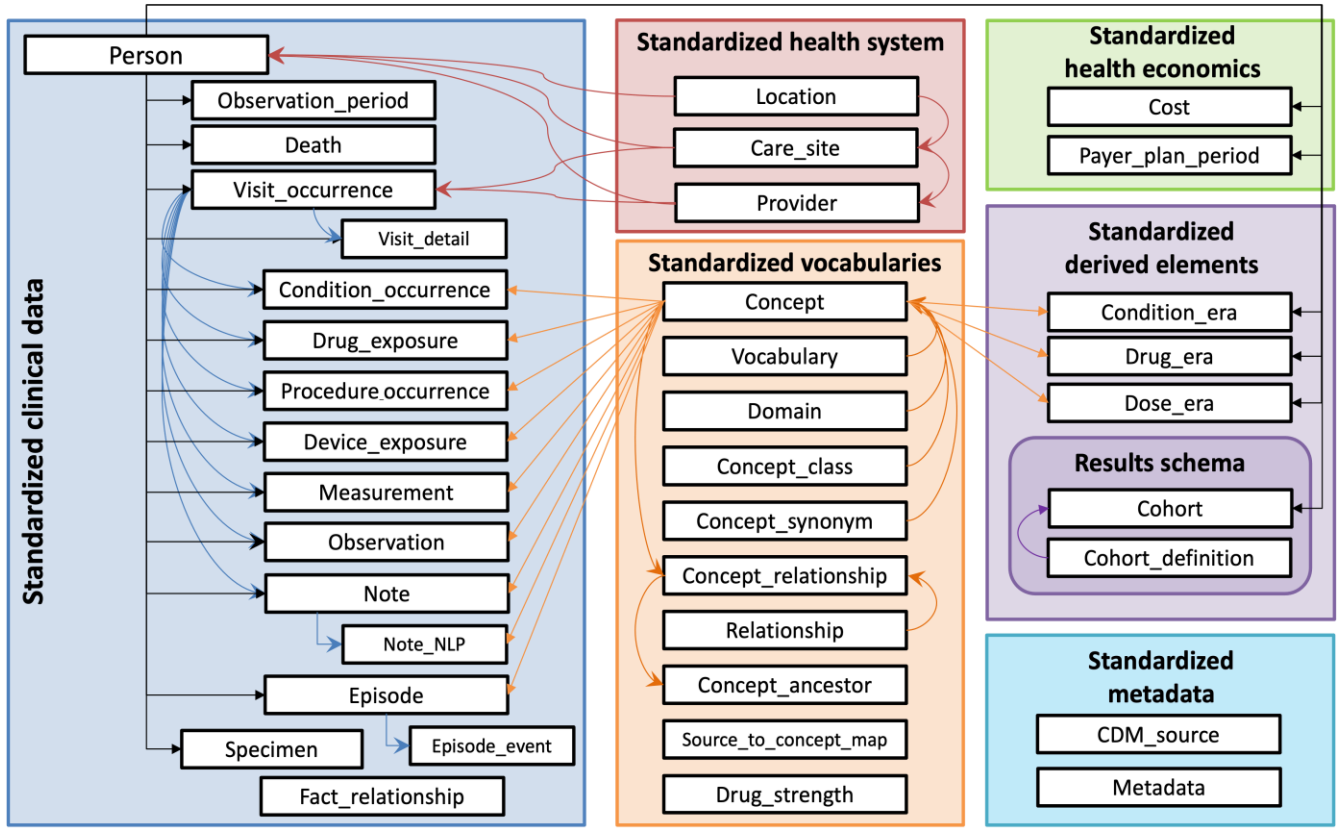
Epic, MSDW, and AIR·MS

Mount Sinai's Epic Electronic Health Record (EHR)

https://labs.icahn.mssm.edu/msdw/

https://labs.icahn.mssm.edu/minervalab/air-ms-artificial-intelligence-ready-mount-sinai/

Main Tables in OMOP Database

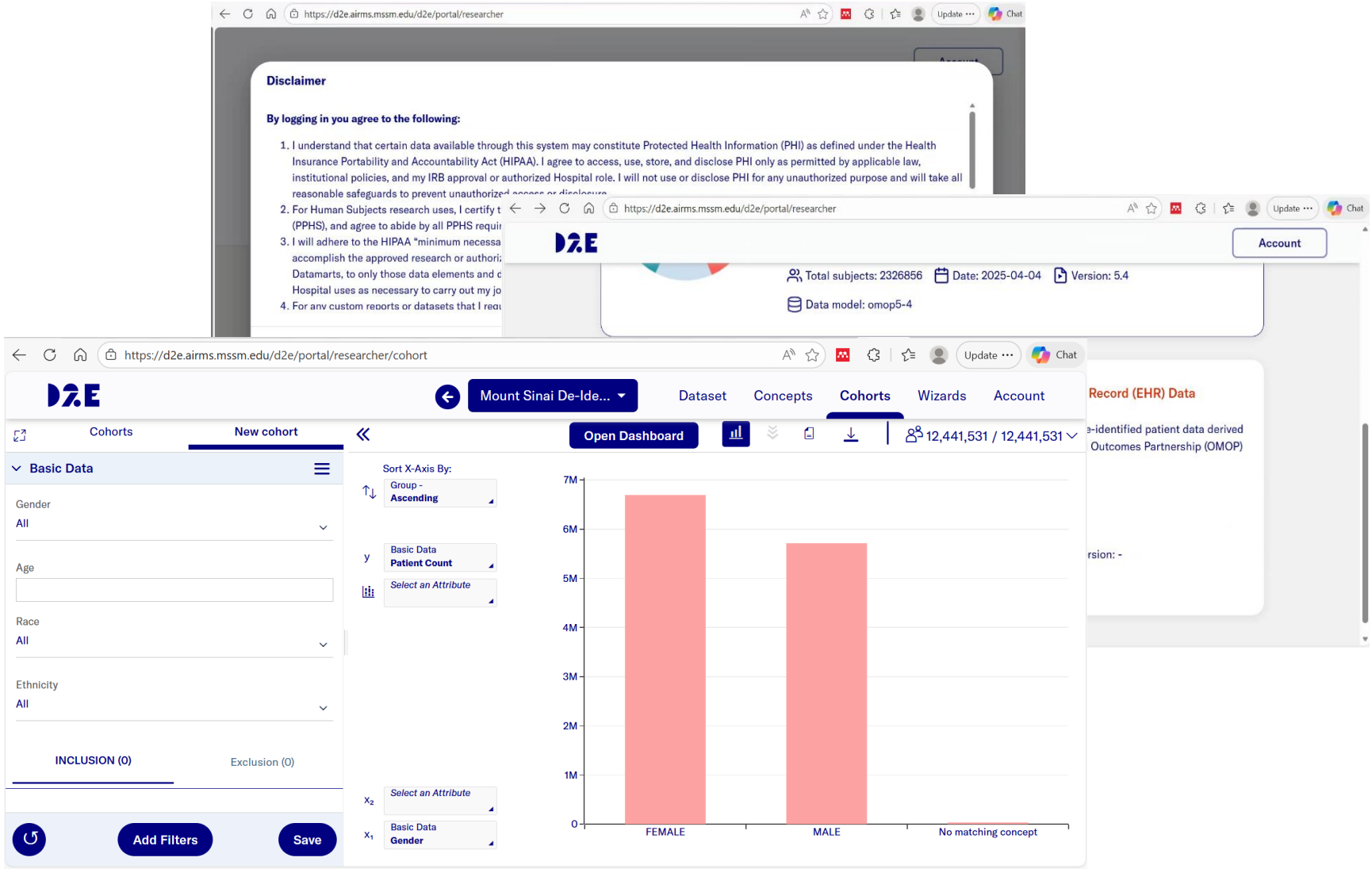


Dataset	OMOP Table	# Records
Patients	person	12,051,324 patients
Encounters	visit_occurrence	111,804,681 encounters
Diagnosis	condition_occurrence	120,323,291 diagnoses
Lab Results	measurements	1,107,913,246 results
Medication	drug_exposure	138,120,557 prescriptions
Procedures	procedure_occurrence	326,372,912 procedures
Observations	observation	145,273,185 social histories
Notes	note	204,925,569 notes
Providers	provider	601,032 specialists
Care Sites	care_site	82,103 beds

As of April, 2025

D2E and Agents

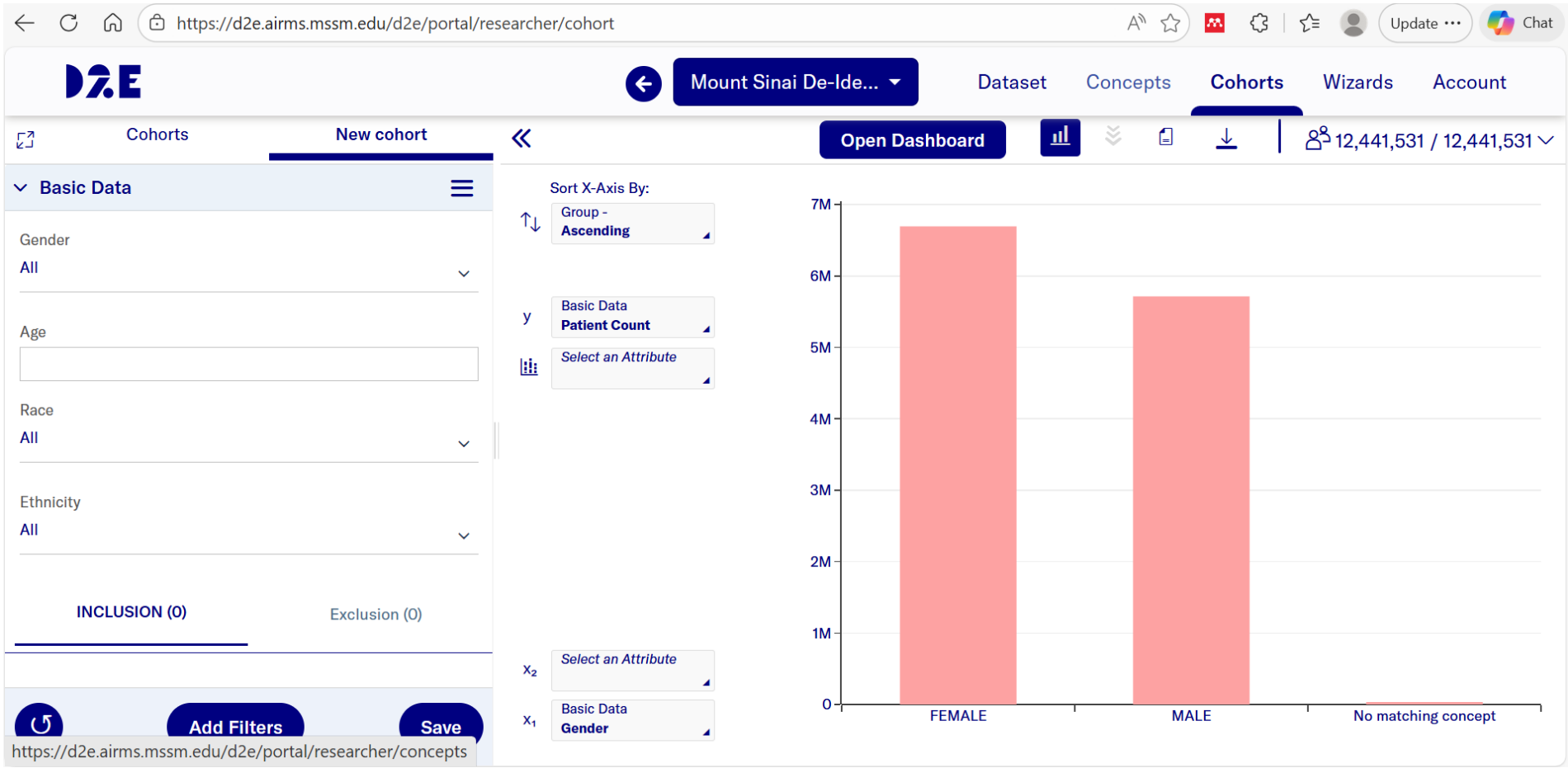
- Data2Evidence (D2E) is a cohort query tool that allows you to build sophisticated cohorts using a graphical user interface
- Currently undergoing testing and finalizing
- We will go over its use in more detail in the next lecture



https://d2e.airims.mssm.edu/

D2E and Agents

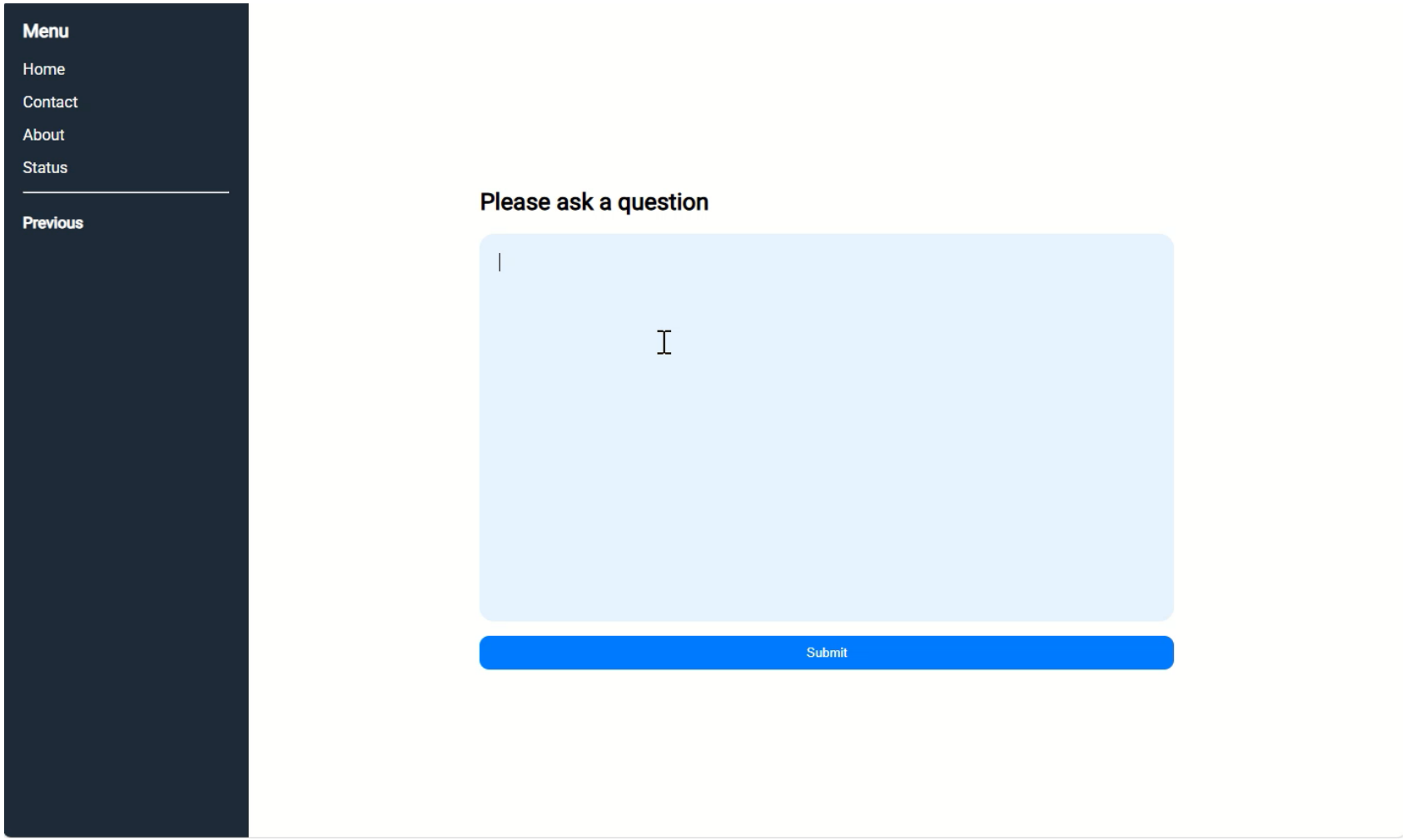
- Data2Evidence (D2E) is a cohort query tool that allows you to build sophisticated cohorts using a graphical user interface
- Currently undergoing testing and finalizing
- We will go over its use in more detail in the next lecture



https://d2e.airms.mssm.edu/

D2E and Agents

- We are building an interactive agent which can create dashboards, operational statistics, and other information on the fly
- If you're interested in testing it please reach out!



Mapping Concepts and Information

From session 1: we have mapping between the coding systems in Epic and the coding systems used by OMOP:

- We can't just use one giant, static mapping table of Epic codes to OMOP codes because some of the coding systems have commercial licenses.
- Don't just perform keyword searches on codes: sometimes the keywords don't appear, consult an expert clinician.
- (ICD = International Classification of Diseases, SNOMED = Systematized Nomenclature of Medicine, NDC = National Drug Code, ATC = Anatomical Therapeutic Classification, LOINC = Logical Observation Identifiers Names and Codes, CPT = Current Procedural Terminology)

Blue = License / Proprietary Coding

Diagnosis

Epic / ICD-10 → SNOMED-CT

Custom

Drug

NDC, ATC → RxNorm

Athena

Labs

LOINC → LOINC

Procedures

CPT → SNOMED-CT

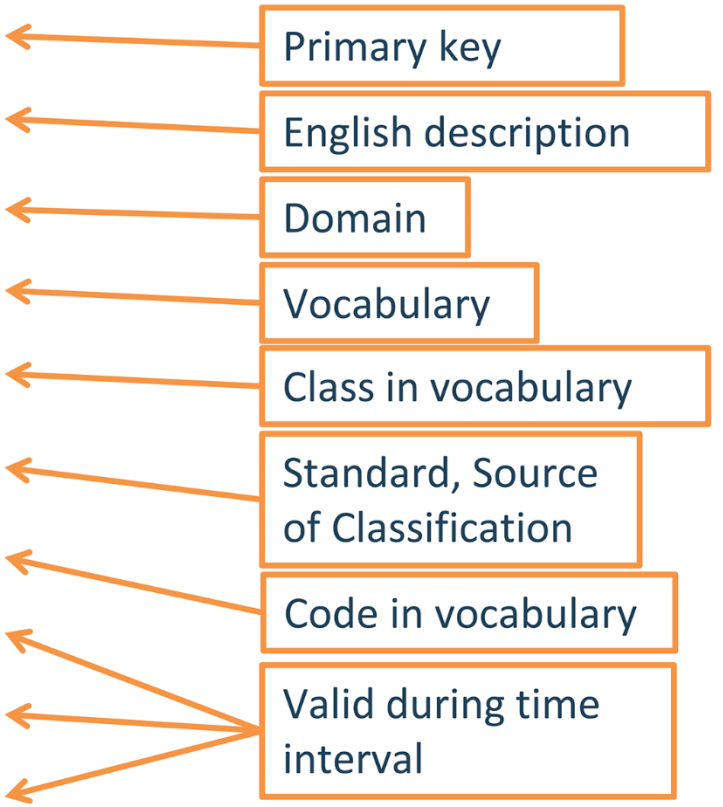
Custom

Using the Data: Querying and Retrieving Data

AIR·MS

- Foundational to OMOP is a “Concept” (stored in the CONCEPT table)
- Concept domains: “Condition,” “Drug,” “Procedure,” “Visit,” “Device,” “Specimen,” etc.

CONCEPT_ID	313217
CONCEPT_NAME	Atrial fibrillation
DOMAIN_ID	Condition
VOCABULARY_ID	SNOMED
CONCEPT_CLASS_ID	Clinical Finding
STANDARD_CONCEPT	S
CONCEPT_CODE	49436004
VALID_START_DATE	01-Jan-1970
VALID_END_DATE	31-Dec-2099
INVALID_REASON	



OMOP Data Example (*synthetic*)

condition_occurrence	Value	Definition
condition_type_concept_id	32827	OMOP's standard record type "EHR encounter record"
xtn_condition_type_source_concept_id	2000000129	MSDW/AIR·MS's source record type "Encounter Diagnosis"
visit_occurrence_id	999888777	Unique identifier for the encounter
person_id	987654321	Unique identifier for the patient
provider_id	123456789	Unique identifier for the provider recording the diagnosis
condition_concept_id	4193704	OMOP's identifier for **SNOMED code 313436004**
condition_source_concept_id	2000602205	MSDW's identifier for **Epic diagnosis code 521601**
condition_start_date	1/1/2022	The date of condition onset or documentation per provider
condition_end_date	NULL	The date on which the condition resolved (if any)

concept_relationship	Value	Definition
concept_1	2000602205	MSDW/AIR·MS's identifier for **Epic diagnosis code 521601**
relationship_id	Maps to non-standard	Text string denoting the type of **mapping relationship** between concept_id_1 & concept_id_2
concept_2	35206882	OMOP's identifier for **ICD-10-CM code E11.9**

OMOP Data Example (*synthetic*)

condition_occurrence	Value
condition_type_concept_id	32827
xtn_condition_type_source_concept_id	2000000129
visit_occurrence_id	999888777
person_id	987654321
provider_id	123456789
condition_concept_id	4193704
condition_source_concept_id	2000602205
condition_start_date	1/1/2022
condition_end_date	NULL

concept_relationship	Value
concept_1	2000602205
relationship_id	Maps to non-standard
concept_2	35206882

concept

concept_id	vocabulary_id	concept_code	concept_name
32827	Type Concept	OMOP4976900	EHR encounter record
2000000129	MSDW Src Rec Type	Encounter Diagnosis	Encounter Diagnosis
4193704	SNOMED	313436004	Type 2 diabetes mellitus without complication
2000602205	EPIC EDG .1	521601	Type 2 diabetes mellitus without complications
2000602205	EPIC EDG .1	521601	Type 2 diabetes mellitus without complications
35206882	ICD10CM	E11.9	Type 2 diabetes mellitus without complications

OMOP Tables

- Have a common naming structure for columns, tables, and can link tables using various IDs

person Column	Description
person_id	Person identifer
gender_concept_id	Gender
race_concept_id	Race
ethnicity_concept_id	Ethnicity
location_id	Location
provider_id	Provider
gender_source_value	(from source)
race_source_value	(from source)
...	

visit_occurrence Column	Description
visit_occurrence_id	ID
person_id	Person ID
visit_start_date	Date
care_site_id	Location
provider_id	Provider
...	

condition_occurrence Column	Description
condition_occurrence_id	ID
person_id	Person ID
condition_concept_id	Concept ID
condition_start_date	Start
condition_end_date	End
stop_reason	Why condition no longer present...
...	

https://www.ohdsi.org/

Patients (`person` Table)

observation		Person	Location		Death	
Patient A	Race	Patient A	Patient A	Home Address A	Patient A	Death date
	Ethnicity					
	Language Preference					
	Sexual Orientation					
Patient B	Race	Patient B	Patient B	Home Address B		
	Marital Status					
	Gender Identity					
Patient C	Ethnicity	Patient C				
	Religious Affiliation					

`person` Column	Description
person_id	Person identifer
gender_concept_id	Gender
race_concept_id	Race
ethnicity_concept_id	Ethnicity
location_id	Location
provider_id	Provider
gender_source_value	(from source)
race_source_value	(from source)
...	

https://www.ohdsi.org/

Encounters (`visit_occurrence` Table)

ED Visit	Hospital Admission	Encounter	visit_occurrence	visit_detail*
	Patient A Inpt Hospitalization 1	Patient A Inpt Hospitalization 1	Patient A Inpt Hospitalization 1	Patient A Care Site Z Care Site Y
Patient B ED Visit 2	Patient B Inpt Hospitalization 2	Patient B Inpt Hospitalization from ED Visit 2	Patient B Inpt Hospitalization from ED Visit 2	Patient B Care Site X Care Site W Care Site V
Patient C ED Visit 3		Patient C ED Visit 3	Patient C ED Visit 3	
Patient A ED Visit 4		Patient A ED Visit 4	Patient A ED Visit 4	
		Patient D Hosp. Outpt Visit 5	Patient D Hosp. Outpt Visit 5	
		Patient E Outpatient Visit 6	Patient E Outpatient Visit 6	
		Patient B Telehealth Visit 7	Patient B Telehealth Visit 7	
		Patient F Mobile Unit Visit 8	Patient F Mobile Unit Visit 8	
		Patient D Patient Contact 9		
		Patient G Documentation 10		

***Under construction**

`visit_occurrence` Column	Description
visit_occurrence_id	ID
person_id	Person ID
visit_start_date	Date
care_site_id	Location
provider_id	Provider
...	

Diagnoses (`condition_occurrence` Table)

visit_occurrence		condition_occurrence			observation	
Patient A	Outpatient Visit 1				Patient A	Past Medical History F6
						Past Medical History K11
Patient B	Outpatient Visit 2	Patient B	Outpatient Visit 2	Encounter Diagnosis D4	Patient B	Past Medical History D4
				Encounter Diagnosis H8		Past Medical History L12
Patient C	Inpt Hospitalization 3	Patient C	Inpt Hospitalization 3	Encounter Diagnosis A1		
				Encounter Diagnosis B2		
				Hospital Problem C3		
				Hospital Problem D4		
				Problem List E5		
Patient D	Inpt Hospitalization 4	Patient D	Inpt Hospitalization 4	Encounter Diagnosis F6		
				Billing Diagnosis G7		
		Patient E		Problem List I9		
				Problem List J10		

`condition_occurrence` Column	Description
condition_occurrence_id	ID
person_id	Person ID
condition_concept_id	Concept ID
condition_start_date	Start
condition_end_date	End
stop_reason	Why condition no longer present...
...	

Lab Results & Vital Signs (`measurement` Table)

Lab Order		Lab Component Result		measurement		Flowsheet	
Patient A	Lab Order 1	Lab Order 1	WBC Count	Patient A	WBC Count		
			RBC Count		RBC Count		
			Hct Result		Hct Result		
			Hgb Result		Hgb Result		
Patient B	Lab Order 2	Lab Order 2	Glucose Result	Patient B	Glucose Result		
			Calcium Result		Calcium Result		
			Sodium Result		Sodium Result		
			BUN Result		BUN Result		
				Patient C	Height	Patient C	Height
					Weight		Weight
					Temp		Temp
				Patient D	Metric 3	Patient D	Metric 3
				Patient E	Metric 4	Patient E	Metric 4

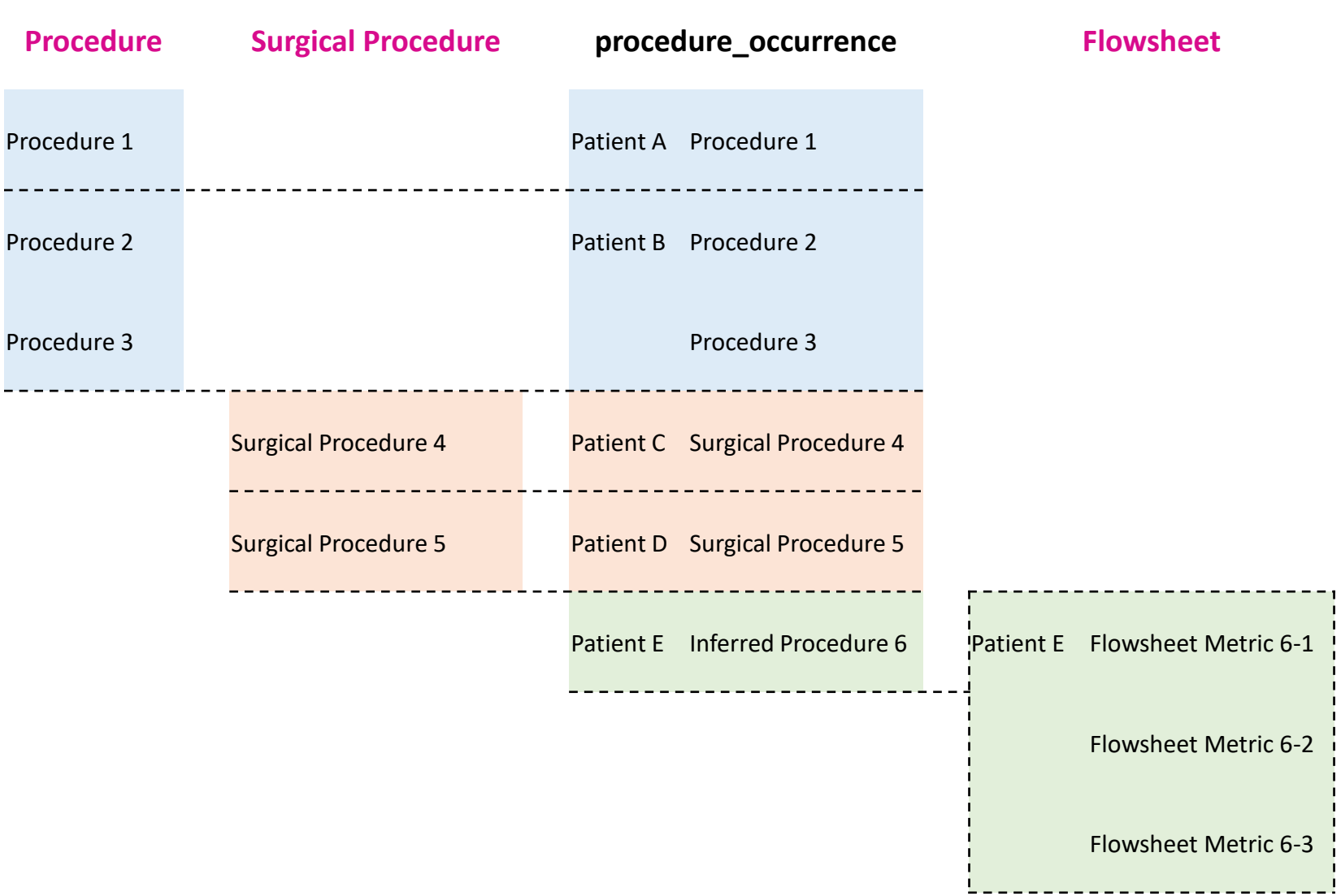
`measurement` Column	Description
measurement_id	ID
person_id	Person ID
measurement_concept_id	Concept ID
measurement_date	Date
value_as_number	Value
value_as_concept_id	Value mapped to vocabulary (ex. High, low, absent, etc)
unit_concept_id	Units
Provider_id	Provider
…	

Medications (drug_exposure Table)

Medication Order		Medication Dispense		Medication Administration		Immunization Event		drug_exposure		
						Patient A	Vaccine Z	Patient A	Immuniz Z1	Vaccine Z
Patient B	Inpt Med Order 2					Inpt Med Order 2	Vaccine Y	Patient B	Inpt Med Order 2	Vaccine Y
Patient C	Inpt Med Order 3			Inpt Med Order 3	Med Admin 3-1			Patient C	Med Admin 3-1	Drug X
					Med Admin 3-2				Med Admin 3-2	Drug X
Patient D	Inpt Med Order 4	Inpt Med Order 4	Dispense 4-1	Inpt Med Order 4	Med Admin 4-1			Patient D	Med Admin 4-1	Drug W
			Dispense 4-2		Med Admin 4-2				Med Admin 4-2	Drug W
Patient E	Inpt Med Order 5	Inpt Med Order 5	Dispense 5-1					Patient E	Dispense 5-1	Drug V
Patient F	Outpt Med Order 6	Outpt Med Order 6	Dispense 6-1					Patient F	Dispense 6-1	Drug U
Patient G	Outpt Med Order 7	Outpt Med Order 7	Dispense 7-1					Patient G	Dispense 7-1	Drug T
			Dispense 7-2						Dispense 7-1	Drug T
Patient H	Outpt Med Order 8							Patient H	Outpt Med Order 8	Drug S

drug_exposure Column	Description
drug_exposure_id	ID
person_id	Person ID
drug_concept_id	Concept ID
drug_exposure_start_date	Date
drug_type_concept_id	Value mapped to vocabulary (ex. High, low, absent, etc)
stop_reason	Units
refills	Provider
…	

Procedures (procedure_occurrence Table)



procedure_occurrence Column	Description
procedure_occurrence_id	ID
person_id	Person ID
procedure_concept_id	Concept ID
procedure_date	Date
procedure_type_concept_id	Procedure mapped to vocabulary (ex. High, low, absent, etc)
quantity	Amount
provider_id	Provider ID
…	

Observations (`observation` Table)

Patient Demographics		Surgical History		Social History		Family History		Patient Allergy	
Patient A	Race								
	Ethnicity								
	Language Preference								
	Sexual Orientation								
Patient B	Race								
	Marital Status								
	Gender Identity								
Patient C	Ethnicity								
	Religious Affiliation								
		Patient A	Procedure Z						
			Procedure Y						
		Patient B	Procedure X						
			Procedure W						
				Patient C	Soc Hx Item 1-1				
					Soc Hx Item 1-2				
					Soc Hx Item 1-3				
						Patient D	Fam Hx 2-1		
							Fam Hx 2-2		
								Patient E	Allergy 3-1

observation	
Patient A	Race
	Ethnicity
	Language Preference
	Sexual Orientation
Patient B	Race
	Marital Status
	Gender Identity
Patient C	Ethnicity
	Religious Affiliation
Patient A	Procedure Z
	Procedure Y
Patient B	Procedure X
	Procedure W
Patient C	Soc Hx Item 1-1
	Soc Hx Item 1-2
	Soc Hx Item 1-3
Patient D	Fam Hx 2-1
	Fam Hx 2-2
Patient E	Allergy 3-1

`observation` Column	Description
observation_id	ID
person_id	Person ID
observation_concept_id	Concept ID
observation_date	Date
…	

Notes (note Table)

Clinical Note	
Patient A	Clinic Note
Patient B	Progress Note
	A&P Note
	Procedure Report
	Periop Record
	Consult Note
	Discharge Summary

Lab Result Text	
Patient B	Pathology Report
	Lab Result Note

Imaging Study Text*	
Patient B	Imaging Report Narrative
	Imaging Report Impression
Patient C	Imaging Report Narrative
	Imaging Report Impression
	Imaging Report Addenda

*Under construction

note	
Patient A	Clinic Note
Patient B	Progress Note
	A&P Note
	Procedure Report
	Periop Record
	Consult Note
	Discharge Summary
	Pathology Report
	Lab Result Note
	Imaging Report Narrative
	Imaging Report Impression
Patient C	Imaging Report Narrative
	Imaging Report Impression
	Imaging Report Addenda

note Column	Description
note_id	ID
person_id	Person ID
note_type_concept_id	Concept ID
note_date	Date
note_text	Text
...	

Providers (`provider` Table)

provider_attribute_xtn		provider	care_site		location	
Physician A	Specialty 1 Specialty 2	Physician A	Physician A	Office A	Office A	Address A
Physician B	Specialty 3 Specialty 4	Physician B	Physician B	Office B	Office B	Address B
Physician C	Specialty 2	Physician C				
Clinician D	Specialty 5 Specialty 6	Clinician D				
Clinician E	Specialty 7	Clinician E				

`provider` Column	Description
provider_id	ID
provider_name	Person ID
npi	National Provider Identifier
specialty_concept_id	Specialty ID
care_site_id	Location ID
...	

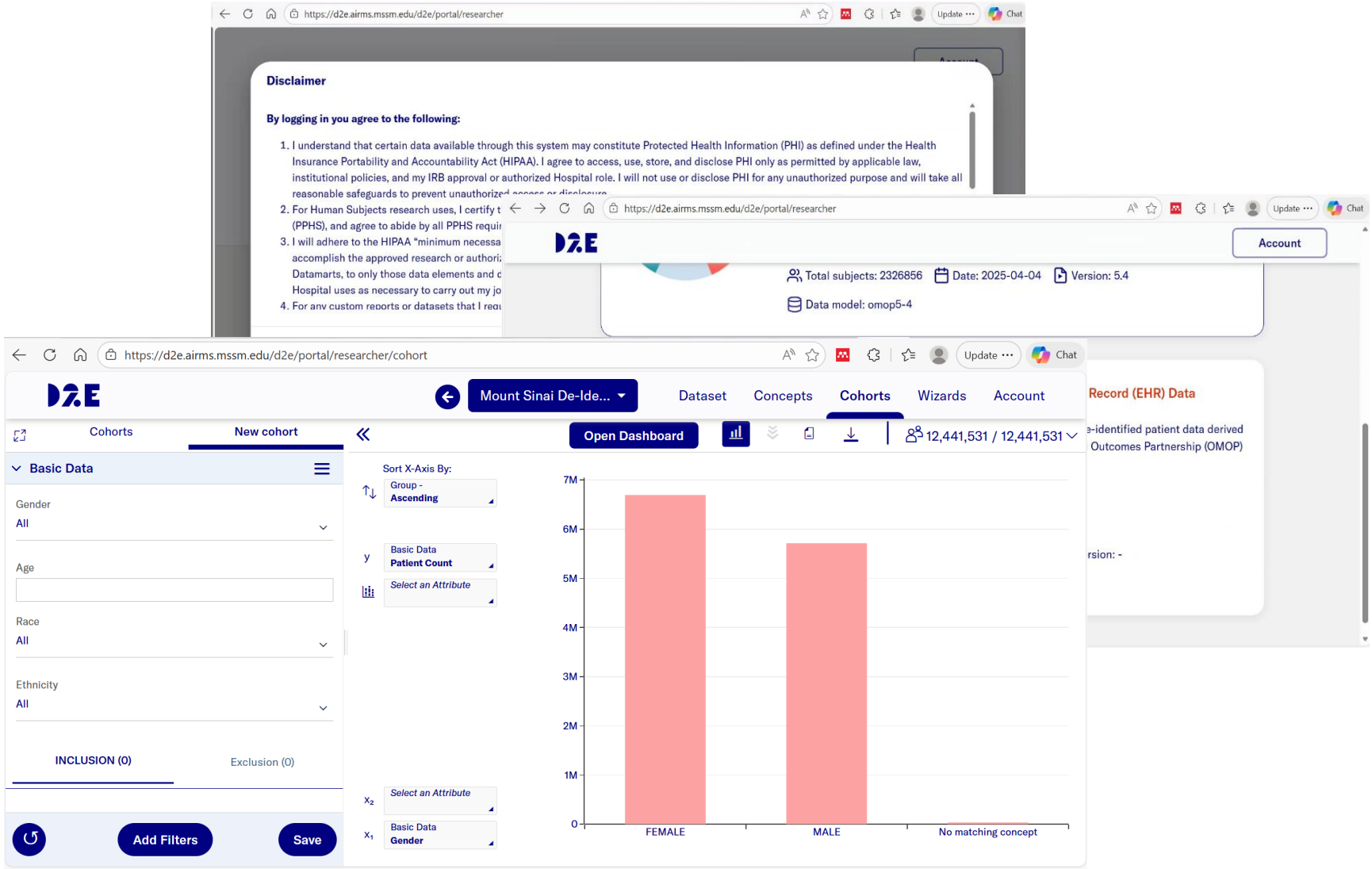
Care Sites (`care_site` Table)

location		care_site	Definition
Service Area A	Address A	Service Area A	Legal entity
Parent Location B	Address B	Parent Location B	Collection of “locations” grouped together for financial accounting & billing purposes
Location C	Address C	Location C	Actual organizational unit, which often (but not always) represents a physical building
Department D	Address D	Department D	Org unit where patient care is actually delivered, according to Epic’s workflows
		Care Area E	Subdivision of a department, usually a grouping of patient rooms
		Room F	Where an inpatient is assigned, used for inpatient census and ADT transactions
		Bed G	Used when a room has more than one bed (i.e., is shared)
Place of Service H	Address H	Place of Service H	Legal entity or subdivision used for billing purposes

Basics of SQL

D2E and Agents

- Data2Evidence (D2E) is a cohort query tool that allows you to build sophisticated cohorts using a graphical user interface
- Currently undergoing testing and finalizing
- We will go over its use in more detail in the next lecture



https://d2e.airims.mssm.edu/

Relational Databases and SQL

What is a Relational Database?

- A collection of **tables** with structured data
- Rows = records (tuples)
- Columns = attributes (fields)
- Tables linked by **relationships** (keys)

Core Concepts

- **Primary Key:** uniquely identifies each row
- **Foreign Key:** links rows across tables
- **Schema:** blueprint of the database structure
- **Constraints:** rules for data integrity

What is SQL?

- Structured Query Language
- Standard for working with relational databases
- Used to:
  - Create and modify tables
  - Insert, update, delete data
  - **Query and filter data**

Schemas

A schema is like a namespace or folder for database objects. In AIR·MS, different schemas store different data modalities. This allows users to get access to the modalities they are allowed to see, while keeping all modalities linkable with each other across the database.

To access a table, you must either:

- Fully quality the table name as schema_name.table_name
- Be inside its schema.

You can check your current schema via:

```
SELECT CURRENT_SCHEMA FROM DUMMY
```

In AIR·MS, your data might be stored in schemas like CDMPHI, CDMDEID, MSMDEID ect.

Basic SQL elements

SQL has two main building blocks:

1.Clauses (keywords)

1. Control the structure of the query.
2. Examples: `SELECT, FROM, WHERE, ORDER BY.`

2.Functions (built-in operations)

1. Perform calculations on values.
2. Examples: `AVG(), COUNT(), MAX().`

Clauses tell the database *what to do*, functions tell it *how to calculate*.

SELECT (…) FROM (…) WHERE (…) CLAUSE

SELECT

- Chooses which columns (or expressions) to display.
- Always the first clause in a query.

FROM

- Tells SQL which **table** (or view) the data comes from.

```
SELECT <column name> as <new column name> FROM <schema>.<table>
```

```
SELECT person_id as id, year_of_birth FROM cdmdeid.person
```

```
SELECT * FROM cdmdeid.condition_occurrence
```

WHERE

- Filters rows **before** grouping or aggregation.
- Only rows meeting the condition are included.

```
SELECT * FROM cdmdeid.person WHERE year_of_birth < 1980
```

GROUP BY CLAUSE

GROUP BY

- Groups rows together for aggregation.
- Without GROUP BY, functions like COUNT or AVG summarize *all rows*; GROUP BY, they summarize *per group*.

```
SELECT gender_concept_id, COUNT(*) AS n_patients
FROM cdmdeid.person
GROUP BY gender_concept_id
```

HAVING

- Filters groups (not rows).
- Used after `GROUP BY` + aggregation.

```
SELECT year_of_birth, COUNT(*) AS n_patients
FROM cdmdeid.person
GROUP BY year_of_birth
HAVING COUNT(*) > 1000
```

ORDER and LIMIT CLAUSE

ORDER BY

- Sorts the results by one or more columns.
- Default = ascending (ASC), descending = DESC.

```
SELECT person_id, year_of_birth
FROM cdmdeid.person
ORDER BY year_of_birth DESC
```

LIMIT / TOP

- Restrict number of results.
- In **HANA**, you can use either:

```
SELECT person_id, year_of_birth FROM cdmdeid.person ORDER BY year_of_birth LIMIT 10
```

```
SELECT TOP 10 person_id, year_of_birth FROM cdmdeid.person ORDER BY year_of_birth
```

Common SQL Functions

COUNT()

- Counts rows.
- COUNT(*) counts all rows.
- COUNT(column) counts only non-null values.

```
SELECT COUNT(*) AS n_patients FROM cdmdeid.person
```

AVG()

- Computes the average of a numeric column.
- Ignores NULL values.

```
SELECT AVG(value_as_number) AS avg_hemoglobin
FROM cdmdeid.measurement
WHERE measurement_concept_id = 3000963
```

To ensure we have only numeric values we can **CAST** the datatype using

```
CAST(VALUE_AS_NUMBER AS DOUBLE)
```

Common SQL Functions

SUM()

- Adds values across rows.

```
SELECT SUM(value_as_number) AS total_lab_values
FROM cdmdeid.measurement
WHERE measurement_concept_id = 3027018
```

MIN() / MAX()

- Get smallest / largest value.

```
SELECT MIN(year_of_birth) AS oldest, MAX(year_of_birth) AS youngest
FROM cdmdeid.person
```

Common SQL Functions

String functions

- `CONCAT(a, b)` → join two strings.
- `UPPER(string)` → convert to uppercase.
- `SUBSTRING(string, start, length)` → extract part of a string.

```
SELECT CONCAT('Patient_', person_id) AS patient_label
FROM cdmdeid.person
LIMIT 5
```

Date functions

- `CURRENT_DATE` → today's date.
- `ADD_DAYS(date, n)` → shift date forward/backward.
- `EXTRACT(YEAR|MONTH|DAY FROM date)` → get part of a date.

```
SELECT person_id, ADD_DAYS(visit_start_date, 30) AS followup_date
FROM cdmdeid.visit_occurrence;
```

Joins

What is a Join?

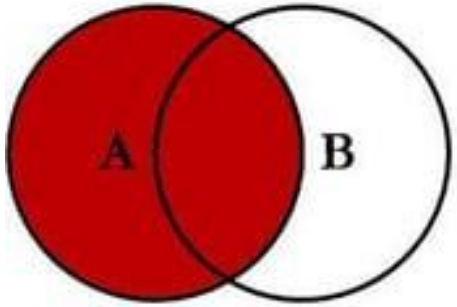
- A **join** combines rows from two (or more) tables based on a related column, usually an **ID** (like person_id or visit__id).
- In OMOP, this is common: e.g., joining **visits** with **conditions**, or **drugs** with **patients**.

Types of Joins (in HANA)

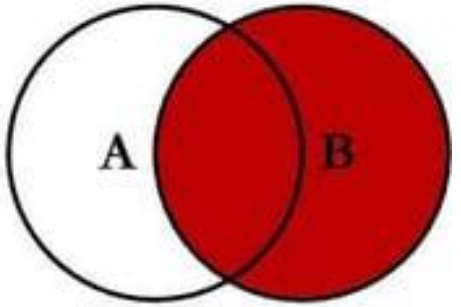
- **`INNER JOIN`** → only rows that match in both tables.
- **`LEFT JOIN`** → all rows from the left table, plus matches from the right (if any).
- **`RIGHT JOIN`** → all rows from the right table, plus matches from the left.
- **`FULL OUTER JOIN`** → all rows from both tables, matched where possible.

In practice, **`INNER JOIN`** and **`LEFT JOIN`** are the most common in OMOP queries.

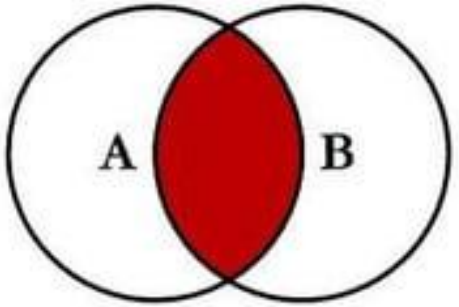
SQL JOINS



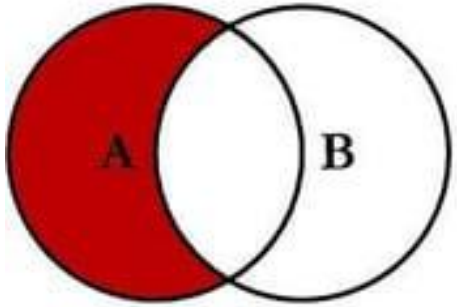

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```



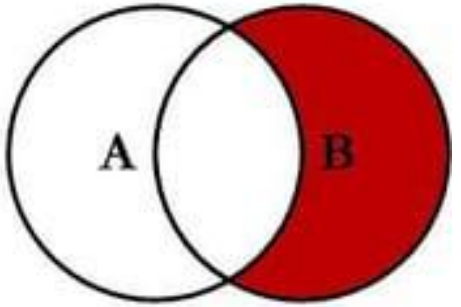

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```



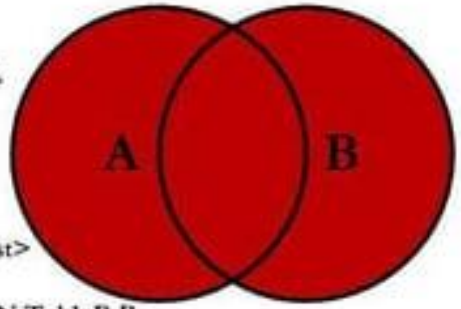

```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```



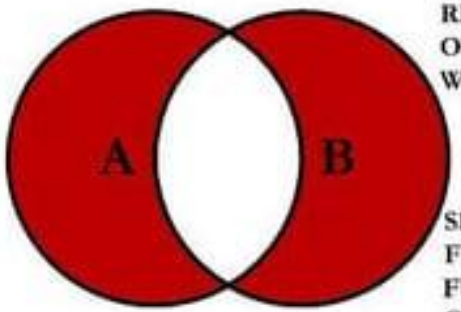

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```




```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```




```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```




```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```

Example Joins

Example 1: Diagnoses with Visit Dates

```
SELECT TOP 10 c.person_id, c.condition_concept_id, v.visit_start_date
FROM cdmdeid.condition_occurrence c
INNER JOIN cdmdeid.visit_occurrence v
ON c.visit_occurrence_id = v.visit_occurrence_id
WHERE v.visit_start_date >= '2020-01-01'
```

Example Joins

Example 2: All Visits, Even Without a Condition (LEFT JOIN)

```
SELECT TOP 10 v.person_id, v.visit_start_date, c.condition_concept_id
FROM cdmdeid.visit_occurrence v
LEFT JOIN cdmdeid.condition_occurrence c
ON v.visit_occurrence_id = c.visit_occurrence_id
```

Example Joins

Example 3: Patients with Metformin and Diabetes

```
SELECT TOP 10 DISTINCT d.person_id
FROM cdmdeid.drug_exposure d
JOIN cdmdeid.condition_occurrence c
ON d.person_id = c.person_id
WHERE d.drug_concept_id = 40164897
AND c.condition_concept_id = 201826
```

Derived Columns

SQL lets you **create new columns on the fly** by applying functions or expressions in the SELECT clause. These are sometimes called **computed/derived columns** or **virtual columns**.

General Syntax

```
SELECT column1, column2, <expression> AS new_column_name FROM table_name
```

expression can be:

- A **function** (`AVG(), CONCAT(), DAYS_BETWEEN()`)
- An **arithmetic operation** (`+, -, *, /`)
- A **case expression** (`CASE WHEN ... THEN ... END`).

Derived Columns Examples

Example 1: Date Differences

```
SELECT person_id,
       visit_start_date,
       visit_end_date,
       DAYS_BETWEEN(visit_start_date, visit_end_date) AS visit_length_days
FROM cdmdeid.visit_occurrence
LIMIT 5
```

Derived Columns Examples

Example 2. Arithmetic on Numeric Columns with 3027018 (glucose)

```
SELECT measurement_id,
       value_as_number,
       unit_source_value,
       value_as_number * 0.18 AS value_in_mmolL
FROM cdmdeid.measurement
WHERE measurement_concept_id = 3000483
AND unit_source_value='mg/dL'
LIMIT 100
```

Derived Columns Examples

Example 3. Conditional New Column

```
SELECT person_id,
       year_of_birth,
       CASE
         WHEN year_of_birth < 1970 THEN 'Older'
         ELSE 'Younger'
       END AS age_group
FROM cdmdeid.person
LIMIT 10
```

Derived Columns Examples

Example 4. Combining Multiple Columns from Multiple Tables

```
SELECT m.person_id,
       m.value_as_number,
       v.visit_start_date,
       DAYS_BETWEEN(p.birth_datetime, v.visit_start_date) AS age_in_days
FROM cdmdeid.measurement m
JOIN cdmdeid.person p ON m.person_id = p.person_id
JOIN cdmdeid.visit_occurrence v ON m.visit_occurrence_id = v.visit_occurrence_id
LIMIT 5
```

LLMs to the rescue: Vibe-querying and vibe-coding

What is vibe-coding?

- Leveraging **Large Language Models (LLMs)** as copilots for coding
- Not strict instruction-following — more like **collaborating with a helpful peer**
- You describe the “vibe” of what you want (intent, not syntax), LLM drafts code

Benefits:

- Lowers the barrier for beginners
- Speeds up iteration for experts
- Helps brainstorm alternative queries or query structures
- Great for learning: compare **LLM suggestion vs. official SQL docs**

Pitfalls:

- LLMs may **hallucinate columns/tables** that don’t exist
- Queries might be correct syntactically but **wrong semantically**
- Should always **validate outputs** on actual schema and data
- Works best as a **copilot, not autopilot**

ChatAI

About Large Language Models

- Large language models (LLMs) represent the culmination of several innovations in artificial intelligence (AI)
- Central to LLMs are transformers – a simple neural network architecture that uses an attention mechanism
- Attention is a mechanism that allows a model to weigh the importance of different parts of an input sequence when processing it
- This innovation allowed for breakthroughs in processing text and performing natural language processing, giving rise to recent LLMs like ChatGPT, Claude, etc.

Attention Is All You Need

Ashish Vaswani[*] Google Brain avaswani@google.com
Noam Shazeer[*] Google Brain noam@google.com
Niki Parmar[*] Google Research nikip@google.com
Jakob Uszkoreit[*] Google Research usz@google.com
Llion Jones[*] Google Research llion@google.com
Aidan N. Gomez[*] [†] University of Toronto aidan@cs.toronto.edu
Łukasz Kaiser[*] Google Brain lukaszkaiser@google.com
Illia Polosukhin[*] [‡] illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

[*]Equal contribution. Listing order is random. Jakob proposed replacing RNNs with self-attention and started the effort to evaluate this idea. Ashish, with Illia, designed and implemented the first Transformer models and has been crucially involved in every aspect of this work. Noam proposed scaled dot-product attention, multi-head attention and the parameter-free position representation and became the other person involved in nearly every detail. Niki designed, implemented, tuned and evaluated countless model variants in our original codebase and tensor2tensor. Llion also experimented with novel model variants, was responsible for our initial codebase, and efficient inference and visualizations. Lukasz and Aidan spent countless long days designing various parts of and implementing tensor2tensor, replacing our earlier codebase, greatly improving results and massively accelerating

About Large Language Models

- There are several LLMs on the market, that perform different tasks, and have different parameter sizes
- **Parameters** = *weightings between pairs and groups of words (simplified definition)* - the more parameters, the more complex (and sometimes more accurate) the LLM

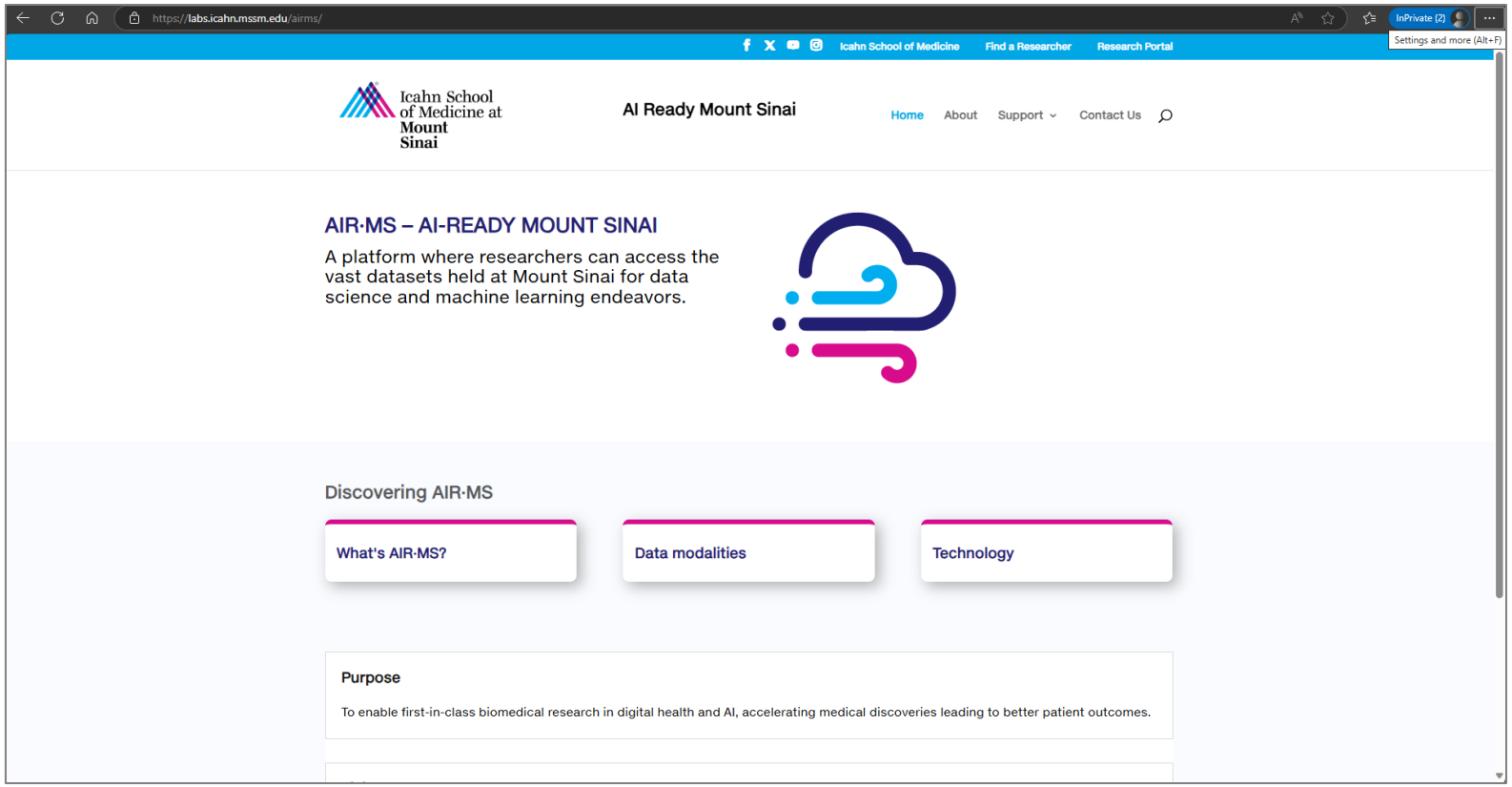
- **ChatGPT LLM** – uses hundreds of billions of parameters; can be used for general tasks
- **Claude LLM** – optimized for coding; about 100 billion parameters
- **Gemma LLM** – have medically optimized versions; a few billion parameters

- Generally, less than ~5 billion parameters can be called a “small language model” (SLM)

- Language models are executed on optimized hardware, such as GPUs (graphics processing units) mostly made by Nvidia (ex. Nvidia A2, T4, V100, etc).

The AIR•MS Chatbot

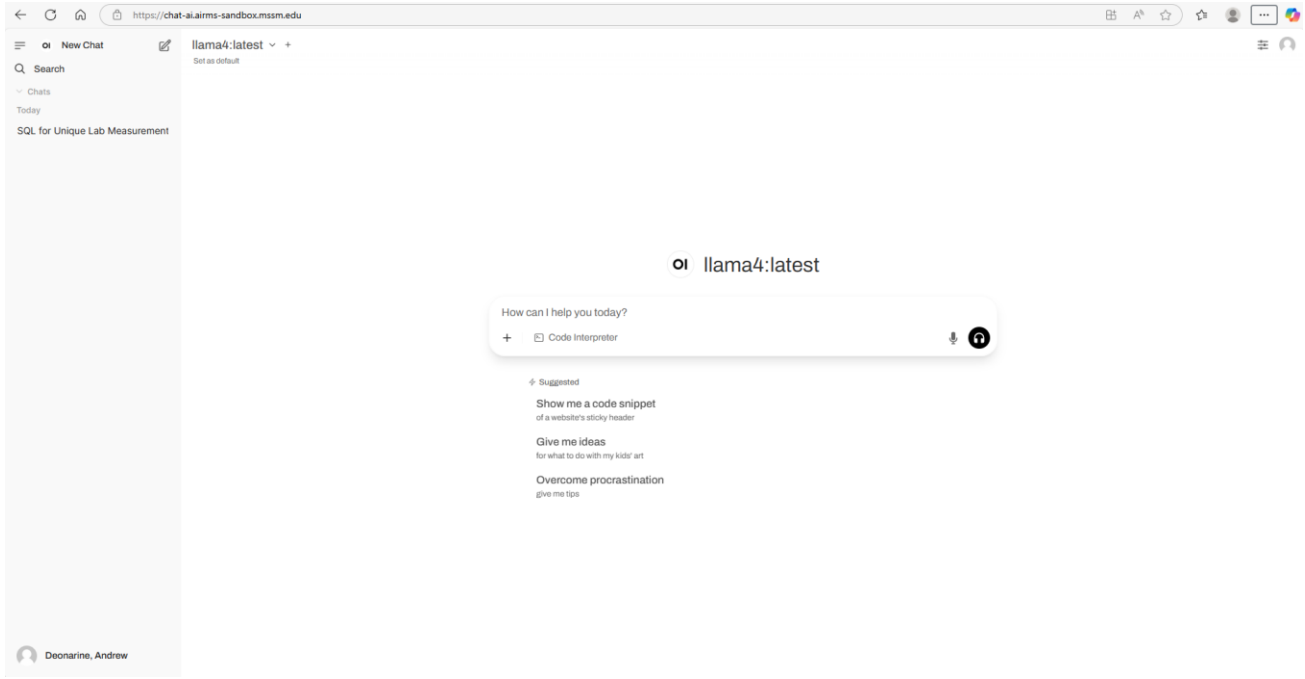
- The chatbot is part of AIR•MS (AI Ready Mount Sinai) which is a centralized platform that enables AI throughout the Mount Sinai System
- Eventually the chatbot will help users answer questions about Mount Sinai clinical data
- An online presence for AIR•MS is currently under construction and will be frequently updated
- Built as part of a collaboration between **Data4Life** (D4L), **Scientific Computing and Data** (SCD), **AI and Human Health** (AIHH), and the **Hasso Plattner Institute** (HPI)



https://labs.icahn.mssm.edu/airms

The AIR•MS Chatbot

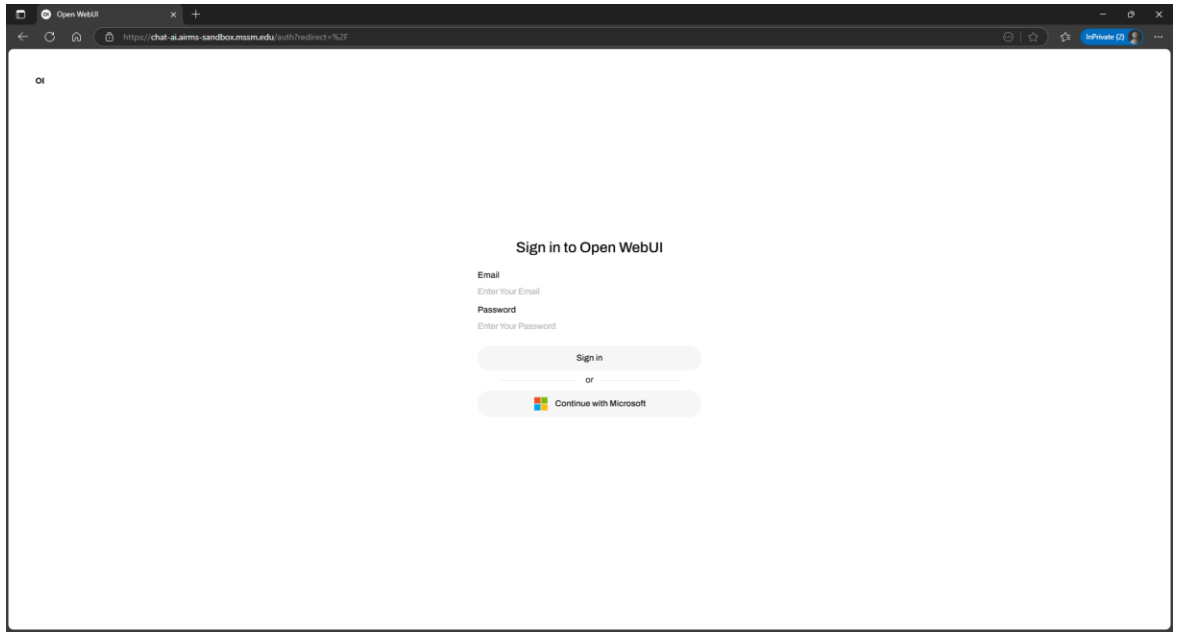
- Using the advanced computational resources on Minerva (consisting of several hundred Nvidia GPUs and large storage capacity), a chatbot has been implemented internally to Mount Sinai
- Runs on Minerva hardware, and several LLMs are integrated into the user interface
- Users will need to log in using their Mount Sinai account and then complete an attestation before starting



https://chat-ai.airms.mssm.edu

The AIR•MS Chatbot

- Currently the AIR•MS chatbot is only meant for internal use, and can only be used on site at Mount Sinai, or through the VPN (virtual private network)
- Users should refrain from entering protected health information
- Once signed in, user prompts and input will be saved into a session, so users can pick up where they left off



https://chat-ai.airms.mssm.edu

Sample Prompts & Use Cases

- LLMs use “prompts” as input

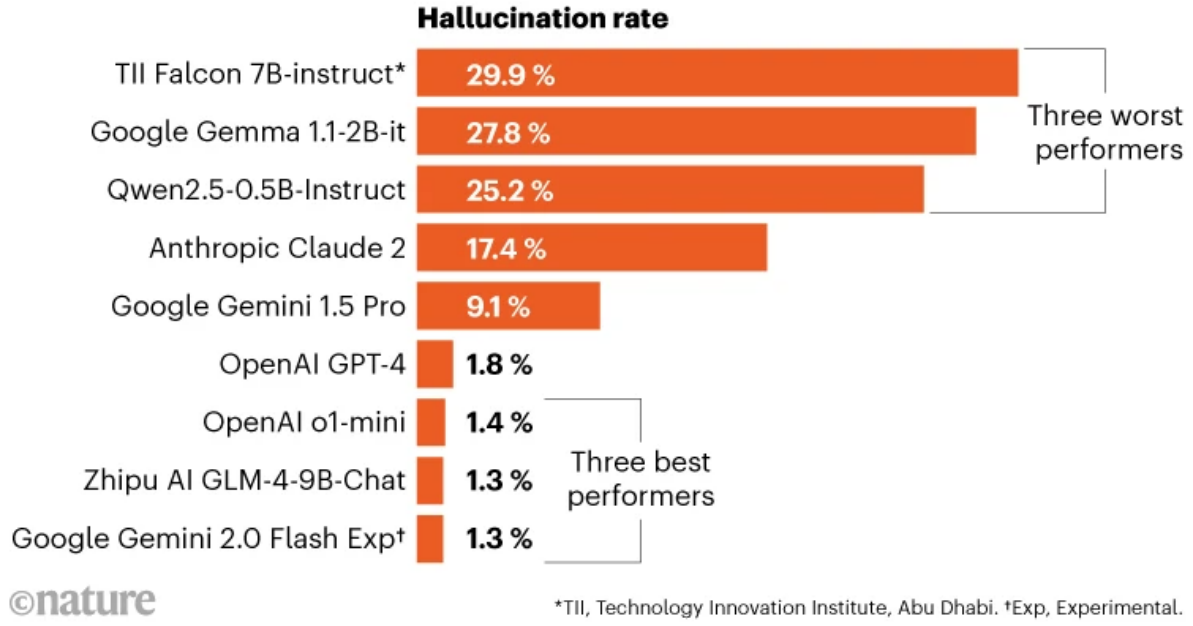
- Prompts are carefully developed statements that contain:
  - Context (ex. “I’m a physician working at a major healthcare organization in New England”)
  - Background information (ex. “I am using a specific database that contains a table on demographic information, and columns containing Name, Age, Sex, Income…”)
  - Specific questions or requests (ex. “I would like you to write an SQL script to retrieve information from the demographics table…”)

- Do I need to be polite to an LLM? Some preliminary evidence shows it might actually help with the quality of the response!

Hallucinations

- Sometimes LLMs will make things up – they will create false references, false facts, etc.
- It is important to verify results from any LLM
- AI hallucinations can't be stopped — but these techniques can limit their damage (https://www.nature.com/articles/d41586-025-00068-5)
- Several computational methods are being used to reduce hallucinations

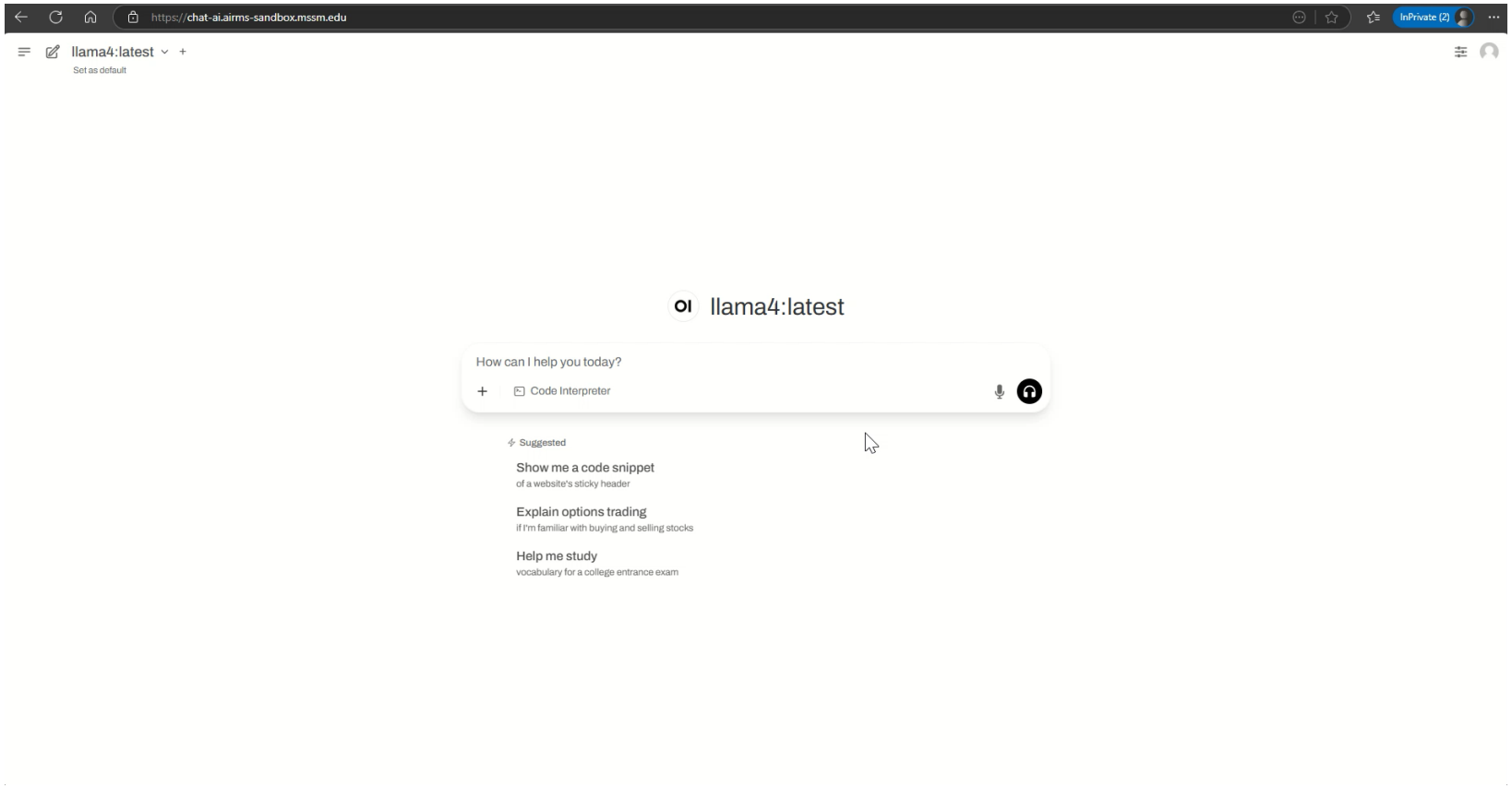
Vectara's Hallucination Leaderboard grades large language models (LLMs) on the specific task of summarizing a provided document, by tracking how many details they make up. Those that performed best (having the lowest hallucination rates) and worst (highest rates) are shown, along with the rates of selected well-known LLMs.



Source: Vectara (https://go.nature.com/4GPQRTT; accessed 11 January 2025)

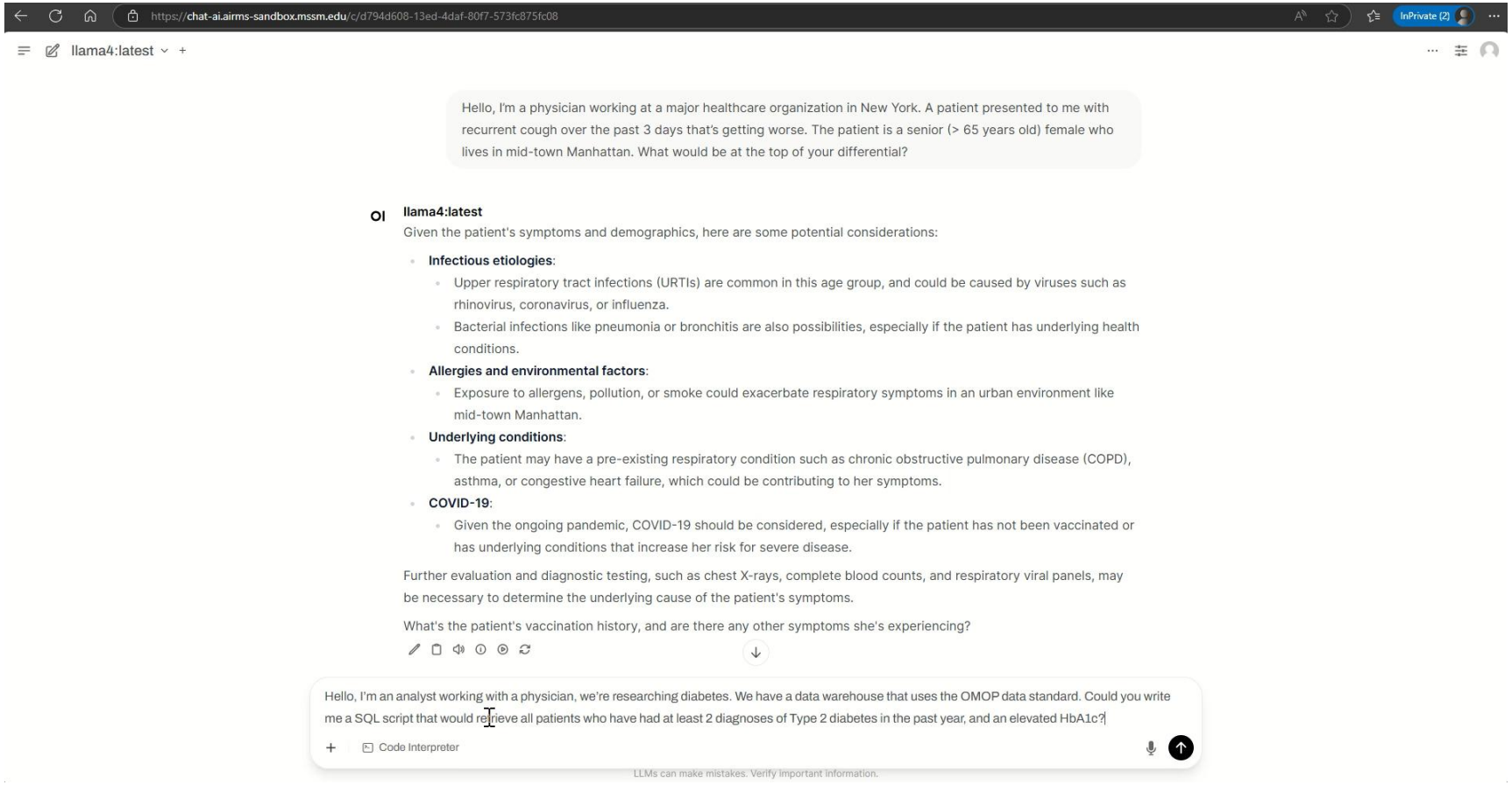
A Simple Prompt

- “Hello, I’m a physician working at a major healthcare organization in New York. A patient presented to me with recurrent cough over the past 3 days that’s getting worse. The patient is a senior (> 65 years old) female who lives in New York City. What would be at the top of your differential?”



A More Complex Prompt

- “Hello, I’m an analyst working with a physician, we’re researching diabetes. We have a data warehouse that uses the OMOP data standard. Could you write me a SQL script that would retrieve all patients who have had at least 2 diagnoses of Type 2 diabetes in the past year, and an elevated HbA1c?”



Chat AI Next Steps

- Over the coming weeks more complex features will be added, and how well the LLM can handle multiple requests (including incorporating Retrieval-Augmented Generation (RAG)-LLMs) will be tested
- The overall aim is to build a robust tool for use across the Mount Sinai Health System
- The LLM will also work closely with AIR•MS (AI Ready Mount Sinai), which will contain integrated clinical datasets
- The combination of AIR•MS and LLMs will accelerate research across the Mount Sinai Health System
- If you'd like to use the chatbot, please reach out to edwin.thrower@mssm.edu
- We would love to hear your feedback: https://app.smartsheet.com/b/form/47ca5aaa5e2a4119bf97476edd0422fe

Ollama and AIR-MS

What is Ollama

Ollama = a platform for running large language models (LLMs) **locally on your computer**

Supports models like **LLaMA, Mistral, Gemma, Phi, and others**

Key features:

- **Local-first:** no internet required once models are downloaded
- **Privacy-preserving:** your data stays on your machine
- **Easy interface:** run models with a single command (ollama run llama3)
- **Integrations:** works with apps, APIs, and developer workflows

Ollama on Minerva

To run ollama on Minerva run the following command from terminal (or Chimera Shell access on OnDemand):

```
$ minerva-ollama-web.sh -o <your ollama preffered location here>
```

After running this command you will see some output including specifications of host and authorization token that you can directly paste in your python session:

```
ollama_client = Client(host='http://10.95.46.94:51101', headers={"Authorization": "Bearer hasans10:d6972ea4292e66e93d4ece5f15831f1"})
```

Using ollama and AIR-MS on Minerva

```
# Setting Up OLLAMA on Minerva
from ollama import Client
### PASTE THE COPIED LINES FROM THE TERMINAL HERE
ollama_client = Client(host='http://10.95.46.94:55602',headers={"Authorization": "Bearer alleve01:fa3efbcefa8e9f73801b7ecb1840d6f1"})
```

To use a open source model you will first need to pull it and load it. You can find a list of available models here Let's start by trying `mistral` out.

```
# download the model
model = 'mistral'
ollama_client.pull(model)
```

```
mistral.ask("""Can you help me write an SQL query to extract all female patients, born between 1990 and 2000? This information
            is stored in the 'person' table of the CDMDEID schema, where females are identified by gender_concept_id = 8532 and
            year_of_birth is the year of birth.
            """)
```

Certainly! Here's an SQL query to extract all female patients born between 1990 and 2000 from the 'person' table in the CDMDEID schema:

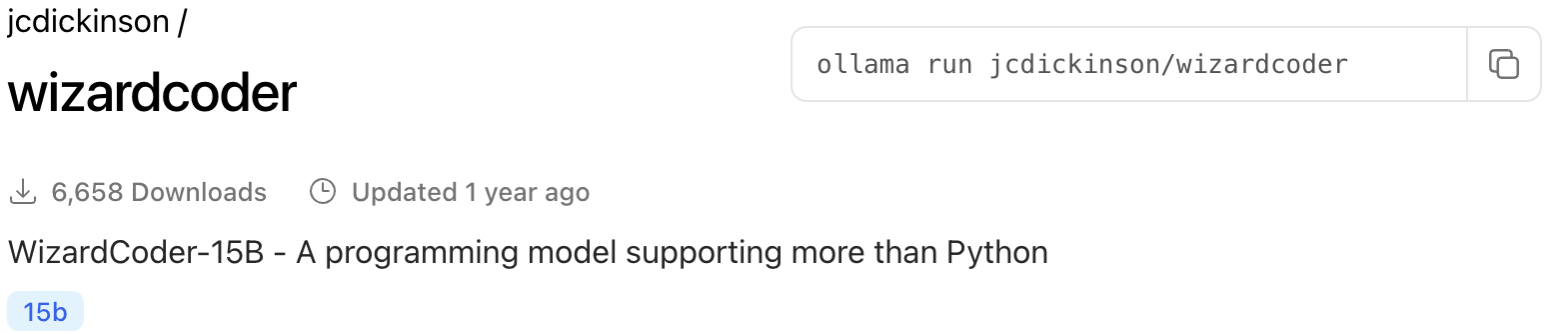
```
SELECT *
FROM cdmddeid.person
WHERE gender_concept_id IN (8532)
AND year_of_birth BETWEEN 1990 AND 2000
```

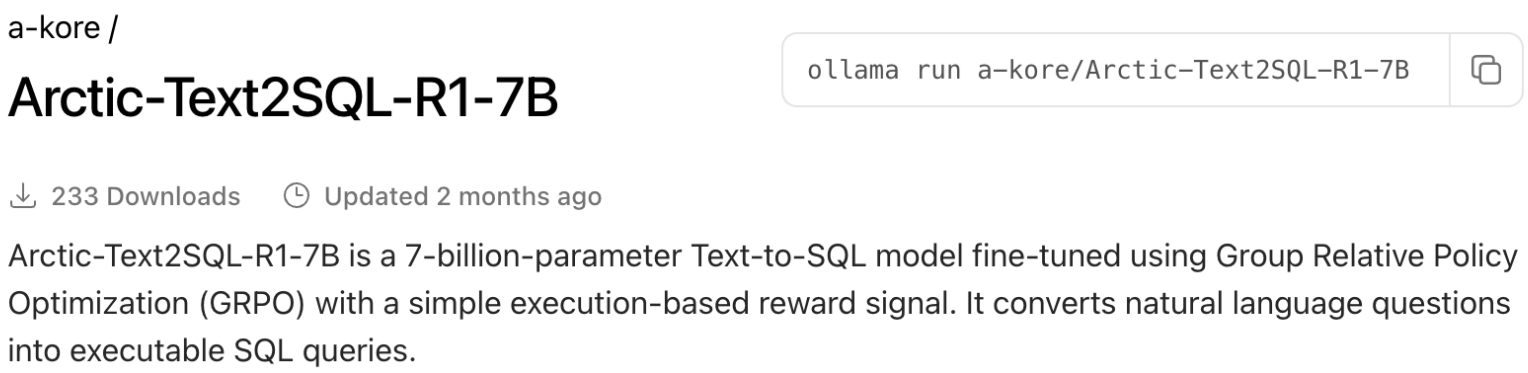
This query uses the `IN` operator to search for records where the gender concept ID is either 8532, which represents females in the OMOP CDM. The `BETWEEN` operator is used to specify the desired year of birth range (1990-2000).

Using optimized models

A list of models available on ollama can be found at https://ollama.com/search

There are multiple models trained for specific tasks, which can be very useful for vibe coding. For example:





Thank You

Icahn School of Medicine at Mount Sinai