IDUG

2022 EMEA **Db2** Tech Conference

Now You See It, Unveil New Insights Through SQL Data Insights

Akiko Hoshikawa

IBM

#IDUGDb2

Platform: Db2 for z/OS

Disclaimer

The information contained in this publication is provided for informational purposes only. While efforts were made to verify the completeness and accuracy of the information contained in this publication, it is provided AS IS without warranty of any kind, express or implied. In addition, this information is based on IBM's current product plans and strategy, which are subject to change by IBM without notice. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, this publication or any other materials. Nothing contained in this publication is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software.

References in this presentation to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates. Product release dates and/or capabilities referenced in this presentation may change at any time at IBM's sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way. Nothing contained in these materials is intended to, nor shall have the effect of, stating or implying that any activities undertaken by you will result in any specific sales, revenue growth or other results.

Agenda

Introduction of Db2 13 SQL Data Insights

Technology behind of SQL Data Insights

Understanding Semantic AI queries

Using Semantic AI queries

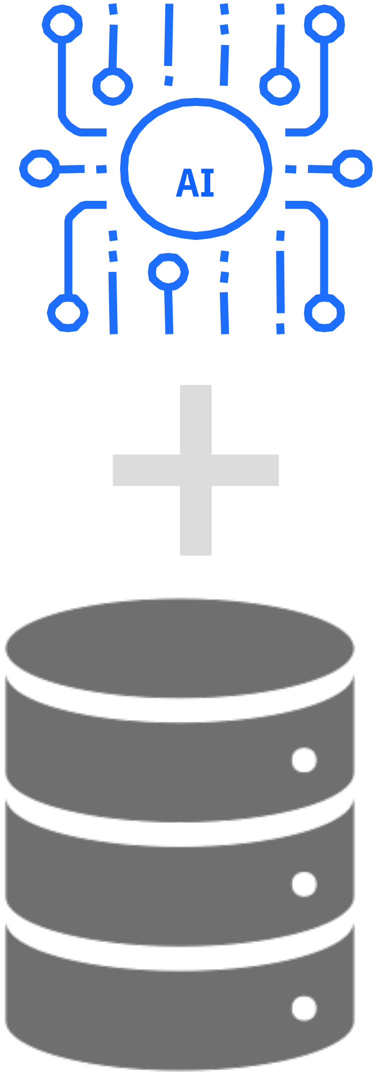
Steps enabling SQL Data Insights

Summary

Q&A

SQL Data Insights

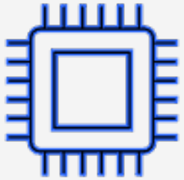
An industry-first relational database with embedded AI capabilities



Infuse NLP directly into your database on existing data to discover hidden information

Minimizes complexity of deploying AI into your applications

Single model used for a range of inferencing task over multiple fields

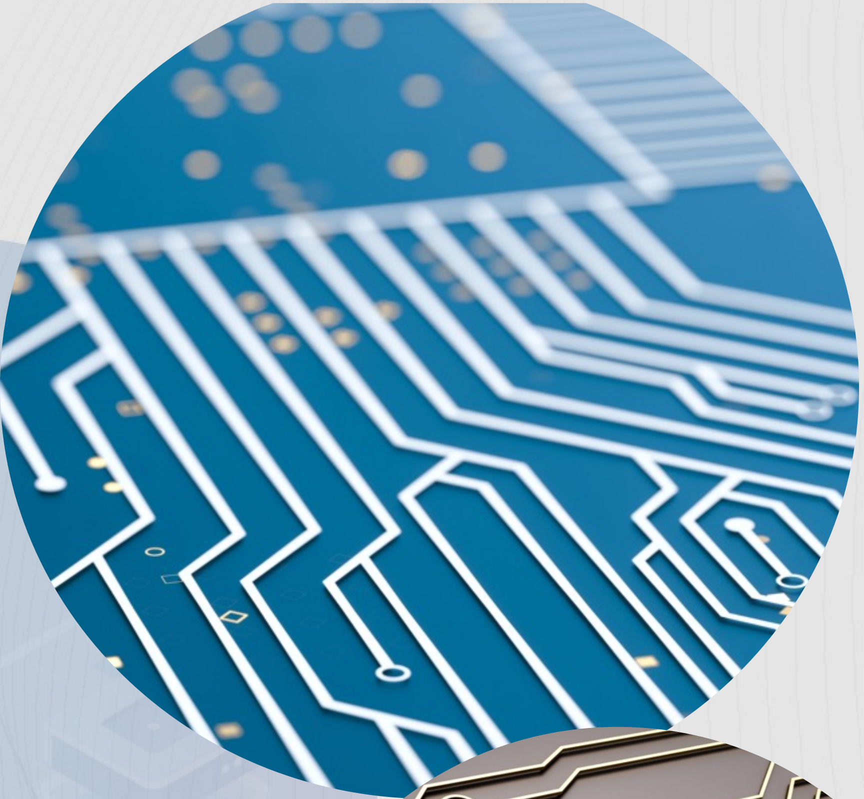


Exploits zIIPs and IBM Z acceleration

Semantic SQL Functions

Initial set of AI Built-In Functions available in Db2 13

Cognitive Intelligence Query	Functional Description	Db2 functions
semantic **similarity and dissimilarities**	• **Matching** rows/entities based on overall meaning (similarity/dissimilarity) • **Suggest** choices for incorrect or missing entities	AI_SIMILARITY
semantic **Clustering**	• **Find** entities/rows based on relationships between attributes in a given set • Example: Find animals similar to (lion, tiger, panther)	AI_SEMANTIC_CLUSTER
Reasoning Analogy	• **Find** entities/rows based on relationships between attributes • Example: Moon : Satellite :: Earth; ?	AI_ANALOGY




Technology Behind of SQL Data Insights

SQL Data Insights: Core Concepts

Unsupervised Neural Network Approach for Natural Language Processing: Word Embedding

- Captures word meaning as collective contributions of words (tokens) in the neighborhood
- Generates semantic representations of words (tokens) using vectors (Vector Embedding)
- Semantic similarities between words (tokens) measured using distance between vectors

Extending Vector Embedding Approach to structured databases: Database Embedding

- Every database column value, irrespective of its column type, converted to a text token
- View a database record as an unordered English-like sentence (bag-of-words) of text tokens
  - Every token is equally related to other tokens in the “sentence”, irrespective of their position
  - Tokens related to unique primary keys and NULL values are treated differently
- Semantic model infers meanings (behavior) of database column values based on their neighboring column values (e.g., within a table row, and across table rows)
- Exploit the trained model to enable new SQL semantic queries that operate on the relational data based on the inferred meaning, not using values

Relationship Hidden in a Table

CustID	Date	Merchant	State	Category	Items	Amount
CustA	9/16	Store-X	NY	Fresh produce	Bananas	80
CustA	9/16	Store-X	NY	Fresh produce	Apples	120
CustD	9/16	Store-Z	NY	Stationary	Crayons	50
CustD	9/16	Store-Z	NY	Stationary	Folders	150
CustC	10/16	Store-X	CT	Fresh produce	Bananas	100
CustC	10/16	Store-X	CT	Fresh produce	Oranges	100

– Which customer's behavior is more similar to Cust-A's behavior ?

– What makes you to think so?

Relationship Hidden in a Table

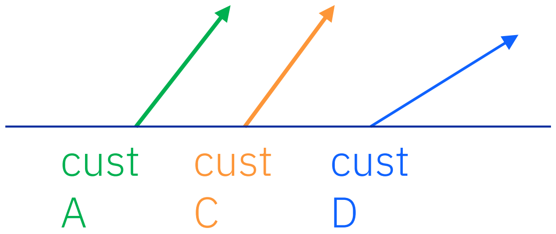
CustID	Date	Merchant	State	Category	Items	Amount
CustA	9/16	Store-X	NY	Fresh produce	Bananas	80
CustA	9/16	Store-X	NY	Fresh produce	Apples	120
CustD	9/16	Store-Z	NY	Stationary	Crayons	50
CustD	9/16	Store-Z	NY	Stationary	Folders	150
CustC	10/16	Store-X	CT	Fresh produce	Bananas	100
CustC	10/16	Store-X	CT	Fresh produce	Oranges	100

Textification : transform values to text token

Txn1 custID_custD Date_9/16 Merchant_ Store-Z State_NY Category_Stationary Items_Folders Amount_1

Generation of "meaning vector" for every column value

custA is similar to custC due to similar purchasing behavior.



- If there is no primary key, row-ID (Txn1 above) will be generated and represent other column values in the same row.
- Meaning vector of the primary key captures the meaning of an entire row.
- Meaning of non-primary key value contributes correctively to its neighbors (e.g. NY is associated with Bananas and Crayons)

Relationship Hidden in a Table

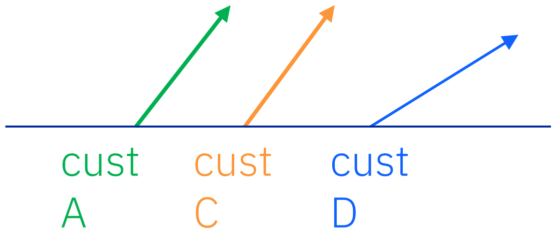
CustID	Date	Merchant	State	Category	Items	Amount
CustA	9/16	Store-X	NY	Fresh produce	Bananas	80
CustA	9/16	Store-X	NY	Fresh produce	Apples	120
CustD	9/16	Store-Z	NY	Stationary	Crayons	50
CustD	9/16	Store-Z	NY	Stationary	Folders	150
CustC	10/16	Store-X	CT	Fresh produce	Bananas	100
CustC	10/16	Store-X	CT	Fresh produce	Oranges	100

Textification : transform values to text token

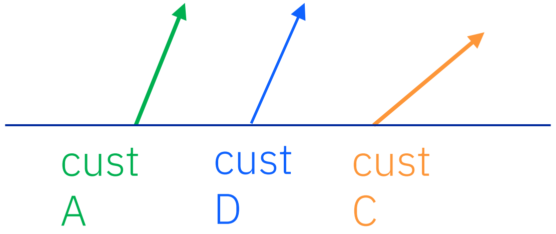
Txn1 custID_custD Date_9/16 Merchant_ Store-Z State_NY Category_Stationary Items_Folders Amount_1

Generation of "meaning vector" for every column value

custA is similar to custC due to similar purchasing behavior.



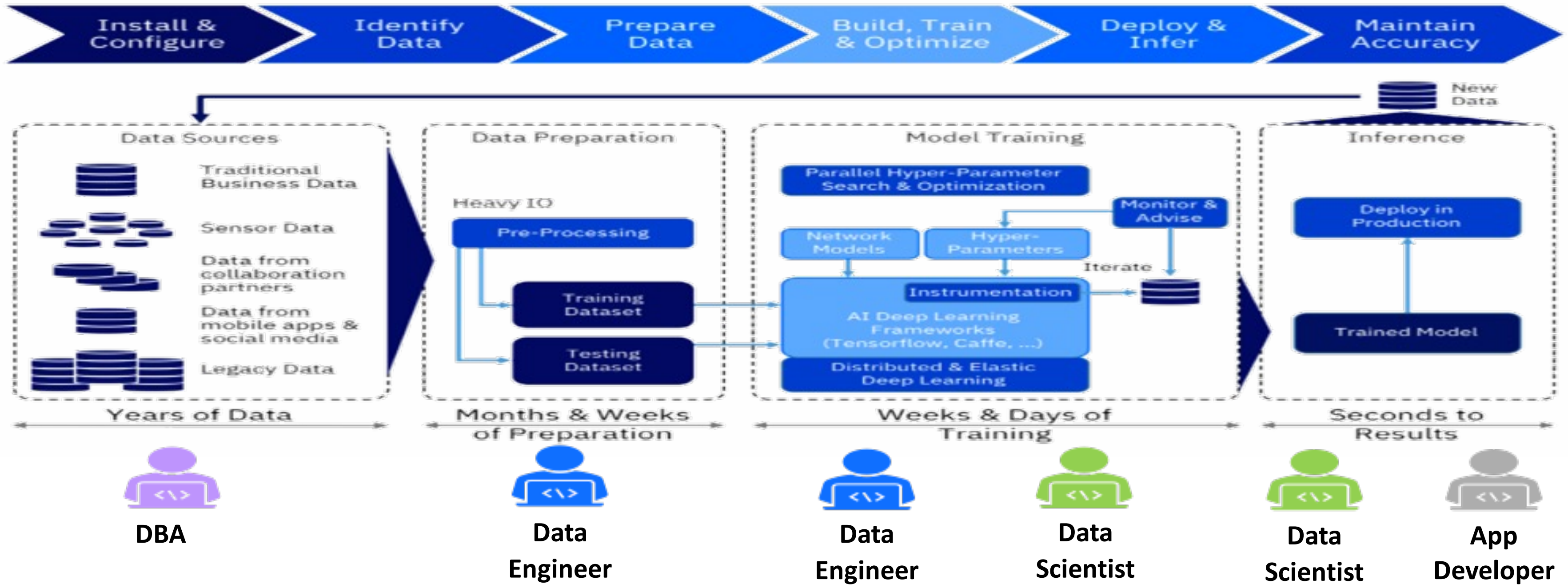
(Withtout Category/Items)
custA is similar to custD due to similar behavior



- If there is no primary key, row-ID (Txn1 above) will be generated and represent other column values in the same row.
- Meaning vector of the primary key captures the meaning of an entire row.
- Meaning of non-primary key value contributes correctively to its neighbors (e.g. NY is associated with Bananas and Crayons)

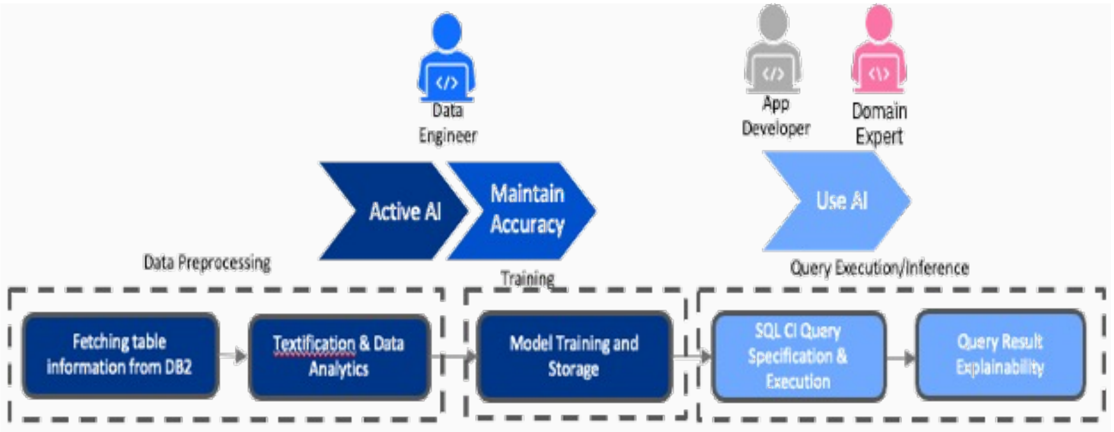
Extract greater value from Db2 for z/OS data

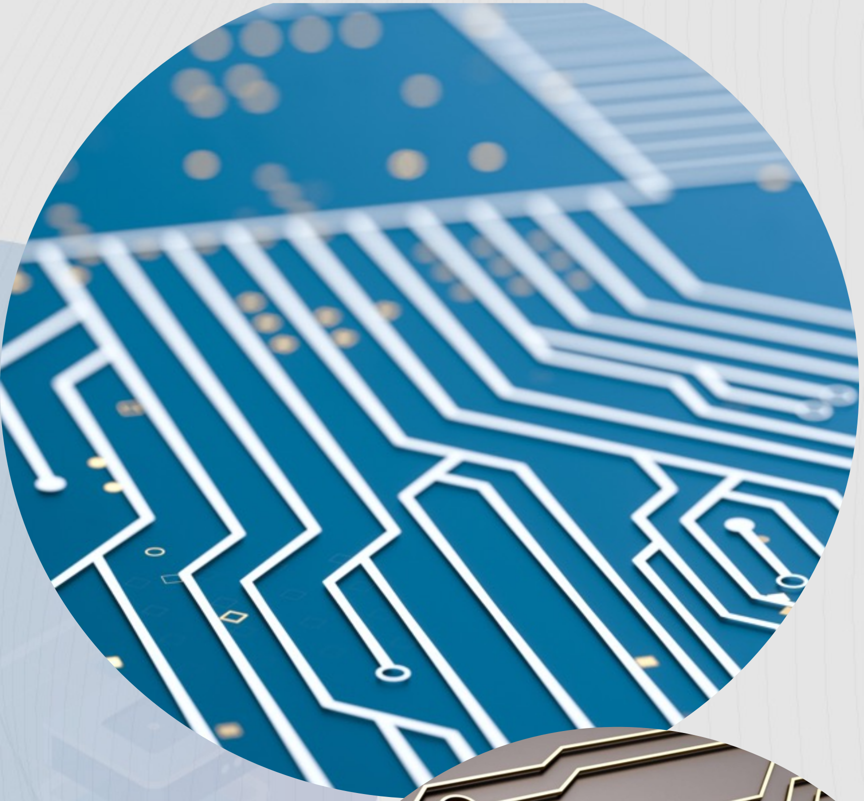
Traditional AI models are complex to build and serve a single narrow purpose



VS.

Build Neural Network powered relationship maps using unsupervised training over (unlabeled) structured data





Semantic AI Functions

AI_SIMILARITY

```
AI_SIMILARITY (expression-1  USING MODEL COLUMN column-name,
               expression-2  USING MODEL COLUMN column-name )
```

AI_SIMILARITY('APPLE', 'RASPBERRY' USING MODEL COLUMN FRUIT)

It computes a similarity score using the values returned by expression-1 and expression-2.

Results of AI_SIMILARITY – floating point number between -1.0 and 1.0

1.0 means very similar or same, -1.0 means very dissimilar

Find top 5 customer IDs that are the most similar to a customer "3668-QPYBJ" who closed his account

note : customerID is defined as a primary key

```
SELECT  AI_SIMILARITY(X.customerID,'3668-QPYBK' USING MODEL
COLUMN customerID )  AS SimilarityScore, X.*
FROM CHURN X
WHERE X.customerID <> '3668-QPYBK'
ORDER BY SimilarityScore DESC
FETCH FIRST 5 ROWS ONLY;
```

SIMILARITYSCORE	CUSTOMERID	GENDER	SENIORCITIZEN	PARTNER	DEPENDENTS	TENURE	PHONESERVICE	MULTIPLELINES	INTERN
0.902809739112854	2207-OBZNX	Male	0	No	No	7	Yes	No	DSL
0.8648061752319336	2108-XWMPY	Male	0	No	No	3	No	No phone service	DSL
0.8551765084266663	6304-IJFSQ	Male	0	No	No	3	Yes	No	DSL
0.8473891615867615	5493-SDRDQ	Male	0	No	No	2	Yes	No	DSL
0.8069272637367249	7580-UGXNC	Female	1	No	No	2	Yes	No	DSL

AI_SIMILARITY – Dissimilarity Query

Find top 5 customer IDs that are the least similar to a customer “3668-QPYBJ” who closed his account
note : customerID is defined as a primary key

```
SELECT  AI_SIMILARITY(X.customerID,'3668-QPYBK’ USING MODEL
COLUMN customerID )  AS SimilarityScore, X.*
FROM CHURN X
WHERE X.customerID <> '3668-QPYBK'
ORDER BY SimilarityScore ASC
FETCH FIRST 5 ROWS ONLY;
```

SIMILARITYSCORE	CUSTOMERID	GENDER	SENIORCITIZEN	PARTNER	DEPENDENTS	TENURE	PHONESERVICE	MULTIPLELINES	INTERNETSERVICE
-0.19289052486419678	6050-FFXES	Female	0	Yes	No	69	Yes	Yes	Fiber optic
-0.1552256941795349	6766-HFKLA	Female	0	Yes	No	56	Yes	Yes	Fiber optic
-0.1492832899093628	8433-WPJTV	Female	1	Yes	Yes	65	Yes	Yes	Fiber optic
-0.13930177688598633	4128-ETESU	Female	1	Yes	No	47	Yes	Yes	Fiber optic
-0.12915533781051636	1400-WIVLL	Male	0	Yes	No	57	Yes	Yes	Fiber optic

Sponsor User’s Test

Find the most similar 5 car manufacturers as Ferrari in the car data base

```
SELECT DISTINCT AI_SIMILARITY(MAKE,'Ferrari’) as SCORE, MAKE
FROM CARS
WHERE MAKE <> 'Ferrari'
ORDER BY 1 DESC
FETCH FIRST 5 ROWS ONLY
---------+---------+---------+---------+---------+-
      Score                  MAKE
---------+---------+---------+---------+---------+-
+0.7351751327514648E+00  Lamborghini
+0.6999126672744751E+00  Rolls-Royce
+0.6649318337440491E+00  Bentley
+0.6472378969192505E+00  Corvette
+0.6257274746894836E+00  McLaren
```

https://www.kaggle.com/datasets/ander289386/cars-germany

Insurance Use Case

Insurance company realizes that they are undercharging a policy holder and want to find customers since 2015 that are similar to him to avoid losses

```
SELECT *
FROM
(SELECT C.*,
AI_SIMILARITY(DRIVERS_LICENSE_NUMBER,
'339 713 155’) AS SIMILARITY
FROM IBM.INSURANCE C)
WHERE
HEATING_LAST_UPDATE_YEAR>’2015’
ORDER BY SIMILARITY
DESC
FETCH FIRST 20 ROWS ONLY
```

IBM Synthetic Data – Insurance Underwriters

AI_SEMANTIC_CLUSTER

AI_SEMANTIC_CLUSTER (member-expression USING MODEL COLUMN column-name,
clustering-expressions)

AI_SEMANTIC_CLUSTER('STRAWBERRY' USING MODEL COLUMN FRUIT, 'RASPBERRY', 'BLACKBERRY', 'BLUEBERRY')

computes a clustering score using the values returned by clustering-expressions
Results of AI_SEMANTIC_CLUSTER – floating point number between -1.0 and 1.0
Higher score means a better clustering of member-expression among the clustering-expressions

Based on a group of customers who have high valued houses and no recent updates, find similar customers to increase premium

```
SELECT  C.*,
AI_SEMANTIC_CLUSTER(C.DRIVERS_LICENSE_NUMBER ,'Q08670943', '543877806', 'T30381936') AS SIMILARITY
FROM AAMININ.INSURANCE C
WHERE C.DRIVERS_LICENSE_NUMBER NOT IN ('Q08670943', '543877806','T30381936')
ORDER BY SIMILARITY DESC
FETCH FIRST 20 ROWS ONLY
```

AI_ANALOGY :

AI_ANALOGY (source-1, target-1, source-2, target-2)

AI_ANALOGY('STRAWBERRY' USING MODEL COLUMN FRUIT, 'RED',
'LEMON', 'YELLOW')

computes an analogy score using the values returned by the arguments. Higher the score, a better analogy than a lower score.
Results of AI_ANALOGY – floating point number, NOT bounded by -1.0 and 1.0

Analyze the relationships between length of contract and internet service subscriptions

```
SELECT DISTINCT
    AI_ANALOGY('Month-to-month' USING MODEL COLUMN CONTRACT,
               'Fiber optic' USING MODEL COLUMN INTERNETSERVICE,
               'Two year',
               INTERNETSERVICE) AS ANALOGY_
       X.INTERNETSERVICE
 FROM CHURN X
 WHERE X.INTERNETSERVICE<>'Fiber optic'
 ORDER BY ANALOGY_SCORE DESC
```

ANALOGY_SCORE	INTERNETSERVICE
0.8413964921922206	DSL
0.6485916530516833	No

Insurance Use Case

Find risky customers in Oklahoma based on a risky customer found in Kansas

```
SELECT * FROM
(SELECT AI_ANALOGY (
'Kansas' USING MODEL COLUMN DRIVERS_LICENSE_STATE,
'Q06-25-5829' USING MODEL COLUMN DRIVERS_LICENSE_NUMBER,
'Oklahoma' USING MODEL COLUMN DRIVERS_LICENSE_STATE,
 DRIVERS_LICENSE_NUMBER) AS ANALOGY_SCORE ,C.*
FROM IBM.INSURANCE C)
ORDER BY ANALOGY_SCORE DESC
FETCH FIRST 20 ROWS ONLY ;
```

IBM Synthetic Data – Insurance Underwriters Use case

SQL Data Insights - Potential Use Cases

Finance (Consumer Banking, Investment Advisors)
- Find customers with similar transactions
- Non-performing Asset Identification (NPA)

Fraud detection
- Anti money laundering
- Account take-over

Insurance
- Identify similar/dissimilar claims
- Evaluate risk profiles by analyzing patient profiles (e.g., symptoms, diagnosis…)

IoT
- Find households/hotel rooms with similar energy consumption patterns

Customer analytics
- Find similar customers based on buying patterns
- Customer Churn Analytics

Advanced sales prediction using external data
- Predict sales of new products to existing customer base

IT incident ticket analysis
- Find accounts with similar ticket patterns

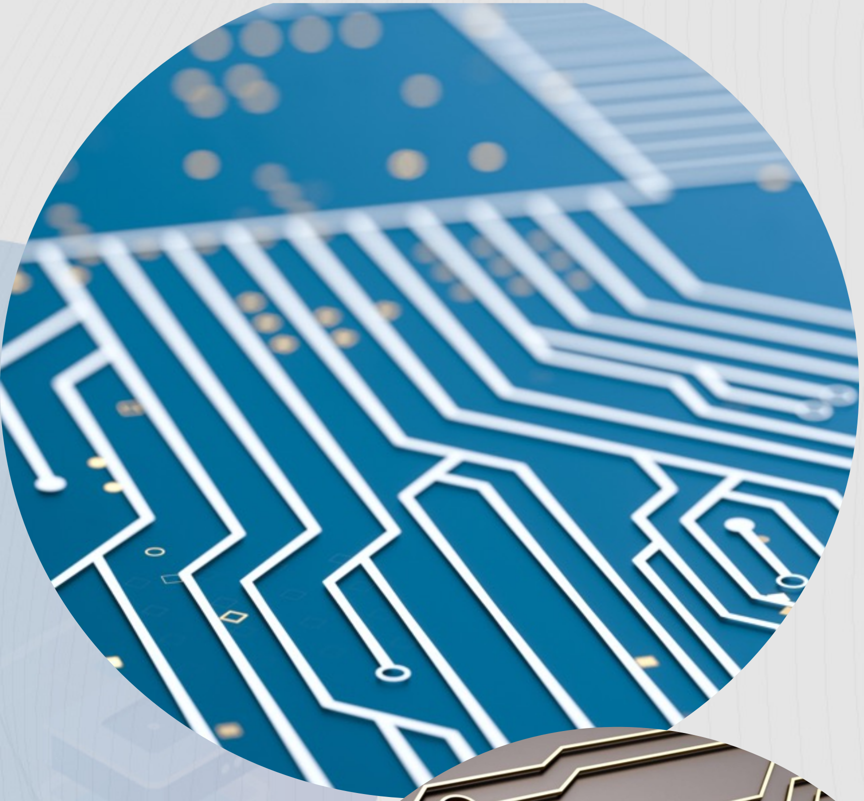
HR
- Find employees with similar skills and similar/different experience

Entity resolution/Data imputation for data quality
- Identify multiple instances of a single customer across multiple data sources

Any use case in your business?

Customer Retention Analysis

- Business needs – retention program at telecom company
  - Reduce the customers who leave the service.
- Data stored in databases
  - Customer information, Service subscription, Billing
- Persona – a business analyst
  - Data analysis skill (SQL skill) – good
  - Data science skill – limited
- Scenario
  - Use AI semantic queries to perform analysis.
    - Identify similar customers who might leave the business based on the customer's record who had already left
    - Identify the common pattern among high-risk customers
    - Identify the set of customers who are not likely leaving and understand the pattern



Using AI Queries (Hint and tips)

SQL Data Insights – Sample Query 1

Based on expenditure transaction data, which 10 vendors are most similar to vendor name 'VERIZON', ranked by the similarity score (desc)

```
SELECT DISTINCT VENDOR_NAME, SIMILARITY_SCORE
FROM
(
SELECT
  VENDOR_NAME,
  AI_SIMILARITY(VENDOR_NAME, 'VERIZON' USING MODEL COLUMN VENDOR_NAME)
                                                AS SIMILARITY_SCORE
  FROM USRT031.VIRG1TB
)
WHERE
        SIMILARITY_SCORE IS NOT NULL
   AND TRIM(VENDOR_NAME) <> 'VERIZON'

ORDER BY SIMILARITY_SCORE DESC

FETCH FIRST 10 ROWS ONLY
```

VENDOR_NAME	SIMILARITY_SCORE
Verizon	1.000000
Nextel Communications Mid-Atlantic Inc	0.589276
AMERICAN ASSOC OF MOTOR VEHICLE ADMIN	0.578739
CAVALIER TELEPHONE LLC	0.577649
American Messaging	0.575459
TELCOVE	0.574863
Cox Communications Northern Virginia	0.574773
MetroCast Co	0.574247
Amtech Inc	0.573625

- SQL Data Insights functions are regular Db2 scalar functions
- Indexes for underlying model related tables automatically created by Db2
- SQL Data Insights functions can return NULL
- Use relative scores (-1 to +1) returned by SQL Data Insights functions
- Strings are internally transformed during training as well as scoring
- Regular SQL tuning practices apply

SQL Data Insights – Sample Query 2

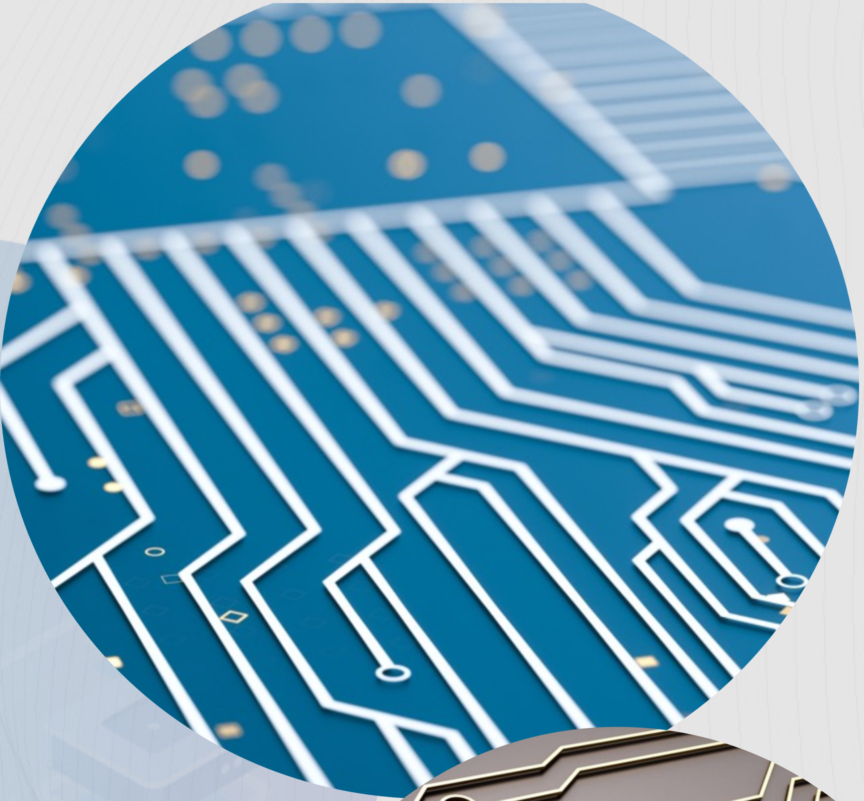
Based on expenditure transaction data, for the agency 'Treasury Board' (AGY_AGENCY_KEY = 125) and its most similar 10 agencies, provide monthly ranking of each agency based on total transaction amount in the month

```
SELECT
  YEAR(VOUCHER_DATE) AS YR,
  MONTH(VOUCHER_DATE) AS MTH,
  SIMILAR.AGY_AGENCY_KEY,
  SIMILAR.AGY_AGENCY_NAME,
  SUM(AMOUNT) AS TOTAL_AMOUNT,
  RANK() OVER (PARTITION BY
       YEAR(VOUCHER_DATE),
       MONTH(VOUCHER_DATE)
       ORDER BY SUM(AMOUNT) DESC
     ) AS RANKING
FROM
USRT031.VIRG1TB EX,
(
  SELECT
   DISTINCT
   EXP.AGY_AGENCY_KEY,
   AGY.AGY_AGENCY_NAME,
   AI_SIMILARITY(EXP.AGY_AGENCY_KEY, 125 USING MODEL COLUMN EXP.AGY_AGENCY_KEY )
                                                                        AS SIMILARITY_SCORE

   FROM USRT031.VIRG1TB EXP
         INNER JOIN USRT031.VIRGAGY AGY  ON EXP.AGY_AGENCY_KEY = AGY.AGY_AGENCY_KEY

   WHERE
     AI_SIMILARITY(
         EXP.AGY_AGENCY_KEY, 125 USING MODEL COLUMN EXP.AGY_AGENCY_KEY
     ) IS NOT NULL

 ORDER BY 3 DESC
 FETCH FIRST 10 ROWS ONLY
) SIMILAR

WHERE EX.AGY_AGENCY_KEY = SIMILAR.AGY_AGENCY_KEY

GROUP BY
  YEAR(VOUCHER_DATE),
  MONTH(VOUCHER_DATE),
  SIMILAR.AGY_AGENCY_KEY,
  SIMILAR.AGY_AGENCY_NAME

ORDER BY YR, MTH, RANKING
```

YR	MTH	AGY_AGEI	AGY_AGENCY_NAME	TOTAL_AMOUNT	RANKING
2015	7	125	Treasury Board	67919790.15	1
2015	7	11	Eastern Virginia Medical School	3985144.32	2
2015	7	56	Virginia Tourism Authority	3401640.34	3
2015	7	48	Virginia Economic Development Partnership	3097507	4
2015	7	397	Innovation & Entreprenuership Investment Authority	1319704.64	5
2015	7	291	Institute for Advanced Learning and Research	1018374.08	6
2015	7	398	Jefferson Science Associates, LLC	436440.23	7
2015	7	161	Interstate Organization Contributions	190940	8

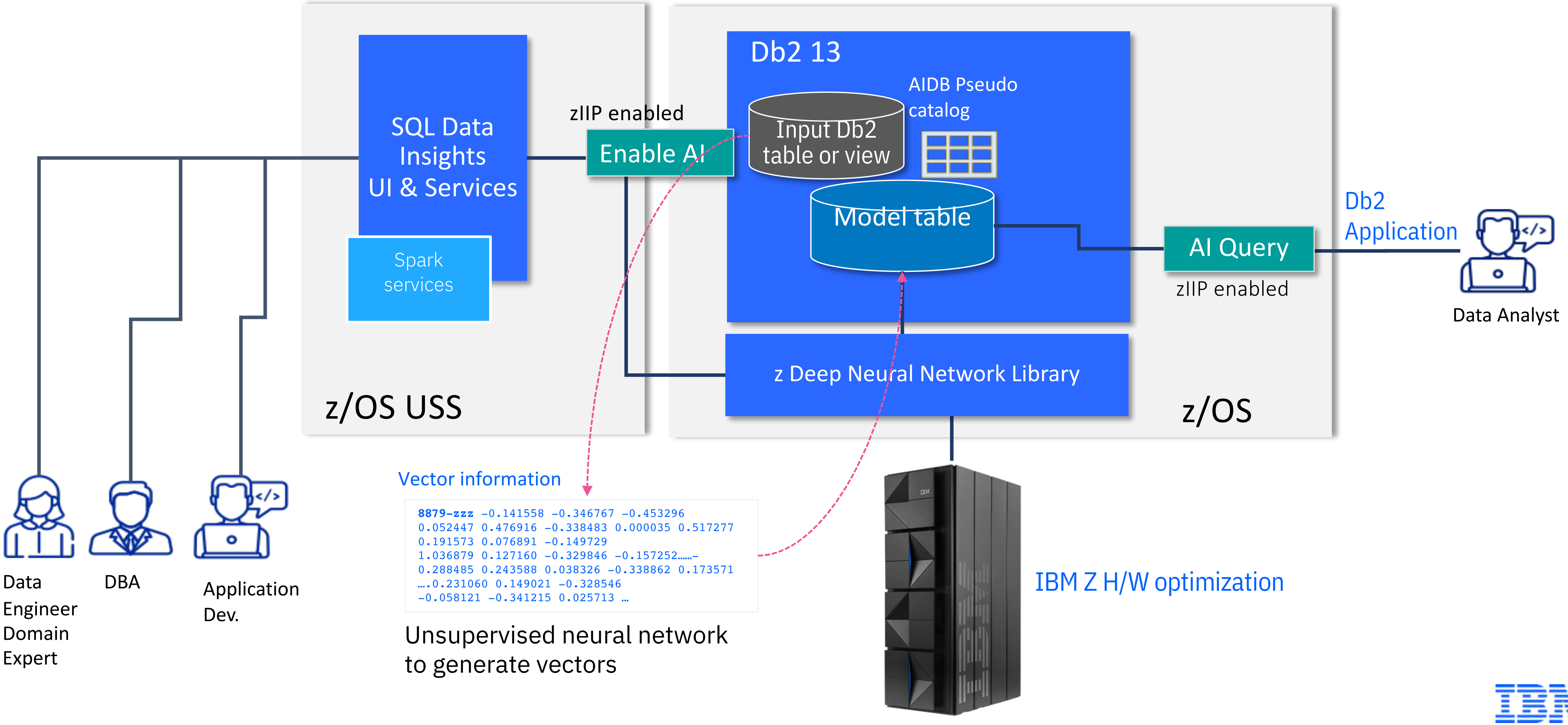
- SQL Data Insights functions augment existing SQL skills that people already use for complex analytical queries
- Results of SQL Data Insights functions can be used to build more advanced SQL based analytics
- Views could be used to simplify training and scoring based on multiple table joins

```
SELECT
  YEAR(VOUCHER_DATE) AS YR,

  MONTH(VOUCHER_DATE) AS MTH,

  SIMILAR.AGY_AGENCY_KEY,
  SIMILAR.AGY_AGENCY_NAME,

  SUM(AMOUNT) AS TOTAL_AMOUNT,

  RANK() OVER (PARTITION BY
       YEAR(VOUCHER_DATE),
       MONTH(VOUCHER_DATE)
       ORDER BY SUM(AMOUNT) DESC

     ) AS RANKING
FROM
USRT031.VIRG1TB EX,
(
  SELECT
    DISTINCT
    EXP.AGY_AGENCY_KEY,
    AGY.AGY_AGENCY_NAME,

    AI_SIMILARITY(EXP.AGY_AGENCY_KEY, 125 USING MODEL COLUMN
EXP.AGY_AGENCY_KEY )

AS SIMILARITY_SCORE
FROM USRT031.VIRG1TB EXP
          INNER JOIN USRT031.VIRGAGY AGY  ON EXP.AGY_AGENCY_KEY =
AGY.AGY_AGENCY_KEY
    WHERE

       AI_SIMILARITY(

            EXP.AGY_AGENCY_KEY, 125 USING MODEL COLUMN
EXP.AGY_AGENCY_KEY
      ) IS NOT NULL

 ORDER BY 3 DESC
 FETCH FIRST 10 ROWS ONLY
) SIMILAR

WHERE EX.AGY_AGENCY_KEY = SIMILAR.AGY_AGENCY_KEY

GROUP BY

  YEAR(VOUCHER_DATE),
  MONTH(VOUCHER_DATE),
  SIMILAR.AGY_AGENCY_KEY,
  SIMILAR.AGY_AGENCY_NAME

ORDER BY YR, MTH, RANKING
```



Enabling SQL Data Insights

SQL Data Insights – High Level Overview



Steps to Enable AI Queries

1 → 2 → 3 → 4 → 5

Step 1

- Db2 & z/OS setup
  - Create pseudo catalog, procedures
  - Create a IVP table
  - Setup z/OS libraries

Step 2

- Install & configure UI
  - UI installation
  - Configure Db2 connection and setup training

Step 3

- Enable AI
  - Pick columns and filtering
  - Trigger training
    - Train CHURN table

Step 4

- Review the training results
  - Model data analysis (Influence and discriminatory metrics)

Step 5

- Run AI queries
  - Test AI Queries
    - Use CHURN table

Step-0 : Software and Hardware Requirement

- Hardware : zEC12 to z16
- Function level V13R1M500 above
  - Technical preview available in V12
- z/OS 2.4 or above with the prerequisite maintenance that installs the following AI libraries with the latest APARs :
  - For z/OS 2.5 with APARs OA62901, OA62902, and OA62903
  - For z/OS 2.4 with APARs OA62849, OA62886, and OA62887
    - IBM Z AI Data Embedding library
    - IBM Z AI Optimization library
    - IBM Z Deep Neural Network library
  - IBM OpenBLAS PH44479 and PH45672 (z/OS 2.4) or PH45663 (z/OS 2.5)
- zIIP eligibility for training requires z/OS support
  - z/OS Supervisor APAR OA62728
  - Java 64 bit SDK V8 SR7 FP6 or later

Step 1 : Db2 Preparation

Db2 function level

- V13R1M500 or higher
- APPLCOMPAT V13R1M500 or higher
  - JDBC driver for GUI

Db2 preparation

- SDSNSAMP member DSNTIJAI
  - Create SQL Data Insights pseudo-catalog (DSNAIDB1)
  - Create DSNAIDB2 for model tables
  - Create stored procedures
  - Db2 permissions for GUI users

Verify Db2 access to z/OS library

- Verify SYS1. SIEALNKE and CEE.SCEERUN2 are APF authorized

Create sample CHURN table (IVP)

- SDSNSAMP member DSNTIJAV
  - Create sample table DS AIDB.CHURN
  - Insert approx. 7000 rows

Notes :

- zLOAD (DRDA fast load) is used to load the vectors to model table
  - Ensure DSNUTILU stored procedure is configured
  - Review load utility setup and control statement (template) in GUI Settings
  - zLOAD retry utility is available. Contact IBM if you need to retry zLOAD without retraining

Notes :

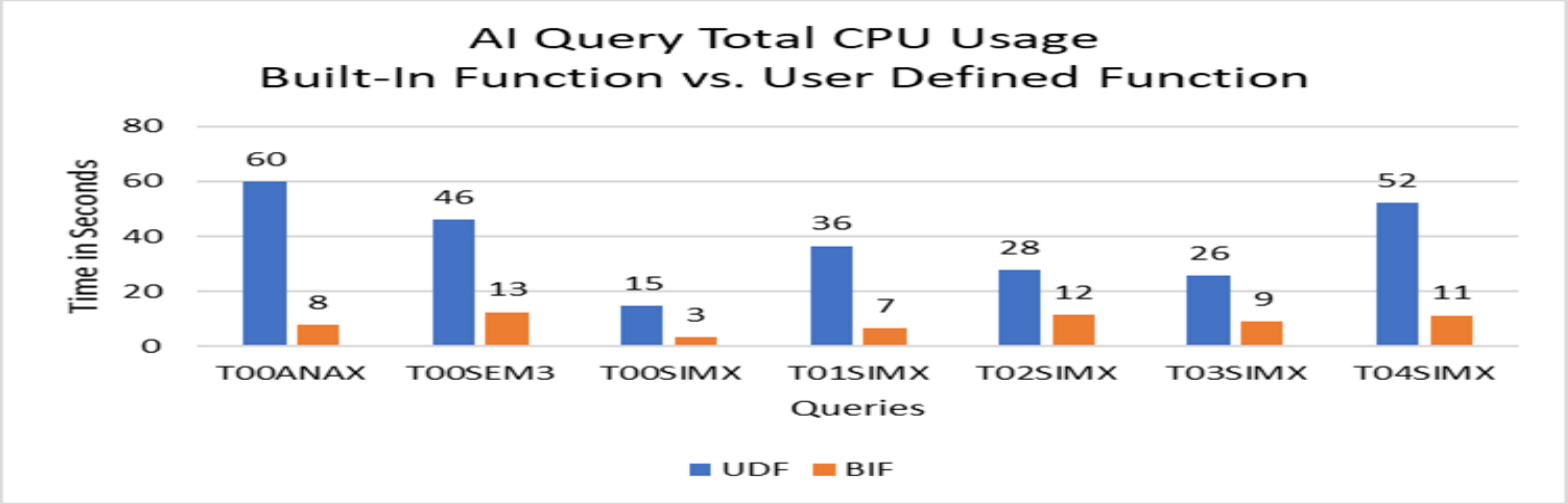
- Db2 12 users can use Beta 2.1
  - Use UDF instead of Built-in-function
    - Similar sample jobs to create pseudo catalog, AIDB, stored procs. , and UDFs
    - WLM application environment definition
    - IVP (DSNTIJAV)

Why Db2z 13?

Semantic queries using UDF vs Built-in-Function

Technology Preview is available in Db2 12 using UDF

- Training process is identical as Db2 13
- Semantic queries do not utilize built-in function nor z/OS optimization

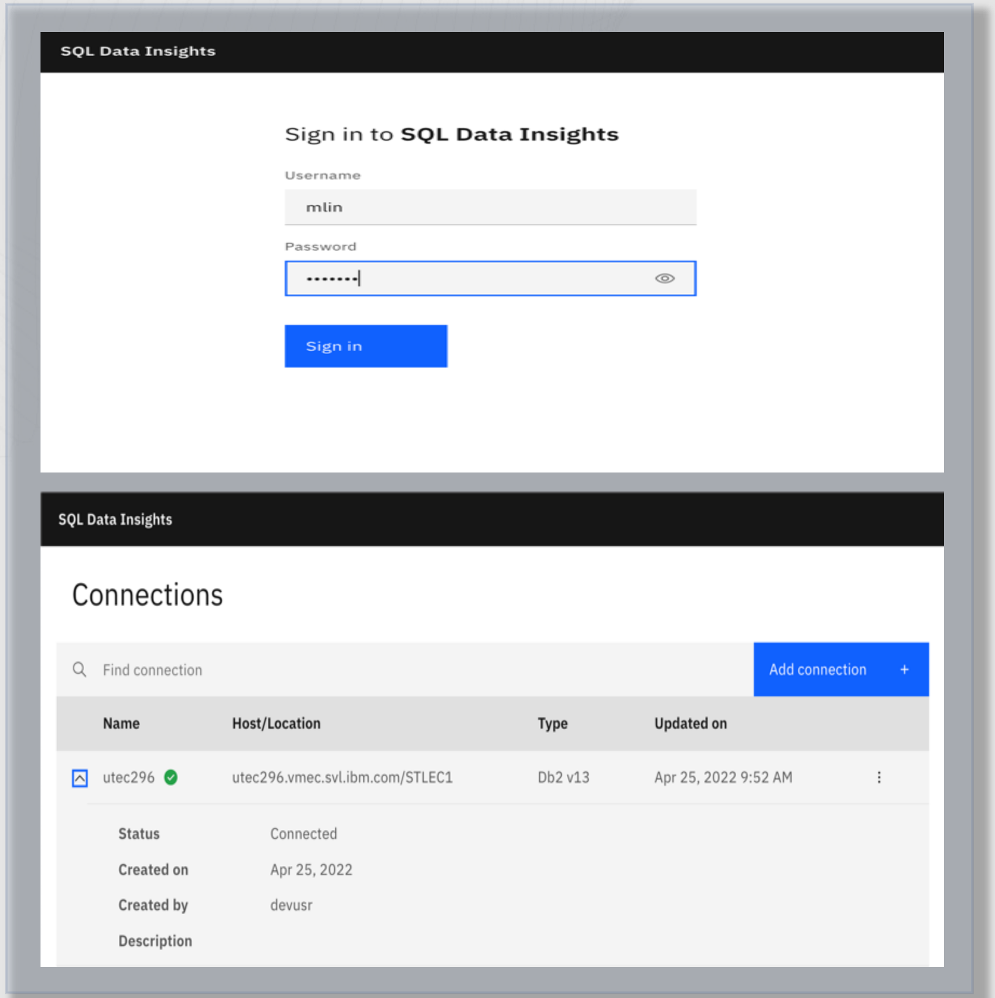


- Elapsed time and CPU time : 2 to 7x less with BIF in IBM z16 due to avoiding UDF + System Z H/W Optimization

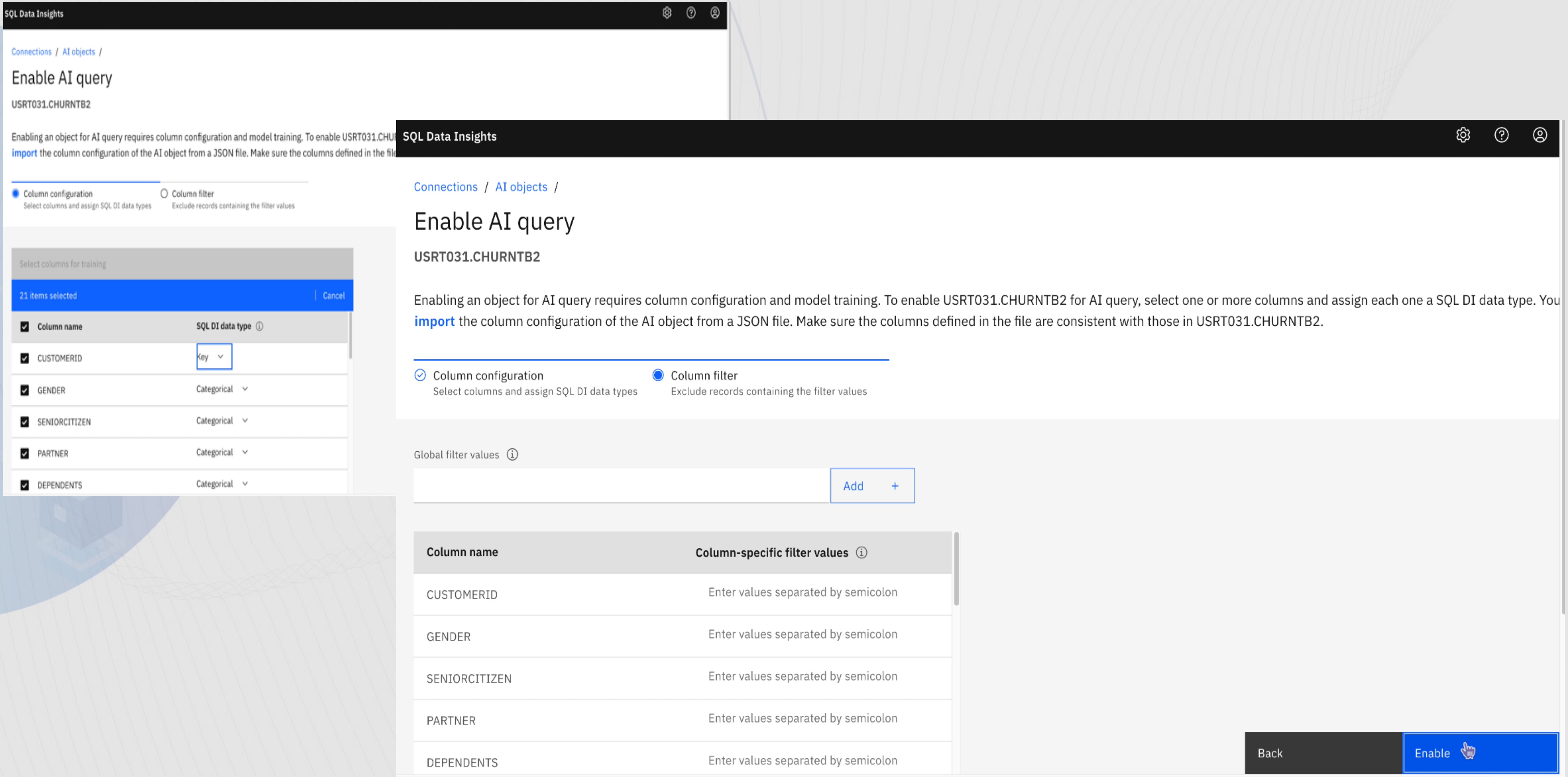
Step 2: Install UI & Training services and connect to Db2

Notes

- Have a z/OS UserID identified as the administrator of SQL Data Insights service
  - The user ID needs to have a OMVS segment defined
  - Recommend to use the provided user profile template
- Prepare a ZFS system (recommend 50 GB)
  - For SQLDI configuration files and log files
- Set up a RACF keyring and certificate/private key
  - For user authentication and SSL communications
- Reserve a range of network ports (recommend to reserve 21 ports, minimum 9 ports)
  - For SQL Data Insights service and Spark cluster
- Configure SQL Data Insights Service
- Identify users who can access SQL Data Insights UI service
  - Define RACF SQLDIGRP
  - Connect the users to SQLDIGRP group

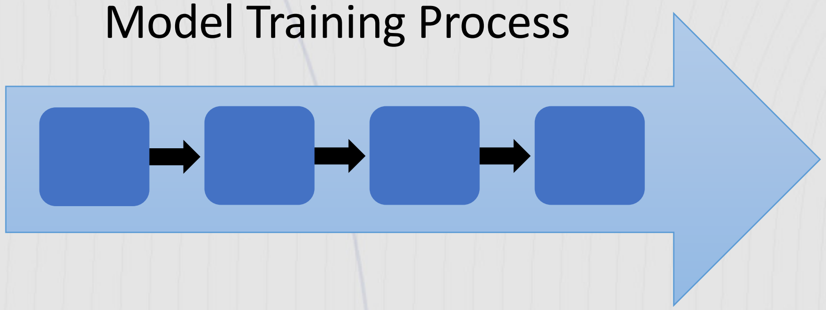


Step 3: Enable AI



Step 3 : Enable AI - Model training Internal

Model Training Process



Step A
Create the vector table

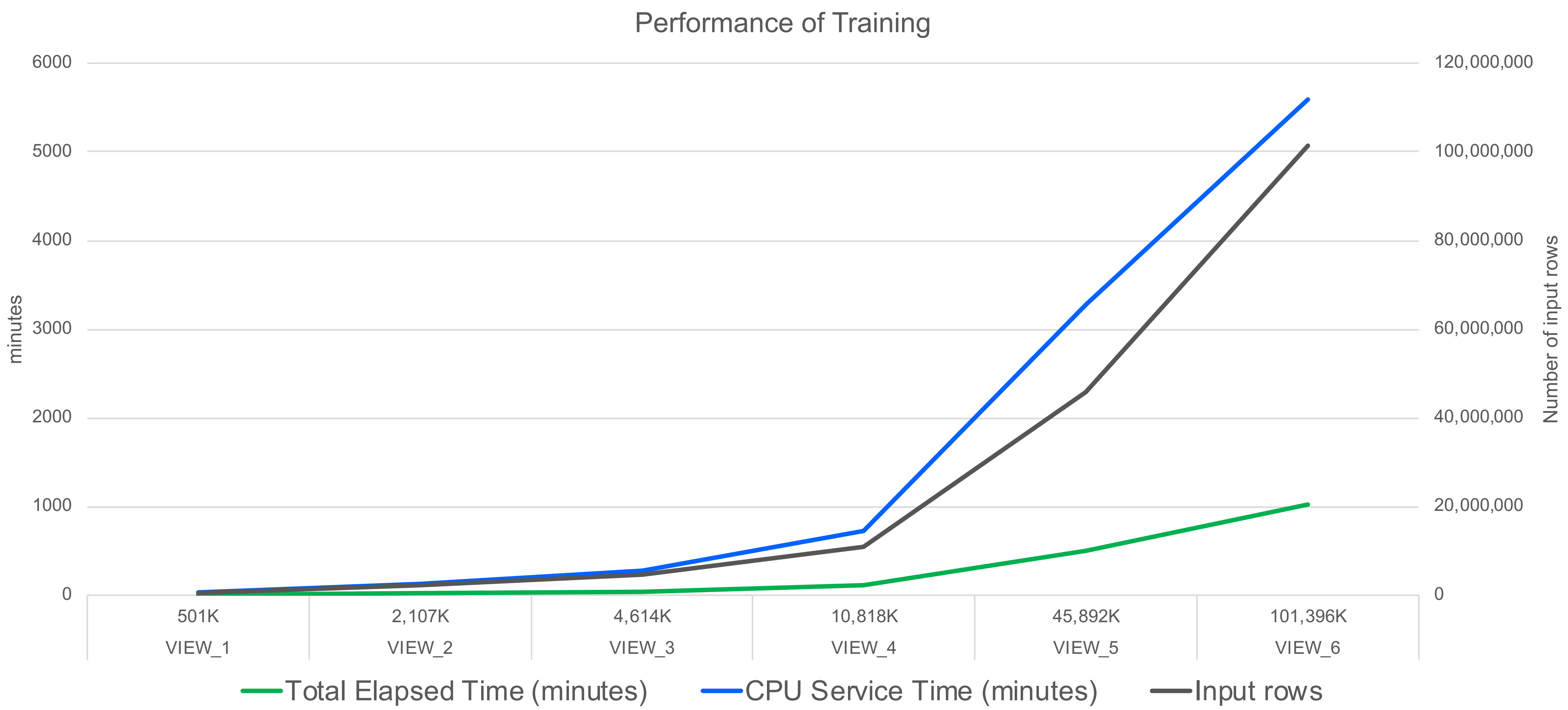
Step B
Read data from the source table
Partially zIIP eligible

Step C
Text Analysis and Training
Full zIIP eligible

Step D
Load the vector table

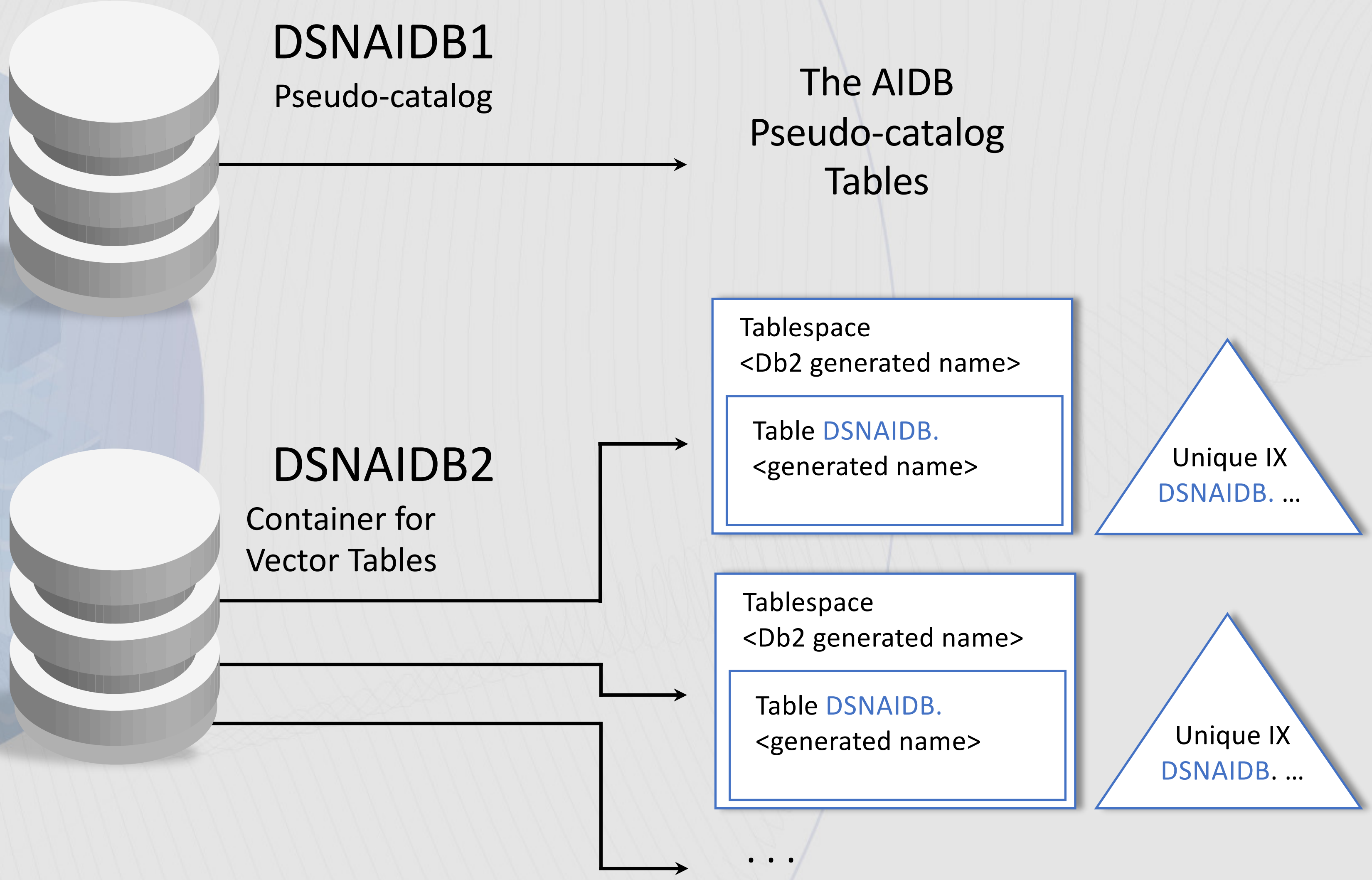
- Runs on z/OS where Db2 SQL Data Insights is installed
- Leverages an imbedded Spark instance
- Interface Db2 through JDBC T4 and stored procedures
- Local or remote loads via zLoad

Training Performance using Freddie Mac Loan Performance Data



Details will be published as a part of Db2 13 Performance Topics (redbook)

Db2 database design



Two Databases

- One for “catalog”
- One for model tables

Pseudo-catalog

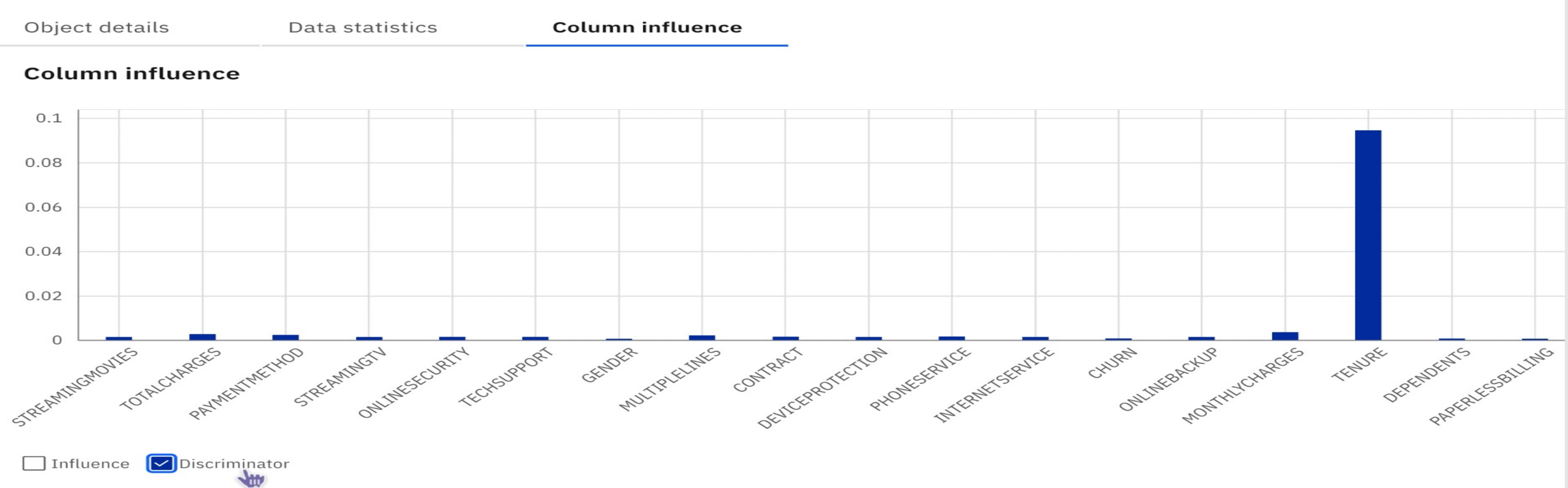
- Metadata tables for model tables
- Not for regular user access

Model tables

- Created by user via Admin UI through Db2 stored procedures
- Table space, table, indexes are given generated names
- Storage and buffer pool attributes inherited from the database

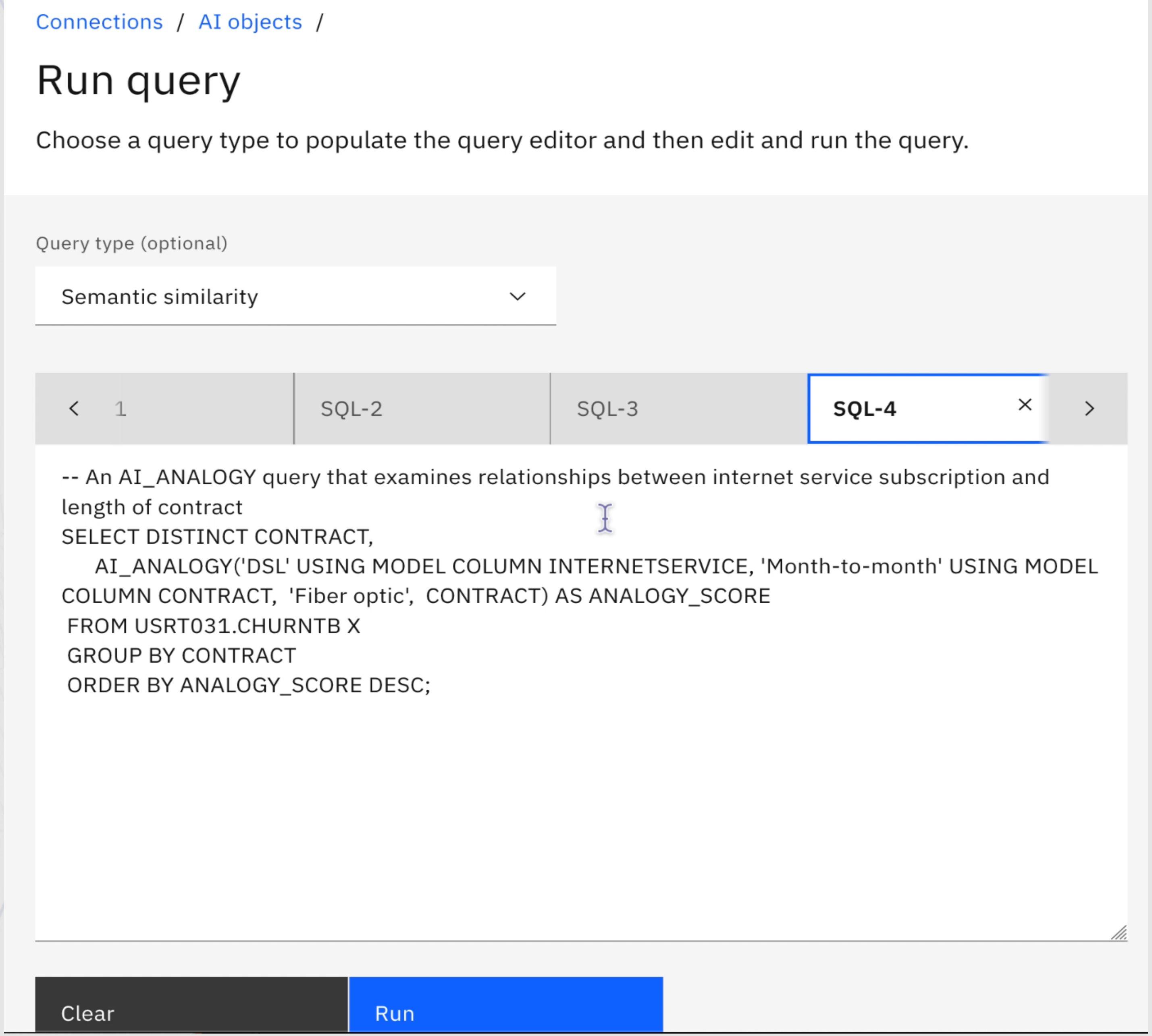
Step 4: Analyze Data

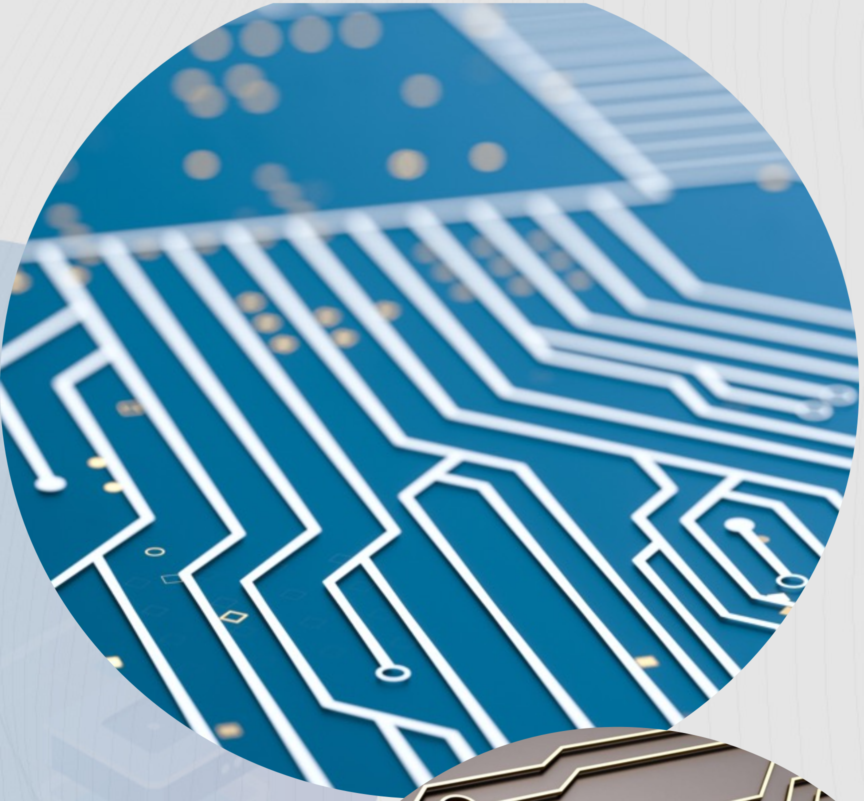
- Influence metrics : influence of a particular column on the training of a model.
  - The influence score for every column is computed as the ratio of NULL and user-specified empty values to the total number of values. The fewer the empty values a column has, the higher its influence score becomes.
- Discriminatory metrics : captures the value distribution of each column in the associated table.
  - The discriminatory score measures the ability of a column (the values in a column) to distinguish its co-occurring entries in rows. The more the unique values a column has, the higher its discriminatory score becomes. The unique primary key column contains unique values only, and its discriminatory score is the highest.



Step 5: Run Queries

Ready to run AI semantic queries





Summary and Future

Summary

SQL Data Insights offers new ways of looking at existing data stored in mainframe.

Utilize existing mainframe data for in place business analytics without going through complex model build process

Sponsor user program is available for Db2 12 to exploit your data!

SQL Data Insights Semantic Queries Beyond Db2 13 GA Level

Cognitive Intelligence Query	Functional Classification	Functional Description	Db2 BIF
semantic **similarity and dissimilarities**	**Entity Matching Recommendation**	• **Matching** rows/entities based on overall meaning (similarity/dissimilarity) • **Suggest** choices for incorrect or missing entities	AI_SIMILARITY
semantic **Clustering**	**Recommendation**	• **Find** entities/rows based on relationships between attributes in a given set • Example: Find animals similar to (lion, tiger, panther)	AI_SEMANTIC_CLUSTER
Reasoning Analogy	**Recommendation**	• **Find** entities/rows based on relationships between attributes • Example: Newspaper:Press :: Cloth:?	AI_ANALOGY
semantic **grouping**	**Entity Matching**	• **Collate** semantically-related entities	AI_GROUPING
profile queries	**Identify Hidden Relationships**	• Given a relational entity of a type, identify entities of other types that are semantically related to the relational entity.	AI_PROFILE
predictive queries	**Prediction over unseen data**	• **Predict** values of unknown attributes • **Predict** values of missing/incorrect attributes	AI_PREDICT

Thank You!

Akiko@us.ibm.com

IDUG

2022 EMEA **Db2** Tech Conference

IDUG

Now You See It, Unveil New Insights Through SQL Data Insights

Please fill out your session evaluation!