Chapter 6 Review Question and Exercises: 6.2, 6.16 (b, c, d, e, f, g, i, j) Chapter 8 Exercises: Specify the queries of exercises 6.16 (b, c, d, e, f, g, i, j) in SQL; 8.16 (b, c, e, f); 8.17 (b, c, d); 8.24(b, d) Chapter 10 Exercises: 10.11, 10.16, 10.19, 10.20, 10.21, 10.26, 10.28, 10.31, 10.33. Chapter 11 Exercises: 11.30. Chapter 17 Exercises: 17.17, 17.22(b, c, d), 17.23 Chapter 18: 18.1.

6.2	Union compatibility describes two relations who have the same degree and whose attribute domains match for all attributes. Required for UNION, INTERSECTION, and DIFFERENCE because these operations examine tuples between the two operand relations and keep or discard the tuple based on the operation. If the two tuples had different degrees or had attributes with different domain values, the comparison would not make logical sense and couldn't be done.

6.16 and SQL

b

$R \leftarrow \pi_{Lname,Fname}(EMPLOYEE \bowtie_{Ssn=Essn\ AND\ Fname=Dependent\_name}\ DEPARTMENT)$

```
SELECT LNAME, FNAME
FROM EMPLOYEE
WHERE EXISTS
(
  SELECT *
  FROM DEPENDENT
  WHERE FNAME=DEPENDENT_NAME AND ESSN=SSN
);

no rows selected
```

c

$WONG\_SSN \leftarrow \pi_{Ssn}\left(\sigma_{Fname='Franklin'\ AND\ Lname='Wong'}(EMPLOYEE)\right)$

$R \leftarrow \pi_{Lname,Fname}(EMPLOYEE \bowtie_{Super\_ssn=Ssn}\ WONG\_SSN)$

```
SELECT LNAME, FNAME
FROM EMPLOYEE
WHERE SUPERSSN IN
(
  SELECT SSN
  FROM EMPLOYEE
  WHERE FNAME='Franklin' AND LNAME='Wong'
);

LNAME           FNAME
--------------- ---------------
Smith           John
Narayan         Ramesh
English         Joyce
```

d

$AVG\_HOURS \leftarrow_{Pno} F_{AVG(Hours)}(WORKS\_ON)$

$R \leftarrow \pi_{Pname,Hours}(PROJECT \bowtie_{Pnumber=Pno}\ AVG\_HOURS)$

```
SELECT PNAME, SUM(HOURS)
FROM PROJECT, WORKS_ON
WHERE PNUMBER=PNO
GROUP BY PNAME
ORDER BY SUM(HOURS) ASC
;

PNAME                SUM(HOURS)
-------------------- ----------
Reorganization               25
```

		ProductY 37.5 ProductZ 50 ProductX 52.5 Newbenefits 55 Computerization 55

	e	$EMP\_PROJ(Ssn, Pno) \leftarrow \pi_{Essn,Pno}(WORKS\_ON)$
		$PROJ\_NUMS(Pno) \leftarrow \pi_{Pnumber}(PROJECT)$
		$EMP\_SSNS \leftarrow EMP\_PROJ \div PROJ\_NUMS$
		$R \leftarrow \pi_{Lname,Fname}(EMPLOYEE * EMP\_SSNS)$
		
		SELECT LNAME, FNAME
		FROM EMPLOYEE
		WHERE NOT EXISTS (
		  SELECT PNUMBER
		  FROM PROJECT
		  WHERE NOT EXISTS (
		    SELECT *
		    FROM WORKS_ON
		    WHERE PNUMBER=PNO AND ESSN=SSN
		  )
		);
		
		no rows selected

	f	$NONWORKER \leftarrow \pi_{Ssn}(EMPLOYEE) - \pi_{Essn}(WORKS\_ON)$
		$R \leftarrow \pi_{Lname,Fname}$
		
		SELECT LNAME, FNAME
		FROM EMPLOYEE
		WHERE NOT EXISTS (
		  SELECT *
		  FROM WORKS_ON
		  WHERE ESSN=SSN
		);
		
		no rows selected

	g	$EMP\_DEP \leftarrow EMPLOYEE \bowtie_{Dno=Dnumber} DEPARTMENT$ $AVG\_HOURS \leftarrow_{Dname} F_{AVG(Salary)}(EMP\_DEP)$ SELECT DNAME, AVG(SALARY) FROM DEPARTMENT, EMPLOYEE WHERE DNUMBER=DNO GROUP BY DNAME ORDER BY AVG(SALARY) ASC DNAME AVG(SALARY) --------------- ----------- Administration 31000 Research 33250 Headquarters 55000
	i	$HOUSTON\_PROJS \leftarrow \pi_{Pnumber}(PROJECT)$ $HOUSTON\_PROJ\_EMPS(Ssn) \leftarrow \pi_{Essn}(WORKS\_ON * HOUSTON\_PROJS)$ $HOUSTON\_DEPTS \leftarrow \pi_{Pnumber}\left(\sigma_{Dlocation='Houston'}(DEPT\_LOCATIONS)\right)$ $NON\_HOUSTON\_DEPT\_EMPS(Ssn) \leftarrow (EMPLOYEE \bowtie_{Dno \neq Dnumber} HOUSTON\_DEPTS)$ $R\_SSNS \leftarrow HOUSTON\_EMPS \cap NON\_HOUSTON\_DEPT\_EMPS$

$R \leftarrow \pi_{Lname,Fname,Address}(R\_SSNS * EMPLOYEE)$

```
SELECT LNAME, FNAME, ADDRESS
FROM EMPLOYEE
WHERE EXISTS (
  SELECT *
  FROM WORKS_ON, PROJECT
  WHERE PLOCATION='Houston' AND PNO=PNUMBER AND SSN=ESSN
)
AND NOT EXISTS (
  SELECT *
  FROM DEPT_LOCATIONS
  WHERE DNO=DNUMBER AND DLOCATION='Houston'
);

LNAME           FNAME           ADDRESS
--------------- --------------- ------------------------------
Wallace         Jennifer        291 Berry, Bellaire, TX
```

j

$DEPT\_MGR(Ssn) \leftarrow \pi_{Mgr\_ssn}(DEPARTMENT)$
$DEPND(Ssn) \leftarrow \pi_{Essn}(DEPENDENT)$
$MGR\_NO\_DEPND \leftarrow DEPT\_MGR * DEPND$
$R \leftarrow \pi_{Lname,Fname}(MGR\_NO\_DEPND * EMPLOYEE)$

```
SELECT LNAME, FNAME
FROM EMPLOYEE
WHERE EXISTS (
  SELECT *
  FROM DEPARTMENT
  WHERE SSN=MGRSSN
)
AND NOT EXISTS (
  SELECT *
  FROM DEPENDENT
  WHERE SSN=ESSN
);

LNAME           FNAME
--------------- ---------------
Borg            James
```

8.16

b

```
SELECT Course_name
FROM COURSE
WHERE Course_number IN
(
  SELECT Course_number
  FROM SECTION
  WHERE Instructor=’King’ AND (Year=’04’ OR Year=’05’)
);
```

c

```
SELECT Course_number, Semester, Year, COUNT(*)
FROM SECTION, GRADE_REPORT
WHERE Instructor='King' AND SECTION.Section_identifier =
GRADE_REPORT.Section_identifier
GROUP BY Course_number, Semester, Year
```

e

```
SELECT Name, Major
FROM STUDENT
WHERE Student_number IN
(
```

		SELECT Student_number FROM GRADE_REPORT WHERE Student_number NOT IN (SELECT Student_number FROM GRADE_REPORT WHERE Grade<>'A'));
	f	SELECT Name, Major FROM STUDENT WHERE Student_number IN (SELECT Student_number FROM GRADE_REPORT WHERE Student_number NOT IN (SELECT Student_number FROM GRADE_REPORT WHERE Grade='A'));
8.24	b	CREATE VIEW RESEARCH_EMPLOYEE_SUMMARY (EmpLname, EmpFname, MgrLname, MgrFname, Salary) AS SELECT… get research employees… SELECT Lname, Fname FROM EMPLOYEE WHERE Dno IN (SELECT Dnumber FROM DEPARTMENT WHERE Dname='Research'); SELECT Lname, Fname FROM EMPLOYEE WHERE
	d	My girlfriend and I broke up last night.
10.11		A minimal set of functional dependencies • Has every FD with a single attribute for its right hand side • Has every FD with its left hand side as a minimal set of attributes (we cannot remove any attributes without causing the new FD to be nonequivalent) • And from which we cannot remove any FD without the new FD being nonequivalent The minimal set of FDs is not always unique. Every set of FDs does have a minimal equivalent set. For example, if the set of FDs is the minimal set, it is equivalent to itself. If the set of FDs isn't the minimal set, it can be decomposed into the minimal set using the algorithms in the book and the closure of the new set is equivalent to the closer of the old set, and the FD sets are thus equivalent.

10.16		Boyce-Codd Normal form is a normal form "higher" than 3NF. A relation is in BCNF if whenever a nontrivial FD X->A holds in R, then X is a superkey of R. This is different from 3NF because 3NF says the FDs are also allowed to have A as a prime attribute in R.
10.19		F = {A->C, AC->D, E->AD, E->H} G = {A->CD, E->AH} F+ = {A, C, E, H, D} G+ = {A, E, C, H, D} G+ == F+ => equivalent.
10.20		G = {Ssn->{Edate, Bdate, Address, Dnumber}, Dnumber->{Dname, Dmgr_ssn}} {Ssn+} = {Edate, Bdate, Address, Dnumber, Dname, Dmgr_ssn} {Dnumber+} = {Dname, Dmgr_ssn}
10.21		No. The set of FDs is not minimal because: • Not every FD has a single attribute for its right hand side The attempted changes are: Change FDs to canonical form (with 1 attribute on rhs): 1) Ssn->Edate 2) Ssn->Bdate 3) Ssn->Address 4) Ssn->Dnumber 5) Dnumber->Dname 6) Dnumber->Dmgr_ssn Remove redundant attributes from LHS: No removals (as LHS is all single attribute). Attempt to remove redundant FDs: Remove i)? => G+ loses attribute. No removals (as this results in loss of attributes in G+). Therefore the list above is minimal. Prove by: G+ = {Ssn+} U {Dnumber+}.
10.26		F = {AB->C, A->DE, B->F, F->GH, D->IJ} The key is AB since there is no single attribute whose closure contains all attributes in R. AB therefore contains a minimal number of attributes and its closure is all attributes in R: {AB}+= ABCDEFGHIJ. To obtain 2NF compute: A+ = A, D, E, I, J B+ = B, F, G, H. So for 2NF: R1(A, B, C) and R2(A, D, E, I, J) and R3(B, F, G, H). To obtain 3NF compute look at the FDs in F which are valid for R1, R2, and R3 and break down any transitive dependencies. R11(A, D, E) and R12(D, I, J) and R21(B, F) and R22(F, G, H) and R3(A, B, C).
10.28	a	A->B does not work because of tuples 1 and 2. B->C could work. C->B does not work because of tuples 1 and 3. B->A does not work because of tuples 1 and 5. C->A does not work because of tuples 1 and 3.
	b	Tuple# is a CK.

		A is not a CK because of tuples 1 and 5. B is a CK because it determines {AC}. C is a CK because it determines {AB}.
10.31		After a natural join the relation looks like: R(O#, I#, Qty_ordered, Total_price, Discount%, Odate, Cust#, Total_amount) and the FDs for it are: {O#, I#}->{Qty_ordered} {O#, I#}->{Total_price} {O#, I#}->{Discount%} {O#}->{Odate} {O#}->{Cust#} {O#}->{Total_amount} This is not in 2NF ecause Odate, Cust#, and Total_amount only partially depend on the joined key {O#, I#}. Since it isn't in 2NF then it isn't in 3NF.
10.33	a	The relation isn't in 3NF because it isn't in 2NF. It isn't in 2NF because none of the attributes are fully functionally dependent on the key.
	b	{Author_name, Book_title} is the key because its closure is all attributes. First decompose to 2NF: Author_name->Author_affil is PD on the PK, so it becomes its own relation. Book_title->Publisher, Book_type is PD on the PK, so it becomes its own relation. Book_type->List_price is not PD on the PK so it does not become its own relation. So we now have: R0(Book_title, Author_name) R1(Author_name, Author_affil) R2(Book_title, Publisher, Book_type, List_price) Further decomposition into 3NF is necessary because of the Book_type->List_price dependency. R0(Book_title, Author_name) R1(Author_name, Author_affil) R21(Book_title, Publisher, Book_type) R2(Book_type, List_price)
11.30	a	{M}+ = {MP, C} which is not all attributes and so M is not a CK. {M, Y}+ = {MP, P, C, Y, M} which IS all attributes and so {M, Y} is a CK. {M, C}+ = {MP, C, M} which is not all attributes and so {M, C} is not a CK.
	b	The relation isn't in 2NF because MP has a partial dependency on only M of the PK. Therefore it is not in 3NF or BCNF.
	c	R(M, Y, P, MP, C) R1(M, Y, P) R2(M, MP, C) Compute the closures of each FD in F as in part a. R1 INTERSECT R2 = M and R2 - R1 = {MP, C}. And so {M}->{MP, C} is part of F+, so this has the lossless property.