```
-------------------------------------------------------------------------------
-- CSE 344 -- Winter 2013
-- AGGREGATES IN SQL
-- Readings: 6.3, 6.4
-------------------------------------------------------------------------------

-- In this lecture we will use the following schema:

create table Purchase
  (pid int primary key,
   product text,
   price float,
   quantity int,
   month varchar(15));

-- download the file data.txt in the current directory
-- use .import to import the data; see .help
-- note that other database systems have different ways to import data

.import data.txt Purchase

update purchase set price = null where price = 'null';

-- the five basic aggregate operations

select count(*) from purchase;

select count(quantity) from purchase;

select sum(quantity) from purchase;

select avg(price) from purchase;

select max(quantity) from purchase;

select min(quantity) from purchase;

-- Null values are not used in the aggregate

insert into Purchase values(12, 'gadget', NULL, NULL, 'april');

select count(*) from purchase;
select count(quantity) from purchase;
select sum(quantity) from purchase;

-- Counting the number of distinct values

select count(product) from purchase;
select count(distinct product) from purchase;

-------------------------------------------------------------------------------
-- Aggregates With Group-by

select product, count(*)
from purchase
group by product;

select month, count(*)
from purchase
group by month;

-- compare the previous two queries:
--   1. for each PRODUCT compute count(*), v.s.
--   2. for each MONTH compute count(*)

-- aggregates over expressions

-- compute the total revenue for each product:
```

```
select product, sum(price*quantity)
from purchase
group by product;


-- compute the average revenue per sale, for each product:
select product, sum(price*quantity)/count(*)
from purchase
group by product;


-- what do these queries do ?
select product, max(month)
from purchase
group by product;

select product, min(month), max(month)
from purchase
group by product;

select product, month
from purchase
group by product;
-- note: sqlite is WRONG on the last query.  why ?

-------------------------------------------------------------------------------
-- Understanding goups

-- 11 tuples:
select * from purchase;

-- 4 groups:
select product, count(*)
from purchase
group by product;

-- 3 groups:
select product, count(*)
from purchase
where price > 2.0
group by product;


-------------------------------------------------------------------------------
-- "DISTINCT" is the same as "GROUP BY"

select month, count(*)
from purchase
group by month;

select month
from purchase
group by month;

select distinct month
from purchase;



-------------------------------------------------------------------------------
-- Ordering results by aggregate

select product, sum(price*quantity) as rev
from purchase
group by product
order by rev desc;

select month, sum(price*quantity)/count(*) as avgrev
```

```
from purchase
group by month
order by avgrev desc;



-------------------------------------------------------------------------------
-- the HAVING clause

select month, count(*)
from purchase
group by month;


select month,  count(*), sum(price*quantity)/count(*)
from purchase
group by month
having sum(price*quantity)/count(*) < 100.0;

-- Rule
--   WHERE condition is applied to individual rows:
--         the rows may or may not contributed to the aggregate
--         no aggregates allowed here
--   HAVING condition is applied to the entire group:
--         entire group is returned, or not al all
--         may use aggregate functions in the group


-------------------------------------------------------------------------------
-- aggregates and joins

create table Product
        (pid int primary key,
         pname text,
         manufacturer text);

insert into product values(1, 'bagel', 'Sunshine Co.');
insert into product values(2, 'banana', 'BusyHands');
insert into product values(3, 'gizmo', 'GizmoWorks');
insert into product values(4, 'gadget', 'BusyHands');
insert into product values(5, 'powerGizmo', 'PowerWorks');


-- number of sales per manufacturer
select x.manufacturer, count(*)
from Product x, Purchase y
where x.pname = y.product
group by x.manufacturer;

-- number of sales per manufacturer and month
select x.manufacturer, y.month, count(*)
from Product x, Purchase y
where x.pname = y.product
group by x.manufacturer, y.month;


-------------------------------------------------------------------------------
--  Semantics of SQL queries with Group By
--
--     SELECT a1, a2, ..., agg1, agg2, ...
--     FROM R1, R2, ...
--     WHERE C
--     GROUP BY g1, g2, ...
--     HAVING D
--
-- Syntactic rules:
--    C is any condition on the attributes in R1, R2, ...
--    D is any condition on the attributes in R1, R2, ... AND aggregates
```

```
--    all attributes a1, a2, ... must occur in the GROUP BY clause (WHY ?)
--
-- Semantics:
--    Step 1. Evaluate the FROM-WHERE part of the query using the "nested loop" semantics
--    Step 2. Group answers by their values of g1, g2, ...
--    Step 3. Compute the aggregates in D for each goup: retain only groups where D is true
--    Step 4. Compute the aggregates in SELECT and return the answer
--
-- Important notes:
--    there is one row in the answer for each group
--    no group can be empty !  In particular, count(*) is never 0

-------------------------------------------------------------------------------
-- Aggregates v.s. nested subqueries

-- normal aggregate query
select month, count(*)
from purchase
group by month;

-- nested query
select distinct x.month, (select count(*) from purchase y where x.month=y.month)
from   purchase x;


-- the previous two queries are equivalent (they return the same answers)
-- HOWEVER:  all things being equal, we prefer a "flat" query over a "nested" query

-------------------------------------------------------------------------------
-- Aggregates on empty groups

-- number of sales per manufacturer: but PowerWorks does not appear !
select x.manufacturer, count(*)
from Product x, Purchase y
where x.pname = y.product
group by x.manufacturer;

-- one way to get the empty group is to use a subquery:
select distinct z.manufacturer,
     (select count(*)
      from Product x, Purchase y
      where z.manufacturer = x.manufacturer and x.pname = y.product)
from Product z;

-- a better way is to use outer joins:
select x.manufacturer, count(y.pid)
from Product x left outer join Purchase y on x.pname = y.product
group by x.manufacturer;

-- In homework 2 ALWAYS use a flat query (with group by) when possible

-------------------------------------------------------------------------------
-- Be careful of what the query means:

select month, count(*)
from purchase
where price > 10.0
group by month;

-- which of the following nested queries is equivalent to the query above ?

select distinct x.month, (select count(*) from purchase y where x.month=y.month)
from   purchase x
where  price > 10.0;

select distinct x.month, (select count(*) from purchase y where x.month=y.month and price >
10.0)
from   purchase x;
```

```
select distinct x.month, (select count(*) from purchase y where x.month=y.month and price >
10.0)
from   purchase x
where  price > 10.0;
```