Name: Jed's Copy

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. You are free to use a prepared 3x5 inch index card with handwritten notes on a single side should you desire. If you have any confusion about question intention or wording, please do not hesitate to ask!

Please restrict yourself on this exam to only SQL keywords that we have discussed in class and which have shown up on homework!

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	5	6	7	8	9	10	Total
Points:	5	8	12	9	3	2	2	12	12	0	65
Score:											

(5) 1. A mysterious table (named `mysterious_table`) has the following query run on it:

```
SELECT
  min(red::INT - cyan) AS new_a,
  percentile_disc(0.5) WITHIN GROUP (ORDER BY red) AS new_b,
  max(2 * red + green) AS new_c
FROM mysterious_table
WHERE blue ILIKE '%odd'
  AND orange BETWEEN '1:00' AND '13:00';
```

and returns a table with the following form:

Column Name	Data Type
new_a	INTEGER
new_b	DOUBLE PRECISION
new_c	NUMERIC

Determine as much information as you can about the columns comprising `mysterious_table`, and explain how you arrived at your conclusions.

Min → same data type as inside, so red::INT − cyan = INT ⇒ cyan must be an integer

Percentile_disc also → same data type as inside, so ⇒ red must be double precision

max → same data type as inside, so 2 * red + green = NUMERIC

INT DOUBLE

DOUBLE + x = NUMERIC ⇒ impossible (s!*t)

blue is being pattern matched, so must be some type of string: TEXT, CHAR, VARCHAR

orange seems to be compared to times, so most likely a TIME type. Could also be a string but a bit weird. Can not be a TIMESTAMP bc no date.

2. You have the below CSV file of names and corresponding addresses.

```
Natalia Church,7789 Ryan Dr.,Englewood,NJ,07631
William Ellison,57 Elizabeth Dr.,Merrillville,IN,46410
Colten Spears,8457 Sycamore Ave.,Amsterdam,NY,12010
Kendra Aguilar,76 North Alton Lane,Tualatin,OR,97062
```

(3) (a) Write out a command to create a table that will hold this information, including appropriate data types. In addition to the fields in the CSV, your table should include an `id` column with the `serial` data type to uniquely identify the individuals.

```
CREATE TABLE addresses (
    id SERIAL,
    name TEXT,
    address TEXT,
    city TEXT,
    state CHAR(2),
    zip CHAR(5)
);
```

(3) (b) Write out a command to import the data from the CSV file into your above created table. You can assume the CSV file is located at `/data/addresses.csv`.

```
COPY addresses (name,address,city,state,zip)
FROM '/data/addresses.csv'
WITH (FORMAT CSV);
```

(2) (c) After importing the data, you realize that your table is still missing the information for Peter Hood, who lives at 73 East Wrangler Street, New Kensington, PA 15068. Write a command to add this information to the end of your table.

```
INSERT INTO addresses (name,address,city,state,zip) VALUES
    ('Peter Hood','73 East Wrangler Street','New Kensington','PA','15068');
```

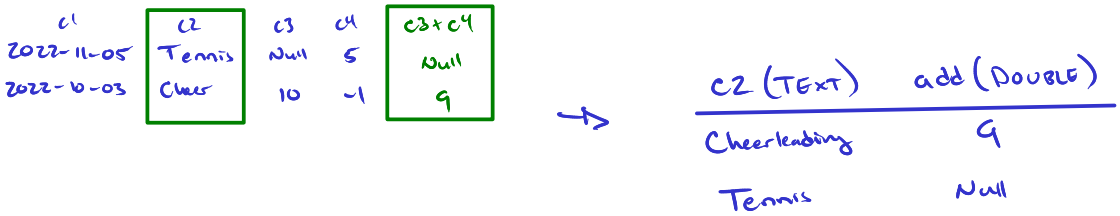
3. You have a table named `amazing` in your database that looks like below.

c1 *DATE*	c2 *TEXT*	c3 *INT*	c4 *DOUBLE PRECISION*
2022-06-29	Curling	16	-0.5
2022-11-05	Tennis	NULL	5
2022-06-12	Baseball	2	-0.5
2022-04-27	Ultimate	1	4
2022-05-15	Surfing	NULL	4
2022-10-03	Cheerleading	10	-1

Use it to determine the output of the below queries, **including column names and type**.

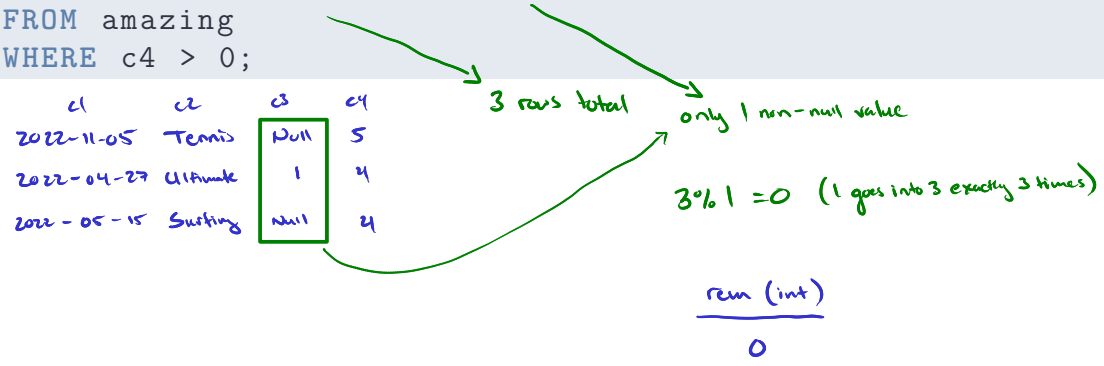
(3) (a)

```
SELECT c2, c3 + c4 AS "add"
FROM amazing
WHERE c2 ILIKE '%e%i%'
ORDER BY c2;
```



(3) (b)

```
SELECT COUNT(*) % COUNT(c3) AS rem
FROM amazing
WHERE c4 > 0;
```



(3) (c)

```
SELECT DISTINCT c4 AS special
FROM amazing
WHERE c1 < 'July 1, 2022' AND c3 >=1
ORDER BY c4;
```

rows 1,3,4,5

rows 1,3,4,6

c1	c2	c3	c4
2022-06-29	Curling	16	-0.5
2022-06-12	Baseball	2	-0.5
2022-04-27	Ultimate	1	4

Distinct

special (DOUBLE)

-0.5

4

(3) (d)

```
SELECT sum(wow.c3 * bazinga.c3) AS "sum"
FROM amazing AS wow
JOIN amazing AS bazinga
  ON wow.c4 ^ 2 = bazinga.c3
WHERE wow.c3 != bazinga.c3
```

c1 & c2 never used so just going to track c3 & c4

wow

c3	c4
16	-0.5 .25
NULL	5 .25
2	-0.5 .25
1	4 16
NULL	4 16
10	-1 1

bazinga

c3	c4
16	-0.5
NULL	5
2	-0.5
1	4
NULL	4
10	-1

on the join, only 16 & 1 match →

more are the same

wow c3	wow c4	bazinga c3	bazinga c4
1	4	16	-0.5
NULL	4	16	-0.5
10	-1	1	4

wow.c3 * bazinga.c3

16

NULL

10

Sum ignores NULLS so 26

sum (int)

26

4. Wordle has taken the world by storm, so suppose you had two tables in your database keeping track of various player's performance. The table named `puzzles` has the following columns:

Name	Type	Description
`id`	`INT`	The unique puzzle id number
`release_date`	`DATE`	The day the puzzle was publically available
`solution`	`CHAR(5)`	The 5 letter solution for that puzzle

While the table named `submissions` has the following columns:

Name	Type	Description
`id`	`SERIAL`	A unique id of this submission
`puzzle_id`	`INT`	The id of the puzzle that was played
`player_name`	`TEXT`	The name of the player
`guesses`	`SMALLINT`	The number of guesses until solving the puzzle. `Null` if they failed to solve the puzzle in less than 7 guesses.

Each puzzle appears once in the `puzzles` table, while each puzzle submission by a player adds a row to the submissions table. So, for example, a **few rows** of each of the tables might look something like:

id	release_date	solution
⋮		
222	2022-01-27	mount
223	2022-01-28	perky
224	2022-01-29	could
225	2022-01-30	wrung
⋮		

id	puzzle_id	player_name	guesses
⋮			
100	222	Frank	5
101	222	Joe	4
102	223	Frank	`NULL`
103	224	Jill	6
⋮			

You can be assured that each puzzle only appears once in the `puzzles` table and that any player can only submit each puzzle once in the `submissions` table. Using these tables, construct queries that would answer the following questions.

(3) (a) When the player named "Bobby" attempts a puzzle, he succeeds in solving it what *percentage* of the time?

No need of puzzles here.

```
SELECT count(guesses) / count(*)::real * 100
FROM submissions
Where player_name = 'Bobby';
```

(3) (b) What is the most common number of guesses it takes any player to complete the puzzle if the letter "a" is the second letter?

mode

Need to join

```
SELECT
    mode() WITHIN GROUP (ORDER BY guesses)
FROM submissions as s
JOIN puzzles as p
  ON p.id = s.puzzle_id
WHERE p.solution ILIKE '_a%';
```

(3) (c) Which wordle puzzles have not been attempted by any players? There should be no duplicates in this list of puzzle ids.

Need join

```
SELECT DISTINCT
    p.id
FROM puzzles as p
LEFT JOIN submissions as s
    ON s.puzzle_id = p.id
WHERE s.id IS NULL
```

(3) 5. Match the below terms to the description that best matches. Each term will only connect to a single description.

B an in-memory map of where to find each key on disk

C tells the computer exactly what should happen and how

F a storage method which enables very quick lookups

A an append-only sequence of records

D tells the computer what you would like to occur and lets it figure out how

E an ordered log segment

A. A log
B. A hash table
C. A imperative language
D. A declarative language
E. A SSTable
F. A B-tree

(2) 6. You have a set of data which contains predominantly many-to-many relationships. What type of storage model would likely be most ideal?

A. A relational model
B. A document model
C. A graph model
D. A runway model

(2) 7. You are worried about tolerating hardware faults in your data system. Which action below would be mostly likely to help?

A. Improving documentation ← reduce user faults

B. Upgrading to a faster CPU ← improve scaling / reduce load

C. Duplicating information across multiple drives ← redundant drives => seamless switches when one drive dies

D. Improving network speeds ← scaling

(12) 8. **Database Models:** Choose **one** of the following prompts to discuss in 4-6 sentences. Circle the prompt please so that it is clear which you are responding to.

(a) Your boss recently read an article about relational databases, and is now convinced they are the end-all-be-all in storage solutions. How would you explain to them some shortcomings of relational databases?

(b) Your boss recently watched a news segment which mentioned a particular document database system. They are confused about how this would compare to your current relational database storage solution. How would you compare and contrast the two in a way that your boss would understand?

a) Major themes:

- Scaling diminishes as systems get huge w/ many tables across many systems
  - ↳ each query needs to 'join' many tables and so needs to lookup on a lot of drives
- Can handle many-to-many, but graph models do it easier
- Schema can be inflexible, complicating changes & upgrades
- Many variants are still commercial & so may be expensive
- "Impedance mismatch" with OOPrograms

b) Major Themes

- Document models store info relating to an "object" all together
  - ↳ can commonly make for faster lookups, especially for huge systems
- Lots more schema flexibility in Document models, simplifying changes
- Document models struggle w/ many-to-many relationships
- Document models have limited support for joins (if any)

(12) 9. **Storage Models:** Choose **one** of the following prompts to discuss in 4-6 sentences. Again, please circle the chose prompt.

(a) Your friend keeps seeing the terms OLTP and OLAP thrown around in documentation that they are reading. Explain to them the differences between the two and in what situations each is used.

(b) A friend keeps seeing the term ETL show up when they search for data engineering positions. Explain to them what it means, entails, and where it tends to show up in common data pipelines.

a) Major themes:

OLTP: Online Transaction Processing
- Handles much of the day to day information storage
- Many different users interface with the db, usually through software
- Writing or reading specific records should be snappy

OLAP: Online Analysis Processing
- For more long term storage or analytics
- A few analysts access it, usually through SQL queries
- Computing aggregate calculations is the focus

b) Major themes:

ETL: Extract, Transform, Load
- Largely describes the process of moving data from short-term application-heavy areas to long term analytic-heavy areas
- Bridges OLTP & OLAP
- Extracts info from smaller dbs, transforms or aggregates for better analytic storage, and the loads to the OLAP db. — Can happen in bulk or stream models

(3 (bonus)) 10. For each major data storage model (relational, document, and graph), name two specific database systems that utilize that particular model.

See Ch2 map