SQL

SQL has several sub-languages:

- Meta-data definition language (CREATE TABLE)
- Mata-data update language (ALTER TABLE)
- Data update language (INSERT, UPDATE, DELETE)
- Query language (SELECT)

1. Meta-data languages manage schema
2. Data languages manipulate set of tuples
3. Query languages are based on relational algebra

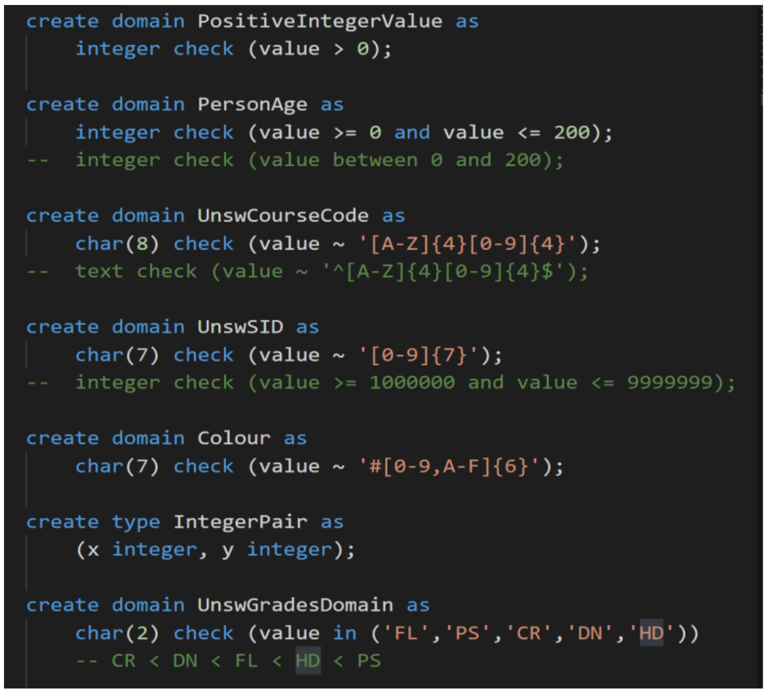
SQL SYNTAX

- Single-quotes are used for strings
- Double-quotes are used for “non-standard” identifiers
- Identifiers are case-insensitive (unless “double quoted”)
    - (Staff = staff = STAFF -> “STAFF” != “Staff” != “staff”

TYPES/ CONSTANTS IN SQL

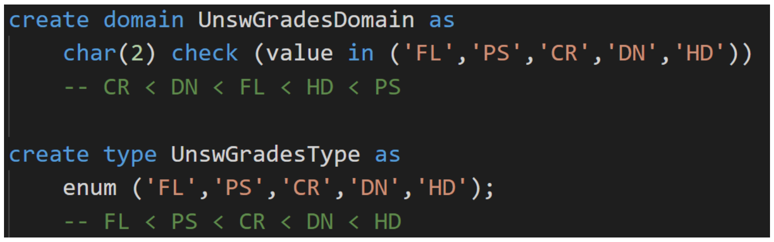
- Numeric types: INTEGER, REAL, NUMERIC
- String types: CHAR(n), VARCHAR(n), TEXT
- Logical type: BOOLEAN, TRUE, FALSE
- Time-related types: DATE, TIME, TIMESTAMP, INTERVAL
    - Date: 2008-04-13
    - Time: 13:30:15
    - Timestamp: 2008-04-13 13:30:15
    - Interval: 10 days
- Users can define their own types in several ways:
    - Domains: constrained version of existing type
- CREATE DOMAIN name AS type CHECK (constraint)
    - Tuple types: defined for each table
- CREATE TYPE name AS (attriName, attriType, …)
    - Enumerated type: specify elements and ordering
- CREATE TYPE name AS ENUM (‘label’, …)

EXERCISE: DEFINING DOMAINS



```
create domain PositiveIntegerValue as
    integer check (value > 0);

create domain PersonAge as
    integer check (value >= 0 and value <= 200);
--  integer check (value between 0 and 200);

create domain UnswCourseCode as
    char(8) check (value ~ '[A-Z]{4}[0-9]{4}');
--  text check (value ~ '^[A-Z]{4}[0-9]{4}$');

create domain UnswSID as
    char(7) check (value ~ '[0-9]{7}');
--  integer check (value >= 1000000 and value <= 9999999);

create domain Colour as
    char(7) check (value ~ '#[0-9,A-F]{6}');

create type IntegerPair as
    (x integer, y integer);

create domain UnswGradesDomain as
    char(2) check (value in ('FL','PS','CR','DN','HD'))
    -- CR < DN < FL < HD < PS
```

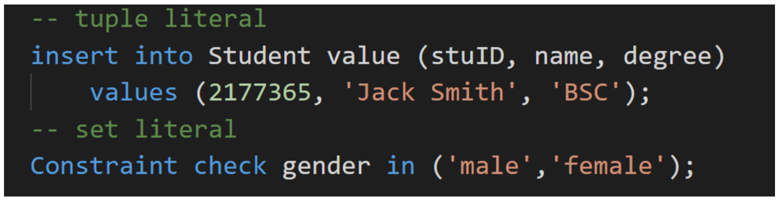
EXERCISE: ENUERATED TYPES



```
create domain UnswGradesDomain as
    char(2) check (value in ('FL','PS','CR','DN','HD'))
    -- CR < DN < FL < HD < PS

create type UnswGradesType as
    enum ('FL','PS','CR','DN','HD');
    -- FL < PS < CR < DN < HD
```

TUPLE AND SET LITERALS

- Tuple and set constants are both written as (val1, val2, val3)
- The correct interpretation in worked out from the context



```
-- tuple literal
insert into Student value (stuID, name, degree)
    values (2177365, 'Jack Smith', 'BSC');
-- set literal
Constraint check gender in ('male','female');
```

SQL OPERATOR

`name LIKE 'Ja%'`	name begins with 'Ja'	`name ~ '^Ja'`	name begins with 'Ja'
`name LIKE '_i%'`	name has 'i' as 2nd letter	`name ~ '^.i'`	name has 'i' as 2nd letter
`name LIKE '%o%o%'`	name contains two 'o's	`name ~ '.*o.*o.*'`	name contains two 'o's
`name LIKE '%ith'`	name ends with 'ith'	`name ~ 'ith$'`	name ends with 'ith'
`name LIKE 'John'`	name equals 'John'	`name ~ 'John'`	name contains 'John'

- Count (attr) = numbers of rows in attr column
- Sum (attr) = sum of values for attr
- Avg (attr) = mean of values for attr
- Min/max(attr) = min/max of values for attr

VIEWS

- A view associates a name with a query:
    - CREATE VIEW viewName [(attributes)] AS query
- Each time the view in involved
    - The query is evaluated, yielding a set of tuples
    - The set of tuples is used as the value of the view