```
-- SELECT Title_ID, avg_rating FROM titles WHERE avg_rating <> 0.0;
```

```
-- SELECT movie_ID, name_Id, name FROM (SELECT movie_Id, name_Id FROM principals WHERE (category='actor' OR category='actress')) as foo NATURAL JOIN (SELECT name_Id, primary_name as name FROM names) as bar;
```

```
-- Step #1: combine the info of movie + actor + names
SELECT movie_Id, title, actor_Id, actor_name, avg_rating
INTO pairs
FROM (
  SELECT movie_Id, name_Id as actor_Id
  FROM principals
  WHERE (
    category='actor'
    OR
    category='actress'
  )
) as foo
NATURAL JOIN (
  SELECT name_Id as actor_Id, primary_name as actor_name
  FROM names
) as bar
NATURAL JOIN (
  SELECT title_id as movie_id, title, avg_rating
  FROM titles
  WHERE (
     avg_rating <> 0.0
  )
)as baz;

--  movie_id  |           title            | actor_id  |    actor_name     |
avg_rating
-- -----------+----------------------------+-----------+-------------------+---------
---
--  tt0001812 | Oedipus Rex                | nm0294276 | Theo Frenkel      |
6.3
--  tt0001008 | The Prince and the Pauper  | nm0874364 | Mabel Trunnelle   |
5.6
--  tt0001240 | Hamlet                     | nm0627274 | Alwin Neuß        |
3.9
--  tt0001240 | Hamlet                     | nm0742120 | Carl Rosenbaum    |
3.9
```

```
--  tt0004249 | Lola                        | nm0290066 | Alec B. Francis    |
5.2
--  tt0004325 | The Merchant of Venice      | nm0806565 | Phillips Smalley   |
5.2
--  tt0004325 | The Merchant of Venice      | nm0916665 | Lois Weber         |
5.2
--  tt0004325 | The Merchant of Venice      | nm0534221 | Jeanie Macpherson  |
5.2
--  tt0004325 | The Merchant of Venice      | nm0933424 | Fred L. Wilson     |
5.2
--  tt0004707 | Tillie's Punctured Romance | nm0000122 | Charles Chaplin    |
6.3
```

```
-- Step #2: get the joint pairs
SELECT a.movie_id, a.actor_name as actor_a, b.actor_name as actor_b, a.avg_rating
INTO movie_pair
FROM pairs a, pairs b
WHERE (
  a.movie_id = b.movie_id
  AND
  a.actor_name <> b.actor_name
);

--  movie_id  |     actor_a     |     actor_b      | avg_rating
-- -----------+-----------------+------------------+------------
--  tt0001812 | Theo Frenkel    | Suzanne de Baere |        6.3
--  tt0001008 | Mabel Trunnelle | Mark Twain       |        5.6
--  tt0001008 | Mabel Trunnelle | William Sorelle  |        5.6
--  tt0001008 | Mabel Trunnelle | Charles Ogle     |        5.6
--  tt0001008 | Mabel Trunnelle | Cecil Spooner    |        5.6
--  tt0001240 | Alwin Neuß      | Oscar Langkilde  |        3.9
--  tt0001240 | Alwin Neuß      | Emilie Sannom    |        3.9
--  tt0001240 | Alwin Neuß      | Einar Zangenberg |        3.9
--  tt0001240 | Alwin Neuß      | Aage Hertel      |        3.9
--  tt0001240 | Alwin Neuß      | Ella La Cour     |        3.9
```

```
-- Step #3: get chem and show the top 10
SELECT actor_a, actor_b, AVG(avg_rating) as chem
FROM movie_pair
GROUP BY actor_a, actor_b
ORDER BY chem DESC LIMIT 10;

--     actor_a     |       actor_b        |        chem
-- ----------------+----------------------+--------------------
```

```
--  Ally Beardsley | Brian Murphy          | 10.0000000000000000
--  Alva Lillie    | Mike Henderson        | 10.0000000000000000
--  Ally Beardsley | Brennan Lee Mulligan  | 10.0000000000000000
--  Akbar Kurtha   | Jacqueline Leonard    | 10.0000000000000000
--  Ally Beardsley | Emily Axford          | 10.0000000000000000
--  Alva Lillie    | Jessika Kwasniak      | 10.0000000000000000
--  Akbar Kurtha   | Christopher Timothy   | 10.0000000000000000
--  Akbar Kurtha   | Mark Frost            | 10.0000000000000000
--  Akbar Kurtha   | Corrinne Wicks        | 10.0000000000000000
--  Alva Lillie    | Terri J. Lee          | 10.0000000000000000
```

Helper commands...

```
SELECT name_id as name_id_a, name_id as name_id_b, primary_name as name_a,
primary_name as name_b
INTO double_names
FROM names;

--  name_id_a | name_id_b |     name_a      |     name_b
-- -----------+-----------+-----------------+-----------------
--  nm0000001 | nm0000001 | Fred Astaire    | Fred Astaire
--  nm0000002 | nm0000002 | Lauren Bacall   | Lauren Bacall
--  nm0000003 | nm0000003 | Brigitte Bardot | Brigitte Bardot
--  nm0000004 | nm0000004 | John Belushi    | John Belushi
--  nm0000005 | nm0000005 | Ingmar Bergman  | Ingmar Bergman
--  nm0000006 | nm0000006 | Ingrid Bergman  | Ingrid Bergman
--  nm0000007 | nm0000007 | Humphrey Bogart | Humphrey Bogart
--  nm0000008 | nm0000008 | Marlon Brando   | Marlon Brando
--  nm0000009 | nm0000009 | Richard Burton  | Richard Burton
--  nm0000010 | nm0000010 | James Cagney    | James Cagney
```

```
SELECT a.name_a, a.name_b, b.chem
-- INTO hollywoodPairs
FROM double_names a JOIN movie_pair b
ON (
  B.actor_a = A.name_id_a
  AND
  B.actor_b = A.name_id_b
)
GROUP BY B.actor_a, B.actor_b
ORDER BY chem DESC LIMIT 10;
```

```sql
-- #problem: transfer id to name
SELECT actor_a, name_a, actor_b, name_b, chem
FROM (
  SELECT actor_a, actor_b, AVG(avg_rating) as chem
  FROM movie_pair
) as foo
NATURAL JOIN (
  SELECT actor_name as name_a, actor_id
  FROM pairs
) as bar
NATURAL JOIN (
  SELECT actor_name as name_b, actor_id
  FROM pairs
) as baz
WHERE (
  foo.actor_a = bar.actor_id
  AND
  foo.actor_b = baz.actor_id
)
GROUP BY actor_a, actor_b
ORDER BY chem DESC LIMIT 10;
```