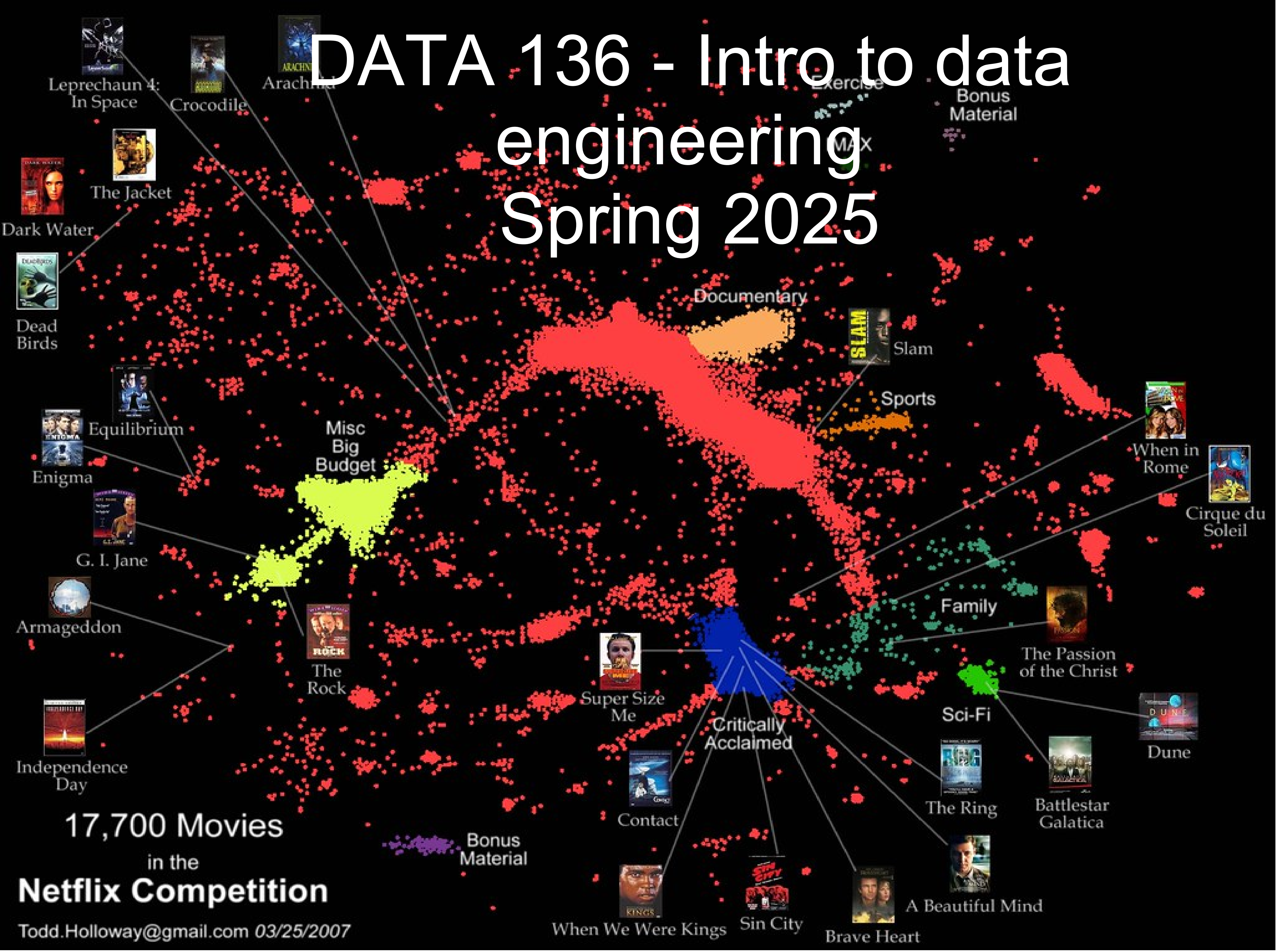
DATA 136 - Intro to data engineering Spring 2025



SQL has two essential parts:

DDL

CREATE
ALTER
DROP

DML

SELECT
INSERT
UPDATE
DELETE

DDL: Create Table Construct

An SQL “relation” is defined using the create table command:

CREATE TABLE r $(A_1\ D_1,\ A_2\ D_2,\ \ldots,\ A_n\ D_n)$ r is the name of the relation

each A_i is an attribute name in the schema of relation r D_i is the data type of values in the domain of attribute A_i

Example:

```
CREATE TABLE instructor (ID char(5), name varchar(20),
dept_name varchar(20),salary float)
```

```
INSERT INTO instructor values ('10212', 'Taylor', 'Biology',
66000);
```

```
INSERT INTO instructor values ('10213', 'Alice', 'CS', 96000);
```

```
INSERT INTO instructor values ('10214', 'Benny', 'Chemistry',
16000);
```

SQL data types...

char(n). Fixed length character string, with user-specified length *n.*
varchar(n). Variable length character strings, with user-specified

maximum length *n.*

int. Integer (a finite subset of the integers that is machine-dependent).

smallint. Small integer (a machine-dependent subset of the integer domain type).

numeric(p,d). Fixed point number, with user-specified precision of *p* digits, with *n* digits to the right of decimal point.

real, double precision. Floating point and double-precision floating point numbers, with machine-dependent precision.

float(n). Floating point number, with user-specified precision of at least *n* digits.

DDL: Domain Constraints

not null **null is a symbol for missing data that is treated special by the DBMS**

primary key (A_1, ..., A_n)

foreign key (A_m, ..., A_n) **references** *r*

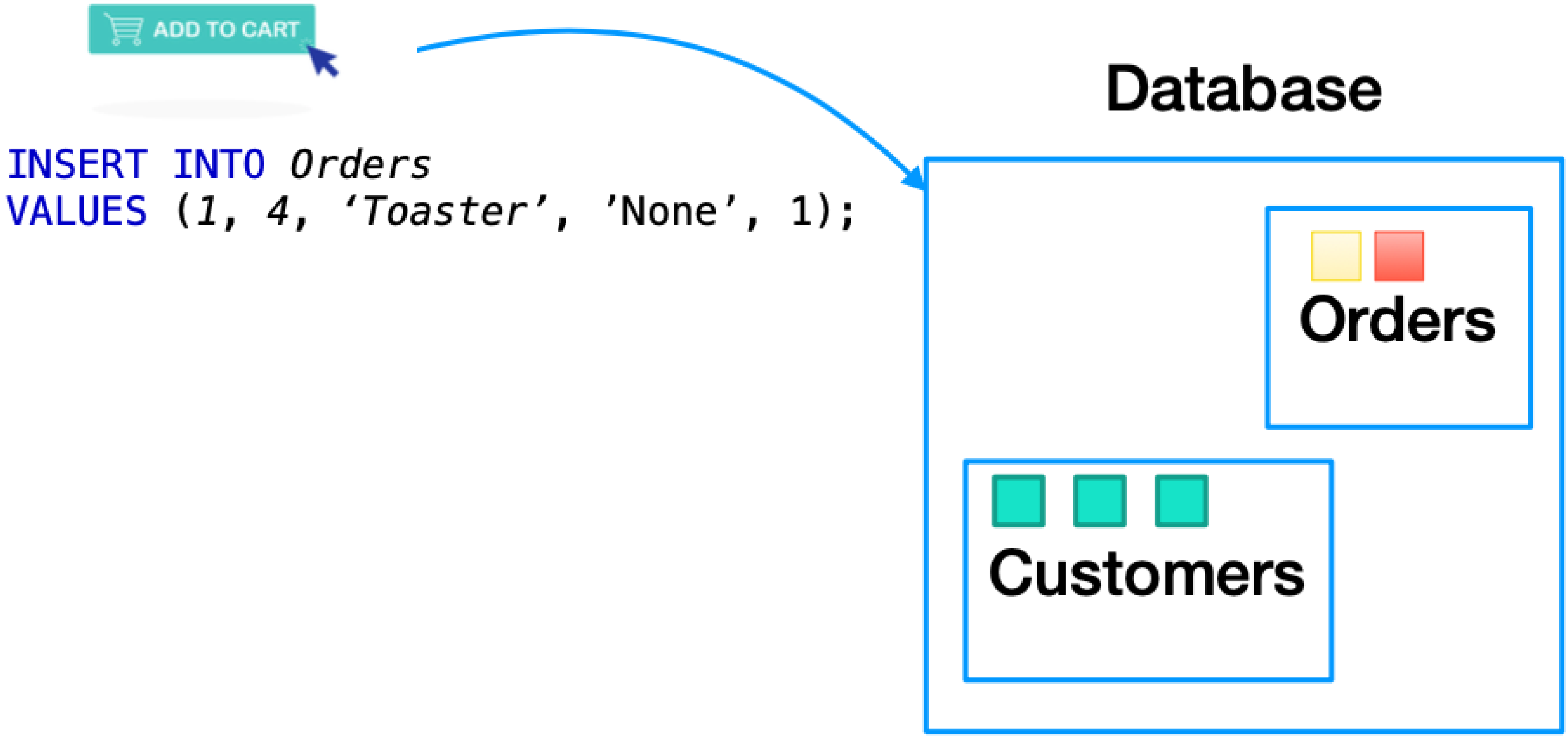
primary key declaration on an attribute automatically ensures **not null**

foreign key declaration on an attribute defines linking conditions

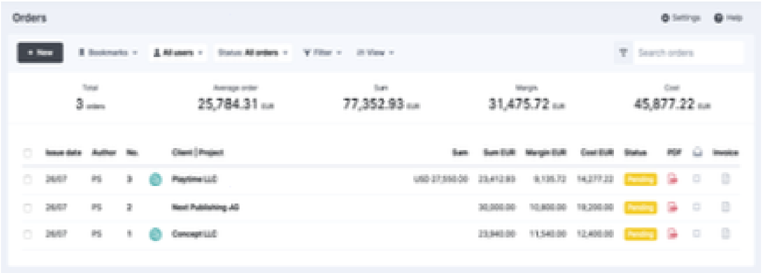
multiple primary key columns allowed.

CREATE TABLE With Constraints

```
CREATE TABLE student (
ID                    varchar(5),
name                  varchar(20) not null,
dept_name             varchar(20),
tot_cred              numeric(3,0),
primary key          (ID),
foreign key (dept_name) references department) );
```



```
INSERT INTO Orders
VALUES (1, 4, ‘Toaster’, ’None’, 1);
```



Orders

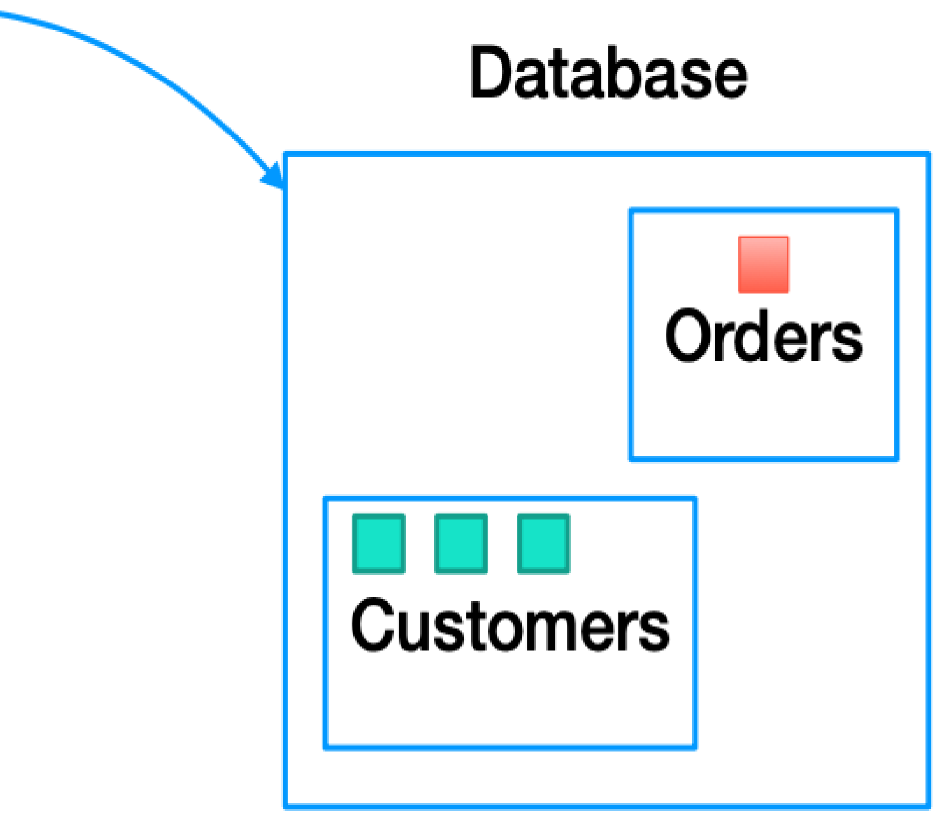
⚙ Settings ❓ Help

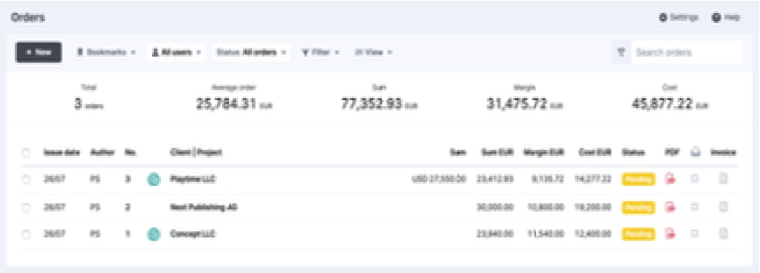
+ New | Bookmarks ▾ | All users ▾ | Status: All orders ▾ | Filter ▾ | View ▾ | Search orders

Total	Average order	Sum	Margin	Cost
3 orders	25,784.31 EUR	77,352.93 EUR	31,475.72 EUR	45,877.22 EUR

□	Issue date	Author	No.	Client | Project	Sum	Sum EUR	Margin EUR	Cost EUR	Status	PDF		Invoice
□	26/07	PS	3	Playtime LLC	USD 27,550.00	23,412.93	9,135.72	14,277.22	Pending		□	
□	26/07	PS	2	Next Publishing AG		30,000.00	10,800.00	19,200.00	Pending		□	
□	26/07	PS	1	Concept LLC		23,940.00	11,540.00	12,400.00	Pending		□	

```
Select *
FROM Orders, Customers
WHERE Orders.CustomerId = 5
```





Orders

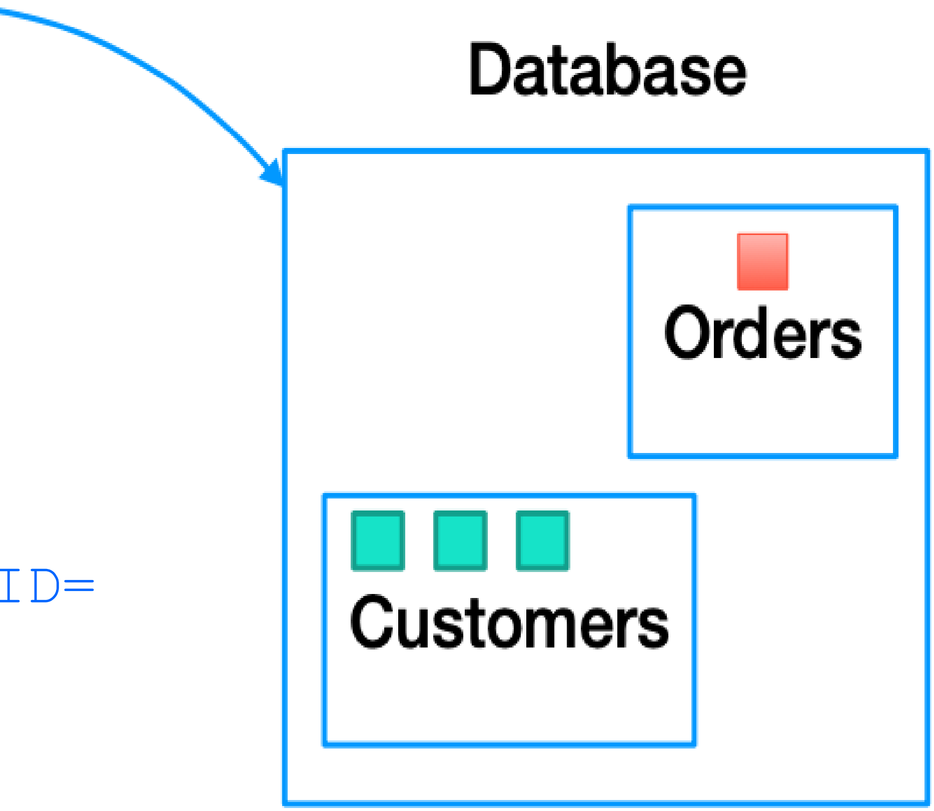
⚙ Settings ❓ Help

Total	Average order	Sum	Margin	Cost
3 orders	25,784.31 EUR	77,352.93 EUR	31,475.72 EUR	45,877.22 EUR

□	Issue date	Author	No.	Client | Project	Sum	Sum EUR	Margin EUR	Cost EUR	Status	PDF		Invoice
□	26/07	PS	3	Playtime LLC	USD 27,550.00	23,412.93	9,135.72	14,277.22	Pending		□	
□	26/07	PS	2	Next Publishing AG		30,000.00	10,800.00	19,200.00	Pending		□	
□	26/07	PS	1	Concept LLC		23,940.00	11,540.00	12,400.00	Pending		□	

```
Select *
FROM Orders, Customers
WHERE Orders.CustomerId = 5
```

```
WHERE Orders.CustomerID=
Customers.CustomerID
```



DML: Basic Query Structure

The SQL data-manipulation language (DML) provides the ability to query information, and insert, delete and update tuples

A typical SQL query has the form:

select A_1, A_2, ..., A_n **from** r_1, r_2, ..., r_m **where** P

A_i represents an attribute (cough, row)

R_i represents a relation (aka table)

P is a predicate (Boolean expression)

The result of an SQL query is a relation (table)

Give us some of the data; slice & dice, maybe add up, maybe join.

SELECT syntax... everything is optional...

```
SELECT <expression that
doens't depend on tables>
```

SELECT syntax... everything is optional...

```
SELECT <columns>
FROM <tables>
```

SELECT syntax... everything is optional...

```
SELECT <columns>
FROM <tables>
WHERE <conditions>
```

SELECT syntax

```
SELECT *
FROM enrolled, student
WHERE enrolled.sid =
student.sid
```

SELECT syntax with compound predicate

```
SELECT *

FROM enrolled, student

WHERE enrolled.sid =
student.sid AND

gpa >= 3.9
```

SELECT from JOIN

```
SELECT *
FROM enrolled JOIN student
ON enrolled.sid = student.sid
```

AGGREGATES

Functions that return a single value from a collection of tuples:

→ **AVG(col)**→ Return the average col value.

→ **MIN(col)**→ Return minimum col value.

→ **MAX(col)**→ Return maximum col value.

→ **SUM(col)**→ Return sum of values in col.

→ **COUNT(col)**→ Return # of values for col.

HAVING

Filters output results

Like a **WHERE** clause for a **GROUP BY**

```
SELECT AVG(s.gpa) AS avg_gpa, e.cid
FROM enrolled AS e, student AS s
WHERE e.sid=s.sid
GROUP BY e.cid
HAVING avg_gpa>3.9;
```

OUTPUT REDIRECTION

```
SELECT *
FROM (SELECT...) as enrolled, student
WHERE enrolled.sid = student.sid AND
gpa >= 3.9
```

OUTPUT CONTROL

ORDER BY <column*> [ASC|DESC]

→ Order the output tuples by the values in one or more of their columns.

```
SELECT FROM enrolled
WHERE cid = '15-721'
ORDER BY grade DESC, sid ASC
```

SQL order of operations...

- FROM (indicate table)
- WHERE (filter rows)
- GROUP BY (apply aggregates)
- HAVING (filter aggregates)
- SELECT (filter columns)
- ORDER BY (order rows)
- LIMIT/OFFSET (filter rows)

SQL syntactical order

- SELECT (filter columns)
- FROM (indicate table)
- WHERE (filter rows)
- GROUP BY (apply aggregates)
- HAVING (filter aggregates)
- ORDER BY (order rows)
- LIMIT/OFFSET (filter rows)

Databases to play around with:

- chinook.db: computer system for a record store. (What's a record store?)
- portal_mammals.sqlite: observations of creatures at an long-term ecological research site
- We can ask for all the rows, rows that have X property in Y column, rows that have Z aggregate property in Y column grouped by W, and, of course, we can join if we can specify the join conditions right.