Name: _____

Please answer the following questions within the space provided on the following pages. Should you need more space, you can use scratch paper, but clearly label on the scratch paper what problem it corresponds to. While you are not required to explain your queries, comments may help me to understand what you were trying to do and thus increase the likelihood of partial credit should something go wrong. If you get entirely stuck somewhere, explain in words as much as possible what you would try.

This is a pen and paper exam, and thus computers and internet capable devices are prohibited. If you have any confusion about question intention or wording, please do not hesitate to ask!

Your work must be your own on this exam, and under no conditions should you discuss the exam or ask questions to anyone but myself. Failure to abide by these rules will be considered a breach of Willamette's Honor Code and will result in penalties as set forth by Willamette's academic honesty policy.

Please sign and date the below lines to indicate that you have read and understand these instructions and agree to abide by them. *Failure to abide by the rules will result in a 0 on the test.* Good luck!!

_____ _____

Signature Date

Question:	1	2	3	4	Total
Points:	4	9	15	12	40
Score:					

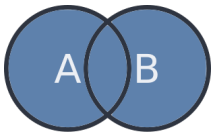
(4) 1. Match each of the below queries with the Venn diagram that best depicts the results of the query. There are more Venn diagrams that queries, but each diagram with a match corresponds to only a single query.

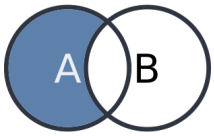
```sql
SELECT *
FROM A
JOIN B
  ON A.key = B.key;
```

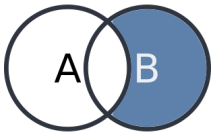
```sql
SELECT *
FROM A
RIGHT JOIN B
  ON A.key = B.key;
```

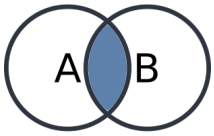
```sql
SELECT *
FROM A
LEFT JOIN B
  ON A.key = B.key
WHERE B.key IS NULL
```

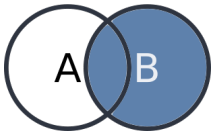
```sql
SELECT *
FROM A
FULL OUTER JOIN B
  ON A.key = B.key
```











(9) 2. Suppose you have the three tables `tab1`, `tab2`, and `tab3` defined and populated as seen below.

```
CREATE TABLE tab1 (
  A text PRIMARY KEY,
  B int,
  C int,
  D int,
  CHECK (B + C < D)
);

CREATE TABLE tab2 (
  E text REFERENCES tab1,
  F text,
  G text,
  PRIMARY KEY (E,F,G),
  FOREIGN KEY (F,G) REFERENCES tab3
);

CREATE TABLE tab3 (
  H date UNIQUE,
  I int NOT NULL,
  J text,
  K text,
  PRIMARY KEY (J,K)
);
```

tab1

A	B	C	D
Bob	3	1	6
Jill	5	0	8
Jane	8	1	10
Greg	3	-2	4
Lilly	4	5	10

tab2

E	F	G
Jill	high	blue
Greg	low	black
Lilly	high	green

tab3

H	I	J	K
2022-10-01	1	low	red
2022-01-30	5	high	blue
2022-12-13	-2	low	blue
2022-01-29	0	low	black
2022-02-14	0	high	green

For each of the below statements, determine whether or not the statement would result in an error when run. If it would not result in an error, state as much, but if it would result in an error, state specifically what constraint is causing the error. All potential errors are related only to constraints, not to syntax. You can assume each of the below commands is run independently on the above tables (they can not affect one another).

(a)
```
INSERT INTO tab2 VALUES
('Greg', 'low', 'blue');
```

Solution: This should run fine

(b)
```
UPDATE tab3
SET H = H - '1 day'::interval
WHERE K = 'blue';
```

Solution: This will error, since subtracting one from the 2022-01-30 date will give 2022-01-29, which is already in the column and thus the column will no longer have unique values (which it is constrained to have).

(c)
```
DELETE FROM tab2
WHERE E = 'Lilly';
```

Solution: This should be fine. `tab2` references other tables, but nothing references it, and nothing references this particular row.

(d)
```
UPDATE tab1
SET B = B + 1
WHERE A ILIKE 'J%';
```

Solution: This will fail, as it will update the valu of B in the Jane row to be 9, which then means that $9 + 1 \not< 10$, which was the constraint placed on `tab1`.

(e)
```
ALTER TABLE tab1 ADD PRIMARY KEY (B);
```

Solution: This will fail, as `tab1` already has a primary key and there can be only one. This does *not* make a compound primary key.

(f)
```
ALTER TABLE tab3 ADD FOREIGN KEY (I) REFERENCES tab1(C);
```

Solution: This will fail, as the contents of C are not unique, so it is impossible to know which row would be referenced when I is 1 (for instance).

3. In the spirit of St. Patrick's Day, you and some friends are looking to plan a fun evening out. You have gathered all the information you have about your friend's houses, local pubs, and bus stops and routes into the database shown in the below ERD. In your town, each bus stop happens to belong to only a single route and bus routes are named by colors.

Given this database, write out queries that could answer each of the following questions. All can be done in a single query, but you are free to use multiple queries if it helps.

(5) (a) Which route (color) has the most pubs along it?

Solution: This would allow us to read off the pubs with the most. I could limit it to 1 but then if there is a tie I might not know. If I really wanted to get all possible ties I could use another query to grab only those with values corresponding to the max.

```
SELECT s.route_color, COUNT(*)
FROM stops as s
JOIN pubs as p
  ON p.bus_stop = s.bus_stop
GROUP BY s.route_color
ORDER BY COUNT(*) DESC
;
```

(5) (b) How many friends live on the most expensive (per stop) bus route?

Solution: Same thing here, I could limit to one but if there was a tie then I wouldn't see it. So this would let me read off the counted number of friends on that route.

```
SELECT r.route_color, r.charge_per_stop, COUNT(*)
FROM routes as r
JOIN stops as s
  ON r.route_color = s.route_color
JOIN friends as f
  ON f.bus_stop = s.bus_stop
GROUP BY r.route_color, r.charge_per_stop
ORDER BY r.charge_per_stop DESC
;
```

(5) (c) If you were to ride the entirety of the "blue" route, hitting every stop and buying one beer at any pub that might be at that stop, what would the total price of your evening out be (including bus fare)?

Solution: This is just doing a bit of joining, filtering, and then aggregating over all the stops and prices. Importantly, there may not be a pub at every stop, so I use left join to ensure that stops without pubs are still in the table (since we'd have to pay for them) but would have `NULL` values for the pub tables values, and thus would not add to the sum of beer prices.

```
SELECT SUM(r.charge_per_stop) + SUM(p.price_of_beer)
FROM routes as r
JOIN stops as s
  ON r.route_color = s.route_color
LEFT JOIN pubs as p
  ON p.bus_stop = s.bus_stop
WHERE r.route_color = 'blue'
;
```

4. Suppose you have the single table below (named `leprechauns`) containing highly classified information about a variety of leprechauns.

name *text*	**beard_color** *text*	**coat_color** *text*	**lucky_num** *int*
Lucky	red	green	4
Connor	blond	red	2
Sean	red	red	8
Aoife	white	green	4
Finley	brown	green	6
Teagan	red	red	10

From this information, determine the output of each of the following SQL queries. *Work must be shown along the way for any chance at partial credit.*

(4) (a)
```
SELECT coat_color, MIN(beard_color)
FROM leprechauns
GROUP BY coat_color
HAVING SUM(lucky_num) > 15;
```

Solution:

coat_color	min
red	blond

(4) (b)

```
SELECT
  lep1.name,
  lep2.name,
  lep1.lucky_num + lep2.lucky_num as total_luck
FROM leprechauns as lep1
JOIN leprechauns as lep2
  ON lep1.beard_color = lep2.coat_color AND
     lep1.name != lep2.name;
```

Solution: The query doesn't have anything about ordering, so the below could be in any order.

name	name	total_luck
Lucky	Connon	6
Lucky	Sean	12
Lucky	Teagan	14
Sean	Connor	10
Sean	Teagan	18
Teagan	Connor	12
Teagan	Sean	18

(4) (c)

```
DELETE FROM leprechauns
WHERE beard_color = coat_color AND
      name NOT ILIKE '%g%';

ALTER TABLE leprechauns ADD COLUMN age text;
UPDATE leprechauns
SET age = (lucky_num + 24)::text;

DELETE FROM leprechauns
WHERE age ILIKE '2_';

ALTER TABLE leprechauns DROP COLUMN coat_color;

SELECT * FROM leprechauns;
```

Solution:

name	beard_color	lucky_num	age
Finley	brown	6	30
Teagan	red	10	34