

1.

- a. This query will return year, semester, sec_id, course_id, and the average of tot_cred for different combinations of year, semester, sec_id, course_id which exists as a valid record and are all in the year 2009 along with the fact that the specific course held in the corresponding semester has had more than 2 enrollments. Hence, joining with section table will only add more details about the taken course since every section_id record in takes and section are one to one.
- b. Yes; the query on the left hand will teachers information with respect to what they have taught and the query on the right hand will return the instructors' information along with the courses of the same department in which the instructor is hired.
- c. Just when one student does not have any record in the takes table i.e. has not taken any courses yet.

2.

- a. $\max(\text{union}(S1, S2)) = \max(\max(S1), \max(S2))$
 $\min(\text{union}(S1, S2)) = \min(\min(S1), \min(S2))$
SUM and COUNT cannot be determined since no information is available about the intersect of these two sets. If they were,
 $\text{SUM}(\text{UNION}(S1, S2)) = \text{SUM}(S1) + \text{SUM}(S2) - \text{SUM}(\text{INTERSECT}(S1, S2))$
 $\text{COUNT}(\text{UNION}(S1, S2)) = \text{COUNT}(S1) + \text{COUNT}(S2) - \text{COUNT}(\text{INTERSECT}(S1, S2))$

- b. $EC = \text{count} \mid ES = \text{sum} \mid E_{Mi} = \text{min} \mid E_{Ma} = \text{max} \mid EA = \text{avg} \mid ESD = SD$
 $S = (A, B, C)$
 $\text{SELECT } A, B, C,$
 $\quad \text{sum}(ES) / \text{sum}(EC) \text{ AS AVG},$
 $\quad \text{sum}((SD + EA * EA) * EC) / \text{sum}(EC) - \text{AVG} * \text{AVG} \text{ AS SD}$
 $\text{FROM aggregate-on-T GROUP BY } (A, B, C)$

3.

- a. $\text{SELECT } a, b, c, d \text{ GROUP BY ROLLUP}(a, b, c, d)$
 UNION
 $\text{SELECT } a, b, c, d \text{ GROUP BY ROLLUP}(b, c, d, a)$
 UNION
 $\text{SELECT } a, b, c, d \text{ GROUP BY ROLLUP}(c, d, a, b)$
 UNION
 $\text{SELECT } a, b, c, d \text{ GROUP BY ROLLUP}(d, a, b, c)$
 UNION
 $\text{SELECT NULL, b, NULL, d \text{ GROUP BY ROLLUP}(b, d)}$
 UNION
 $\text{SELECT } a, \text{NULL}, c, \text{NULL} \text{ GROUP BY ROLLUP}(a, c)$

```

SELECT a,b,c,d GROUP BY (a,b,c,d)
UNION
SELECT a,b,c,NULL GROUP BY (a,b,c)
UNION
SELECT NULL,b,c,d GROUP BY (b,c,d)
UNION
SELECT a,b,NULL,d GROUP BY (a,b,d)
UNION
SELECT a,NULL,c,d GROUP BY (a,c,d)
UNION
SELECT a,b,NULL,NULL GROUP BY (a,b)
UNION
SELECT a,NULL,c,NULL GROUP BY (a,c)
UNION
SELECT a,NULL,NULL,d GROUP BY (a,d)
UNION
SELECT NULL,b,c,NULL GROUP BY (b,c)
UNION
SELECT NULL,b,NULL,d GROUP BY (b,d)
UNION
SELECT NULL,NULL,c,d GROUP BY (c,d)
UNION
SELECT a,NULL,NULL,NULL GROUP BY (a)
UNION
SELECT NULL,b,NULL,NULL GROUP BY (b)
UNION
SELECT NULL,NULL,c,NULL GROUP BY (c)
UNION
SELECT NULL,NULL,NULL,d GROUP BY (d)

```

b.

```

SELECT D,
       NTILE(20) OVER (
         ORDER BY A) HIST
FROM r

```

c. Not any rollup can be rewritten by except and unions.

d.

Building	Room_Number	Time_slot_id	
ECE_Dept	32	A	1
ECE_Dept	32	NULL	1
ECE_Dept	30	B	1
ECE_Dept	30	NULL	1

ECE_Dept	NULL	NULL	2
MEC_Dept	25	A	1
MEC_Dept	25	NULL	1
MEC_Dept	20	C	1
MEC_Dept	20	NULL	1
MEC_Dept	NULL	NULL	2
IND_Dept	17	D	1
IND_Dept	17	NULL	1
IND_Dept	34	C	1
IND_Dept	34	NULL	1
IND_Dept	NULL	NULL	2
NULL	NULL	NULL	6

4. H. By using INNER JOIN for Supervisor and Employee table on ID .

5.

```
WITH Tmp(actor_id, store_id, counted_movies) AS (
    SELECT actor_id, store_id, COUNT(DISTINCT film_id)
    FROM actor INNER JOIN film_actor USING(actor_id)
    INNER JOIN film USING(film_id)
    INNER JOIN inventory USING(film_id)
    GROUP BY (actor_id, store_id)
)

SELECT store_id, actor.first_name, actor.last_name FROM Tmp
INNER JOIN actor USING(actor_id)
WHERE Tmp.counted_movies = (SELECT max(counted_movies) FROM Tmp AS T
                           WHERE T.store_id = Tmp.store_id)
```

Data Output

	store_id smallint	first_name character varying (45)	last_name character varying (45)
1	2	Walter	Torn
2	1	Gina	Degeneres

```

SELECT country, city, COUNT(inventory_id), SUM(amount)
FROM inventory
INNER JOIN rental USING(inventory_id)
INNER JOIN customer USING(customer_id)
INNER JOIN address USING(address_id)
INNER JOIN city USING(city_id)
INNER JOIN country USING(country_id)
INNER JOIN payment USING(rental_id)
GROUP BY ROLLUP(country, city)
ORDER BY country

```

Data Output

	country character varying (50)	city character varying (50)	count bigint	sur nur
1	Afghanistan	Kabul	18	
2	Afghanistan	[null]	18	
3	Algeria	Batna	27	
4	Algeria	Bchar	23	
5	Algeria	Skikda	32	
6	Algeria	[null]	82	
7	American Samoa	Tafuna	15	
8	American Samoa	[null]	15	
9	Angola	Benguela	20	
10	Angola	Namibe	25	
11	Angola	[null]	45	

Data Output Messages

dvdrental/postgres@Local

Query Editor Query History

```

1 WITH Tmp AS
2 (
3   SELECT film_id, rental_rate, replacement_cost
4   NTILE(6) OVER(ORDER BY length) AS Group
5   FROM film
6 )
7
8 SELECT film_id, Tmp.Group, rental_rate, replacement_cost,
9 SUM(replacement_cost) OVER
10 (PARTITION BY Tmp.Group ORDER BY rental_rate ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING)
11 FROM Tmp
12 ORDER BY Tmp.Group, rental_rate
13

```

Data Output

	film_id integer	group integer	rental_rate numeric (4,2)	replacement_cost numeric (5,2)	sum numeric
1	586	1	0.99	27.99	27.99
2	581	1	0.99	16.99	44.98
3	516	1	0.99	18.99	63.97
4	97	1	0.99	24.99	88.96
5	634	1	0.99	14.99	103.95
6	947	1	0.99	16.99	120.94
7	866	1	0.99	28.99	149.93
8	894	1	0.99	23.99	173.92
9	598	1	0.99	22.99	196.91
10	799	1	0.99	26.99	223.90
11	18	1	0.99	27.99	251.89

Data Output Messages

Notification:

```

INNER JOIN city USING(city_id)
INNER JOIN country USING(country_id)
GROUP BY country, rating)

SELECT *
FROM (SELECT country, ct AS G
      FROM Tmp
      WHERE rating = 'G') AS s1
INNER JOIN (SELECT country, ct AS PG
            FROM Tmp
            WHERE rating = 'PG') AS s2 USING (country)
INNER JOIN (select country, ct AS PG13
            FROM Tmp
            WHERE rating = 'PG-13') AS s3 USING (country)
INNER JOIN (select country, ct AS R
            FROM Tmp
            WHERE rating = 'R') AS s4 USING (country)
INNER JOIN (SELECT country, ct AS NC17
            FROM Tmp
            WHERE rating = 'NC-17') AS s5 USING (country)

```

Data Output

	country character varying (50)	g bigint	pg bigint	pg13 bigint
1	Malaysia	19	21	1
2	Iran	51	41	4
3	South Africa	51	48	6
4	Malawi	6	6	
5	Afghanistan	3	2	
6	United Kingdom	40	36	4
7	Peru	10	27	2
8	American Samoa	3	5	
9	French Guiana	4	3	
10	Runion	8	12	
11	Lithuania	5	5	

Data Output Messages

✔ Successfully run. Total query runtime: 126 msec. 107 rows aff