1.

1.1.
select m.DIR from Movie m, Cast c where m.title = c.title and c.actor = 'Harrison Ford';
\[ \pi_{m.DIR}(\sigma_{m.title=c.title \text{ and } c.actor='Harrison Ford'}(\text{Movie } m \bowtimes \text{ Cast } c)) \]

1.2.
select c1.actor from Cast c1 where c1.title = 'Star Wars' and not exists
(select * from Cast c2
    where c1.actor = c2.actor and c2.title = 'Return of the Jedi');
\[ \pi_{c1.actor}(\sigma_{c1.title='Star Wars'}(\text{Cast } c1) \bowminus \pi_{c2.actor}(\sigma_{c2.title='Return of the Jedi'}(\text{Cast } c2)) \]

1.3.
select distinct c0.actor from Cast c0
where not exists
(select * from Cast c1
    where c1.actor = 'Harrison Ford' and not exists
    (select * from Cast c2 where c2.title = c1.title and c2.actor = c0.actor));
\[ \text{Cast } \bowminus \pi_{c.title}(\sigma_{c.actor='Harrison Ford'}(\text{Cast } c)) \text{ (here } \bowminus \text{ means quotient)} \]

1.4.
select c.actor, count(*)
from Movie m, Cast c
where m.dir='Steven Spielberg' and m.title=c.title
group by c.actor

1.5.
create view AvgBudget as
select avg(budget) as amt from Movie;

create view OverBudget as
select m.title as title
from Movie m
where m.budget >= ALL
    (select amt from AvgBudget);

create view Num as
(select c.actor as actor, count(*) as ct
from Cast c, Overbudget o
where c.title=o.title
group by c.actor)
union
(select distinct c.actor as actor, 0 as ct
from Cast c
where not exists
    (select * from Overbudget o
        where o.title = c.title));
select c.actor from Num
where c.ct in
    select max(ct) from Num;

2.
2.1.
select origin, count(*) from {
(select distinct c0.origin, c1.destination from Connection c0, Connection c1
where c0.destination = c1.origin and c0.airline = c1.airline and c0.origin <> c1.destination)
union
(select distinct c3.origin, c3.destination from Connection c3)
} group by origin;

2.2.
with recursive Airpath(origin, destination) as
(select origin, destination from Connection)
union
(select p.origin, c.destination
from Airpath p, Connection c
where p.destination = c.origin and p.destination <> 'Boston')
select destination from Airpath where origin = 'Buffalo';

3.
3.1.
selects all actors that starred in a movie which was directed by Steven Spielberg and in which Harrison Ford starred.

$$\pi_{c2.actor}(\sigma_{m.DIR='Steven Spielberg' \text{ and } c1.actor='Harrison Ford' \text{ and } m.title=c1.title=c2.title}
(Movie m \times Cast c1 \times Cast c2))$$

3.2.
selects all directors together with date of first movie they directed

$$\pi_{m0.DIR, m0.year (Movie m0 )} - \pi_{m2.DIR, m2.year (\sigma_{m1.DIR=m2.DIR \text{ and } m1.year<m2.year(Movie m1 \times Movie m2)})}$$

4.
Both queries select employees from each department that have lowest salary in the department.

To show that they are equivalent it's enough to show that their respective sub-queries always both return empty result set or both return non-empty result set (for every e1).
Let's denote:

```sql
SELECT * FROM EMP e2
WHERE e1.DEPT=e2.DEPT AND e1.SALARY>e2.SALARY as Q1
and
SELECT * FROM EMP e2, EMP e3
WHERE e1.DEPT=e2.DEPT AND e2.DEPT=e3.DEPT AND e1.SALARY>e3.SALARY as Q2.
```

Without loss of generality we can assume that Q2 is

```sql
(SELECT * FROM EMP e2, EMP e3
WHERE e1.DEPT=e2.DEPT AND e2.DEPT=e3.DEPT AND e1.DEPT = e3.DEPT AND e1.SALARY>e3.SALARY)
```

because of transitivity of equality.

If Q1 returns empty result set it means that there is no employee in e1.Dept with smaller salary than e1, therefore in Q2 there also won't be any e3 which would be in e1.DEPT and would have smaller salary than e1, so Q2 will return empty set too.

If Q1 returns non-empty result set it means that there is an employee in e1.Dept with smaller salary than e1, therefore in Q2 when that employee will be set as e2 and e3 then Q2 will also return a row.

5.

```sql
create view temp as
select a1, a2, ..., an, count(*) as num
from(
    select a1, a2, ..., an, 1 as num, a1 as id from f union
    select a1, a2, ..., an, 2 as num, a2 as id from f union
    ...
    select a1, a2, ..., an, n as num, an as id from f)
where id=1
group by a1, a2, ..., an;

select a1, a2, ..., an from temp where num = (select max(num) from temp);
```