COP 4710: Database Systems
Fall 2013

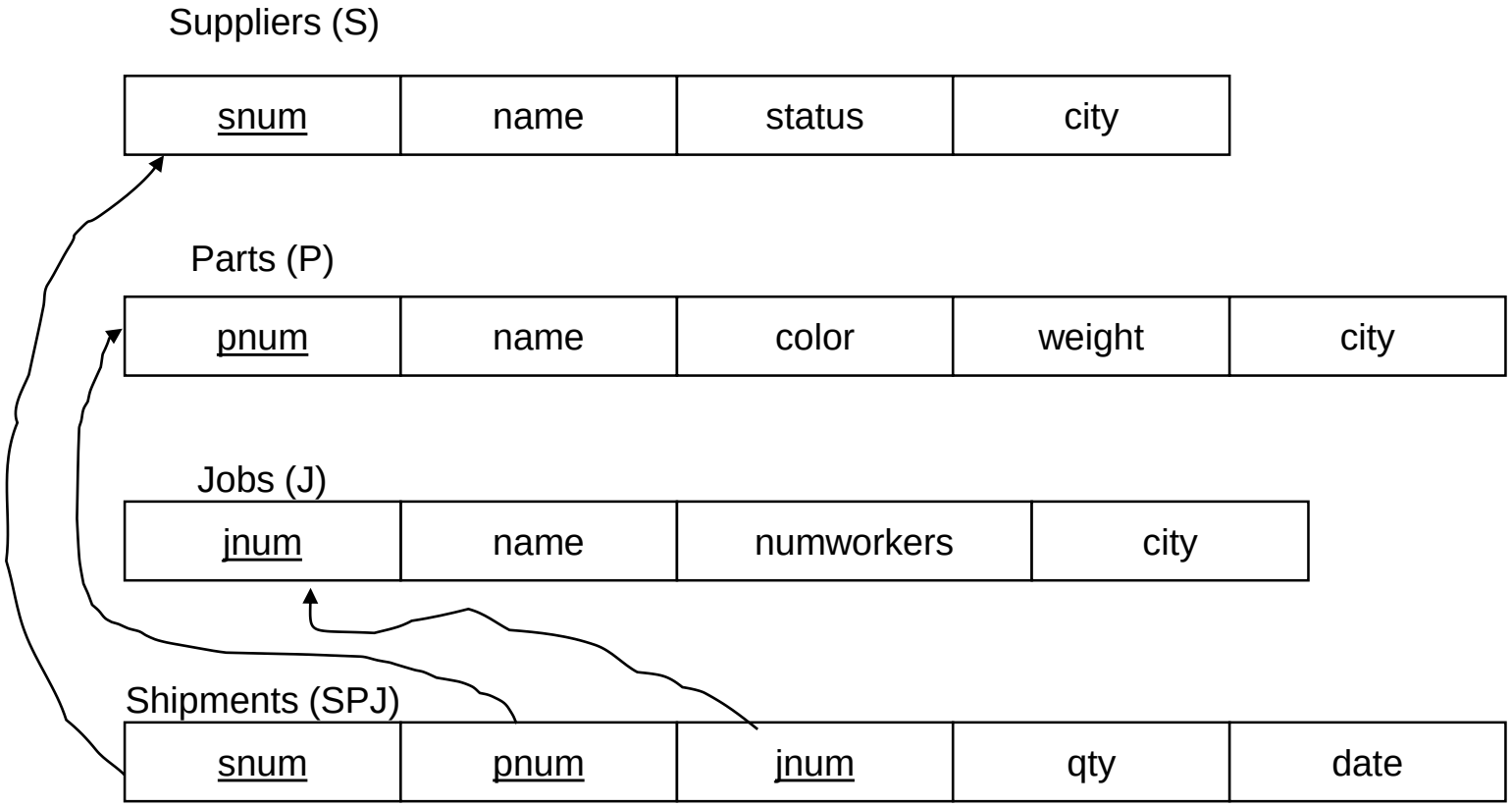
SQL – Practice Problems

Instructor : Dr. Mark Llewellyn
markl@cs.ucf.edu
HEC 236, 407-823-2790
http://www.cs.ucf.edu/courses/cop4710/fall2013

Department of Electrical Engineering and Computer Science
Computer Science Division
University of Central Florida

SQL In Class Exercises

- Use the following database scheme for the problems in this exercise.



- Develop SQL expressions for each of the following queries:

1. List only the names of those suppliers who ship a part that weighs more than 200.

Solutions

```
SELECT sname
FROM suppliers NATURAL JOIN shipments CROSS JOIN parts
WHERE weight > 200 AND shipments.pnum = parts.pnum;
```

Note that a second natural join won't work here since the join would also occur on the city attribute, which would be a more restrictive query.

WRONG

```
SELECT sname
FROM suppliers NATURAL JOIN shipments NATURAL JOIN parts
WHERE weight > 200 AND shipments.pnum = parts.pnum;
```

- or -

```
SELECT sname
FROM suppliers
WHERE snum IN (SELECT snum
               FROM shipments
               WHERE pnum IN (SELECT pnum
                              FROM parts
                              WHERE weight > 200) ) );
```


1. List only the names of those suppliers who ship a part that weighs more than 200.

Solutions

Relational Algebra Version

$$\pi_{name}\left(\text{suppliers} \triangleright\!\triangleleft \left(\text{shipments} \triangleright\!\triangleleft \left(\pi_{pnum}\left(\sigma_{weight > 200}\left(\text{parts}\right)\right)\right)\right)\right)$$

Tuple Calculus Version

{ t.name | t ∈ suppliers AND ∃u (u ∈ shipments AND u.snum = t.snum AND

∃v(v ∈ parts AND v.pnum=u.pnum AND v.weight > 200)) }

2. List the names of those cities in which both a supplier and a job are located.

Solutions

```
SELECT supplier.city
FROM suppliers NATURAL JOIN jobs;
```

- or -

```
SELECT supplier.city
FROM suppliers JOIN jobs
WHERE suppliers.city = jobs.city;
```

- or -

```
SELECT supplier.city
FROM suppliers
WHERE city IN (SELECT city
               FROM jobs);
```

- or -

```
SELECT supplier.city
FROM suppliers, jobs
WHERE suppliers.city = jobs.city;
```


2. List the names of those cities in which both a supplier and a job are located.

Solutions

Relational Algebra Version

$$\left[\pi_{city}(\text{suppliers})\right] \cap \left[\pi_{city}(\text{jobs})\right]$$

Tuple Calculus Version

{r.city | r ∈ suppliers AND ∃t (t ∈ jobs AND r.city = t.city) }

3. List the names of those jobs that receive a shipment from supplier number S1.

WRONG

```
SELECT jname
FROM jobs
WHERE jnum = (SELECT jnum
              FROM shipments
              WHERE snum = "S1");
```

Solutions

```
SELECT jname
FROM jobs
WHERE jnum IN (SELECT jnum
               FROM shipments
               WHERE snum = "S1");
```

- or -

```
SELECT jname
FROM jobs NATURAL JOIN shipments
WHERE snum = "S1";
```

- or -

```
SELECT jname
FROM jobs, shipments
WHERE snum = "S1"
    AND jobs.jnum = shipments.jnum;
```


3. List the names of those jobs that receive a shipment from supplier number S1.

Solutions

Relational Algebra Version

$$\pi_{name}\left(\text{jobs} \triangleright\!\triangleleft \left(\pi_{jnum}\left(\sigma_{snum="S1"}\left(\text{shipments}\right)\right)\right)\right)$$

Tuple Calculus Version

{t.name | t ∈ jobs AND ∃r (r ∈ shipments AND r.jnum = t.jnum AND r.snum = “S1”) }

4. List the names of those parts that are not shipped to any job.

Solutions

```
SELECT pname
FROM parts
WHERE pnum NOT IN (SELECT pnum
                   FROM shipments);
```

- or -

```
SELECT pname
FROM parts
WHERE NOT EXISTS (SELECT *
                  FROM shipments
                  WHERE shipments.pnum = parts.pnum);
```


4. List the names of those parts that are not shipped to any job.

Solutions

Relational Algebra Version

$$\left(\pi_{pnum}\left(\text{parts}\right)\right) - \left(\pi_{pnum}\left(\text{shipments}\right)\right)$$

Tuple Calculus Version

{t.pnum | t ∈ parts AND NOT∃r (r ∈ shipments AND r.pnum = t.pnum) }

5. List the names of those suppliers who ship part number P2 to any job.

Solutions

```
SELECT sname
FROM suppliers
WHERE snum IN (SELECT snum
               FROM shipments
               WHERE pnum = “P2”);
```

- or -

```
SELECT sname
FROM suppliers NATURAL JOIN shipments
WHERE pnum = “P2”;
```


5. List the names of those suppliers who ship part number P2 to any job.

Solutions

Relational Algebra Version

$$\pi_{name}\left(\text{suppliers} \rhd\lhd \left(\sigma_{pnum="P2"}\left(\text{shipments}\right)\right)\right)$$

Tuple Calculus Version

{t.name | t ∈ supliers AND ∃r (r ∈ shipments and r.snum = t.snum AND r.pnum = “P2”) }

6. List the names of those suppliers who do not ship part number P2 to any job.

Note that neither of the following are correct!

```
SELECT sname
FROM suppliers
WHERE snum = (SELECT snum
      FROM shipments
      WHERE pnum ≠ “P2”);
```

-or-

```
SELECT sname
FROM suppliers
WHERE snum IN (SELECT snum
    FROM shipments
    WHERE snum ≠ “P2”);
```

Solutions

```
SELECT sname
FROM suppliers
WHERE snum NOT IN (SELECT snum
                   FROM shipments
                   WHERE pnum = “P2”);
```

- or -

```
SELECT sname
FROM suppliers
WHERE NOT EXISTS (SELECT *
                  FROM shipments
                  WHERE shipments.snum = suppliers.snum AND shipments.pnum = “P2”);
```


6. List the names of those suppliers who do not ship part number P2 to any job.

Solutions

Relational Algebra Version

$$\pi_{name}\left(\text{suppliers} \triangleright\!\triangleleft \left[\left(\pi_{snum}\left(\text{suppliers}\right)\right) - \left(\pi_{snum}\left(\sigma_{pnum="P2"}\left(\text{shipments}\right)\right)\right)\right]\right)$$

Tuple Calculus Version

{t.name | t ∈ supliers AND NOT∃r (r ∈ shipments and r.snum = t.snum AND r.pnum = “P2”) }

7. List the names of those suppliers who ship part at least one red part to any job.

Solutions

```
SELECT sname
FROM suppliers
WHERE snum IN (SELECT snum
                    FROM shipments
                     WHERE pnum IN (SELECT pnum
                                         FROM parts
                                         WHERE color = “red” ));
```

- or -

```
SELECT sname
FROM suppliers NATURAL JOIN shipments
WHERE pnum IN (SELECT pnum
                     FROM parts
                     WHERE color = “red”);
```


7. List the names of those suppliers who ship part at least one red part to any job.

Solutions

Relational Algebra Version

$$\pi_{name}\left(\text{suppliers} \rhd\lhd \left(\pi_{snum}\left(\text{shipments} \rhd\lhd \left(\sigma_{color="red"}\left(\text{parts}\right)\right)\right)\right)\right)$$

Tuple Calculus Version

{t.name | t ∈ suppliers AND ∃r (r ∈ shipments AND r.snum = t.snum AND ∃u (u ∈ parts
AND u.color = “red” AND u.pnum = r.pnum)) }

8. List the part number for every part that is shipped more than once (the part must be shipped more than one time).

Solution

```
SELECT pnum
FROM shipments
GROUP BY pnum
HAVING COUNT (snum) > 1;
```

WHERE clause restricts by rows

HAVING clause restricts by groups

Relational Algebra Version

Tuple Calculus Version

Since this query is expressed using an aggregate operation (you need to count tuples), this query is not expressible in either relational algebra nor tuple calculus.

8A. List the part number for every part that is shipped by more than one supplier (the part must be shipped by different suppliers).

Solution

```
SELECT pnum
FROM shipments
GROUP BY pnum
HAVING COUNT ( DISTINCT snum) > 1;
```

This ensures that the same supplier would appear only once, since DISTINCT is applied first, the count will look only at unique snums

Relational Algebra Version

Tuple Calculus Version

Since this query is expressed using an aggregate operation (you need to count tuples), this query is not expressible in either relational algebra nor tuple calculus.

9. List the names of those suppliers who ship every part.

Solutions

```
SELECT sname
FROM suppliers
WHERE  NOT EXISTS (SELECT *
                   FROM parts
                   WHERE NOT EXITS (SELECT *
                                    FROM shipments
                                    WHERE shipments.snum = suppliers.snum
                                      AND shipments.pnum = parts.pnum ) );
```

This solution is correct if the participation of parts in shipments is optional or mandatory.

- or -

```
SELECT sname
FROM suppliers
WHERE (SELECT COUNT (shipments.pnum)
       FROM shipments
       WHERE shipments.snum = suppliers.snum)
      =
      (SELECT COUNT (parts.pnum)
       FROM parts);
```

This solution is correct only if the participation of parts in shipments is mandatory. It is incorrect if the participation of parts in shipments is optional.

9. List the names of those suppliers who ship every part.

Solutions

Relational Algebra Version

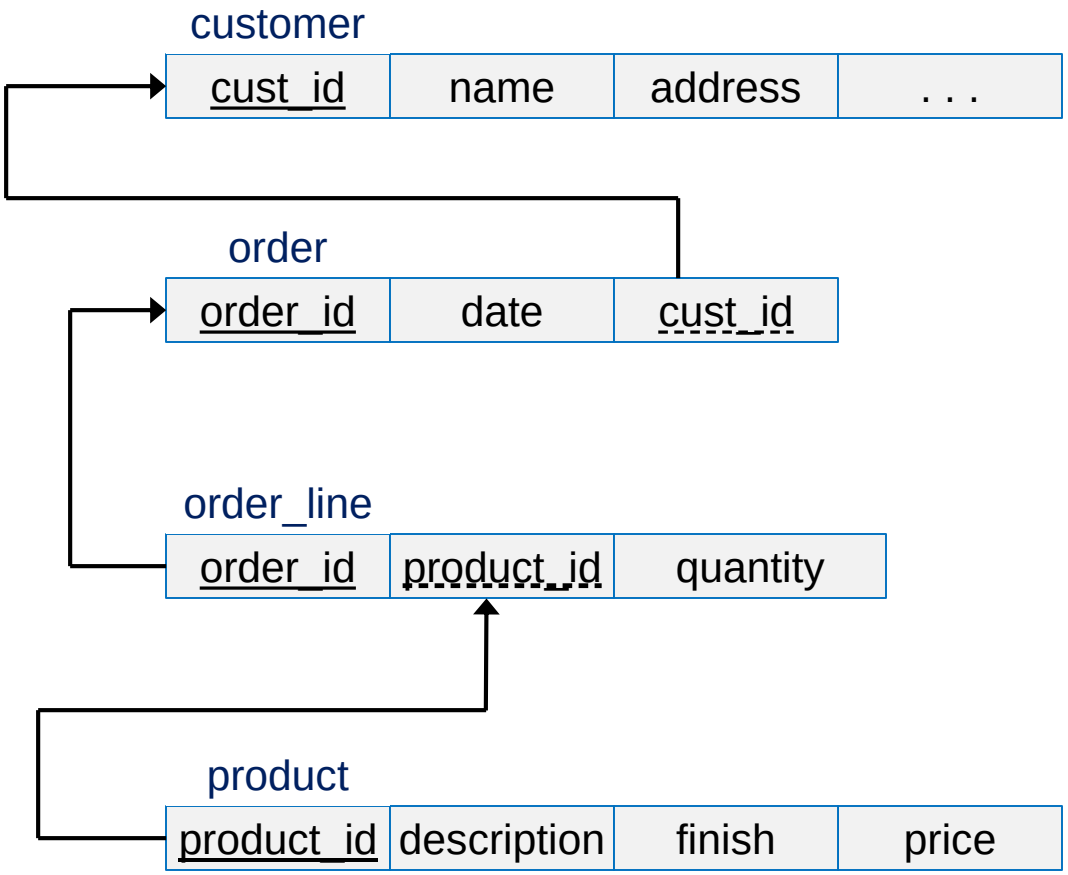
$$\pi_{name}\left(\text{suppliers} \triangleright\triangleleft \left[\left(\pi_{snum,\,pnum}\left(\text{shipments}\right)\right) \div \left(\pi_{pnum}\left(\text{parts}\right)\right)\right]\right)$$

Tuple Calculus Version

{t.name | t ∈ suppliers AND ∀u (u ∈ parts AND ∃ r (r ∈ shipments AND u.pnum = r.pnum AND
r.snum = t.snum)) }

SQL In Class Exercises

- For the remainder of the problems use the following database schema. I did only SQL expressions for the remaining examples.



10. List the date of every order placed by customer 5.

Solutions

```
SELECT date
FROM order
WHERE cust_id = 5;
```

- or -

```
SELECT DISTINCT date
FROM order
WHERE cust_id = 5;
```


11. List all the cities from which a customer placed an order on March 11th.

Solutions

```
SELECT DISTINCT city
FROM customer NATURAL JOIN order
WHERE date = “March 11”;
```

- or -

```
SELECT DISTINCT city
FROM customer
WHERE cust_id  IN (SELECT cust_id
                   FROM order
                   WHERE date = “March 11”);
```


12. List the dates for every order placed that included part number 6.

Solutions

```
SELECT DISTINCT date
FROM order NATURAL JOIN order_line
WHERE product_id = 6;
```

- or -

```
SELECT DISTINCT date
FROM order
WHERE order_id  IN (SELECT order_id
                    FROM order_line
                    WHERE product_id = 6);
```


13. List the names of those customers who have not placed any orders.

Solution

```
SELECT name
FROM customer
WHERE cust_id NOT IN (SELECT cust_id
                      FROM order);
```


14. List the names of those customers who have never ordered part number 6.

Solution

```
SELECT DISTINCT name
FROM customer
WHERE cust_id NOT IN (SELECT cust_id
                      FROM order
                      WHERE order_id IN (SELECT order_id
                                         FROM order_line
                                         WHERE product_id = 6)
                      );
```


15. List the names of those customers who have ordered both part number 5 and part number 6.

Solution

```
SELECT DISTINCT name
FROM customer
WHERE  (cust_id IN (SELECT cust_id
                    FROM order
                     WHERE order_id IN (SELECT order_id
                                        FROM order_line
                                        WHERE product_id = 5)  )
)
AND
(cust_id IN (SELECT cust_id
                    FROM order
                     WHERE order_id IN (SELECT order_id
                                        FROM order_line
                                        WHERE product_id = 6)  )
);
```


16. List the names of those customers who have ordered part number 5 and not ordered part number 6.

Solution

```
SELECT DISTINCT name
FROM customer
WHERE  (cust_id IN (SELECT cust_id
                    FROM order
                    WHERE order_id IN (SELECT order_id
                                       FROM order_line
                                       WHERE product_id = 5)  )
)
AND
(cust_id  NOT IN (SELECT cust_id
                  FROM order
                  WHERE order_id IN (SELECT order_id
                                     FROM order_line
                                     WHERE product_id = 6)  )
);
```


17. List the names of those customers who have ordered either part number 5 or part number 6.

Solution

```
SELECT DISTINCT name
FROM customer
WHERE  cust_id IN (SELECT cust_id
                   FROM order
                    WHERE order_id IN (SELECT order_id
                                       FROM order_line
                                        WHERE product_id = 5
                                           OR product_id = 6)  );
```

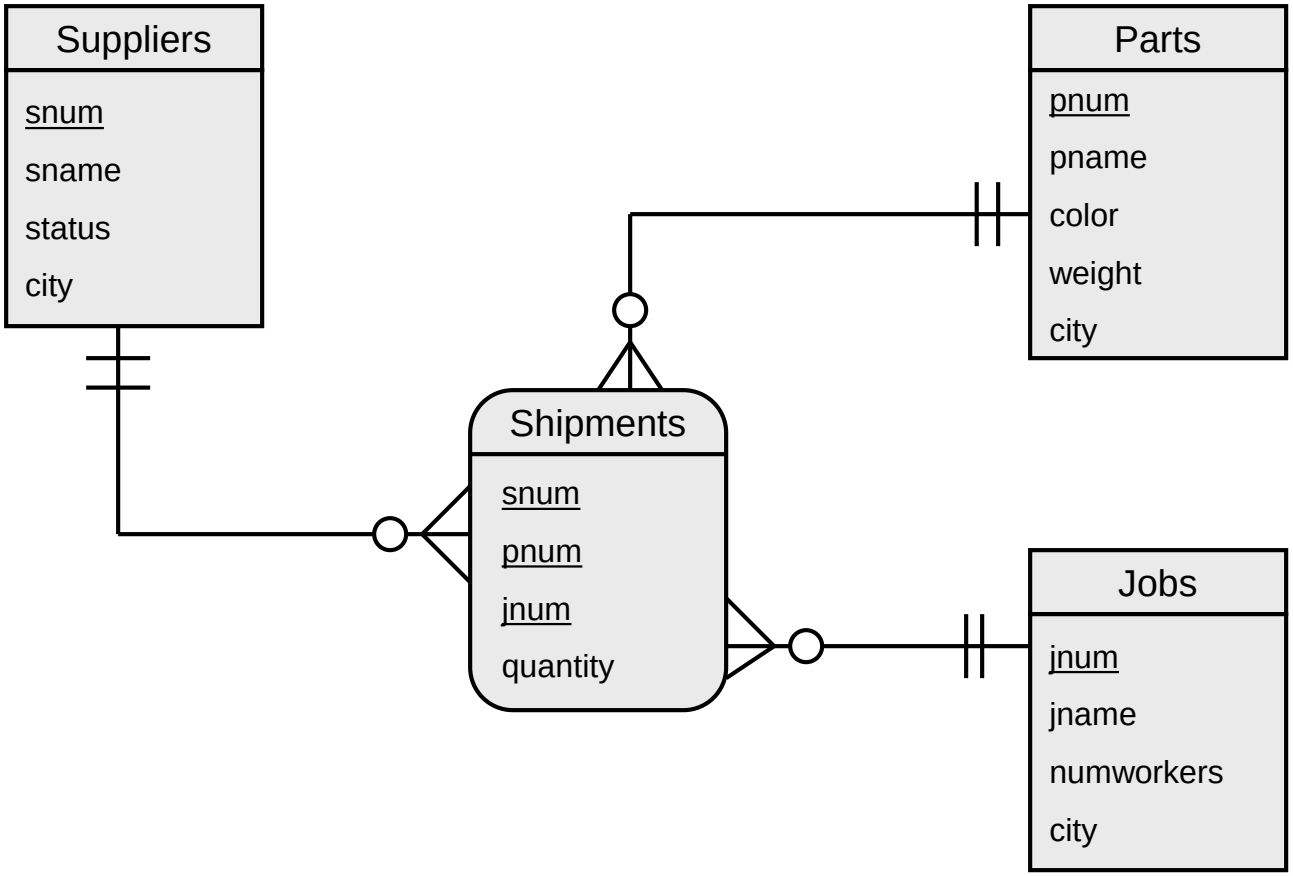

18. List the names of those customers who have ordered only part number 6.

Solution

```
SELECT DISTINCT name
FROM customer
WHERE  (cust_id IN (SELECT cust_id
                    FROM order
                    WHERE order_id IN (SELECT order_id
                                       FROM order_line
                                       WHERE product_id = 6)  )
)
AND
(cust_id  NOT IN (SELECT cust_id
                  FROM order
                  WHERE order_id IN (SELECT order_id
                                     FROM order_line
                                     WHERE product_id <> 6)  )
);
```


SQL In Class Exercises

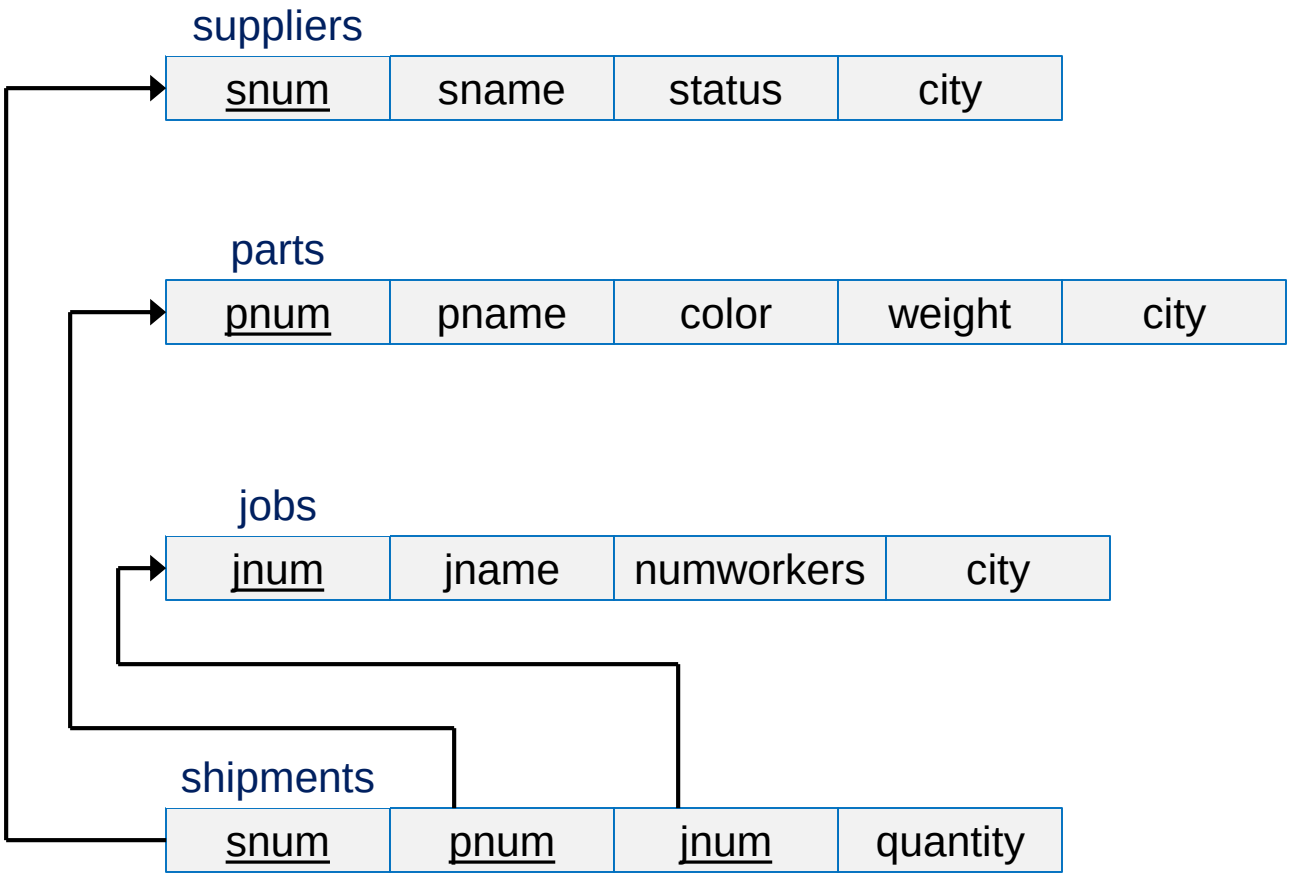
- Use the following database scheme for problems 19-30 in this exercise.



- Develop SQL expressions for each of the following queries:

SQL In Class Exercises

- The schema version of the database for problems 19-30.



19. Create the table definition for the suppliers schema. Do not allow a supplier's information to be deleted if they have a shipment.

Solution

```
CREATE TABLE suppliers
(snum VARCHAR2(5) NOT NULL,
 sname VARCHAR2(25),
 status INTEGER,
 city VARCHAR2(20),
 CONSTRAINT supplier_PK PRIMARY KEY (snum)
   ON UPDATE RESTRICT);
```

20. Assuming that the tables for the parts and jobs were created in a similar fashion to that of the suppliers table on the previous page, create the table definition for the shipments schema.

Solution

```
CREATE TABLE shipments
(snum VARCHAR2(5) NOT NULL,
 pnum VARCHAR2(5) NOT NULL,
 jnum VARCHAR2(5) NOT NULL,
 quantity INTEGER,
 CONSTRAINT ship_PK PRIMARY KEY (snum, pnum, jnum),
 CONSTRAINT ship_FK1 FOREIGN KEY (snum) REFERENCES suppliers(snum),
 CONSTRAINT ship_FK2 FOREIGN KEY (pnum) REFERENCES parts(pnum),
 CONSTRAINT ship_FK3 FOREIGN KEY (jnum) REFERENCES jobs(jnum) );
```


21. Insert a new supplier's information into the suppliers table.

Solution

```
INSERT INTO suppliers VALUES
("S1","Kristy",14,"Orlando);
```


22. Delete from the shipment table every row where the quantity is less than 10.

Solution

```
DELETE FROM shipments
WHERE quantity < 10;
```


23. Update the suppliers table by modifying the status of every supplier whose current status is 10 by increasing the status by 5.

Solution

```
UPDATE suppliers
  SET status = status + 5
  WHERE status = 10;
```


24. Update the parts table by modifying the weight of part number 6 to its current weight + 20.

Solution

```
UPDATE parts
   SET weight = weight + 20
   WHERE pnum = 6;
```


25. Modify the data in the parts table so that every part that was blue is now colored green.

Solution

```
UPDATE parts
    SET color = "green"
    WHERE color = "blue";
```

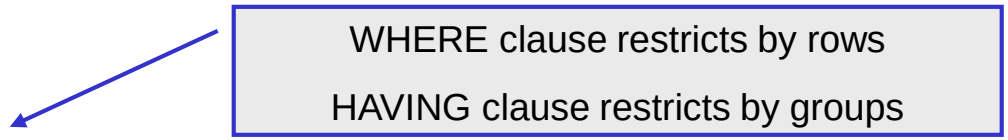

26. List only the names of those suppliers who are located in Orlando.

Solution

```
SELECT sname
FROM suppliers
WHERE city = "Orlando";
```


27. List the part number for every part that is shipped by more than one supplier.

Solution




```
SELECT pnum
FROM shipments
GROUP BY pnum
HAVING COUNT (snum) > 1;
```


28. Find the average weight of all parts.

Solution

```
SELECT AVG(weight)
FROM parts;
```


29. For each part list the part number and the total quantity in which that part is shipped and order the results in descending order of the total quantity shipped. Name the total quantity shipped in the result as totalShipped.

Solution

```
SELECT pnum, SUM(quantity) AS totalShipped
FROM shipments
GROUP BY pnum
ORDER BY SUM(quantity) DESC;
```


30. List the supplier number and the total quantity of parts that supplier ships and group the results by supplier number in descending order of the total quantity supplied.

Solution

```
SELECT snum, SUM(shipments.quantity)
       AS totalShipped
FROM shipments
GROUP BY snum
ORDER BY sum(quantity) DESC;
```