Product (maker, model, type) PC (model, speed, ram, hdisk, price) Laptop (model, speed, ram, hdisk, screen, price)

Printer (model, color, type, price)

1. Find the model number, speed, and hard-disk size for all PC ’s whose price is under $1000. SELECT model, speed, hdisk

FROM PC

WHERE price < 1000;

+-------+-------+-------+

| model | speed | hdisk |

+-------+-------+-------+

| 1002 | 2.10 | 250 |

| 1003 | 1.42 | 80 |

| 1004 | 2.80 | 250 |

| 1005 | 3.20 | 250 |

| 1007 | 2.20 | 200 |

| 1008 | 2.20 | 250 |

| 1009 | 2.00 | 250 |

| 1010 | 2.80 | 300 |

| 1011 | 1.86 | 160 |

| 1012 | 2.80 | 160 |

| 1013 | 3.06 | 80 |

+-------+-------+-------+

11 rows in set (0.01 sec)

2. List the manufacturers of all printers. SELECT DISTINCT maker

FROM Product, Printer

WHERE Product.model = Printer.model;

+-------+

| maker |

+-------+

| D |

| E |

| H |

+-------+

3 rows in set (0.00 sec)

3. Find the model number, memory size, and screen size for laptops costing more than $1500. SELECT model, ram, screen

FROM Laptop

WHERE price > 1500;

+-------+------+--------+

| model | ram | screen |

+-------+------+--------+

| 2001 | 2048 | 20.10 |

| 2005 | 1024 | 17.00 |

| 2006 | 2048 | 15.40 |

| 2010 | 2048 | 15.40 |

+-------+------+--------+

4 rows in set (0.00 sec)

4. List all color printers.

SELECT model

FROM Printer

WHERE color=true;

+-------+

| model |

+-------+

| 3001 |

| 3003 |

| 3004 |

| 3006 |

| 3007 |

+-------+

5 rows in set (0.01 sec)

5. Find the model number and hard-disk size for those PC ’s that have a speed of 3.2 and a price less than $2000. SELECT model, hdisk

FROM PC

WHERE speed = 3.2 and price < 2000;

+-------+-------+

| model | hdisk |

+-------+-------+

| 1005 | 250 |

| 1006 | 320 |

+-------+-------+

2 rows in set (0.00 sec)

6. Give the manufacturer and speed of laptops with a hard disk of at least 30. SELECT DISTINCT maker, speed

FROM Laptop, Product

WHERE Product.model = Laptop.model and hdisk >= 30;

+-------+-------+

| maker | speed |

+-------+-------+

| A | 2.00 |

| A | 2.16 |

| B | 1.83 |

| E | 2.00 |

| E | 1.73 |

| E | 1.80 |

| F | 1.60 |

| G | 2.00 |

+-------+-------+

8 rows in set (0.00 sec)

7. Find the model number and price of all products (of any type) made by manufacturer B. SELECT DISTINCT model, price

FROM Laptop

UNION SELECT model, price

FROM PC

UNION SELECT model, price

FROM Printer

WHERE EXISTS (SELECT model FROM Product WHERE maker = ‘B’);

+-------+---------+

| model | price |

+-------+---------+

| 2001 | 3673.00 |

| 2002 | 949.00 |

| 2003 | 549.00 |

| 2004 | 1150.00 |

| 2005 | 2500.00 |

| 2006 | 1700.00 |

| 2007 | 1429.00 |

| 2008 | 900.00 |

| 2009 | 680.00 |

| 2010 | 2300.00 |

| 1001 | 2114.00 |

| 1002 | 995.00 |

| 1003 | 478.00 |

| 1004 | 649.00 |

| 1005 | 630.00 |

| 1006 | 1049.00 |

| 1007 | 510.00 |

| 1008 | 770.00 |

| 1009 | 650.00 |

| 1010 | 770.00 |

| 1011 | 959.00 |

| 1012 | 649.00 |

| 1013 | 529.00 |

| 3001 | 99.00 |

| 3002 | 239.00 |

| 3003 | 899.00 |

| 3004 | 120.00 |

| 3005 | 120.00 |

| 3006 | 100.00 |

| 3007 | 200.00 |

+-------+---------+

30 rows in set (0.00 sec)

8. Find those manufacturers that sell Laptops, but not PC ’s. SELECT DISTINCT maker

FROM Product

WHERE type = ‘laptop’ AND maker NOT IN

(SELECT DISTINCT maker

FROM Product

WHERE type = ‘pc’);

+-------+

| maker |

+-------+

| F |

| G |

+-------+

2 rows in set (0.00 sec)9. Find those hard-disk sizes that occur in two or more PC ’s. SELECT DISTINCT PC.hdisk FROM PC, PC P WHERE PC.hdisk = P.hdisk;

+-------+

| hdisk |

+-------+

| 250 |

| 80 |

| 320 |

| 200 |

| 300 |

| 160 |

+-------+

6 rows in set (0.00 sec)

10. Find those pairs of PC models that have both the same speed and RAM. A pair should be listed only once; e.g., list (i , j ) but not (j,i).

SELECT PC.model, P.model

FROM PC, PC P

WHERE PC.speed = P.speed AND PC.ram = P.RAM AND PC.model < P.model;

+-------+-------+

| model | model |

+-------+-------+

| 1004 | 1012 |

+-------+-------+

1 row in set (0.00 sec)

11. Find those manufacturers of at least two different computers (PC’s or laptops) with speeds of at least 3.0.

SELECT DISTINCT P.maker

FROM Product P, Product P1

WHERE P.maker = P1.maker AND P.model in

(SELECT PC.model from PC WHERE PC.speed > 3 UNION

SELECT Laptop.model FROM Laptop WHERE Laptop.speed > 3)

AND P1.model in (SELECT PC.model from PC WHERE PC.speed > 3 UNION

SELECT Laptop.model FROM Laptop WHERE Laptop.speed > 3)

AND P1.model <> P.model;

+-------+

| maker |

+-------+

| B |

+-------+

1 row in set (0.01 sec)

12. Find the makers of PC ’s with a speed of at least 3.0. SELECT DISTINCT maker

FROM Product, PC

WHERE PC.model = Product.model AND PC.speed > 3;

+-------+

| maker |

+-------+

| B |

| E |

+-------+

2 rows in set (0.00 sec)

13. Find the printers with the highest price. SELECT DISTINCT model

FROM Printer

WHERE price =

(SELECT MAX(price) FROM Printer);

+-------+

| model |

+-------+

| 3003 |

+-------+

1 row in set (0.00 sec)

14. Find the laptops whose speed is slower than that of any PC.

SELECT DISTINCT model

FROM Laptop

WHERE speed <

(SELECT MIN(speed) FROM PC);

Empty set (0.00 sec)

15. Find the model number of the item (PC, laptop, or printer) with the highest price. SELECT DISTINCT model FROM PC

WHERE price =

(SELECT Max(P.price) FROM

(SELECT price FROM PC UNION SELECT price FROM Laptop

UNION SELECT price FROM Printer) P) UNION

SELECT DISTINCT model FROM Laptop

WHERE price =

(SELECT Max(P.price) FROM

(SELECT price FROM PC UNION SELECT price FROM Laptop

UNION SELECT price FROM Printer) P) UNION

SELECT DISTINCT model FROM Printer

WHERE price =

(SELECT Max(P.price) FROM

(SELECT price FROM PC UNION SELECT price FROM Laptop

UNION SELECT price FROM Printer) P);

+-------+

| model |

+-------+

| 2001 |

+-------+

1 row in set (0.00 sec)

16. Find the maker of the least expensive color printer.

SELECT maker FROM Product, Printer WHERE Product.model = Printer.model AND

Printer.price = (SELECT Min(price) FROM Printer);

+-------+

| maker |

+-------+

| E |

+-------+

1 row in set (0.00 sec)17. Find the maker(s) of the PC(s) with the fastest processor among all those PC’s that have the smallest amount of RAM.

SELECT DISTINCT maker FROM Product, PC WHERE Product.model = PC.model AND

PC.model =

(SELECT Min(model) FROM

(SELECT model, speed FROM PC WHERE PC.ram =

(SELECT Min(ram) FROM PC)) P1 WHERE P1.speed = (SELECT Max(speed) FROM

(SELECT model, speed FROM PC WHERE PC.ram =

(SELECT Min(ram) FROM PC)) P2));

+-------+

| maker |

+-------+

| B |

+-------+

1 row in set (0.01 sec)

18. Find the average speed of PC ’s. SELECT AVG(speed) FROM PC;

+------------+

| AVG(speed) |

+------------+

| 2.484615 |

+------------+

1 row in set (0.01 sec)

19. Find the average speed of laptops costing over $1000. SELECT AVG(speed) FROM (SELECT speed FROM Laptop WHERE price > 1000) P1;

+------------+

| AVG(speed) |

+------------+

| 1.998333 |

+------------+

1 row in set (0.00 sec)

20. Find the average price of PC’s made by manufacturer “A.”

SELECT AVG(price) FROM

(SELECT price FROM PC, PRODUCT WHERE Product.model = PC.model AND Product.maker = ‘A’) P1;

+-------------+

| AVG(price) |

+-------------+

| 1195.666667 |

+-------------+

1 row in set (0.01 sec)

21. Find the average price of PC’s and laptops made by manufacturer “D.”

SELECT AVG(price) FROM

((SELECT price FROM PC, Product WHERE Product.model = PC.model AND Product.maker = ‘D’) UNION

(SELECT price FROM Laptop, Product WHERE Product.model = Laptop.model AND Product.maker = ‘D’)) P1;

+------------+

| AVG(price) |

+------------+

| 710.000000 |

+------------+

1 row in set (0.00 sec)

22. Find, for each different speed, the average price of a PC. SELECT AVG(price), speed FROM

(SELECT speed, price FROM PC) P1

GROUP BY speed;

+-------------+-------+

| AVG(price) | speed |

+-------------+-------+

| 478.000000 | 1.42 |

| 959.000000 | 1.86 |

| 650.000000 | 2.00 |

| 995.000000 | 2.10 |

| 640.000000 | 2.20 |

| 2114.000000 | 2.66 |

| 689.333333 | 2.80 |

| 529.000000 | 3.06 |

| 839.500000 | 3.20 |

+-------------+-------+

9 rows in set (0.00 sec)

23. Find for each manufacturer, the average screen size of its laptops. SELECT AVG(screen), maker FROM

(SELECT screen, maker FROM Laptop, Product

WHERE Product.model = Laptop.model) P1

GROUP BY maker;

+-------------+-------+

| AVG(screen) | maker |

+-------------+-------+

| 15.233333 | A |

| 13.300000 | B |

| 17.500000 | E |

| 14.750000 | F |

| 15.400000 | G |

+-------------+-------+

5 rows in set (0.00 sec)

24. Find the manufacturers that make at least three different models of PC. SELECT DISTINCT Product.maker FROM Product, Product P1, Product P2

WHERE Product.maker = P1.maker AND P1.maker = P2.maker

AND Product.model != P1.model AND Product.model != P2.model

AND P1.model != P2.model AND Product.type = ‘PC’;

+-------+

| maker |

+-------+

| A |

| B |

| D |

| E |

+-------+

4 rows in set (0.00 sec)

25. Find for each manufacturer who sells PC’s the maximum price of a PC. SELECT maker FROM product WHERE Product.model =

(SELECT model FROM PC WHERE price =

(SELECT MAX(price) FROM PC)) P2

+-------+

| maker |

+-------+

| A |

+-------+

1 row in set (0.00 sec)

26. Find, for each speed of PC above 2.0, the average price. SELECT AVG(price), speed FROM

(SELECT price, speed FROM PC WHERE speed > 2.0) P1

GROUP BY speed;

+-------------+-------+

| AVG(price) | speed |

+-------------+-------+

| 995.000000 | 2.10 |

| 640.000000 | 2.20 |

| 2114.000000 | 2.66 |

| 689.333333 | 2.80 |

| 529.000000 | 3.06 |

| 839.500000 | 3.20 |

+-------------+-------+

6 rows in set (0.00 sec)

27. Find the average hard disk size of a PC for all those manufacturers that make printers.

SELECT AVG(hdisk) FROM PC,

(SELECT model FROM Product,

(SELECT DISTINCT maker FROM Product WHERE type = ‘printer’) P2

WHERE Product.maker = P2.maker and type = ‘PC’) P3

WHERE PC.model = P3.model;

+------------+

| AVG(hdisk) |

+------------+

| 200.0000 |

+------------+

1 row in set (0.01 sec)

Database 2: Battleship A database concerning World War II capital ships. It involves the following relations: Classes (class, type, country, guns, bore, displacement) Ships (name, class, launched) Battles (name, bdate) Outcomes (ship, battle, result) 28. Find the class name and country for all classes with at least 10 guns. SELECT DISTINCT country, class FROM Classes WHERE guns >= 10;

+---------+-----------+

| country | class |

+---------+-----------+

| USA | Tennessee |

+---------+-----------+

1 row in set (0.00 sec)

29. Find the names of all ships launched prior to 1918, but call the resulting column shipName. SELECT name FROM Ships WHERE launched < 1918;

+-----------------+

| name |

+-----------------+

| Haruna |

| Hiei |

| Kirishima |

| Kongo |

| Ramillies |

| Renown |

| Repulse |

| Resolution |

| Revenge |

| Royal Oak |

| Royal Sovereign |

+-----------------+

11 rows in set (0.00 sec)

30. Find the names of ships sunk in battle and the name of the battle in which they were sunk. SELECT ship, battle FROM outcomes WHERE result = ‘sunk’;

+-------------+----------------+

| ship | battle |

+-------------+----------------+

| Arizona | Pearl Harbor |

| Bismarck | Denmark Strait |

| Fuso | Surigao Strait |

| Hood | Denmark Strait |

| Kirishima | Guadalcanal |

| Scharnhorst | North Cape |

| Yamashiro | Surigao Strait |

+-------------+----------------+

7 rows in set (0.00 sec)

31. Find all ships that have the same name as their class. SELECT name FROM Ships WHERE name = class;

+----------------+

| name |

+----------------+

| Iowa |

| Kongo |

| North Carolina |

| Renown |

| Revenge |

| Tennessee |

| Yamato |

+----------------+

7 rows in set (0.00 sec)

32. Find the names of all ships that begin with the letter “R.” SELECT name FROM ships WHERE name LIKE ‘%r%’;

+-----------------+

| name |

+-----------------+

| Missouri |

| New Jersey |

| Haruna |

| Kirishima |

| North Carolina |

| Renown |

| Repulse |

| Ramillies |

| Resolution |

| Revenge |

| Royal Oak |

| Royal Sovereign |

| California |

+-----------------+

13 rows in set (0.00 sec)

33. Find the names of all ships from the Outcomes table whose name consists of three or more words (e.g., King George V). SELECT ship FROM outcomes WHERE ship LIKE ‘% % %’;

+-----------------+

| ship |

+-----------------+

| Duke of York |

| King George V |

| Prince of Wales |

+-----------------+

3 rows in set (0.00 sec)

34. Find the ships heavier than 35,000 tons.

SELECT name FROM ships,(SELECT class FROM classes WHERE displacement > 35000) P1

WHERE Ships.class = P1.class;

+----------------+

| name |

+----------------+

| Iowa |

| Missouri |

| New Jersey |

| Wisconsin |

| North Carolina |

| Washington |

| Musashi |

| Yamato |

+----------------+

8 rows in set (0.00 sec)

35. List the name, displacement, and number of guns of the ships engaged in the battle of Guadalcanal. SELECT ship, displacement, guns FROM classes,

(SELECT ship, class FROM ships,

(SELECT ship FROM outcomes WHERE battle = ‘Guadalcanal’) P1

WHERE ships.name = P1.ship) P2

WHERE P2.class = classes.class;

+------------+--------------+------+

| ship | displacement | guns |

+------------+--------------+------+

| Kirishima | 32000 | 8 |

| Washington | 37000 | 9 |

+------------+--------------+------+

2 rows in set (0.00 sec)

36. Find those countries that have both battleships and battlecruisers. SELECT DISTINCT P1.country FROM classes, classes P1

WHERE classes.type = ‘bb’ AND P1.type = ‘bc’;

+-------------+

| country |

+-------------+

| Japan |

| Gt. Britain |

+-------------+

2 rows in set (0.00 sec)

37. Find those ships that were damaged in one battle, but later fought in. SELECT P2.ship FROM

(SELECT ship, bdate FROM battles, outcomes

WHERE battles.name = outcomes.battle) P2,

(SELECT ship, bdate FROM battles, outcomes

WHERE battles.name = outcomes.battle AND outcomes.result = ‘sunk’) P1

WHERE P2.bdate > P1.bdate AND P2.ship = P1.ship;

Empty set (0.00 sec)

38. Find those battles with at least three ships of the same country.

SELECT DISTINCT P5.battle FROM

(SELECT DISTINCT P4.battle, C1.country as country1, C2.country as country2, C3.country as country3 FROM classes C1, classes C2, classes C3,

(SELECT DISTINCT P3.battle, S1.class as class1, S2.class as class2, S3.class as class3

FROM ships S1, ships S2, ships S3,(SELECT DISTINCT outcomes.battle, outcomes.ship as ship1, P1.ship as ship2, P2.ship as ship3

FROM outcomes, outcomes P1, outcomes P2

WHERE outcomes.ship != P1.ship and outcomes.ship != P2.ship

and P1.ship != P2.ship and outcomes.battle = P1.battle and P1.battle = p2.battle) P3

WHERE S1.name = P3.ship1 AND P3.ship2 = S2.name AND P3.ship3 = S3.name) P4

WHERE C1.class = P4.class1 and C2.class = P4.class2 and C3.class = P4.class3) P5

WHERE P5.country1 = P5.country2 AND P5.country2 = P5.country3;

Empty set (0.00 sec)

39. Find the countries whose ships had the largest number of guns. SELECT country FROM classes

WHERE classes.guns = (SELECT MAX(guns) FROM classes);

+---------+

| country |

+---------+

| USA |

+---------+

1 row in set (0.00 sec)

40. Find the classes of ships, at least one of which was sunk in a battle.

SELECT class FROM ships,(SELECT ship FROM outcomes WHERE result = ‘sunk’) P1

WHERE ships.name = P1.ship;

+-------+

| class |

+-------+

| Kongo |

+-------+

1 row in set (0.00 sec)

41. Find the names of the ships with a 16-inch bore.

SELECT name FROM ships,

(SELECT class FROM classes WHERE bore = 16 ) P1

WHERE ships.class = P1.class;

+----------------+

| name |

+----------------+

| Iowa |

| Missouri |

| New Jersey |

| Wisconsin |

| North Carolina |

| Washington |

+----------------+

6 rows in set (0.00 sec)

42. Find the battles in which ships of the Kongo class participated. SELECT DISTINCT battle FROM outcomes,

(SELECT name FROM ships WHERE class = ‘Kongo’) P1

WHERE P1.name = outcomes.ship;

+-------------+

| battle |

+-------------+

| Guadalcanal |

+-------------+

1 row in set (0.00 sec)

43. Find the names of the ships whose number of guns was the largest for those ships of the same bore. SELECT name, P2.bore FROM ships,

(SELECT class, P1.bore FROM classes,

(SELECT MAX(guns) as guns, bore FROM classes GROUP BY bore) P1

WHERE classes.guns = P1.guns AND classes.bore = P1.bore) P2

WHERE ships.class = P2.class;

+-----------------+------+

| name | bore |

+-----------------+------+

| Iowa | 16 |

| Missouri | 16 |

| New Jersey | 16 |

| Wisconsin | 16 |

| North Carolina | 16 |

| Washington | 16 |

| Ramillies | 15 |

| Resolution | 15 |

| Revenge | 15 |

| Royal Oak | 15 |

| Royal Sovereign | 15 |

| California | 14 |

| Tennessee | 14 |

| Musashi | 18 |

| Yamato | 18 |

+-----------------+------+

15 rows in set (0.01 sec)

44. Find the number of battleship (type=’bb’) classes. SELECT COUNT(class) FROM classes WHERE type = ‘bb’;

+--------------+

| COUNT(class) |

+--------------+

| 6 |

+--------------+

1 row in set (0.00 sec)

45. Find the average number of guns of battleship (type=’bb’) classes.

SELECT AVG(guns) FROM classes WHERE type =’bb’;

+-----------+

| AVG(guns) |

+-----------+

| 9.1667 |

+-----------+

1 row in set (0.00 sec)

46. Find the average number of guns of battleships (type=’bb’). Note the difference to the previous question. In this question, you need to weight a class by the number of ships of that class.

SELECT AVG(guns) FROM (SELECT name, guns FROM ships, classes

WHERE ships.class = classes.class AND classes.type = ‘bb’) P1;

+-----------+

| AVG(guns) |

+-----------+

| 9.0667 |

+-----------+

1 row in set (0.00 sec)

47. Find for each class the year in which the first ship of that class was launched.

SELECT class, MIN(launched) FROM ships GROUP BY class;

+----------------+---------------+

| class | MIN(launched) |

+----------------+---------------+

| Iowa | 1943 |

| Kongo | 1913 |

| North Carolina | 1941 |

| Renown | 1916 |

| Revenge | 1916 |

| Tennessee | 1920 |

| Yamato | 1941 |

+----------------+---------------+

7 rows in set (0.00 sec)

48. Find for each class the number of ships of that class sunk in battle.

SELECT DISTINCT COUNT(ship), class FROM outcomes, ships

WHERE ships.name = outcomes.ship AND result = ‘sunk’GROUP BY class;

+-------------+-------+

| COUNT(ship) | class |

+-------------+-------+

| 1 | Kongo |

+-------------+-------+

1 row in set (0.01 sec)

49. Find for each class with at least three ships the number of ships of that class sunk in battle. SELECT COUNT(name), class FROM outcomes,

(SELECT name, Ships.class FROM ships,(SELECT DISTINCT S1.class FROM ships S1, ships S2, ships S3

WHERE S1.name != S2.name AND S2.name != S3.name AND S1.name != S3.name) P1

WHERE P1.class = ships.class) P2

WHERE outcomes.ship = P2.name AND outcomes.result = ‘sunk’ GROUP BY class;

+-------------+-------+

| COUNT(name) | class |

+-------------+-------+

| 1 | Kongo |

+-------------+-------+

1 row in set (0.02 sec)

50. The weight (in pounds) of the shell fired from a naval gun is approximately one half the cube of the bore (in inches). Find the average weight of the shell for each country’s ships. SELECT DISTINCT country, ((bore \* bore \* bore) / 2) as Weight FROM classes;

+-------------+-----------+

| country | Weight |

+-------------+-----------+

| Germany | 1687.5000 |

| USA | 2048.0000 |

| Japan | 1372.0000 |

| Gt. Britain | 1687.5000 |

| USA | 1372.0000 |

| Japan | 2916.0000 |

+-------------+-----------+

6 rows in set (0.00 sec)

Database 3: Product Catalog This database contains the following relations: Suppliers(sid:integer, sname:string, address:string) Parts(pid:integer, pname:string, color:string) Catalog(sid:integer, pid:integer, cost:real) The Catalog relation lists the prices charged for parts by suppliers. Write the following queries in SQL: 51. Find the pnames of parts for which there is some supplier. SELECT DISTINCT pname FROM parts, catalog WHERE catalog.pid = parts.pid;

+-----------------------------------+

| pname |

+-----------------------------------+

| Left Handed Bacon Stretcher Cover |

| Acme Widget Washer |

| I Brake for Crop Circles Sticker |

| Anti-Gravity Turbine Generator |

| Fire Hydrant Cap |

| 7 Segment Display |

+-----------------------------------+

6 rows in set (0.00 sec)

52. Find the snames of suppliers who supply every part.

SELECT DISTINCT sname FROM suppliers,(SELECT C.sid

FROM Catalog C

WHERE NOT EXISTS (

SELECT P.pid FROM Parts P

WHERE NOT EXISTS (

SELECT C1.sid FROM Catalog C1 WHERE C1.sid = C.sid AND C1.pid = P.pid))) P1

WHERE suppliers.sid = P1.sid;

Empty set (0.00 sec)

53. Find the snames of suppliers who supply every red part.

SELECT S.sname FROM Suppliers S

WHERE NOT EXISTS

(SELECT P.pid FROM Parts P

WHERE P.color = ‘red’ AND P.pid NOT IN

(SELECT C.pid FROM Catalog C

WHERE C.sid = S.sid AND C.pid = P.pid))

+----------------------+

| sname |

+----------------------+

| Big Red Tool and Die |

+----------------------+

1 row in set (0.00 sec)

54. Find the pnames of parts supplied by "Acme Widget Suppliers" and no one else.

SELECT pname FROM parts,

(SELECT C.pid as pid FROM Catalog C, suppliers

WHERE C.sid = suppliers.sid AND suppliers.sname = ‘Acme Widget Suppliers’ AND

C.pid NOT IN

(SELECT DISTINCT pid FROM Catalog C, suppliers S

WHERE C.sid = S.sid and S.sname != ‘Acme Widget Suppliers’)) P1

WHERE parts.pid = P1.pid;

+--------------------+

| pname |

+--------------------+

| Acme Widget Washer |

+--------------------+

1 row in set (0.00 sec)

55. Find the sids of suppliers who charge more for some part than the average cost of that part (averaged over all the suppliers who supply that part).

SELECT sid FROM Catalog,(SELECT AVG(cost) as averageCost, pid FROM CATALOG GROUP BY pid) P1

WHERE Catalog.cost > averageCost and P1.pid = Catalog.pid;

+-----+

| sid |

+-----+

| 2 |

| 1 |

| 3 |

+-----+

3 rows in set (0.00 sec)

56. For each part, find the sname of the supplier who charges the most for that part.

SELECT sname, pid FROM Suppliers,

(SELECT sid, Catalog.pid FROM Catalog,

(SELECT MAX(cost) as highest\_cost, pid FROM Catalog GROUP BY pid) P1

WHERE Catalog.pid = P1.pid and Catalog.cost = P1.highest\_cost) P2

WHERE P2.sid = Suppliers.sid;

+-----------------------+-----+

| sname | pid |

+-----------------------+-----+

| Acme Widget Suppliers | 4 |

| Big Red Tool and Die | 1 |

| Big Red Tool and Die | 3 |

| Perfunctory Parts | 8 |

| Perfunctory Parts | 9 |

| Alien Aircaft Inc. | 5 |

| Alien Aircaft Inc. | 6 |

| Alien Aircaft Inc. | 7 |

+-----------------------+-----+

8 rows in set (0.00 sec)

57. Find the sids of suppliers who supply only red parts.

SELECT DISTINCT catalog.sid FROM catalog

WHERE sid NOT IN

(SELECT DISTINCT catalog.sid FROM catalog, parts

WHERE parts.pid = catalog.pid and parts.color != ‘red’)

+-----+

| sid |

+-----+

| 2 |

+-----+

1 row in set (0.01 sec)

58. Find the sids of suppliers who supply a red part or a green part.

SELECT DISTINCT sid FROM Catalog, parts WHERE Catalog.pid = parts.pid AND (color = ‘red’ OR color = ‘green’);

+-----+

| sid |

+-----+

| 2 |

| 1 |

| 3 |

+-----+

3 rows in set (0.00 sec)

59. Find the sids of suppliers who supply a red part and a green part.

SELECT DISTINCT C2.sid FROM (SELECT sid, C1.pid, color FROM Catalog C1, parts WHERE C1.pid = parts.pid) C2, (SELECT sid, S1.pid, color FROM Catalog S1, parts WHERE S1.pid = parts.pid) C3

WHERE C2.color = ‘red’ and C3.color = ‘green’ AND C2.sid = C3.sid;

+-----+

| sid |

+-----+

| 3 |

+-----+

1 row in set (0.00 sec)

60. Find the sids of suppliers who supply at least two red parts and at least two green parts.

SELECT C1.sid FROM

(SELECT parts.pid, catalog.sid as sid, color

FROM catalog, parts WHERE catalog.pid = parts.pid AND color = ‘red’) C1,

(SELECT parts.pid, catalog.sid as sid, color

FROM catalog, parts WHERE catalog.pid = parts.pid AND color = ‘red’) C2,

(SELECT parts.pid, catalog.sid as sid, color

FROM catalog, parts WHERE catalog.pid = parts.pid AND color = ‘green’) C3,

(SELECT parts.pid, catalog.sid as sid, color

FROM catalog, parts WHERE catalog.pid = parts.pid AND color = ‘green’) C4

WHERE C1.sid = C2.sid AND C2.sid = C3.sid AND C3.sid = C4.sid AND C1.pid != C2.pid

AND C3.pid != C4.pid;

Empty set (0.00 sec)

61. For every supplier that only supplies green parts, print the name of the supplier and the total number of parts that she supplies.

SELECT sname, COUNT(pid) FROM Catalog,

(SELECT sid, sname FROM suppliers WHERE sid NOT IN (

SELECT sid FROM catalog WHERE pid IN (SELECT pid FROM parts WHERE color = ‘green’))) P1

WHERE P1.sid = Catalog.sid GROUP BY P1.sid;

+-----------------------+------------+

| sname | COUNT(pid) |

+-----------------------+------------+

| Acme Widget Suppliers | 3 |

| Big Red Tool and Die | 3 |

| Alien Aircaft Inc. | 3 |

+-----------------------+------------+

3 rows in set (0.00 sec)

62. For every supplier that supplies a green part and a red part, print the name and price of the most expensive part that she supplies.

SELECT sname, MAX(cost) FROM Suppliers, Catalog,

(SELECT DISTINCT S1.sid FROM Catalog S1, Catalog S2,

(SELECT C1.pid FROM parts C1, parts C2

WHERE C1.color = ‘red’ and C2.color = ‘green’) C3,

(SELECT C2.pid FROM parts C1, parts C2

WHERE C1.color = ‘red’ and C2.color = ‘green’) C4

WHERE C3.pid = S1.pid AND S2.pid = C4.pid AND S1.sid = S2.sid) P1

WHERE Suppliers.sid = P1.sid AND Catalog.sid = P1.sid;

+-------------------+-----------+

| sname | MAX(cost) |

+-------------------+-----------+

| Perfunctory Parts | 12.50 |

+-------------------+-----------+

1 row in set (0.00 sec)

Database 4: Company Consider the following relational schema. An employee can work in more than one department; the pct\_time field of the Works relation shows the percentage of time that a given employee works in a given department. Emp(eid:integer, ename:string, age:integer, salary:real)

Works(eid:integer, did:integer, pct\_time:integer)

Dept(did:integer, dname:string, budget:real, managerid:integer) 63. Print the names and ages of each employee who works in both the Hardware department and the Software department. SELECT P1.ename, P1.age FROM

(SELECT works.eid, works.did, ename, dname, age

FROM works, emp, dept

WHERE works.eid = emp.eid AND dept.did = works.did) P1,

(SELECT works.eid, works.did, ename, dname, age

FROM works, emp, dept

WHERE works.eid = emp.eid AND dept.did = works.did) P2

WHERE P1.eid = P2.eid AND P1.dname = ‘Hardware’ AND P2.dname = ‘Software’;

+----------------+------+

| ename | age |

+----------------+------+

| Mary Johnson | 44 |

| Stanley Browne | 23 |

+----------------+------+

2 rows in set (0.00 sec)

64. For each department with more than 20 full-time-equivalent employees (i.e., where the part-time and full-time employees add up to at least that many full-time employees), print the did together with the number of employees that work in that department.

SELECT works.did, COUNT(eid) FROM works,(SELECT did FROM

(SELECT did, SUM(pct\_time) AS time FROM works GROUP BY did) P1

WHERE P1.time >= 2000) P2

WHERE works.did = P2.did GROUP BY did;

+-----+------------+

| did | COUNT(eid) |

+-----+------------+

| 2 | 26 |

| 6 | 22 |

+-----+------------+

2 rows in set (0.01 sec)

65. Print the name of each employee whose salary exceeds the budget of all of the departments that he or she works in. SELECT ename FROM emp E

WHERE salary >

(SELECT SUM(budget) FROM works, dept WHERE

works.eid = E.eid and works.did = dept.did);

+-------------+

| ename |

+-------------+

| Linda Davis |

+-------------+

1 row in set (0.00 sec)

66. Find the managerids of managers who manage only departments with budgets greater than $1 million. SELECT managerid FROM dept

WHERE budget > 1000000;

+-----------+

| managerid |

+-----------+

| 141582651 |

| 287321212 |

| 578875478 |

+-----------+

3 rows in set (0.00 sec)

67. Find the enames of managers who manage the departments with the largest budgets.

SELECT ename FROM emp,

(SELECT managerid FROM dept

WHERE budget = (SELECT MAX(budget) FROM dept)) P1

WHERE emp.eid = P1.managerid;

+----------------+

| ename |

+----------------+

| Michael Miller |

| Edward Baker |

+----------------+

2 rows in set (0.00 sec)

68. If a manager manages more than one department, he or she controls the sum of all the budgets for those departments. Find the managerids of managers who control more than $5 million.

SELECT managerid FROM

(SELECT SUM(P3.budget) as total, P3.managerid FROM

(SELECT DISTINCT P1.budget, P1.did, P1.managerid

FROM dept P1, dept P2

WHERE P1.managerid = P2.managerid) P3

GROUP BY P3.managerid) P4

WHERE total > 5000000 ;

+-----------+

| managerid |

+-----------+

| 287321212 |

| 578875478 |

+-----------+

2 rows in set (0.00 sec)

69. Find the managerids of managers who control the largest amounts.

SELECT managerid FROM dept

WHERE budget = (SELECT MAX(total) FROM

(SELECT SUM(P3.budget) as total, P3.managerid FROM

(SELECT DISTINCT P1.budget, P1.did, P1.managerid

FROM dept P1, dept P2

WHERE P1.managerid = P2.managerid) P3

GROUP BY P3.managerid) P4);

+-----------+

| managerid |

+-----------+

| 287321212 |

| 578875478 |

+-----------+

2 rows in set (0.00 sec)

70. Find the enames of managers who manage only departments with budgets larger than $1 million, but at least one department with budget less than $5 million. SELECT ename FROM emp,

(SELECT DISTINCT P1.managerid FROM dept P1, dept P2

WHERE P1.budget > 1000000 AND P2.budget < 5000000

AND P1.managerid = P2.managerid AND P1.did != P2.did) P3

WHERE emp.eid = P3.managerid; +--------------+

| ename |

+--------------+

| Mary Johnson |

+--------------+

1 row in set (0.00 sec)

Database 5: University The university database has the following relations: Student(snum:integer, sname:string, major:string, level:string, age:integer) Class(cname:string, meets\_at:string, room:string, fid:integer) Enrolled(snum:integer, cname:string) Faculty(fid:integer, fname:string, deptid:integer) The meaning of these relations is straightforward; for example, Enrolled has one record per student-class pair such that the student is enrolled in the class. Write the following queries in SQL. No duplicates should be printed in any of the answers. 71. Find the names of all junior students (level = JR) who are enrolled in a class taught by Ivana Teach.

SELECT sname FROM student,

(SELECT snum FROM enrolled,

(SELECT cname FROM class,(SELECT fid FROM faculty WHERE fname = ‘Ivana Teach’) P1

WHERE class.fid = P1.fid) P2

WHERE enrolled.cname = P2.cname) P3

WHERE P3.snum = student.snum AND level = ‘JR’;

+--------------------+

| sname |

+--------------------+

| Christopher Garcia |

| Paul Hall |

+--------------------+

2 rows in set (0.00 sec)

72. Find the names of students not enrolled in any class. SELECT sname FROM student WHERE snum NOT IN

(SELECT DISTINCT snum FROM enrolled);

+-----------------+

| sname |

+-----------------+

| Maria White |

| Charles Harris |

| Angela Martinez |

| Thomas Robinson |

| Margaret Clark |

| Dorthy Lewis |

| Daniel Lee |

| Nancy Allen |

| Mark Young |

| Donald King |

| George Wright |

| Steven Green |

| Edward Baker |

+-----------------+

13 rows in set (0.01 sec)

73. Find the students who are enrolled in two classes taught by the same faculty. List student names together with the two class names. SELECT C1.snum, class1, class2 FROM enrolled C1, enrolled C2,

(SELECT P1.cname as class1, P2.cname as class2 FROM class P1, class P2

WHERE P1.fid = P2.fid AND P1.cname < P2.cname) P3

WHERE C1.cname = P3.class1 and C2.cname = P3.class2 and C1.snum = C2.snum;

+-----------+-----------------+-------------------------+

| snum | class1 | class2 |

+-----------+-----------------+-------------------------+

| 567354612 | Data Structures | Operating System Design |

+-----------+-----------------+-------------------------+

1 row in set (0.00 sec)

74. Find the names of faculty members who teach in every room in which some class is taught.

SELECT fname FROM faculty,

(SELECT C.fid

FROM faculty C

WHERE NOT EXISTS (

SELECT P.room FROM class P

WHERE NOT EXISTS (

SELECT C1.fid FROM class C1 WHERE C1.fid = C.fid AND C1.room = P.room))) P1

WHERE P1.fid = faculty.fid;

+-----------------+

| fname |

+-----------------+

| Richard Jackson |

+-----------------+

1 row in set (0.00 sec)

75. Print the level and the average age of students for that level, for each level.

SELECT level, AVG(age) FROM student GROUP BY level;

+-------+----------+

| level | AVG(age) |

+-------+----------+

| FR | 17.6667 |

| JR | 19.5000 |

| SO | 18.4000 |

| SR | 20.7143 |

+-------+----------+

4 rows in set (0.01 sec)

76. Print the level and the average age of students for that level, for all levels except JR.

SELECT level, AVG(age) FROM student

WHERE level != ‘JR’ GROUP BY level;

+-------+----------+

| level | AVG(age) |

+-------+----------+

| FR | 17.6667 |

| SO | 18.4000 |

| SR | 20.7143 |

+-------+----------+

3 rows in set (0.00 sec)

77. For each faculty member that has taught classes only in room R128, print the faculty member's name and the total number of classes she or he has taught.

SELECT faculty.fname, class.fid, COUNT(cname) FROM class, faculty,

(SELECT class.fid FROM class, (SELECT fid FROM faculty F1 WHERE NOT EXISTS

(SELECT class.room FROM class WHERE F1.fid = class.fid and class.room != ‘R128’)) P1

WHERE class.fid = P1.fid) P2

WHERE P2.fid = class.fid AND faculty.fid = P2.fid GROUP BY class.fid;

+--------------+-----------+--------------+

| fname | fid | COUNT(cname) |

+--------------+-----------+--------------+

| Robert Brown | 356187925 | 1 |

+--------------+-----------+--------------+

1 row in set (0.00 sec)

78. Find the names of students enrolled in the maximum number of classes. SELECT S1.snum FROM enrolled S1

WHERE NOT EXISTS

(SELECT E1.cname FROM class E1 WHERE NOT EXISTS

(SELECT S2.snum FROM enrolled S2

WHERE S2.snum = S1.snum AND E1.cname = S2.cname));

Empty set (0.00 sec)

79. Find the names of faculty members for whom the combined enrollment of the courses that they teach is less than five. SELECT fname FROM faculty,

(SELECT SUM(numStudent) as totalTaught, fid FROM

(SELECT P1.numStudent, fid, P1.cname FROM class,

(SELECT COUNT(snum) as numStudent, cname FROM enrolled GROUP BY cname) P1

WHERE class.cname = P1.cname) P2 GROUP BY fid) P3

WHERE P3.totalTaught < 5 AND faculty.fid = P3.fid;

+------------------+

| fname |

+------------------+

| John Williams |

| Elizabeth Taylor |

| Mary Johnson |

| William Moore |

| Patricia Jones |

| Richard Jackson |

| Jennifer Thomas |

+------------------+

7 rows in set (0.00 sec)

80. Find the age of the oldest student who is either a History major or enrolled in a course taught by Ivana Teach.

SELECT MAX(age) FROM

(SELECT snum, age FROM student

WHERE major = ‘History’

UNION

SELECT P3.snum as snum, age FROM student,

(SELECT snum FROM enrolled,

(SELECT cname FROM class,

(SELECT fid FROM faculty

WHERE fname = ‘Ivana Teach’) P1

WHERE class.fid = P1.fid) P2

WHERE P2.cname = enrolled.cname) P3

WHERE P3.snum = student.snum) P4;

+----------+

| MAX(age) |

+----------+

| 20 |

+----------+

1 row in set (0.00 sec)

81. Find the names of all classes that either meet in room R128 or have five or more students enrolled.

SELECT cname FROM class WHERE room =’R128’

UNION

SELECT cname FROM

(SELECT cname, COUNT(snum) as count FROM enrolled GROUP BY cname) P1

WHERE P1.count >= 5; +--------------------------+

| cname |

+--------------------------+

| Archaeology of the Incas |

| Dairy Herd Management |

| Data Structures |

| Intoduction to Math |

| Patent Law |

| Database Systems |

| Operating System Design |

+--------------------------+

7 rows in set (0.00 sec)

82. For each age value that appears in the Student table, find the level value that appears mostoften. For example, if there are more FR level students aged 18 than SR, JR, or SO students aged 18, you should print the pair (18, FR).

SELECT MAX(count) as maxCount, age, level FROM

(SELECT COUNT(snum) as count, age, level FROM student

WHERE level = ‘FR’ GROUP BY age

UNION

SELECT COUNT(snum) as count, age, level FROM student

WHERE level = ‘SO’ GROUP BY age

UNION

SELECT COUNT(snum) as count, age, level FROM student

WHERE level = ‘JR’ GROUP BY age

UNION

SELECT COUNT(snum) as count, age, level FROM student

WHERE level = ‘SR’ GROUP BY age) P1 GROUP BY age;

+----------+------+-------+

| maxCount | age | level |

+----------+------+-------+

| 2 | 17 | FR |

| 4 | 18 | FR |

| 3 | 19 | SO |

| 4 | 20 | JR |

| 4 | 21 | SR |

| 1 | 22 | SR |

+----------+------+-------+

6 rows in set (0.00 sec)

Database 6: Flights The flights database keeps track of airline flight information: Flights(flno:integer, origin:string, destination:string, distance:integer, departs:datetime, arrives:datetime, price:integer)

Aircraft(aid:integer, aname:string, cruisingrange:integer) Certified(eid:integer, aid:integer) Employees(eid:integer, ename:string, salary:integer) Note that the Employees relation describes pilots and other kinds of employees as well; every pilot is certified for some aircraft, and only pilots are certified to fly. Write each of the following queries in SQL: 83. Find the names of aircraft such that all pilots certified to operate them earn more than $80,000.

SELECT DISTINCT C.aid FROM certified C WHERE C.aid NOT IN

(SELECT DISTINCT aid FROM certified C1,

(SELECT eid FROM employees WHERE salary < 80000) P1

WHERE C1.eid = P1.eid);

+-----+

| aid |

+-----+

| 1 |

| 2 |

| 3 |

| 4 |

| 8 |

| 9 |

| 10 |

| 11 |

| 12 |

| 13 |

| 14 |

| 15 |

+-----+

12 rows in set (0.01 sec)

84. For each pilot who is certified for more than three aircraft, find the eid and the maximum cruising range of the aircraft for which she or he is certified.

SELECT P2.eid, MAX(P2.cruisingrange) FROM

(SELECT P1.eid, P1.aid, cruisingrange FROM aircraft,

(SELECT C1.eid, C1.aid FROM certified C1, certified C2, certified C3

WHERE C1.eid = C2.eid AND C2.eid = C3.eid AND C1.aid != C2.aid AND C2.aid != C3.aid

AND C1.aid != C3.aid) P1

WHERE aircraft.aid = P1.aid) P2

GROUP BY P2.eid;

+-----------+-----------------------+

| eid | MAX(P2.cruisingrange) |

+-----------+-----------------------+

| 141582651 | 6475 |

| 142519864 | 8430 |

| 269734834 | 8430 |

| 390487451 | 7120 |

| 552455318 | 3383 |

| 556784565 | 7120 |

| 567354612 | 8430 |

| 573284895 | 7120 |

+-----------+-----------------------+

8 rows in set (0.02 sec)

85. Find the names of pilots whose salary is less than the price of the cheapest route from Los Angeles to Honolulu.

SELECT ename FROM employees

WHERE employees.salary < (SELECT MIN(price) FROM flights WHERE origin = ‘Los Angeles’ AND destination = ‘Honolulu’) AND employees.eid IN (SELECT DISTINCT eid FROM certified);

+-------------+

| ename |

+-------------+

| Milo Brooks |

+-------------+

1 row in set (0.00 sec)

86. For all aircraft with cruising range over 1000 miles, find the name of the aircraft and the average salary of all pilots certified for this aircraft.

SELECT P3.aid, AVG(P3.salary) FROM

(SELECT P2.aid, P2.eid, E1.salary FROM employees E1,

(SELECT C1.aid, C1.eid FROM certified C1,(SELECT aid FROM aircraft WHERE cruisingrange > 1000) P1

WHERE C1.aid = P1.aid) P2

WHERE E1.eid = P2.eid) P3 GROUP BY aid;

+-----+----------------+

| aid | AVG(P3.salary) |

+-----+----------------+

| 1 | 244776.750000 |

| 2 | 191700.250000 |

| 3 | 217597.666667 |

| 4 | 220251.333333 |

| 5 | 182838.200000 |

| 6 | 118113.500000 |

| 8 | 205001.250000 |

| 9 | 242685.750000 |

| 10 | 189508.571429 |

| 11 | 257973.333333 |

| 12 | 209557.000000 |

| 13 | 243198.333333 |

| 14 | 201283.666667 |

| 15 | 273215.500000 |

+-----+----------------+

14 rows in set (0.01 sec)

87. Find the aids of all aircraft that can be used on routes from Los Angeles to Chicago.

SELECT aid FROM aircraft WHERE cruisingrange >= (SELECT distance FROM flights WHERE origin = ‘Los Angeles’ AND destination = ‘Chicago’);

+-----+

| aid |

+-----+

| 1 |

| 2 |

| 3 |

| 6 |

| 8 |

| 9 |

| 10 |

| 11 |

| 12 |

| 13 |

| 14 |

+-----+

11 rows in set (0.00 sec)

88. Identify the routes that can be piloted by every pilot who makes more than $100,000.

SELECT DISTINCT flno FROM flights

WHERE distance <=

(SELECT MIN(cruisingrange) FROM aircraft,

(SELECT DISTINCT aid FROM certified, (SELECT eid FROM employees WHERE salary > 100000) P1

WHERE certified.eid = P1.eid) P2

WHERE aircraft.aid = P2.aid);

+------+

| flno |

+------+

| 149 |

| 701 |

| 2223 |

| 4884 |

| 5694 |

| 7789 |

+------+

6 rows in set (0.01 sec)

89. Print the names of pilots who can operate aircrafts with cruising range greater than 3000 miles but are not certified on any Boeing aircraft.

SELECT ename FROM employees,(SELECT DISTINCT eid FROM certified,

(SELECT aid FROM aircraft WHERE cruisingrange >= 3000) P1

WHERE certified.aid = P1.aid AND certified.aid NOT IN

(SELECT aid FROM aircraft WHERE aname LIKE ‘Boeing%’)) P2

WHERE employees.eid = P2.eid;

+-----------------+

| ename |

+-----------------+

| Betty Adams |

| George Wright |

| Lawrence Sperry |

| Mark Young |

| Lisa Walker |

| Eric Cooper |

| Joseph Thompson |

| Angela Martinez |

| William Jones |

+-----------------+

9 rows in set (0.00 sec)

90. Compute the difference between the average salary of pilots and the average salary of all employees (including pilots).

SELECT (pilotSalary - otherSalary) as salaryDifference FROM

(SELECT AVG(salary) as otherSalary FROM employees) P1,(SELECT AVG(salary) as pilotSalary FROM employees

WHERE eid IN (SELECT DISTINCT eid FROM certified)) P2;

+------------------+

| salaryDifference |

+------------------+

| 11449.645161 |

+------------------+

1 row in set (0.01 sec)

91. Print the name and salary of every non-pilot whose salary is more than the average salary for pilots.

SELECT ename FROM employees

WHERE salary >

(SELECT AVG(salary) as pilotSalary FROM employees

WHERE eid IN (SELECT DISTINCT eid FROM certified))

AND eid NOT IN (SELECT DISTINCT eid FROM certified);

+----------------+

| ename |

+----------------+

| David Anderson |

+----------------+

1 row in set (0.00 sec)

92. Print the names of employees who are certified only on aircrafts with cruising range longer than 1000 miles. SELECT ename FROM employees,

(SELECT DISTINCT eid FROM certified WHERE eid NOT IN

(SELECT eid FROM certified WHERE aid IN

(SELECT aid FROM aircraft WHERE cruisingrange < 1000))) P1

WHERE employees.eid = P1.eid;

+------------------+

| ename |

+------------------+

| John Williams |

| Elizabeth Taylor |

| Mary Johnson |

| James Smith |

| Michael Miller |

| Joseph Thompson |

| Angela Martinez |

| Robert Brown |

| Lawrence Sperry |

| Karen Scott |

| Mark Young |

| Eric Cooper |

| William Jones |

+------------------+

13 rows in set (0.00 sec)

93. Print the names of employees who are certified only on aircrafts with cruising range longer than 1000 miles, but on at least two such aircrafts.

SELECT ename FROM employees,

(SELECT DISTINCT C1.eid FROM certified C1, certified C2 WHERE

C1.aid NOT IN (SELECT aid FROM aircraft WHERE cruisingrange < 1000)

AND C2.aid NOT IN (SELECT aid FROM aircraft WHERE cruisingrange < 1000)

AND C1.eid = C2.eid AND C1.aid != C2.aid AND C1.eid NOT IN

(SELECT DISTINCT eid FROM certified WHERE eid NOT IN

(SELECT eid FROM certified WHERE aid IN

(SELECT aid FROM aircraft WHERE cruisingrange < 1000)))) P1

WHERE employees.eid = P1.eid;

+---------------+

| ename |

+---------------+

| Betty Adams |

| George Wright |

| Lisa Walker |

| Larry West |

+---------------+

4 rows in set (0.00 sec)

94. Print the names of employees who are certified only on aircrafts with cruising range longer than 1000 miles and who are certified on some Boeing aircraft.

SELECT ename FROM employees,

(SELECT DISTINCT eid FROM certified WHERE eid NOT IN(SELECT DISTINCT eid FROM certified WHERE eid NOT IN

(SELECT eid FROM certified WHERE aid IN

(SELECT aid FROM aircraft WHERE cruisingrange < 1000)))

AND aid IN (SELECT aid from aircraft WHERE aname LIKE ‘Boeing%’)) P1

WHERE employees.eid = P1.eid;

+---------------+

| ename |

+---------------+

| Betty Adams |

| George Wright |

| Lisa Walker |

| Larry West |

+---------------+

4 rows in set (0.00 sec)

Again, to make grading easier, put your answers in a single plain text file (.txt) and submit to e-learning. For each question, include in your submission: 1. the question itself; 2. your SQL statement; and 3. query result from MySQL. Failure to follow this format may cause your homework not graded.