Data Definition Language

1. Consider the following table creation.

```sql
CREATE TABLE q1 (
  attr1 INT PRIMARY KEY,
  attr2 TEXT UNIQUE
);
```

Which of the following instance is a valid instance? In other words, all the rows satisfies the constraint specified.

A.

attr1	attr2
2	'a'
2	'b'

B.

attr1	attr2
2	'a'
1	'a'

C.

attr1	attr2
2	'a'
1	NULL

D.

attr1	attr2
2	'a'
NULL	'b'

Notes: C. `attr2` can be `null` and `attr1` has to be unique and not `null` .

2. Consider the following table creation.

```sql
CREATE TABLE q2 (
  attr1 INT CHECK (attr1 > 1),
  attr2 TEXT
);
```

Which of the following instance is a valid instance? In other words, all the rows satisfies the constraint specified.

A.

attr1	attr2
2	'a'

B.

attr1	attr2
1	'b'

C.

attr1	attr2
0	''

D.

attr1	attr2
-1	'c'

Notes: A. 2 is the only value strictly greater than 1.

3. Consider the following table creation.

```sql
CREATE TABLE q3 (
  attr1 INT UNIQUE,
  attr2 TEXT NOT NULL
);
```

Which of the following i `INSERT` statement will **NOT** cause an error. Treat each insertion separately starting from an empty table.

A. `INSERT INTO q3 VALUES (10, 'a'), (10, 'b');`

B. `INSERT INTO q3 VALUES (10), (11);`

C. `INSERT INTO q3 (attr2, attr1) VALUES (10, 'a'), (11, 'b');`

D. `INSERT INTO q3 (attr2) VALUES ('a');`

Notes: D. It is better to see other choices one by one.

A. `attr1` is not unique.

B. `attr2` is not given so it will be `NULL` but it cannot be `NULL`.

C. The second element is `attr1`. In this case, `'a'` and `'b'` are not `INT`.

D. We do not have `attr1` and it will be `NULL`, but it is allowed to be `NULL`.

4. Consider the following relation instance.

student	course	grade
3118	101	3.0
3118	112	4.0
5609	112	1.0
1423	311	4.5

The table was created with the following `CREATE TABLE` with the `PRIMARY KEY` missing.

```
CREATE TABLE q4 (
  student   INT,
  course    INT,
  grade     NUMERIC,
  -- PRIMARY KEY( ... )
);
```

Which of the following `PRIMARY KEY` constraint does not hold on the table above? Note, simply look at the table and do not worry about the meaning.

A. `PRIMARY KEY (student)`

B. `PRIMARY KEY (grade)`

C. `PRIMARY KEY (student, course)`

D. `PRIMARY KEY (student, grade)`

Select `X` if there is no answer.

Notes: A. The first two rows is a violation.

For the next 4 questions, we will use the following schema.

```
CREATE TABLE employee (
  eid   INT PRIMARY KEY,
  name  VARCHAR(256) NOT NULL
);

CREATE TABLE room (
  name  VARCHAR(32) PRIMARY KEY,
  num   INT NOT NULL,
  floor INT NOT NULL,
  UNIQUE (num, floor),
  CHECK (num > 0),
  CHECK (floor > 0)
);

CREATE TABLE booked_room (
  name    VARCHAR(32) REFERENCES room(name),
  date    DATE,
  purpose VARCHAR(256) NOT NULL,
  PRIMARY KEY (name, date)
);

CREATE TABLE booking (
  eid   INT NOT NULL REFERENCES employee(eid),
  name  VARCHAR(32),
  date  DATE,
  FOREIGN KEY (name, date) REFERENCES booked_room(name, date),
  PRIMARY KEY (name, date)
);
```

We further consider the following **preliminary data**.

employee Table

eid	name
101	'Amy'
102	'Dieter'
103	'Isidor'

booked_room Table

name	date	purpose
'Meeting Room 1'	2025-03-11	'Meeting
'Meeting Room 2'	2025-03-10	'Meeting
'Meeting Room 2'	2025-03-09	'Meeting
'Meeting Room 3'	2025-03-09	'Meeting

room Table

name	num	floor
'Meeting Room 1'	1	1
'Meeting Room 2'	1	2
'Meeting Room 3'	2	1
'Meeting Room 4'	2	3

booking Table

eid	name	date
101	'Meeting Room 2'	2025-03-10
103	'Meeting Room 1'	2025-03-11

5. Which of the following `INSERT` statement will cause an error. Treat each insertion separately starting from the given preliminary data.

    A. `INSERT INTO room VALUES ('Meeting Room 5A', 2, 2);`

    B. `INSERT INTO room VALUES ('Meeting Room 5B', 1, 3);`

    C. `INSERT INTO room VALUES ('Meeting Room 5C', 3, -1);`

    D. `INSERT INTO room VALUES ('Meeting Room 5D', 3, 1);`

Notes: C. `floor` must be greater than 0.

6. We focus on the table `room`. We want to reimplement the table `room` differently. Assume that by modifying `room`, we also modify the other tables referencing `room`.

    Which of the following alternative capture exactly the same constraint as the schema above?

A.
```
CREATE TABLE room (
  name  VARCHAR(32) UNIQUE NOT NULL,
  num   INT,
  floor INT,
  PRIMARY KEY (num, floor),
  CHECK (num > 0 AND floor > 0)
);
```

B.
```
CREATE TABLE room (
  name  VARCHAR(32) UNIQUE,
  num   INT,
  floor INT,
  PRIMARY KEY (num, floor),
  CHECK (num > 0),
  CHECK (floor > 0)
);
```

C.
```
CREATE TABLE room (
  name  VARCHAR(32) UNIQUE NOT NULL,
  num   INT,
  floor INT,
  PRIMARY KEY (num, floor),
  CHECK (num > 0 OR floor > 0)
);
```

D.
```
CREATE TABLE room (
  name  VARCHAR(32),
  num   INT,
  floor INT,
  PRIMARY KEY (name, num, floor),
  CHECK (num > 0),
  CHECK (floor > 0)
);
```

Notes: A. `UNIQUE` and `NOT NULL` is equivalent to `PRIMARY KEY`. Additionally, two different `CHECK` must both be satisfied. Hence, it is equivalent to `AND`.

7. Which of the following constraint is not enforced by the given schema?

    A. Rooms can be identified by the floor number (`floor`) and room number (`num`).

    B. Rooms can be identified by the room name (`name`).

    C. No two different employees can book the same room on the same day (`date`).

    D. No two different rooms can be booked by the same employee on the same day (`date`).

Notes: D. The rest are enforced.

We can enumerate the choices and see how they are enforced.

A. Enforced by having `NOT NULL` on `room.num` and `room.floor` as well as `UNIQUE (num, floor)`.

B. Enforced by having `PRIMARY KEY` on `room.name`.

C. Enforced by having `PRIMARY KEY` on the pair `(booking.name, booking.date)`. Consider a triple `(eid1, name, date)`. We cannot have another triple `(eid2, name, date)`.

D. Not enforced. To be enforced, we need to prevent the following pair of triple from coexisting: `(eid, name1, date)` and `(eid, name2, date)`.

E. *None of the above.*

8. Which of the following constraints are not enforced by the given schema?

A. There can be a booked room (*i.e.,* according to `booked_room`) but not by any employee.

B. It is possible that an employee does not book any room at all.

C. It is possible that a room is not booked at all.

D. *None of the above.* In other words, all are enforced.

Notes: D. All are enforced.

We can enumerate the choices and see how they are enforced.

A. Enforced by having `PRIMARY KEY` on `employee.eid`.

B. Enforced by allowing insert into `booked_room` but not inserting to `booking`.

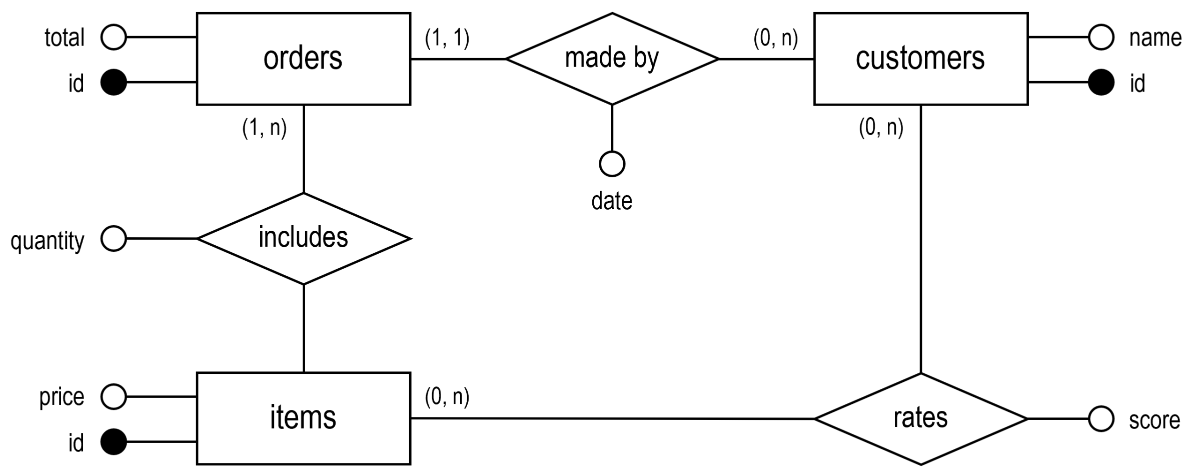
C. Enforced by allowing insert into `employee` but not inserting to `booking`.

D. Enforced by allowing insert into `room` but not inserting to `booking_room`.

E. *None of the above.*

Entity-Relationship Diagram

Consider the following entity-relationship diagram.



1. Consider the following constraints.

   (i) An order is made by exactly one customer.

   (ii) A customer can only rate the item they have bought at least once.

   (iii) An order must include at least one item.

   Which constraints above are enforced by the entity-relationship diagram? Select the most appropriate option.

   ✗ only (i)

   ✔ only (i) and (iii)

   ✗ only (i) and (ii)

   ✗ only (ii) and (iii)

   Notes: For (i), it is enforced by cardinality of (1,1) between `orders` and `made by`. For (iii), it is enforced by cardinality of (1,n) between `orders` and `includes`.

   For (ii), this is not enforced by the entity-relationship diagram as `rates` connect only `customers` and `items`. Hence, it has no information on `orders`.

For the next 3 question, consider a schema translation from entity-relationship diagram into `CREATE TABLE` statements. Constraints should only be enforced by the schema using only " `PRIMARY KEY` ", " `FOREIGN KEY` ", " `UNIQUE` ", " `NOT NULL` ", and " `CHECK` ".

(A) `customers(id, name)` with the primary key of `id`.

(B) `orders(total, id, cid, date)` with the primary key of `id`.

(C) `items(id, price)` with the primary key of `id`.

(D) `includes(iid, oid, quantity)` with the primary key of `(iid, oid)`.

(E) `rates(iid, cid, score)` with the primary key of `(iid, cid)`.

2. Consider the following constraints.

    (i) An order is made by exactly one customer.

    (ii) A customer can only rate the item they have bought at least once.

    (iii) An order must include at least one item.

    Which constraints above can be enforced by the schema? Select the most appropriate option.

    - ✔ only (i)
    - ✗ only (i) and (iii)
    - ✗ only (i) and (ii)
    - ✗ only (ii) and (iii)

    Notes: For (i), it is enforced because the primary key of `orders` is `id`, hence, it uniquely identifies `cid`.

    For (ii), we do not have information regarding the `orders.id`, so we cannot enforce this. For (iii), since the tables are different (i.e., `orders` and `items`), we cannot enforce that insertion to `orders` is followed by insertion to `items`.

3. Consider the following constraints.

    (i) The total price (i.e., `total`) of an order must be the sum of the prices of all included items.

    (ii) The score rating can only be an integer 1, 2, 3, 4, or 5.

    (iii) The quantity must be greater than 0.

    Which constraints above can be enforced by the schema? Select the most appropriate option.

    - ✗ only (i)
    - ✗ only (i) and (iii)
    - ✗ only (i) and (ii)
    - ✔ only (ii) and (iii)

    Notes: For (i), this cannot be enforced with a `CHECK` as we need to query multiple rows.

    For (ii), this is a simple `CHECK (score >= 1 AND score <= 5)`.

    For (iii), this is a simple `CHECK (quantity > 0)`.

4. We use the symbol `(t1.attr1, t1.attr2)` $\rightsquigarrow$ `(t2.attr3, t2.attr4)` to be equivalent to the following constraint.

    ```
    FOREIGN KEY (attr1, attr2) REFERENCES t2 (attr3, attr4)
    ```

    Which of the following foreign keys are required by the schema translation?

    - ✗ `(customers.id)` $\rightsquigarrow$ `(orders.cid)`
    - ✔ `(orders.cid)` $\rightsquigarrow$ `(customers.id)`

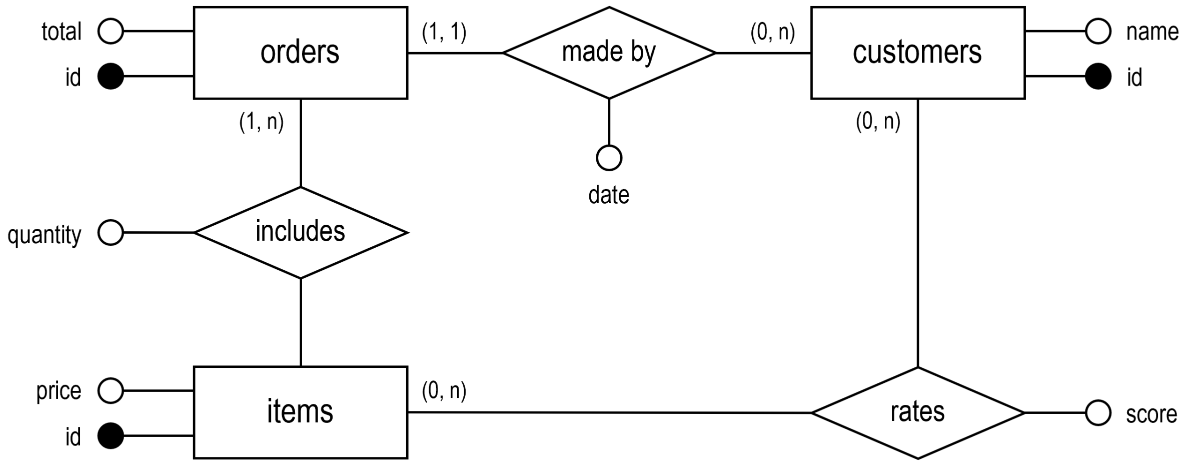
✗ `(rates.cid)` $\rightsquigarrow$ `(orders.cid)`

✗ *None of the above.*

Notes: Should a straightforward foreign key to the primary key.

SQL Query

Consider the following entity-relationship diagram.



Consider the following tables obtained from schema translation.

(A) `customers(id, name)` with the primary key of `id`.

(B) `orders(total, id, cid, date)` with the primary key of `id`.

(C) `items(id, price)` with the primary key of `id`.

(D) `includes(iid, oid, quantity)` with the primary key of `(iid, oid)`.

(E) `rates(iid, cid, score)` with the primary key of `(iid, cid)`.

Assume that the correct foreign keys have been added.

1. Consider the following problem.

   Find the name of the different customers that made at least one order with a total price over 1000.

   Which of the SQL queries below solve the problem above?

   ✗
   ```
   SELECT c.name
   FROM customers c INNER JOIN orders o
     ON o.cid = c.id AND o.total > 1000;
   ```

   ✗
   ```
   SELECT name
   FROM customers NATURAL JOIN orders
   WHERE total > 1000;
   ```

✗
```
SELECT c.name
FROM customers c LEFT JOIN orders o
  ON o.cid = c.id
WHERE o.id IS NOT NULL AND o.total > 1000;
```

✓ *None of the above.*

Notes: `INNER JOIN` and `LEFT JOIN` may contain different customer and not just different customer names. This is because the same customer can have multiple orders. Hence, neither A nor C can an answer.

`NATURAL JOIN` cannot work here because the attributes do not match. Hence, B cannot be an answer.

`EXCEPT` cannot work here as it may remove duplicates. But we want the names of the different customers. We may have two different customers (i.e., different `id`) but the same name. So, we cannot remove duplicates. Hence, D cannot be an answer.

2. Consider the following SQL query.

```
SELECT tmp.name
FROM (
  SELECT c1.id, c1.name
  FROM customers c1
  EXCEPT
  SELECT r2.cid, c2.name
  FROM customers c2, rates r2
  WHERE c2.id = r2.cid
) tmp;
```

Which of the following description is true about the query above?

- ✗ The query finds the **distinct** name of the customers that have not rated any items.
- ✔ The query may produce duplicate rows.
- ✗ The query will not produce duplicate rows.
- ✗ *None of the above.*

Notes: The inner query find the different customers as the `customers.id` is part of the columns. So, the `id` is guaranteed to be distinct. Hence, we find the **different** students and not the different names. The names can be the same as we will only remove the `id` column later on. Because the names can be the same, we may have duplicate rows.

3. **(Fill in the Blank)** Consider the following problem.

Find all the items with the highest average score for its rating (i.e., in the table `rates`). Exclude items that has never been rated before. Output the item id and the average score.

You came up with a solution but accidentally spilled coffee all over your query. After erasing the stain, you need to fill in the blank again. Here is what you can recover.

```
SELECT r.iid, _____1_____(r.score) AS average
FROM rates r
GROUP BY _____2_____
HAVING _____3_____ >= ALL (
  SELECT _____4_____
  FROM rates r
  GROUP BY _____5_____
)
```

Your task is to fill in the blanks from _____1_____ to _____5_____ so that the query solves the problem above. Write your answer on the answer sheet.

Notes:

```
SELECT r.iid, AVG(r.score) AS average
FROM rates r
GROUP BY r.iid
HAVING average >= ALL (    -- or HAVING AVG(r.score)
  SELECT AVG(r.score)
  FROM rates r
  GROUP BY r.iid
)
```

For the following SQL queries, you are **NOT** allowed to use **Common Table Expressions** (i.e., CTE).

4. **(SQL)** Write an SQL query to solve the following problem.

   Find the different items that are the most expensive. Output only the item id and the price. Sort your result in descending order of price followed by ascending order of item id.

   You are **NOT** allowed to use **aggregate queries**.

   Notes: As noted by one of the student, we do not need to sort in descending order of `price` as the most expensive item can only be a single price!

```
SELECT i1.id, i1.price
FROM items i1
WHERE i1.price >= ALL (
  SELECT i2.price
  FROM items i2
)
ORDER BY i1.price DESC, i1.id ASC;
```

5. (6 points) **(SQL)** Write an SQL query to solve the following problem.

   For each order, find the total quantity of all items in the order. Output only the order id and the total quantity. Sort your result in descending order of quantity followed by ascending order of order id.

   Notes:

```
SELECT o.id, SUM(i.quantity) AS total
FROM orders o, includes i
WHERE o.id = i.oid
GROUP BY o.id
ORDER BY total DESC, o.id ASC;
```

For the next ?? questions, consider the following schema.

```
CREATE TABLE employee (
  eid   INT PRIMARY KEY,
  name  VARCHAR(256) NOT NULL
);

CREATE TABLE room (
  name  VARCHAR(32) PRIMARY KEY,
  num   INT NOT NULL,
  floor INT NOT NULL,
  UNIQUE (num, floor),
  CHECK (num > 0),
  CHECK (floor > 0)
);

CREATE TABLE booked_room (
  name    VARCHAR(32) REFERENCES room(name),
  date    DATE,
  purpose VARCHAR(256) NOT NULL,
  PRIMARY KEY (name, date)
);

CREATE TABLE booking (
  eid   INT NOT NULL REFERENCES employee(eid),
  name  VARCHAR(32),
  date  DATE,
  FOREIGN KEY (name, date) REFERENCES booked_room(name, date),
  PRIMARY KEY (name, date)
);
```

We further consider the following **preliminary data**.

`employee` Table

eid	name
101	`'Amy'`
102	`'Dieter'`
103	`'Isidor'`

`booked_room` Table

name	date	purpose
`'Meeting Room 1'`	2025-03-11	`'Meeting`
`'Meeting Room 2'`	2025-03-10	`'Meeting`
`'Meeting Room 2'`	2025-03-09	`'Meeting`
`'Meeting Room 3'`	2025-03-09	`'Meeting`

`room` Table

name	num	floor
`'Meeting Room 1'`	1	1
`'Meeting Room 2'`	1	2
`'Meeting Room 3'`	2	1
`'Meeting Room 4'`	2	3

`booking` Table

eid	name	date
101	`'Meeting Room 2'`	2025-03-10
103	`'Meeting Room 1'`	2025-03-11

6. Which of the following query **cannot be answered** given the schema above? Note that there are no other constraints except those specified by the schema.

   A. Find the different employees (`employee.eid`) that have never booked any room (`room.name`).

   B. Find the different floors (`room.floor`) that have no room (`room.name`).

C. Find the different room (`room.name`) that has never become a booked room (*i.e.*, according to `booked_room`).

D. Find the different room (`room.name`) that has never been booked (*i.e.*, according to `booking`).

Notes: B. We do not know what other floors are there. Consider the data above, we do not know if there are floor 4 or not.

7. Which of the following query is guaranteed to have no duplicate? Note that you should consider any possible valid instance satisfying the schema and not just the current data.

(i)
```
SELECT b.date, r.num, r.floor
FROM room r, booked_room b
WHERE b.name = r.name;
```
(ii)
```
SELECT b.date, r.name
FROM room r, booked_room b
WHERE b.name = r.name;
```
(iii)
```
SELECT e.eid, b.name, b.date
FROM employee e, booking b
WHERE b.eid = e.eid;
```

A. (i) only
B. (i) and (ii) only
C. (ii) and (iii) only
D. (i), (ii), and (iii)

Notes: D.

For (i) and (ii), note that `b.name = r.name` , so the primary key of both `room` an d `booked_room` are equal. If the primary key or candidate key appears in the `SELECT` clause, then the query has no duplicate values.

For (iii), we only have `b.eid = e.eid` . So the query has no duplicate only if the primary key or candidate key of BOTH `employee` and `booking` are in the `SELECT` clause.

8. **(Fill in the Blank)** What is the result of the following query?

```
SELECT br.name, br.date, e.name AS employee
FROM booked_room br, booking b, employee e
WHERE br.name = b.name AND b.eid = e.eid
  AND br.name IN (
        SELECT r.name
        FROM room r
        WHERE r.floor = 1
);
```

You do not have to use all rows. The order of the rows does not matter.

name	date	employee

Notes: Here, the inner subquery is *uncorrelated*. So, we can evaluate it on its own separately from the outer query. The inner subquery finds all the room name located on floor number 1. There are 2 of them: (i) `'Meeting Room 1'` and (ii) `'Meeting Room 3'` .

We then look at `booked_room` table to find the rows corresponding to this. However, when looking at the outer query, we will only consider the rows for which the room has been booked by an employee. Hence, we ignore `'Meeting Room 3'` because it is a booked room but no employee is booking it.

Finally, we obtain the name of the meeting room, the date the room become a booked room, and the name of the employee booking the room. This gives us the following table.

name	date	employee
'Meeting Room 1'	2025-03-11	'Isidor'

9. **(SQL)** Find the different room that is not booked (*i.e.*, according to `booked_room`) on 2025-03-09. Order the result in **descending order** of room name. Recap that you can write a date by enclosing it in single quote like a string (*e.g.*, `'2025-03-09'`).

Using the data above, we will have the following result.

name	num	floor
'Meeting Room 1'	1	1
'Meeting Room 4'	2	3

(a) (6 points) Use algebraic query in your solution.

Notes:

```
-- Using FULL JOIN (here, left)
SELECT r.name, r.num, r.floor
FROM room r FULL JOIN booked_room b
  ON r.name = b.name AND b.date = '2025-03-09'
WHERE b.name ISNULL
ORDER BY r.name DESC;

-- Using EXCEPT
SELECT * FROM room r
EXCEPT
SELECT r1.name, r1.num, r1.floor
FROM room r1, booked_room br
WHERE r1.name = br.name AND br.date = '2025-03-09'
ORDER BY r.name DESC;
```

(b) (6 points) Use nested query in your solution. Note that you cannot use algebraic query for this part.

Notes:

```
-- Using NOT IN
SELECT * FROM room r
WHERE r.name NOT IN (
  SELECT b.name FROM booked_room b
  WHERE b.date = '2025-03-09'
)
ORDER BY r.name DESC;
```

```
-- Using NOT EXISTS
SELECT * FROM room r
WHERE NOT EXISTS (
  SELECT b.name FROM booked_room b
  WHERE b.date = '2025-03-09' AND b.name = r.name  -- correlation
)
ORDER BY r.name DESC;
```

For the next 5 question, we will use the following table `student` , `course` , and `enrolment` .

student

student	name
3118	'Alice'
5609	'Bob'
1423	'Charlie'

course

course	name
101	'Intro'
112	'Algo'
311	'Advanced'

enrolment

student	course	grade
3118	101	3.0
3118	112	4.0
5609	112	1.0
1423	311	4.5
1423	112	4.0

Notes: If you wish to try it out on your own, you can use the following simplified schema.

```
CREATE TABLE student (
  student INT,
  name TEXT
);

CREATE TABLE course (
  course INT,
  name TEXT
);

 CREATE TABLE enrolment (
   student INT,
   course INT,
   grade NUMERIC
 );

 INSERT INTO student VALUES (3118, 'Alice');
 INSERT INTO student VALUES (5609, 'Bob');
 INSERT INTO student VALUES (1423, 'Charlie');

 INSERT INTO course VALUES (101, 'Intro');
 INSERT INTO course VALUES (112, 'Algo');
 INSERT INTO course VALUES (311, 'Advanced');

 INSERT INTO enrolment VALUES (3118, 101, 3.0);
 INSERT INTO enrolment VALUES (3118, 112, 4.0);
 INSERT INTO enrolment VALUES (5609, 112, 1.0);
 INSERT INTO enrolment VALUES (1423, 311, 4.5);
 INSERT INTO enrolment VALUES (1423, 112, 4.0);
```

10. What is the result of the following query?

```
SELECT DISTINCT e.course, e.grade
FROM enrolment e
WHERE e.grade >= 2.0;
```

You do not have to use all rows. The order of the rows does not matter.

course	grade

Notes: The keyword `DISTINCT` is to be applied to the entire (course, grade) pair. We have a duplicate (112, 4.0). Only one will be printed. Note that you may put 3 or 3.0 for the grade.

course	grade
101	3.0
112	4.0
311	4.5

11. What is the result of the following query?

```
SELECT s.name AS student, c.name AS course, e.grade
FROM enrolment e, student s, course c
WHERE s.student = e.student
  AND c.course = e.course
  AND s.name = 'Alice';
```

You do not have to use all rows. The order of the rows does not matter.

student	course	grade

Notes:

student	course	grade
'Alice'	'Intro'	3.0
'Alice'	'Algo'	4.0

12. What is the result of the following query?

```
SELECT c.name, MAX(e.grade) AS max
FROM enrolment e, course c
WHERE grade < 4.5
  AND e.course = c.course
GROUP BY e.course, c.name;
```

You do not have to use all rows. The order of the rows does not matter.

name	max

Notes:

name	max
'Intro'	3.0
'Algo'	4.0

13. Find the different student who has taken the `'Algo'` course with a grade higher than or equal to `2.0`. Return the student name and the grade. Order the output in **ascending** order of grade followed by **descending** order of name.

    You must use only **simple query**. Using the data above, we will have the following result.

name	grade
'Charlie'	4.0
'Alice'	4.0

Notes:

```sql
SELECT s.name, e.grade
FROM student s, enrolment e, course c
WHERE e.grade >= 2.0
  AND c.name = 'Algo'
  AND s.student = e.student
  AND c.course = e.course
ORDER BY e.grade ASC, s.name DESC;
```

14. For each student, find their average grade. Return the student name and the average grade. Order the output in **descending** order of average grade followed by **descending** order of name.

    Do not forget to rename the attributes but you do not have to truncate/round the average. Using the data above, we will have the following result.

name	average
'Charlie'	4.25
'Alice'	3.50
'Bob'	1.00

Notes:

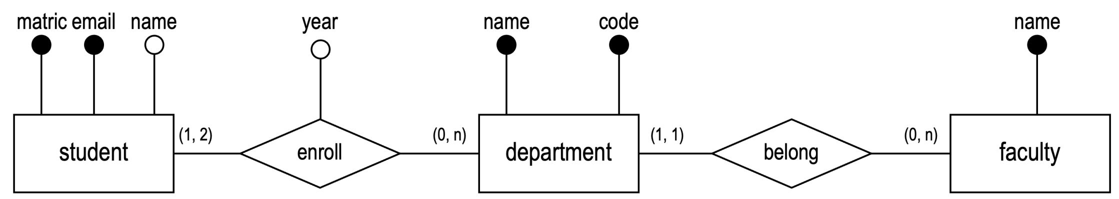
```sql
SELECT s.name, AVG(e.grade)
FROM student s, enrolment e
WHERE s.student = e.student
GROUP BY s.student, s.name
ORDER BY AVG(e.grade) DESC, s.name DESC;
```

Entity-Relationship Diagram

1. **(ERD)** NUS IT is hiring! In your first interview, you are asked to design the student database for the university. Your design should be drawn as entity-relationship diagram following the convention used in this semester. Additionally, it must satisfies the following constraints.

    1. A student is uniquely identified by their matric number.
    2. A student is uniquely identified by their email address.
    3. A student name must be recorded.
    4. A student must be enrolled to at least 1 department.
    5. A student may be enrolled to at most 2 department (e.g., double major program).
    6. A student year of enrolment to a department must be recorded. Note that the same student may be enrolled to two different departments at different year.
    7. A department is uniquely identified by the department name.
    8. A department is uniquely identified by the department code.
    9. A department belong to exactly one faculty.
    10. A department may have any number of students (including 0).
    11. A faculty only has a name.
    12. A faculty may have any number of department (including 0).

Notes:



Consider the following schema.

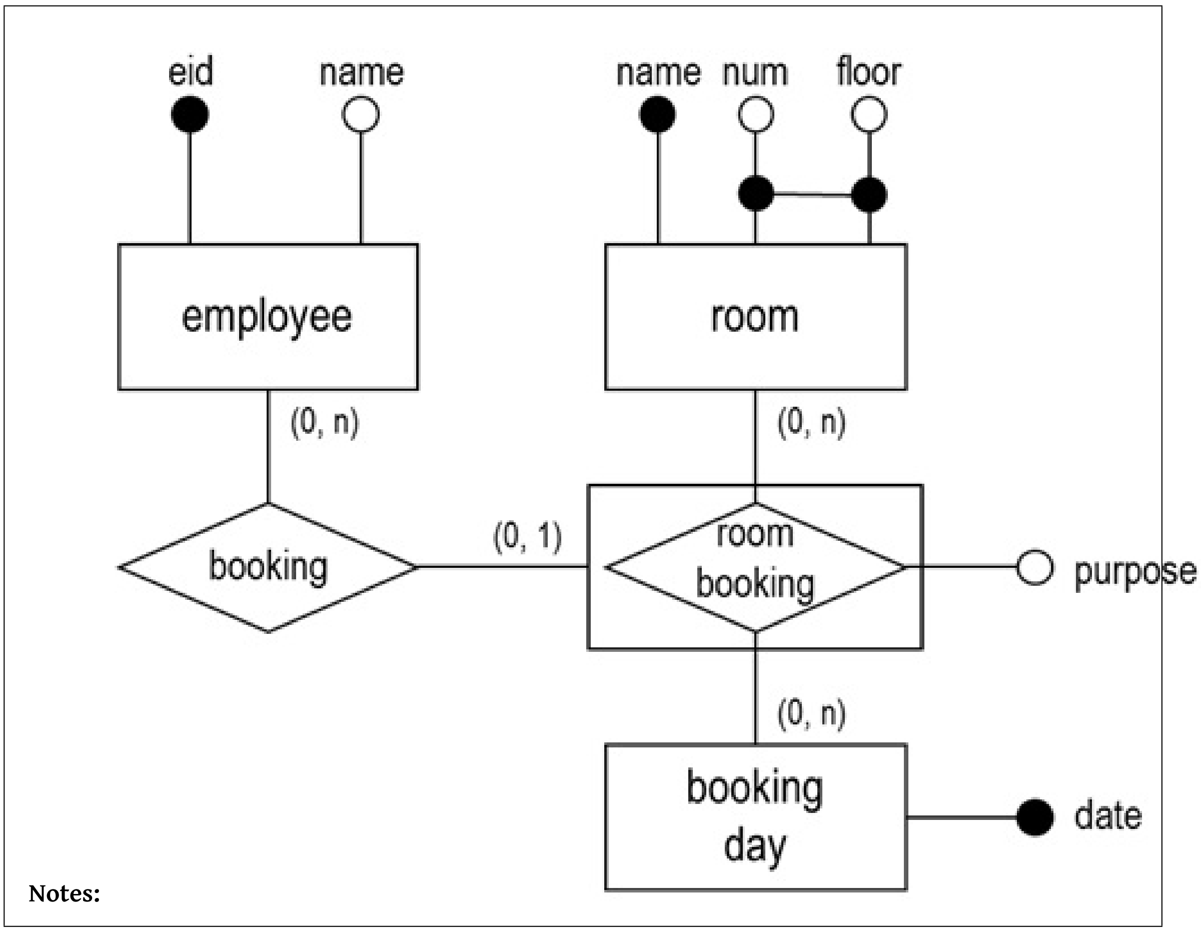
```
CREATE TABLE employee (
  eid   INT PRIMARY KEY,
  name  VARCHAR(256) NOT NULL
);

CREATE TABLE room (
  name  VARCHAR(32) PRIMARY KEY,
  num   INT NOT NULL,
  floor INT NOT NULL,
  UNIQUE (num, floor),
  CHECK (num > 0),
  CHECK (floor > 0)
);

CREATE TABLE booked_room (
  name    VARCHAR(32) REFERENCES room(name),
  date    DATE,
  purpose VARCHAR(256) NOT NULL,
  PRIMARY KEY (name, date)
);

CREATE TABLE booking (
  eid   INT NOT NULL REFERENCES employee(eid),
  name  VARCHAR(32),
  date  DATE,
  FOREIGN KEY (name, date) REFERENCES booked_room(name, date),
  PRIMARY KEY (name, date)
);
```

2. **(ERD)** Given the schema, draw one possible entity-relationship diagram such that by using schema translation rule in the lecture, we will get the schema.

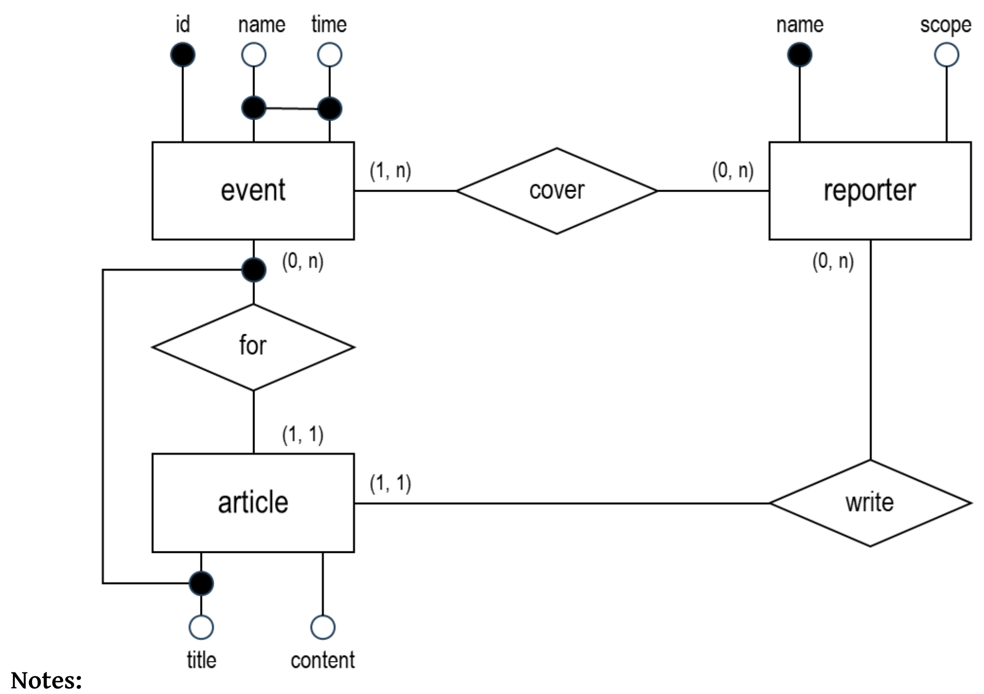


Notes:

3. You are starting your work at a newspaper helping to build their database system. In this newspaper, reporters write articles about events. Events are covered by reporters. You are given the following constraints.

    1. Each event can be uniquely identified by either the event id or the combination of event name and event time.
    2. Each reporter can be uniquely identified by their name. The job scope of the reporter is also recorded.
    3. The title of an article can uniquely identify the article only among the article written for a given event. The content of the article is also recorded.
    4. A reporter may write 0 or more articles.
    5. An article must be written by exactly one reporter.
    6. An event may be covered by different reporters but it must be covered by at least one reporter.
    7. A reporter may cover 0 or more event.
    8. An event may be written in 0 or more articles.
    9. An article must be written for exactly one event.
    10. An event may have 0 or more article written about them.

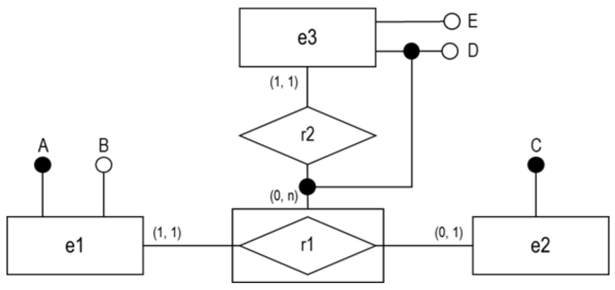
    Draw the ER diagram in **our convention** that captures as much constraints as possible.


id
name
time
name
scope
event
(1, n)
cover
(0, n)
reporter
(0, n)
(0, n)
for
(1, 1)
article
(1, 1)
write
title
content

Notes:

Functional Dependencies

Consider the following entity-relationship diagram.



1. Which of the following functional dependency does not hold on the entity-relationship diagram?
   - [✓ $\{C\} \rightarrow \{B\}$]
   - B. $\{E\} \rightarrow \{A\}$
   - C. $\{E\} \rightarrow \{B\}$
   - D. $\{B\} \rightarrow \{A\}$
2. Consider $R(A, B, C, D, E)$ with S being the set of functional dependencies that holds on the entity-relationship above. Which of the following is the candidate key (*i.e., keys*) of R?
   - A. $\{E\}$
   - [✓ $\{A, D\}$]
   - C. $\{A, E\}$
   - D. $\{E, D\}$

The next **three (3)** questions uses the same relation R and the same set of functional dependencies S.

- $R(A, B, C, D, E, F, G)$
- $S = \{\ \{A, B\} \rightarrow \{C\},\ \{C, D\} \rightarrow \{E\},\ \{F, G\} \rightarrow \{A, B, C\},\ \{B\} \rightarrow \{A, F\},\ \{A\} \rightarrow \{B, E\}\ \}$

3. Find all the keys of R with S.

Notes: $\{A, D, G\}, \{B, D, G\}, \{D, F, G\}$

4. Derive a lossless-join decomposition of R with S in BCNF using the BCNF decomposition algorithm introduced in the lecture.

Notes: One possible decomposition is the following.
$R1 = \{A, B, C, E, F\}, R2 = \{B, D, G\}$

5. Does your BCNF decomposition from the previous question preserve all functional dependencies in S?

Notes: The decomposition above does not preserve the following.
$\{C, D\} \rightarrow \{E\}$ and $\{F, G\} \rightarrow \{A, B, C\}$.

The next **three (3)** questions uses the same relation R and the same set of functional dependencies S.

- $R(A, B, C, D, E, F)$
- $S = \{\ \{D\} \rightarrow \{B\},\ \{A, B\} \rightarrow \{C\},\ \{B, C\} \rightarrow \{A\},\ \{C, D\} \rightarrow \{E\},\ \{B, D\} \rightarrow \{E\},\ \{A, D, E\} \rightarrow \{C\}\ \}$

6. Find all the keys of R with S.

Notes: $\{A, D, F\}, \{C, D, F\}$

7. Derive one possible minimal cover of S.

Notes: One possible minimal cover is the following.
$\{\{A, B\} \rightarrow \{C\}, \{B, C\} \rightarrow \{A\}, \{D\} \rightarrow \{B\}, \{D\} \rightarrow \{E\}\}$

8. Derive a 3NF decomposition of R with S using the 3NF decomposition algorithm introduced in the lecture.

Notes: One possible decomposition is the following.
$R1 = \{A, D, F\}, R2 = \{A, B, C\}, R3 = \{B, D, E\}$.

For the next 7 questions, consider the following relation and set of functional dependencies.

$R = \{A, B, C, D, E\}$
$\Sigma = \{\ \{B, C, D\} \rightarrow \{E\}, \{B, E\} \rightarrow \{A, B\}, \{B, D, E\} \rightarrow \{B, E\}, \{B, D\} \rightarrow \{B, C\}, \{E\} \rightarrow \{D\}\ \}$

Notes: We start with the following computation:

- **Keys:** $\{\ \{B, D\},\ \{B, E\}\ \}$
- **Minimal Cover:** $\{\ \{B, D\} \rightarrow \{C\},\ \{B, D\} \rightarrow \{E\},\ \{B, E\} \rightarrow \{A\},\ \{E\} \rightarrow \{D\}\ \}$

9. Select the *trivial* functional dependencies in Σ.

A. $\{B, C, D\} \rightarrow \{E\}$
B. $\{B, E\} \rightarrow \{A, B\}$
[✓ C. $\{B, D, E\} \rightarrow \{B, E\}$]
D. $\{B, D\} \rightarrow \{B, C\}$

Notes: $\{B, E\} \subseteq \{B, D, E\}$

10. Select the *non-trivial* functional dependencies in Σ that are not *completely non-trivial.*
    - A. $\{B, C, D\} \rightarrow \{E\}$
    - ✔ $\{B, E\} \rightarrow \{A, B\}$
    - C. $\{B, D, E\} \rightarrow \{B, E\}$
    - D. $\{E\} \rightarrow \{D\}$

Notes: **B:** $\{A, B\} \not\subseteq \{B, E\} \land \{A, B\} \cap \{B, E\} \neq \emptyset$

11. Select the *completely non-trivial* functional dependencies in Σ.
    - ✔ $\{B, C, D\} \rightarrow \{E\}$
    - B. $\{B, E\} \rightarrow \{A, B\}$
    - C. $\{B, D, E\} \rightarrow \{B, E\}$
    - D. $\{B, D\} \rightarrow \{B, C\}$

Notes: **A:** $\{E\} \cap \{B, C, D\} = \emptyset$; **E:** $\{E\} \cap \{D\} = \emptyset$

12. Select the *superkeys* of R with Σ from the choice below.
    - A. $\{A, B, C\}$
    - ✔ $\{B, C, D\}$
    - C. $\{C, D, E\}$
    - D. $\{A, D, E\}$

13. Select the *candidate keys* (i.e., keys) of R with Σ from the choice below.
    - A. $\{A, B\}$
    - B. $\{D, E\}$
    - C. $\{C, D\}$
    - ✔ $\{B, D\}$
    - E. $\{A, E\}$

Notes: There are two keys: $\{B, D\}$ and $\{B, E\}$.

14. Select the functional dependencies below that violates the BCNF property of R with Σ.
    - A. $\{B, E\} \rightarrow \{A\}$
    - B. $\{B, D, E\} \rightarrow \{B\}$
    - ✔ $\{E\} \rightarrow \{D\}$
    - D. None of the above.

Notes: $\{C\} \not\subseteq \{E\}$ and $\{E\}$ is not a candidate key.

15. Select the functional dependencies below that violates the 3NF property of R with Σ.

    A. $\{B, C, D\} \rightarrow \{E\}$

    B. $\{B, E\} \rightarrow \{A\}$

    C. $\{B, D, E\} \rightarrow \{B\}$

    D. $\{B, D\} \rightarrow \{C\}$

    E. $\{E\} \rightarrow \{D\}$

Notes: Similar to above, but now we cannot use $\{E\} \rightarrow \{D\}$ because D is a prime attribute.

For the next **ten (10)** questions, we will be using the following relation and set of functional dependencies.

$$R = \{A, B, C, D, E\}$$
$$\Sigma = \{\, \{C, E\} \rightarrow \{A, B\}, \{A, B, C\} \rightarrow \{A, D\}, \{D, E\} \rightarrow \{A, B, E\}, \{D\} \rightarrow \{A, C\},$$
$$\{A, C\} \rightarrow \{A, B\}, \{A, D\} \rightarrow \{B\}, \{C, D\} \rightarrow \{A, B\}\}$$

Notes: We start with the following computation:

- **Keys:** $\{\{C, E\}, \{D, E\}\}$
- **Minimal Cover:** $\{\{A, C\} \rightarrow \{B\}, \{A, C\} \rightarrow \{D\}, \{C, E\} \rightarrow \{A\}, \{D\} \rightarrow \{A\}, \{D\} \rightarrow \{C\}\}$

16. Select the attributes not in the *closure* of $\{A, C\}$ (i.e., $\{A, C\}^+$).

    A. B

    B. D

    [✓ E]

    D. None of the above

Notes: $\{A, C\}^+ = \{A, B, C, D\}$

17. Select the functional dependency that is *logically entailed* by Σ.

    A. $\{A, B\} \rightarrow \{C\}$

    B. $\{B, C\} \rightarrow \{D\}$

    C. $\{C, D\} \rightarrow \{E\}$

    [✓ $\{D, E\} \rightarrow \{A\}$]

Notes: D

18. Select the *superkeys* of R with Σ from the choice below.
    - A. $\{A, B, C\}$
    - ✔ $\{C, D, E\}$
    - C. $\{A, B, E\}$
    - D. None of the above.

Notes: $\{C, D, E\}$ is the superset of keys (see below).

19. Select the *candidate keys* of R with Σ from the choice below.
    - A. $\{A, B\}$
    - B. $\{A, E\}$
    - C. $\{B, E\}$
    - ✔ $\{C, E\}$

Notes: D: $\{C, E\}$.

20. Select the functional dependency below that does not violates the BCNF property of R with Σ.
    - A. $\{A, B, C\} \rightarrow \{D\}$
    - ✔ $\{D, E\} \rightarrow \{B\}$
    - C. $\{D\} \rightarrow \{C\}$
    - D. $\{C, D\} \rightarrow \{B\}$

Notes: ACD violates as the LHS are not superset of keys (see above) **AND non-trivial**. Note that we should also check if the functional dependencies are logically entailed by Σ. But a quick look shows that they are. We simply make the RHS singular from some of the FD in Σ.

21. Select the functional dependency below that violates the 3NF property of R with Σ.
    - A. $\{A, B, C\} \rightarrow \{D\}$
    - B. $\{D, E\} \rightarrow \{B\}$
    - C. $\{D\} \rightarrow \{C\}$
    - D. $\{C, D\} \rightarrow \{B\}$

Notes: D: Similar as above but we further exclude cases where the RHS is a prime attributes (i.e., not C, D, or E).