ICDT1202Y Database Systems

Lecture 3. SQL DDL

Anisah Ghoorah
a.ghoorah@uom.ac.mu
Dept. Digital Technologies
Faculty of ICDT
University of Mauritius

Acknowledgment and Reading

- **Chapter 4** in Elmasri, R. and Navathe, S. B. (2010) Fundamental of Database Systems, **6th edition**, Pearson.

- **Chapter 7** in Connolly, T. and Begg, C. (2014) Database Systems – A practical approach to design, implementation, and management, **6th edition**, Addison-Wesley.

Learning Outcomes

- Create databases and their tables specifying appropriate data types and constraints using SQL
  - Distinguish between DDL, DML and DCL statements
  - Use DDL statements to create,alter and drop a database, its tables and other database objects and specify data types and constraints
  - Explain the importance of table index

Content

- SQL Identifiers
- Data types supported by SQL
- How to define integrity constraints using SQL
- CREATE TYPE
- CREATE RULE
- CREATE TABLE, ALTER TABLE and DROP TABLE
- CREATE SCHEMA, DROP SCHEMA
- CREATE INDEX, DROP INDEX

Database Language

Ideally, database language should allow user to:

- create the database and relation structures
- perform insertion, modification, deletion of data from relations;
- perform simple and complex queries

Structured Query Language - SQL

- SQL has 2 major components:
  - A **DDL** for defining database structure
  - A **DML** for retrieving and updating data
- SQL **DCL** for granting and revoking permissions
- SQL is relatively easy to learn:
  - It is non-procedural – you specify *what* information you require, rather than *how* to get it
  - It is essentially free-format

SQL

Consists of standard English words:

```
CREATE TABLE Staff( staffNo VARCHAR(5),
                    lName   VARCHAR(15),
                    salary  DECIMAL(7,2) );
```

```
INSERT INTO Staff VALUES
    ('SG16', 'Brown', 8300);
```

```
SELECT staffNo, lName, salary
FROM Staff
WHERE salary > 10000;
```

Writing SQL Commands

- SQL statement consists of :
  - **Reserved words** are a fixed part of SQL and must be spelt exactly as required and cannot be split across lines
  - **User-defined words** are made up by user and represent names of various database objects such as relations, columns and views
- Most components of an SQL statement are ***case insensitive***, except for literal character data
- More readable with indentation and lineation

Data Definition Language

- SQL DDL allows database objects such as tables, and views to be created and destroyed
- Examples of main SQL DDL statements are:
  - CREATE DATABASE
  - CREATE TYPE
  - CREATE RULE
  - CREATE TABLE, ALTER TABLE, DROP TABLE
  - CREATE SCHEMA, DROP SCHEMA
  - CREATE INDEX, DROP INDEX
  - CREATE VIEW, DROP VIEW
    - Will be covered in another lecture

Case Study: Dreamhome

Branch

branchNo	street	city	postcode
B005	22 Deer		
B007	16 Argy		
B003	163 Ma		
B004	32 Man		
B002	56 Clov		

Client

clientNo	fName	lName	telNo	prefType	maxRent
CR76	John	Kay	0207-774-5632	Flat	425
			3-1825	Flat	350
			92178	House	750
			96720	Flat	600

Staff

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manag				
SG37	Ann	Beech	Assista				
SG14	David	Ford	Supery				

PrivateOwner

ownerNo	fName	lName	address	telNo
CO46	Joe	Keogh	2 Fergus Dr, Aberdeen AB2 7SX	01224-861212
CO87	Carol	Farrel	6 Achray St, Glasgow G32 9DX	0141-357-7419
CO40	Tina	Murphy	63 Well St, Glasgow G42	0141-943-1728
		Shaw	12 Park Pl, Glasgow G4 0QR	0141-225-7025

Viewing

clientNo	propertyNo	viewDate	comment
CR56	PA14	24-May-04	too small
CR76	PG4		
CR56	PG4		
CR62	PA14		
CR56	PG36		

Registration

clientNo	branchNo	staffNo	dateJoined
CR76	B005	SL41	
CR56	B003	SG37	
CR74	B003	SG37	
CR62	B007	SA9	

PropertyForRent

propertyNo	street	city	postcode	type	rooms	rent	ownerNo	staffNo	branchNo
PA14	16 Holhead	Aberdeen	AB7 5SU	House	6	650	CO46	SA9	B007
PL94	6 Argyll St	London	NW2	Flat	4	400	CO87	SL41	B005
PG4	6 Lawrence St	Glasgow	G11 9QX	Flat	3	350	CO40		B003
PG36	2 Manor Rd	Glasgow	G32 4QX	Flat	3	375	CO93	SG37	B003
PG21	18 Dale Rd	Glasgow	G12	House	5	600	CO87	SG37	B003
PG16	5 Novar Dr	Glasgow	G12 9AX	Flat	4	450	CO93	SG14	B003

Example of CREATE TABLE

```
CREATE DATABASE dreamhome;


CREATE TABLE Branch (
 branchNo   char(4),
 street     varchar(25),
 city       varchar(15),
 postcode   varchar(8),
 PRIMARY KEY (branchNo)
);
```

SQL Identifiers

- Identify objects in the database
- E.g. table names, view names, and columns
- Characters that can be used in a user-defined SQL identifier must appear in a **character set**
- Default character set consists of the upper-case letters A...Z, the lower-case letters a...z, the digits 0...9, and the underscore (_) character
- Restrictions on an SQL identifier
  - An identifier must start with a letter, underscore (_), at sign (@), or number sign (#)
  - An identifier cannot contain spaces
  - https://learn.microsoft.com/en-us/sql/relational-databases/databases/database-identifiers?view=sql-server-ver16#rules-for-regular-identifiers

SQL Data type

- While creating table, for each column, we specify a data type, i.e. what kind of data we want to store.
- Examples
  - Numeric (smallint, int, bigint, decimal, numeric, …)
  - Approximate numeric (float, real)
  - Character (char, varchar, text)
  - Date, Time (date, time, datetime, …)
- For full list, visit
  - https://learn.microsoft.com/en-us/sql/t-sql/language-reference?view=sql-server-ver16
  - https://learn.microsoft.com/en-us/sql/t-sql/data-types/data-types-transact-sql?view=sql-server-ver16

Numeric

Name	Description	Storage Size	Range
smallint	Stores whole numbers, small range.	2 bytes	-32768 to +32767
integer	Stores whole numbers.Use this when you want to store typical integers.	4 bytes	-2147483648 to +2147483647
bigint	Stores whole numbers, large range.	8 bytes	-9223372036854775808 to 9223372036854775807
decimal	user-specified precision, exact	variable	up to 131072 digits before the decimal point; up to 16383 digits after the decimal point.
numeric	user-specified precision, exact	variable	up to 131072 digits before the decimal point; up to 16383 digits after the decimal point.
real	variable-precision, inexact	4 bytes	6 decimal digits precision.
double precision	variable-precision, inexact	8 bytes	15 decimal digits precision

2^{15}

2^{31}

2^{63}

Numeric (contd)

```
SELECT 100 * 0.08875 ;          -- 8.87500
SELECT 100 * CAST(0.08875 AS DECIMAL(7,2)) ;          -- 9.00
SELECT CAST( (100 * 0.08875) AS DECIMAL(7,2))   --8.88
```

Character

Name	Description
character varying(n), varchar(n)	variable-length with limit
character(n), char(n)	fixed-length, blank padded
text	variable unlimited length

- char : stores a single (exactly 1) character.
- char(n) : stores exactly n characters in length.
- varchar(n) : stores strings up to n characters in length. Without specified length it stores strings of any size.

Identity (serial)

- Identity columns can be used for generating key values. The identity property on a column guarantees the following conditions:
  - Each new value is generated based on the current seed and increment.
  - Each new value for a particular transaction is different from other concurrent transactions on the table.
  - E.g. An automatically incrementing identification number.

```
CREATE TABLE new_employees (
    id_num INT IDENTITY(1, 1),
    fname VARCHAR(20),
    minit CHAR(1),
    lname VARCHAR(30)
);
```

Bit

- The **bit** data type can be used to store Boolean values.
- The string values `TRUE` and `FALSE` can be converted to **bit** values:
  - `TRUE` is converted to `1`
  - `FALSE` is converted to `0`.
- E.g.
  - ```
    INSERT INTO dbo.test(id, absence) VALUES
    (123, 'True')
    ```
    ```

Date

Syntax	`DATE`

Usage	`DECLARE @MyDate DATE`
	
	`CREATE TABLE Table1 (Column1 DATE)`

Default string literal format (used for down-level client)	`yyyy-MM-dd` For more information, see the Backward compatibility for down-level clients section.
Range	`0001-01-01` through `9999-12-31` (`1582-10-15` through `9999-12-31` for Informatica) January 1, 1 CE (Common Era) through December 31, 9999 CE (October 15, 1582 CE through December 31, 9999 CE for Informatica)
Element ranges	`yyyy` is four digits from `0001` to `9999` that represent a year. Informatica limits `yyyy` to the range `1582` to `9999`. `MM` is two digits from `01` to `12` that represent a month in the specified year. `dd` is two digits from `01` to `31`, depending on the month, which represents a day of the specified month.

Examples of attributes and their data type

- branchNo CHAR(4)
- address VARCHAR(30)
- rooms SMALLINT
- salary DECIMAL(7,2)
- viewDate DATE

User-defined data type

- To create a user-defined data type alias

```
CREATE TYPE ssn
FROM varchar(11) NOT NULL ;
```

User-defined data type

- To create a user-defined data type alias **with a constraint**
- https://learn.microsoft.com/en-us/sql/t-sql/statements/create-rule-transact-sql?view=sql-server-ver16

```
CREATE TYPE SexType
FROM CHAR(1) ;

CREATE RULE SexTypeRule AS @list IN ('M', 'F') ;

EXEC SP_BINDRULE 'SexTypeRule', 'SexType';
```

Integrity Enhancement Feature (IEF)

- Consider five types of integrity constraints
  - required data (NOT NULL)
  - domain constraints (data type + allowable values)
  - entity integrity (Primary Key)
  - referential integrity (Foreign Key)
  - general constraints

IEF – Required Data & Domain Constraints

Required Data

```
CREATE TABLE Staff (

position   VARCHAR(10)   NOT NULL
```

Column Name — Data Type — Required Data

Domain Constraints using CHECK

```
CREATE TABLE Staff (

sex   CHAR NOT NULL CHECK (sex IN ('M', 'F'))
```

Column Name — Data Type — Required Data — Constraint

IEF – Entity Integrity

- Primary key of a table must contain a unique, non-null value for each row
- Entity integrity is supported with the PRIMARY KEY clause in the CREATE and ALTER TABLE statements

```
CREATE TABLE Staff (
PRIMARY KEY (staffNo)
```

```
CREATE TABLE Viewing (
PRIMARY KEY (clientNo, propertyNo)
```

- Ensure uniqueness for alternate keys

```
UNIQUE (clientNo, propertyNo)
```

IEF - Referential Integrity

- Referential integrity means that, if the foreign key (FK) contains a value, that value must refer to an existing row in the parent table
- FKs are supported with FOREIGN KEY clause in CREATE and ALTER TABLE

```
CREATE TABLE Staff (
FOREIGN KEY(branchNo) REFERENCES Branch
```

- SQL rejects any INSERT/UPDATE operation that attempts to create FK value in a child table without a matching candidate key (CK) value in parent

IEF - Referential Integrity (2)

- Action SQL takes for any UPDATE/DELETE operation that attempts to update/delete a CK value in parent table that has some matching rows in the child table, is dependent on the referential action specified using ON UPDATE and ON DELETE subclauses of the FOREIGN KEY clause
- When the user attempts to delete a row from a parent table, and there are one or more matching rows in the child table, SQL supports four options regarding the action to be taken:
  - CASCADE, SET NULL, SET DEFAULT, NO ACTION

IEF - Referential Integrity (3)

- CASCADE
  - Delete row from parent and delete matching rows in child, and so on in cascading manner
- SET NULL
  - Delete row from parent and set FK column(s) in child to NULL. Only valid if FK columns are NOT NULL
- SET DEFAULT
  - Delete row from parent and set each component of FK in child to specified default. Only valid if DEFAULT specified for FK columns
- NO ACTION
  - Reject delete from parent. Default

IEF - Referential Integrity (4)

- E.g. In the PropertyForRent table:

CREATE TABLE PropertyForRent (

```
FOREIGN KEY (staffNo) REFERENCES Staff
                        ON DELETE SET NULL
```

```
FOREIGN KEY (ownerNo) REFERENCES Owner
                        ON UPDATE CASCADE
```

IEF - General Constraints

- E.g. To define the general constraint that prevents a member of staff from managing more than 100 properties at the same time (table PropertyForRent)

```
staffNo varchar(5) CONSTRAINT StaffNotHandlingTooMuch
          CHECK (NOT EXISTS (SELECT staffNo
                             FROM PropertyForRent
                             GROUP BY staffNo
                             HAVING COUNT(*) > 100
                             )
            )
```

CREATE TABLE -- Syntax

- Use extended form of Backus Naur Form (BNF) notation:
  - Upper-case letters represent reserved words
  - Lower-case letters represent user-defined words
  - Vertical bar (|) indicates a *choice* among alternatives
  - Curly braces indicate a *required element*
  - Square brackets indicate an *optional element*
  - … indicates *optional repetition* (0 or more)

CREATE TABLE -- Syntax

```
CREATE TABLE TableName
    {(colName dataType [NOT NULL] [UNIQUE]
    [DEFAULT defaultValue][CHECK searchCondition] [,...]}
    [PRIMARY KEY (listOfColumns),]
    {[UNIQUE (listOfColumns),] […,]}
    {[FOREIGN KEY (listOfFKColumns)
          REFERENCES ParentTableName [(listOfCKColumns)]
          [ON UPDATE referentialAction]
          [ON DELETE referentialAction ]] [,…]}
       {[CHECK (searchCondition)] [,…] })
```

CREATE TABLE (2)

- Creates a table with one or more columns of the specified *dataType*
- With NOT NULL, system rejects any attempt to insert a null in the column
- Can specify a DEFAULT value for the column
- Primary keys should always be specified as NOT NULL
- FOREIGN KEY clause specifies FK along with the referential action

Example – CREATE TABLE

- Create any types

```
CREATE TYPE OwnerNumber FROM VARCHAR(5);

CREATE TYPE StaffNumber FROM VARCHAR(5);

CREATE TYPE BranchNumber FROM CHAR(4);

CREATE TYPE PropertyNumber FROM VARCHAR(5);

CREATE TYPE PropertyType FROM VARCHAR(5);

CREATE TYPE PropertyRooms FROM SMALLINT ;

CREATE TYPE PropertyRent FROM DECIMAL(6,2)  ;
```

Example – CREATE TABLE

- Create any rules

```
CREATE RULE PropertyTypeRule
AS @list IN ('Flat', 'House');

CREATE RULE PropertyRoomsRule
AS @range>=1 AND @range<5;

CREATE RULE PropertyRentRule
AS @range>=100 AND @range<10000;
```

Example – CREATE TABLE

- Bind rules to types using SP_BINDRULE

```
EXEC SP_BINDRULE 'PropertyTypeRule', 'PropertyType'

EXEC SP_BINDRULE 'PropertyRoomsRule', 'PropertyRooms'

EXEC SP_BINDRULE 'PropertyRentRule', 'PropertyRent'
```

Example: CREATE TABLE

```
CREATE TABLE PropertyForRent(
propertyNo PropertyNumber   NOT NULL,
street     VARCHAR(25)      NOT NULL,
city       VARCHAR(15)      NOT NULL,
postcode   VARCHAR(8),
type       PropertyType     NOT NULL DEFAULT 'Flat',
rooms      PropertyRooms    NOT NULL DEFAULT 4,
rent       PropertyRent     NOT NULL DEFAULT 600,
ownerNo    OwnerNumber      NOT NULL,
staffNo    StaffNumber,
branchNo   BranchNumber     NOT NULL,
PRIMARY KEY (propertyNo),
FOREIGN KEY (staffNo) REFERENCES Staff
                          ON DELETE SET NULL
                          ON UPDATE CASCADE,
FOREIGN KEY (ownerNo) REFERENCES PrivateOwner
                          ON DELETE NO ACTION
                          ON UPDATE CASCADE,
FOREIGN KEY (branchNo) REFERENCES Branch
                          ON DELETE NO ACTION
                          ON UPDATE CASCADE
);
```

More examples on CREATE TABLE

- https://learn.microsoft.com/en-us/sql/t-sql/statements/create-table-transact-sql?view=sql-server-ver16

ALTER TABLE

- Changing a table definition
  - Add a new column to a table
  - Drop a column from a table
  - Add a new table constraint
  - Drop a table constraint
  - Set a default for a column
  - Drop a default for a column

Example:ALTER TABLE – Add/remove column

- Examples

```
ALTER TABLE distributor
  ADD address varchar(30);
```

```
ALTER TABLE distributor
 DROP column address;
```

Example: ALTER TABLE - Add column

E.g add column with constraint

```
ALTER TABLE distributor
  ADD nid VARCHAR(20) NULL
    CONSTRAINT nid_unique UNIQUE ;
```

Example: ALTER TABLE - Add constraint

E.g. adds a constraint to an existing column in the table. The column has a value that violates the constraint. Therefore, `WITH NOCHECK` is used to prevent the constraint from being validated **against existing rows,** and to allow for the constraint to be added.

```
ALTER TABLE distributor WITH NOCHECK
 ADD CONSTRAINT qty_check
     CHECK (qty > 5) ;
```

Example: ALTER TABLE – add default

E.g. Add a DEFAULT constraint to an existing column

```
ALTER TABLE distributor
 ADD CONSTRAINT col_b_def
 DEFAULT 50 FOR column_b ;
```

Example:ALTER TABLE - Add/drop FK

```
CREATE TABLE ContactBackup
   (ContactID INT) ;


ALTER TABLE ContactBackup
ADD CONSTRAINT FK_ContactBackup_Contact FOREIGN KEY (ContactID)
   REFERENCES Person (BusinessEntityID) ;


ALTER TABLE ContactBackup
  DROP CONSTRAINT FK_ContactBackup_Contact ;
```

More examples on ALTER TABLE

- https://learn.microsoft.com/en-us/sql/t-sql/statements/alter-table-transact-sql?view=sql-server-ver16#add

Rename a column

```
EXEC sp_rename 'tablename.colname', newcolcname, 'COLUMN';
```

Refer to

https://learn.microsoft.com/en-us/sql/relational-databases/tables/rename-columns-database-engine?view=sql-server-ver16

DROP TABLE

```
DROP TABLE TableName
```

- DROP TABLE cannot be used to drop a table that is referenced by a FOREIGN KEY constraint. The referencing FOREIGN KEY constraint or the referencing table must first be dropped. If both the referencing table and the table that holds the primary key are being dropped in the same DROP TABLE statement, the referencing table must be listed first.
- Multiple tables can be dropped in any database. If a table being dropped references the primary key of another table that is also being dropped, the referencing table with the foreign key must be listed before the table holding the primary key that is being referenced.
- When a table is dropped, rules or defaults on the table lose their binding, and any constraints or triggers associated with the table are automatically dropped. If you re-create a table, you must rebind the appropriate rules and defaults, re-create any triggers, and add all required constraints.
- Example

```
DROP TABLE PropertyForRent;
```

More examples on DROP TABLE

- https://learn.microsoft.com/en-us/sql/t-sql/statements/drop-table-transact-sql?view=sql-server-ver16

Schema

- Why? To group together tables and other database objects that belong to the same database application
- A **schema** is identified by a **schema name**, and includes an **authorization identifier** to indicate the user or account who owns the schema, as well as **descriptors** for each element in the schema
- Schema elements include tables, constraints, views, domains, and other constructs (such as authorization grants) that describe the schema
- The privilege to create schemas, tables, and other constructs must be explicitly granted to the relevant user accounts by the system administrator or DBA

Create and Drop Schema

Syntax

```
CREATE SCHEMA [Name | AUTHORIZATION CreatorIdentifier]

DROP SCHEMA Name
```

Example

```
CREATE SCHEMA Sales;

CREATE TABLE Sales.Region  (
  Region_id INT NOT NULL,
  Region_Name CHAR(5) NOT NULL
) ;
```

dbo schema

- The dbo schema is the default schema of every database.
- By default, users created with the CREATE USER Transact-SQL command have dbo as their default schema.
- The dbo schema is owned by the dbo user account.
- When database objects are referenced by using a one-part name, SQL Server first looks in the user's default schema. If the object is not found there, SQL Server looks next in the dbo schema. If the object is not in the dbo schema, an error is returned.

More examples on CREATE SCHEMA

- https://learn.microsoft.com/en-us/sql/t-sql/statements/create-schema-transact-sql?view=sql-server-ver16
- https://learn.microsoft.com/en-us/sql/relational-databases/security/authentication-access/ownership-and-user-schema-separation?view=sql-server-ver16

Index

- Why? An index is a structure that provides **accelerated access** to the rows of a table based on the values of one or more columns
- The presence of an index can significantly improve the performance of a query.
- However, since indexes may be updated by the system every time the underlying tables are updated, **additional overheads** may be incurred.
- Indexes are usually created to satisfy particular **search criteria** after the table has been in use for some time and has grown in size.

Create and Drop Index

Syntax

```
CREATE INDEX IndexName ON schema1.table1 (column1);

DROP INDEX IndexName ;
```

Example

```
CREATE INDEX StaffNoInd ON Staff (staffNo);
```

Index name

Table name

Column name

More examples on CREATE INDEX

- https://learn.microsoft.com/en-us/sql/t-sql/statements/create-index-transact-sql?view=sql-server-ver16

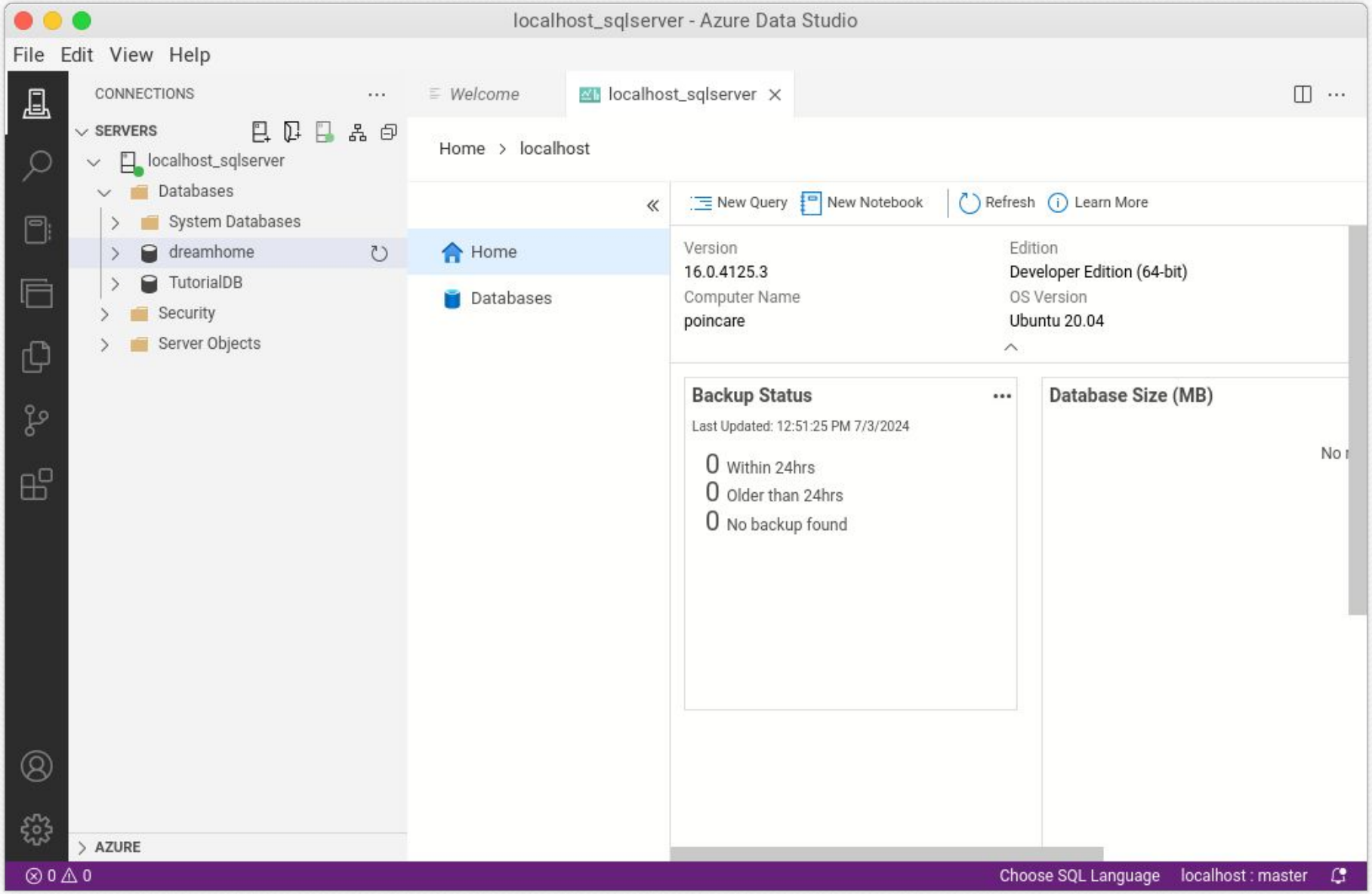
To install on your workstation

- Install SQL Sever 2022
  - Support available for Windows, Linux
  - https://www.microsoft.com/en-us/sql-server/sql-server-downloads
- Install Azure Data Studio
  - Support available for Windows, Linux
  - https://learn.microsoft.com/en-us/azure-data-studio/download-azure-data-studio?tabs=win-install%2Cwin-user-install%2Credhat-install%2Cwindows-uninstall%2Credhat-uninstall

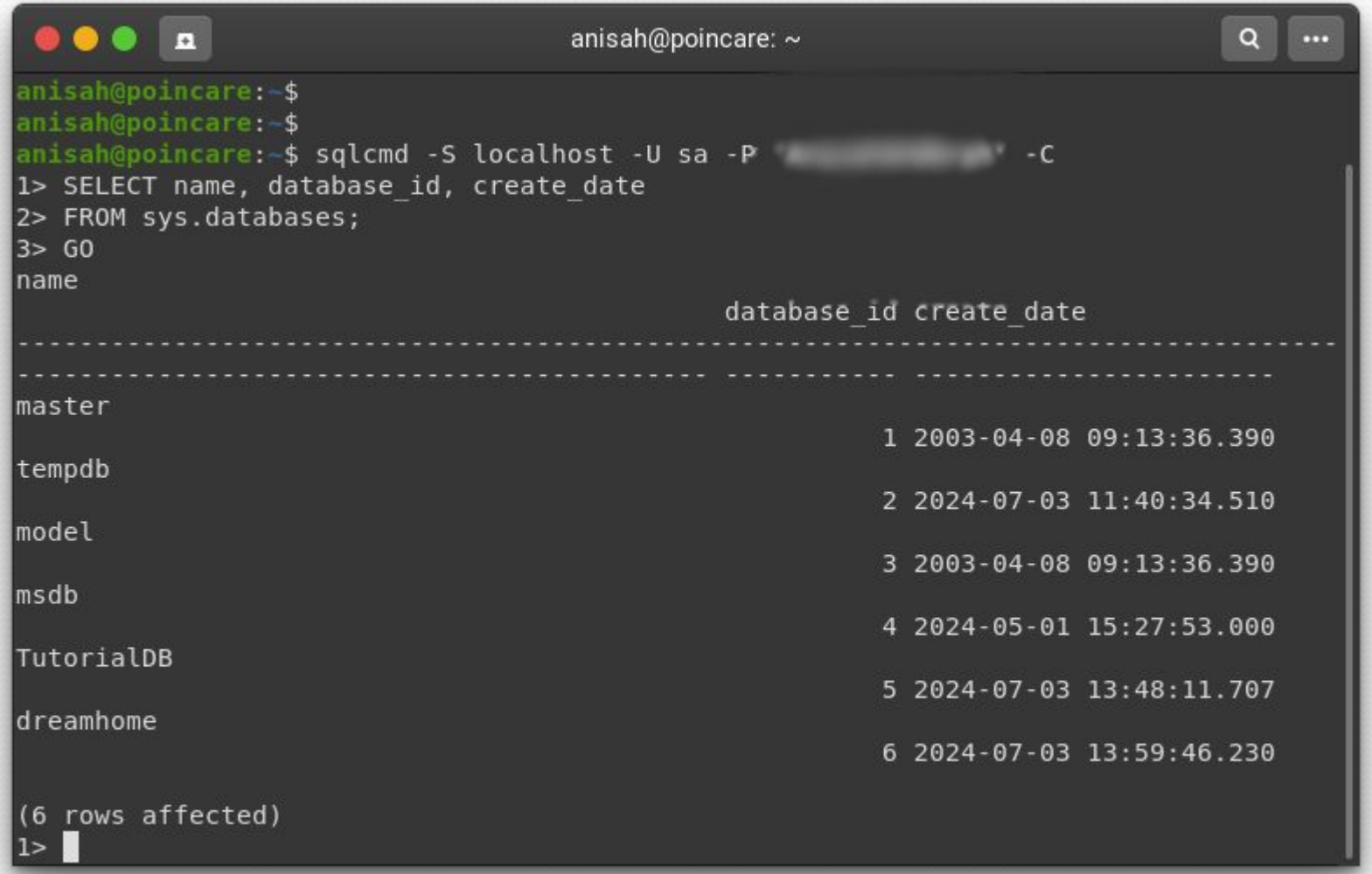
AZURE Data Studio **versus** SQL Server Management Studio (SSMS)

- https://learn.microsoft.com/en-us/azure-data-studio/what-is-azure-data-studio

AZURE Data Studio



SQL Server


anisah@poincare: ~
anisah@poincare:~$
anisah@poincare:~$
anisah@poincare:~$ sqlcmd -S localhost -U sa -P '[redacted]' -C
1> SELECT name, database_id, create_date
2> FROM sys.databases;
3> GO
name
database_id create_date
master
1 2003-04-08 09:13:36.390
tempdb
2 2024-07-03 11:40:34.510
model
3 2003-04-08 09:13:36.390
msdb
4 2024-05-01 15:27:53.000
TutorialDB
5 2024-07-03 13:48:11.707
dreamhome
6 2024-07-03 13:59:46.230
(6 rows affected)
1>

Review Questions

- Attempt exercises posted on Google Classroom