CSE 344 Introduction to Data Management

Section 2: More SQL
Creating Tables

CREATE TABLE Population ( rank INTEGER, 
country VARCHAR(30) PRIMARY KEY, 
population INTEGER, 
percentage FLOAT );

CREATE TABLE GDP ( rank INTEGER, 
country VARCHAR(30) PRIMARY KEY, 
gdp INTEGER );

CREATE TABLE Airport ( code VARCHAR(30) PRIMARY KEY, 
name VARCHAR(30), 
country VARCHAR(30) );
importing files

.mode csv
.import ./population.csv Population
.import ./gdp.csv GDP
.import ./airport.csv Airport

# this will make it easier to see:
.headers ON
.mode column

# this will make it easier to see:
# fname   lname   years_dancing
# --------------- ---------------
# Siena    Dumas Ang NULL
SQL 3-valued logic

• SQL has 3-valued logic

  FALSE = 0 (ex. price<25 is FALSE when price = 99)
  UNKNOWN = 0.5 (ex. price <25 is UNKNOWN when price=NULL)
  TRUE = 1 (ex. price<25 is TRUE when price = 19)
SQL 3-valued logic

C1 AND C2 means min(C1,C2)
C1 OR C2 means max(C1,C2)
not C means 1-C

The rule for SELECT ... FROM ... WHERE C is the following:
if C = TRUE then include the row in the output
if C = FALSE or C = unknown then do not include it
Example Setup

```sql
erville NULL
CREATE TABLE Dancer (fname VARCHAR(20),
    lname VARCHAR(20),
    years_dancing INTEGER);
INSERT INTO Dancer (fname, lname) 
    VALUES ('Siena', 'Dumas Ang');

CREATE TABLE product (pname varchar(30),
    price integer, category varchar(30),
    manufacturer varchar(40));
```
First Example

```sql
select fname from Dancer where years_dancing = 10;
# Returns nothing
```

```sql
select fname from Dancer where lname = 'Dumas Ang';
# Returns
# fname lname years_dancing
# ---------- ---------- ----------
# Siena Dumas Ang NULL
```

```sql
select fname from Dancer
    where years_dancing = 10 and lname = 'Dumas Ang';
# (years_dancing = 10) = 0.5
# (lname = 'Dumas Ang') = 1
# 0.5 and 1 = min(0.5, 1) = 0.5 or UNKNOWN
```

```sql
select fname from Dancer
    where years_dancing = 10 or lname = 'Dumas Ang';
# (years_dancing = 10) = 0.5
# (lname = 'Dumas Ang') = 1
# 0.5 or 1 = max(0.5, 1) = 1 or TRUE
```
Another Example

```sql
select * from Dancer where years_dancing = NULL;
select * from Dancer where years_dancing <> NULL;
# Returns nothing, because = or <> comparisons
# with NULL BOTH return UNKNOWN!

select * from Dancer where years_dancing is NULL;
# Returns
# fname lname years_dancing
# ----------- ----------- -----------
# Siena Dumas Ang NULL
Compute the value of the condition with NULL

```
INSERT INTO product(pname, price, category, manufacturer)
VALUES ('NullProduct', 19.00, null, null);

SELECT * FROM product WHERE (price < 25)
AND (category = 'gadget')
OR (manufacturer = 'Apple');

SELECT * FROM product WHERE (price < 25)
OR (category = 'gadget')
OR (manufacturer = 'Apple');
```
Review: Order BY

ORDER BY column_name DESC
# ASC (ascending) is default

# Multiple columns
ORDER BY column_name1 DESC, column_name2 ASC

# Or even
ORDER BY column_name1,
       column_name2 DESC,
       column_name3

# Sort order for final case:
# column_name1 ASCending
# column_name2 DESCending
# column_name3 ASCending
Aggregates

• What is the average population of the countries?
  – select avg(population) from Population;
• How many Airports are there total in the DB?
  – select count(*) from Airport;
• How many Airports in each country?
  – select country, count(*) from Airport;
    • Does not work! How should we do this?
    • GROUP BY comes later. 😊
• What is the longest country name?
  – select Population.country, max(length(Population.country))
    from Population;
  – select * from Population order by length(Population.country)
    DESC limit 1;
  – Max and Min can also be found through ordering and limiting in
    some cases
Inner and Outer Joins

INNER JOIN

Table A  
Intersection  
Table B

Select all records from Table A and Table B, where the join condition is met.

LEFT OUTER JOIN

Table A  
Intersection  
Table B

Select all records from Table A, along with records from Table B for which the join condition is met (if at all).

Simple Examples

```
select * from a INNER JOIN b
on a.a = b.b;
# Cardinality: 2
```

```
select * from a LEFT OUTER JOIN b
on a.a = b.b;
# Cardinality: 4
```

```
select * from a RIGHT OUTER JOIN b
on a.a = b.b;
# Cardinality: 4 (but not the same 4 as with the left outer join!)
```
Inner and Outer Joins Data Set Example

- Population does not have a few countries:
  - French Polynesia, Russian Federati, Alaska, Cote D'Ivoire
- How could we count these?
  - `select count(*) from Airport, Population where Airport.country = Population.country;`
  - Only count 8697 but there are 9186 airports!
- Inner Join is the default when unspecified
  - So the unmatched Airports just didn’t get counted!
  - Equivalent to: `select count(*) from Airport INNER JOIN Population on Airport.country = Population.country;`
- Outer join includes tuples from both
  - `select count(*) from Airport LEFT OUTER JOIN Population on Airport.country = Population.country;`
  - 9186 Returned!
GROUP BY

1. Compute the FROM and WHERE clauses.
   - What table?
   - What constraints on each column need you enforce?

2. Group by the attributes in the GROUP BY
   - Create groups of rows that have the same value for that column

3. Compute the SELECT clause:
   grouped attributes and aggregates.
   - Select needs to be the GROUP BY target or an aggregate, such as average, max, or min
Group By Examples

• What 10 countries have the most airports?
  – select country, count(*) from Airport group by country order by count(*) desc limit 10;

• What 10 airport names reoccur the most times?
  – select name, count(*) from Airport group by name having count(*) > 1 order by count(*) desc limit 10;
Don’t forget!

LIMIT
COUNT(*)
DISTINCT
AS
SUM
MAX/MIN