Data 101/CS 187: Data Engineering
Midterm Exam Solutions

UC Berkeley, Fall 2025

October 16, 2025

Name: ______________________________

Email: ______________________________@berkeley.edu

Student ID: ______________________________

Examination room: ______________________________

Name of the student on your left: ______________________________

Name of the student on your right: ______________________________

Instructions

Do not open the examination until you are instructed to do so.

This exam consists of **101** points spread over **6 questions** (including the Honor Code), and must be completed in the 110-minute time period on October 16, 2025, 7:10pm – 9:00pm unless you have pre-approved accommodations otherwise.

For multiple-choice questions, select **one choice** for circular bubble options, and select **all choices that apply** for box bubble options. In either case, please indicate your answer(s) by **fully** shading in the corresponding circle/box.

Make sure to write your SID on each page to ensure that your exam is graded.

Honor Code **[1 pt]**

As a member of the UC Berkeley community, I act with honesty, integrity, and respect for others. I am the person whose name is on the exam, and I completed this exam in accordance with the Honor Code.

Signature: ______________________________

Chapter 1: This One's for the Books [24 pt]

```
CREATE TABLE students {
 id INT,
 name VARCHAR(30) NOT NULL,
 email VARCHAR(50) NOT NULL,
 major VARCHAR(40) NOT NULL,
 second_major VARCHAR(40),
 minor VARCHAR(40),
 PRIMARY KEY id
};
```

```
CREATE TABLE libraries {
 name VARCHAR(60),
 hours VARCHAR(10) NOT NULL,
 study_spaces BOOLEAN,
 max_capacity INT,
 snacks_allowed BOOLEAN,
 PRIMARY KEY name
};
```

`study_spaces` attribute indicates whether the library can be used for studying.

1.1. [5 pt] UC Berkeley knows that its students love the libraries, so it wanted to conduct a survey to figure out which libraries were students' favorites. Each student filled out the survey once and was only allowed to select one library as their favorite. This resulted in the following survey_results relation:

```
CREATE TABLE survey_results {
 id INT UNIQUE,
 fav_lib VARCHAR(60),

 FOREIGN KEY (id) REFERENCES students (id),
 FOREIGN KEY (fav_lib) REFERENCES libraries (name)
};
```

Write a query to return the 5 most popular libraries according to the survey (aka the libraries that received the greatest number of votes). When multiple libraries have received the same number of votes, break ties by sorting the library names in alphabetical order.

SELECT ____

FROM ____

____

____

____;

Solution:

```
SELECT fav_lib

FROM survey_results

GROUP BY fav_lib

ORDER BY COUNT(*) DESC, fav_lib ASC

LIMIT 5;
```

1.2. You create a materialized view called `students_lib` that performs a join on the students and `survey_results` relations to save the combined students and survey data.

```
CREATE MATERIALIZED VIEW students_lib AS (
SELECT s.id AS id, name, email, major, second_major, minor, fav_lib
FROM students AS s
JOIN survey_results AS sr
 ON s.id = sr.id);
```

You are not sure if it would have made more sense to create a new table, a view, or to use a CTE instead. Given the following conditions, select the best ways to save the results of your query to reduce query cost or the amount of code you have to write (**select all that apply for each setting**).

i. [1 pt] The survey is halfway completed and students are still responding. You want to include newly added responses in the queries you write referencing the joined tables.

□ A. TABLE

□ B. MATERIALIZED VIEW

□ **C. VIEW**

□ D. CTE

□ E. SUBQUERY

ii. [1 pt] You only need to join students and survey_results for one query, with all subsequent queries operating on students or survey_results alone.

□ A. TABLE

□ B. MATERIALIZED VIEW

□ C. VIEW

□ **D. CTE**

□ **E. SUBQUERY**

iii. [1 pt] The survey has closed, and now you want to do analysis on the results.

□ **A. TABLE**

□ **B. MATERIALIZED VIEW**

□ C. VIEW

□ D. CTE

□ E. SUBQUERY

1.3. According to the survey, the majority of Data Science (DS) majors have the same favorite library. The UC Berkeley Library wants to let DS students know about other libraries that are a good alternative to their favorite library so that library isn't overrun with students.

Find the names of all libraries with the exact same opening and closing hours as the DS majors' favorite library and that also have study spaces available. Use the `students_lib` materialized view created in Question 1.2.

You can assume that there is a single library that received the most votes; you must include this favorite library in the final result. Additionally, remember that DS could be a student's primary or second major. The relevant schema and materialized view are repeated below for convenience.

```
CREATE MATERIALIZED VIEW students_lib AS (
SELECT s.id AS id, name, email, major, second_major, minor, fav_lib
FROM students AS s
JOIN survey_results AS sr
 ON s.id = sr.id);
```

```
students_lib(id, name, email, major, second_major, minor, fav_lib)
libraries(name, hours, study_spaces, max_capacity, snacks_allowed)
```

```
SELECT ____________________ A ____________________

FROM ____________________ B ____________________

WHERE __________ C __________ AND __________ D __________

      (SELECT __________________ E __________________

       FROM students_lib AS sl

 ________________ F ________________ JOIN libraries AS l

          ON __________________ G __________________

       WHERE __________________ H __________________

       GROUP BY sl.fav_lib

       ORDER BY COUNT(*) DESC

       LIMIT 1);
```

i. [1 pt] Fill in section A

Solution: libraries.name

ii. [1 pt] Fill in section B

Solution: libraries

iii. [1 pt] Fill in section C

Solution: study_spaces = TRUE

iv. [1 pt] Fill in section D

Solution: hours =

v. [1 pt] Fill in section E

Solution: l.hours

vi. [1 pt] Select **one** of the following to fill in section F

[✓ ○ **A. INNER**]

○ B. LEFT

○ C. RIGHT

○ D. OUTER

vii. [1 pt] Select **all** that apply to fill section G

□ **A. sl.fav_lib = l.name**

□ B. name = fav_lib

□ C. libraries.name = students_lib.name

□ **D. libraries.name = students_lib.fav_lib**

viii. [1 pt] Fill in section H

____

Solution: major = 'Data Science' OR second_major = 'Data Science'

1.4. When examining the libraries relation later, you notice that a few libraries' hours got messed up at some point after you performed the queries about the survey from the previous questions (this new development will not impact any of your previous queries). Library hours are now formatted in two different ways. You want to use string manipulation to change the hours so that they all follow the format:

```
'{opening hour} {a.m.|p.m.}-{closing hour} {a.m.|p.m.}'
 ex. 10 a.m.-5 p.m.
```

To test out your commands you select a subsection of the libraries relation and call it `libraries_test`. The new `libraries_test` relation is shown below:

name	**hours**
Kresge Engineering & Mathematical Sciences Library	9 a.m.−11 p.m.
C. V. Starr East Asian Library	9 a.m.−10 p.m.
Environmental Design Library	9:00am−10:00pm
Main (Gardner) Stacks	9 a.m.−2 a.m.
Ethnic Studies Library	10:00am−5:00pm

You write the following query to alter the hours:

```
SELECT SUBSTRING(hours, 1, 1) || ‘ a.m.-’ || SUBSTRING(hours, 8, 2) || ‘ p.m.’
AS hours
FROM libraries_test
WHERE hours NOT LIKE ‘%a.m.%p.m.’;
```

i. [1 pt] How many rows are selected from the libraries_test relation after we run the query?

○ A. 0

○ B. 1

○ C. 2

○ **D. 3**

ii. [1 pt] After the query is run, the new ‘hours’ value for the ‘Environmental Design Library’ tuple will be the same as the ‘hours’ value for which library?

○ A. Kresge Engineering & Mathematical Sciences Library

○ **B. C. V. Starr East Asian Library**

○ C. Main (Gardner) Stacks

○ D. None

1.5. The Earth Sciences & Map Library allows students to get 10 free scans of maps in its collection! Not many students know about this opportunity, and the library wants to advertise this service to students who have not used their scans.

```
CREATE TABLE map_scans {
 student_id INT NOT NULL,
 map_id INT NOT NULL,
 request_date CHAR(9),

 FOREIGN KEY (student_id) REFERENCES students (id)
};
```

Write a query to return the names of all students who have used less than 2 of their free map scans, and include student id, name, and email in the output. Fill in each subsection of the query below.

Schema for your reference:

```
students(id, name, email, major, second_major, minor)
map_scans(student_id, map_id, request_date)
```

```
SELECT ____________________ A ____________________

FROM map_scans AS ms

RIGHT JOIN ____________________ B ____________________

 ON ____________________ C ____________________

 GROUP BY ____________________ D ____________________

 ____________________ E ____________________
```

i. [1 pt] Fill in section A

Solution: student.id, student.name, student.email

ii. [1 pt] Fill in section B

Solution: students

iii. [1 pt] Fill in section C

Solution: ms.student_id = students.id

iv. [1 pt] Fill in section D

Solution: students.id, students.name, students.email

v. [2 pt] Select all that options that result in a functioning query given the rest of your responses to fill in section E

□ A. WHERE COUNT(s.id) < 2

□ **B. HAVING COUNT(ms.student_id) < 2**

□ **C. HAVING COUNT(ms.map_id) < 2**

□ D. WHERE COUNT(ms.student_id) < 2

Chapter 2: Don’t Bag My Labubu [23 pt]

Labubus are a collectible line of toys produced by Popmart. Each Labubu figure is part of a series. In each series, there is a special “secret” one that is rarer and harder to find. In the context of this database, the owners table represents all individuals who own a labubu in the labubus table based on a unique identifier. Multiple labubus can have the same `product_id` that references the Popmart table. The Popmart table is a catalog of all the products that Popmart sells, not just labubus.

```
 CREATE TABLE owners (
    id INT PRIMARY KEY,
    name VARCHAR(255),
    age INT,
    labubu_id INT,
    FOREIGN KEY (labubu_id) REFERENCES labubus(id));

CREATE TABLE labubus (
    id INT PRIMARY KEY,
    name VARCHAR(255),
    series VARCHAR(255),
    secret BOOLEAN,
    birthday DATE,
    product_id INT,
    FOREIGN KEY (product_id) REFERENCES Popmart(id));

CREATE TABLE Popmart (
    id INT PRIMARY KEY,
    product_name VARCHAR(255),
    price DECIMAL(10, 2),
    inStock BOOLEAN);
```

2.1. Identify all the relational algebra expressions that satisfy the given descriptions. **Select all that apply**.

i. [3 pt] Find the names of owners who own a Labubu from the series "BigIntoEnergy"

□ **A.** $\pi_{owners.name}(owners \bowtie_{owners.labubu\_id=labubus.id} (\sigma_{series='BigIntoEnergy'}(labubus)))$

□ **B.** $\pi_{owners.name}(owners \bowtie_{owners.labubu\_id=labubus.id \wedge labubus.series='BigIntoEnergy'} labubus)$

□ C. $\pi_{owners.name}(\sigma_{series='BigIntoEnergy'}(owners \bowtie labubus))$

□ D. $\pi_{owners.name}\left((\pi_{owners.name}(owners)) \bowtie_{owners.labubu\_id=labubus.id \wedge labubus.series='BigIntoEnergy'} labubus\right)$

ii. [3 pt] Find product IDs that are in Popmart, but are not labubus

□ A. $\pi_{product\_id}(labubus) - \pi_{id}(Popmart)$

□ B. $\pi_{id}(Popmart) - \pi_{id}(labubus)$

□ C. $\rho_{id \rightarrow product\_id}(\pi_{id}(Popmart)) \cap \pi_{product\_id}(labubus)$

□ D. $\rho_{id \rightarrow product\_id}(\pi_{id}(Popmart)) - \pi_{product\_id}(labubus)$

iii. [3 pt] Find ids of Labubus that are secret editions but their product is not in stock at Popmart.

□ A. $\Big(\pi_{labubus.id}(\sigma_{secret='true'}(labubus)) -$
$\pi_{labubus.id}\big((labubus \bowtie_{labubus.product\_id=Popmart.id} (\sigma_{inStock='true'}(Popmart))\big)\Big)$

□ B. $\pi_{labubus.id}(\sigma_{secret='true' \wedge inStock='false'}(labubus \bowtie Popmart))$

□ C. $\pi_{labubus.id}(\sigma_{secret='true'}(labubus)) \bowtie \pi_{id}(\sigma_{inStock='false'}(Popmart))$

□ D. $\pi_{labubus.id}(labubus) - \pi_{labubus.id}(labubus \bowtie Popmart)$

2.2. [2 pt] What is the following relational algebra expression doing? Write one sentence in plain English. Do not use any relational algebra terminology. γ refers to GROUP BY.

$\rho_{name,series,\text{COUNT}(*)\rightarrow num\_products}(\gamma_{owners.name,labubus.series,\text{COUNT}(*)}(owners \bowtie_{owners.labubu\_id=labubus.id} labubus))$

____

____

____

Solution: Find the number of labubus that each owner has in each series

2.3. Write this query using only primitive operators $\{\sigma, \pi, \rho, \times, \cup, -\}$: **Find the owner names who have a labubu worth more than \$60**.

π ____________ A ____________ (σ ____________ B ____________ $\wedge$

____________ C ____________ $\wedge$ ____________ D ____________

______ (owners ______ E ______ labubus ______ F ______ Popmart))

i. [1 pt] Fill in the blank for section A.

○ **A. owners.name**

○ B. Popmart.price > 60

○ C. labubus.name

○ D. owners.price > 60

ii. [1 pt] What operator can go in blanks E and F?

○ A. −

○ B. ∪

○ **C. ×**

iii. [3 pt] Which of the following conditions must be part of the selection clause (sections B, C, or D)? **Select all that apply**.

□ **A. Popmart.price > 60**

□ **B. owners.labubu_id = labubus.id**

□ **C. labubus.product_id = Popmart.id**

□ D. owners.age > 60

2.4. The TAs and Tutors love Labubus. Consider the following collections of tuples with the same schema (owner_id, name, labubu_id).

- **TAs** = { (1, ’Vicky’, 101), (1, ’Vicky’, 101), (1, ’Vicky’, 107), (6, ’Elizabeth’, 103), (6, ’Elizabeth’, 108) }
- **TALabubus** = { (1, ’Happiness’, 101), (1, ’Happiness’, 101), (1, ’DuoDuo’, 107), (6, ’Lychee Berry’, 103), (6, ’Hope’, 108), (NaN, ’SiSi’, 105), (NaN, ’ZiZi’, 106) }
- **Tutors** = { (2, ’Sydney’, 102), (3, ’Sofia’, 103), (4, ’Shashwat’, 104), (5, ’Jiajun’, 105), (8, ’Andrea’, NaN) }
- **TutorLabubus** = { (2, ’Luck’, 102), (3, ’Lychee Berry’, 103), (4, ’Thanh’, 104), (5, ’SiSi’, 105), (NaN, ’SiSi’, 105), (NaN, ’ZiZi’, 106) }

Option	Result Set / Bag
A	{(NaN, ’SiSi’, 105), (NaN, ’SiSi’, 105), (NaN, ’ZiZi’, 106), (NaN, ’ZiZi’, 106)}
B	{ (1, ’Happiness’, 101), (1, ’Happiness’, 101), (1, ’DuoDuo’, 107), (2, ’Luck’, 102), (3, ’Lychee Berry’, 103), (4, ’Thanh’, 104), (5, ’SiSi’, 105), (NaN, ’SiSi’, 105), (NaN, ’ZiZi’, 106), (6, ’Lychee Berry’, 103), (6, ’Hope’, 108) }
C	{ (1, ’Vicky’, 101), (1, ’Vicky’, 107), (6, ’Elizabeth’, 103), (6, ’Elizabeth’, 108) }
D	{ (1, ’Happiness’, 101), (1, ’DuoDuo’, 107), (2, ’Luck’, 102), (3, ’Lychee Berry’, 103), (4, ’Thanh’, 104), (5, ’SiSi’, 105), (6, ’Lychee Berry’, 103), (6, ’Hope’, 108) }
E	{ (1, ’Happiness’, 101), (1, ’Happiness’, 101), (1, ’DuoDuo’, 107), (2, ’Luck’, 102), (3, ’Lychee Berry’, 103), (4, ’Thanh’, 104), (5, ’SiSi’, 105), (NaN, ’SiSi’, 105), (NaN, ’SiSi’, 105), (NaN, ’ZiZi’, 106), (NaN, ’ZiZi’, 106), (6, ’Lychee Berry’, 103), (6, ’Hope’, 108) }
F	{ (1, ’Vicky’, 101), (1, ’Vicky’, 101), (1, ’Vicky’, 107), (2, ’Sydney’, 102), (3, ’Sofia’, 103), (4, ’Shashwat’, 104), (5, ’Jiajun’, 105), (6, ’Elizabeth’, 103), (6, ’Elizabeth’, 108), (8, ’Andrea’, NaN) }
G	{ (1, ’Vicky’, 101), (1, ’Vicky’, 101), (1, ’Vicky’, 107), (6, ’Elizabeth’, 103), (6, ’Elizabeth’, 108) }
H	{(NaN, ’SiSi’, 105), (NaN, ’ZiZi’, 106)}
I	{ (1, ’Happiness’, 101), (1, ’Happiness’, 101), (1, ’DuoDuo’, 107), (6, ’Lychee Berry’, 103), (6, ’Hope’, 108) }
J	{ (1, ’Vicky’, 101), (6, ’Elizabeth’, 103) }

Match the expression with the corresponding result set/bag. You can use an option more than once, and not all options may be used. For each operator, semantics are specified as subscript. `Set(.)` or `Bag(.)` converts the input to a set or bag respectively. Fill **only** the option character (A–J) in the provided box.

i. [1 pt] Set(TAs)

____

Solution: C

ii. [1 pt] TALabubus $\cup_{\text{bag}}$ TutorLabubus

____

Solution: E

iii. [1 pt] TAs $-_{\text{set}}$ Tutors

____

Solution: C

iv. [1 pt] TALabubus $\cap_{\text{bag}}$ TutorLabubus

____

Solution: H

v. [1 pt] Bag(TAs)

____

Solution: G

vi. [2 pt] (TALabubus $\cup_{\text{set}}$ TutorLabubus) $-_{\text{set}}$ (TALabubus $\cap_{\text{set}}$ TutorLabubus)

____

Solution: D

Chapter 3: DML/DDL [14 pt]

Consider the following tables: `Teams` and `Drivers`.

```
CREATE TABLE Teams (
    TeamID INT PRIMARY KEY,
    TeamName VARCHAR(50) NOT NULL UNIQUE,
    BaseCountry VARCHAR(50) NOT NULL
);


CREATE TABLE Drivers (
    DriverID INT PRIMARY KEY,
    DriverName VARCHAR(50) NOT NULL,
    Nationality VARCHAR(50) NOT NULL,
    TeamID INT,
    Points INT CHECK (Points >= 0),
    FOREIGN KEY (TeamID)
        REFERENCES Teams(TeamID)
        ON DELETE SET NULL ON UPDATE CASCADE);
```

TeamID	TeamName	BaseCountry
1	Red Bull Racing	Austria
2	Ferrari	Italy
3	Mercedes	Germany

DriverID	DriverName	Nationality	TeamID	Points
44	Lewis Hamilton	UK	3	200
33	Max Verstappen	Netherlands	1	350
16	Charles Leclerc	Monaco	2	250
11	Sergio Perez	Mexico	1	180

3.1. [2 pt] Which of the following ALTER statement(s) is valid? **Select all that apply**.

□ **A. ALTER TABLE Drivers ADD COLUMN Podiums INT;**

□ B. ALTER TABLE Teams DROP COLUMN TeamID;

□ C. ALTER TABLE Drivers ADD COLUMN Points INT;

□ D. ALTER TABLE Drivers DROP COLUMN DriverID;

3.2. [2 pt] Which of the following INSERT statement(s) will fail? Assume that the current state of the Driver table. **Select all that apply**.

□ A. INSERT INTO Drivers VALUES (77, ’Valtteri Bottas’, ’Finland’, 3, 120);

□ **B. INSERT INTO Drivers VALUES (44, ’Hamilton’, ’UK’, 3, 100);**

□ **C. INSERT INTO Drivers VALUES (22, NULL, ’Germany’, 3, 90);**

□ **D. INSERT INTO Drivers VALUES (5, ’Daniel Ricciardo’, ’Australia’, 10, 0);**

□ **E. INSERT INTO Drivers VALUES (55, ’Carlos Sainz’, ’Spain’, 2, -50);**

3.3. [2 pt] For the given table, is it possible to make the combination of `(DriverName, Nationality, TeamID)` a primary key? Answer True/False and provide a 1 line justification.

○ True ○ **False**

Solution: There can exist multiple drivers with the same name, nationality, and belonging to the same team, so the combination does not guarantee uniqueness.

Since TeamID in Drivers allows NULL (because of the ON DELETE SET NULL clause), this combination cannot serve as a primary key because all columns in a primary key must be NOT NULL.

3.4. [2 pt] If a team's TeamID is updated, all drivers referencing that team will automatically update their TeamID. Answer true or false, and provide a 1 line justification.

○ **True** ○ False

Solution: ON UPDATE CASCADE clause updates the TeamID in the drivers table.

3.5. Suppose we have the following data now:

TeamID	TeamName	BaseCountry
1	Red Bull Racing	Austria
2	Ferrari	Italy
3	Mercedes	Germany

DriverID	DriverName	Nationality	TeamID	Points
44	Lewis Hamilton	UK	3	200
33	Max Verstappen	Netherlands	1	350

You execute the following query:

```
DELETE FROM Teams WHERE TeamID = 1;
UPDATE Teams SET TeamID = 5 WHERE TeamID = 3;
```

After these commands, what are the TeamID values in Drivers for the following drivers? For each DriverID, please write down only the value of the TeamID, nothing else. If it can't be determined, put NA.

i. [1 pt] DriverID 44

____

Solution: 5

ii. [1 pt] DriverID 33

____

Solution: NULL

iii. [1 pt] Suppose instead the Drivers.TeamID foreign key was defined as:

```
FOREIGN KEY (TeamID) REFERENCES Teams(TeamID) ON DELETE CASCADE
ON UPDATE CASCADE
```

After executing the same (UPDATE and DELETE) commands above, what is the TeamID value for DriverID 44?

____

Solution: 5

iv. [1 pt] Suppose a backup of the original drivers table was not taken. Assuming the foreign key was defined with ON DELETE CASCADE (as in the previous question), what data has been lost permanently after executing the given UPDATE and DELETE commands ?

○ A. No data loss

○ B. Team names for some drivers

○ C. Only team ids for drivers

○ **D. Entire row data for some drivers**

3.6. [2 pt] The Racing Director wants to increase each driver's points by 10. If a driver's total would exceed 15 after the increase, it should be set to 15. Complete the SQL statement below so that it correctly applies this rule.

```
UPDATE Drivers
SET Points = __________
```

```
WHERE Points < 15;
```

____

Solution: CASE WHEN Points + 10 > 15 THEN 15 ELSE Points + 10 END

Chapter 4: Query Optimization [13 pt]

The course staff for Data 101 is considering restaurants and days for a staff social. The staff, restaurants, and days that people are available are stored in tables with schema as follows:

```
CREATE TABLE staff (
 staff_id INT PRIMARY KEY,
 name VARCHAR(40),
 favorite_restaurant VARCHAR(40));

CREATE TABLE availability (
 date DATE,
 staff_name VARCHAR(40),
 restaurant_name VARCHAR(40),
 restaurant_address VARCHAR(30));
```

```
CREATE TABLE restaurants (
 restaurant_id INT PRIMARY KEY,
 restaurant_name VARCHAR(40),
 capacity INT,
 restaurant_address VARCHAR(30));
```

The staff table contains information about all of the staff members. The restaurants table contains information about restaurants in Berkeley. For each day a staff member is available for a social, a row containing information about that staff member and each restaurant open that day is present in the availability table. Suppose the **staff table** is **small** (less than 20 rows), while the **restaurants** and **availability** tables are **large** (>1000 rows). You may assume that hash partitions are similarly sized when hashing on any attribute.

4.1. For each of the following queries, select which join type is preferred. You may assume that no indexes exist when choosing the joins, other than those automatically made by PostgreSQL.

i. [1 pt]
```
SELECT staff.name, restaurants.restaurant_name
FROM staff INNER JOIN restaurants
ON staff.favorite_restaurant = restaurants.restaurant_name
LIMIT 5;
```

○ A. Hash Join

○ B. Sort Merge Join

○ **C. Nested Loop Join**

ii. [1 pt]
```
SELECT staff.name, restaurants.restaurant_name
FROM staff INNER JOIN restaurants
ON staff.favorite_restaurant = restaurants.restaurant_name
ORDER BY staff.name
LIMIT 5;
```

○ **A. Hash Join**

○ B. Sort Merge Join

○ C. Nested Loop Join

iii. [1 pt]
```
SELECT restaurants.restaurant_name, availability.date
FROM availability INNER JOIN restaurants
ON availability.restaurant_name = restaurants.restaurant_name
ORDER BY availability.date
```

○ **A. Hash Join**

○ B. Sort Merge Join

○ C. Nested Loop Join

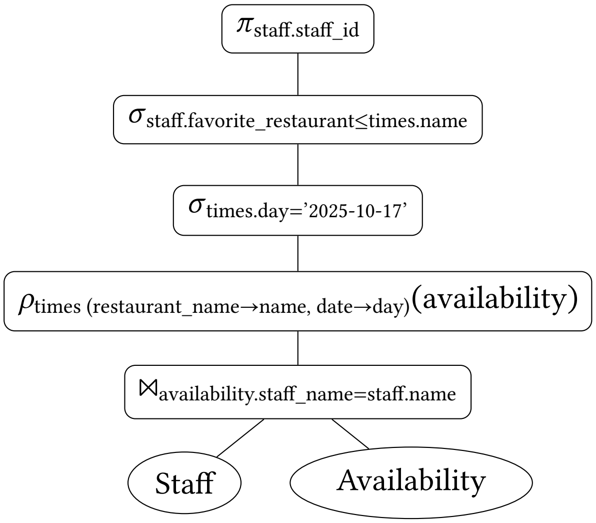
iv. [1 pt]
```
SELECT restaurants.restaurant_name, availability.date
FROM availability INNER JOIN restaurants
ON availability.restaurant_name = restaurants.restaurant_name
ORDER BY restaurants.restaurant_name
```

○ A. Hash Join

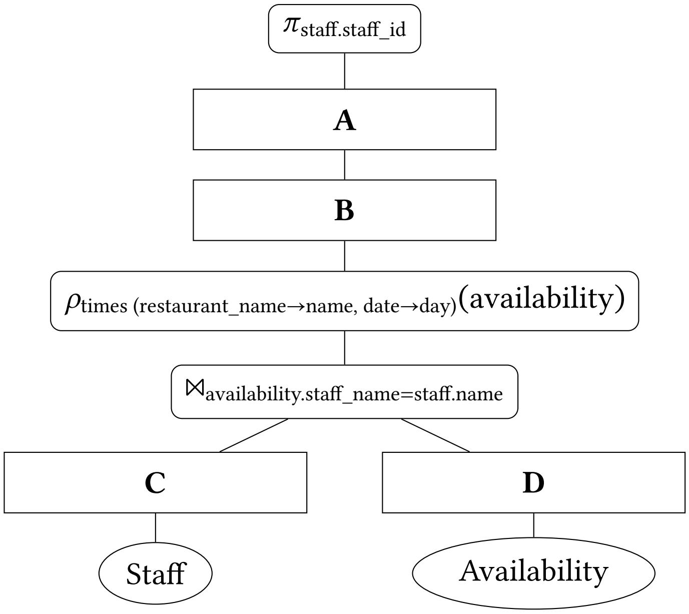
○ **B. Sort Merge Join**

○ C. Nested Loop Join

4.2. Consider the following unoptimized query plan:


$\pi_{\text{staff.staff\_id}}$
$\sigma_{\text{staff.favorite\_restaurant} \leq \text{times.name}}$
$\sigma_{\text{times.day='2025-10-17'}}$
$\rho_{\text{times (restaurant\_name} \to \text{name, date} \to \text{day)}}(\text{availability})$
$\bowtie_{\text{availability.staff\_name=staff.name}}$
Staff
Availability

Fill in the blanks below to represent the query plan after selection pushdown has been applied. You may not need to use all of the blanks. Write **NA** in the space provided if you think it should be blank.


$\pi_{\text{staff.staff\_id}}$
A
B
$\rho_{\text{times (restaurant\_name→name, date→day)}}(\text{availability})$
$\bowtie_{\text{availability.staff\_name=staff.name}}$
C
D
Staff
Availability

i. [1 pt] Fill in the blank for Section A.

Solution: N/A

ii. [1 pt] Fill in the blank for Section B.

Solution: $\sigma_{\text{staff.favorite\_restaurant} \leq \text{name}}$

iii. [1 pt] Fill in the blank for Section C.

Solution: N/A

iv. [1 pt] Fill in the blank for Section D.

Solution: $\sigma_{\text{availability.date} = \text{'2025-10-17'}}$

4.3. [2 pt] Suppose we run the following query, and currently have no indexes, other than the indices automatically made by PostgreSQL based on schema:

```
SELECT restaurants.restaurant_name
FROM availability INNER JOIN restaurants
ON availability.restaurant_address = restaurants.restaurant_address
WHERE restaurants.capacity = 42
ORDER BY availability.date ASC
```

Creating an index on which of the following fields would be expected to improve query performance? Assume that only one restaurant in Berkeley has a capacity of exactly 42. **Select all that apply**.

□ A. restaurants.restaurant_name

□ **B. restaurants.restaurant_address**

□ **C. availability.restaurant_address**

□ **D. restaurants.capacity**

□ **E. availability.date**

4.4. [3 pt] Which of the following are true about query optimization? **Select all that apply**.

□ A. Adding or deleting rows of data will not affect the query optimizer's query plan, but changing the schema of tables or adding indices may change its plan.

□ B. Data can be simultaneously clustered on any number of indices available.

□ C. Scanning over a clustered index is always preferable over scanning over an unclustered index if both indices are on attributes in selection predicates for a given query.

□ **D. Query optimizers may not choose a globally optimal plan for executing every query.**

□ **E. Materialized views can affect query performance, but views cannot.**

Chapter 5: Big Back 101 [16 pt]

You are a current analyst for the Data Infrastructure Committee at UC Berkeley. As part of your end-of-semester project, your team has partnered with Golden Bear Café (GBC) — a popular restaurant on Upper Sproul Plaza that serves thousands of students every day. GBC recently began recording detailed point-of-sale (POS) data for every order, and you've been given access to a dataset containing all orders placed by Data 101 staff members this semester. You've been asked to evaluate how this data should be represented in a database system, and estimate its storage requirements.

```
CREATE TABLE gbc_orders (
  order_id    CHAR(6),
  staff_101   VARCHAR(32),
  order_dt    VARCHAR(10),
  main_dish   VARCHAR(40),
  qty         INTEGER,
  price       DOUBLE);
```

order_id	staff_101	order_dt	main_dish	qty	price
104932	Sahil	2025-08-25	CHEESEBURGER	3	9.99
104933	Bing	2025-08-25	CHICKEN TENDERS	8	8.99
104934	Elizabeth	2025-08-26	CHICKEN SANDWICH	2	7.00
104935	Jiajun	2025-08-26	NA	1	10.01
104936	Pranav	2025-08-27	SALAD BOWL	3	11.49
104937	Joshua	2025-08-27	VEGGIE BURGER	6	8.50
104938	Thanh	2025-08-28	SALMON SANDWICH	2	11.99
104939	Sofia	2025-08-28	BULGOGI FRIES	2	12.00

5.1. Suppose Data 101 staff placed exactly 500 orders this semester. The café's data analyst team wants to estimate how much digital storage their ordering system would require in the worst-case scenario, assuming each VARCHAR column is filled to its maximum possible length. This helps ensure that their database can handle future growth in data volume and prevent potential storage overflow issues. Using the byte rules below, compute the total storage required in bytes. Assume ASCII characters. Storage rules (use these exactly):

- CHARs store 1 byte per character
- VARCHARs store 1 byte per character plus 1 byte of length metadata
- INTEGERs store 4 bytes
- DOUBLEs store 8 bytes

i. [5 pt] Per attribute per row (fill each):

Data Field	Size in Bytes
order_id	____________ bytes
order_dt	____________ bytes
main_dish (worst case)	____________ bytes
qty	____________ bytes
price	____________ bytes

Solution:

- order_id = 6 bytes
- order_dt = 11 bytes
- main_dish = 41 bytes
- price = 8 bytes

ii. [1 pt] The database schema defines the order_dt column using the VARCHAR(10) data type. While this can store dates in the 'YYYY-MM-DD' format, it is not considered best practice. In one or two sentences, describe a significant drawback of using the VARCHAR data type to store date information.

____

____

Solution: When date information is stored as a VARCHAR, we lose access to date-time properties such as sorting, filtering, and date arithmetic.

5.2. To reduce storage costs and improve query efficiency, the café's data engineering team is now exploring ways to optimize how menu items are stored in the ordering system. Instead of saving the full string for each main_dish, they decide to represent every unique menu item using a **fixed-length binary encoding**—where each unique dish is assigned a unique binary code. Suppose there are exactly **50 unique menu items** (main_dish) available this semester. Using this encoding approach, answer the following questions

i. [1 pt] How many bits would be required to store a single main_dish using this encoding?

____________ bits

Solution: 6 bits

ii. [1 pt] How many bits would be required in total to store the main_dish column for all 500 orders placed during the semester?

____________ bits

Solution: 3000 bits

5.3. The café's data analytics team has begun cleaning and preparing the order data for downstream analysis. However, they've discovered that some of the rows in the dataset appear to be corrupted — particularly in the price column, which contains a mixture of numeric and non-numeric entries To determine how best to represent this column and handle these inconsistencies, the team decides to apply the Minimum Description Length (MDL) principle to compare whether the data should be stored as integers or floats. Below is a snapshot of GBC orders for a few Data 101 staff members. For the purposes of this question, consider it the full dataset.

order_id	staff_101	order_dt	main_dish	qty	price
104932	Sahil	2025-08-25	CHEESEBURGER	3	9.99
104933	Bing	2025-08-25	CHICKEN TENDERS	8	1.00
104934	Elizabeth	2025-08-26	CHICKEN SANDWICH	2	7.00
104935	Jiajun	2025-08-26	NA	1	10.01
104936	Pranav	2025-08-27	SALAD BOWL	3	free
104937	Joshua	2025-08-27	VEGGIE BURGER	6	8.50
104938	Thanh	2025-08-28	SALMON SANDWICH	2	&^
104939	Sofia	2025-08-28	BULGOGI FRIES	2	12.00

We will use MDL to determine the optimal type for the **price** column. Assume floats are 32 bits, integers are 16 bits for this question, and a character is 8 bits for this question.

i. [3 pt] Determine whether the integer or float model is a better representation for the price column. Provide a 1 line explanation for which model MDL selects.

○ **INTEGER** ○ FLOAT

____

____

ii. [1 pt] Let's now suppose that instead of the invalid entry &^, the value in that row is replaced with 8. Considering the data type you selected above, would the updated sum of the new MDL (total number of bits) change?

○ Yes ○ **No**

5.4. After cleaning the corrupted entries and ensuring that all rows are now consistent and properly formatted, the café's analytics team wants to better understand how this data could be represented numerically for future modeling or machine learning tasks. Let's now suppose that we fixed our relation to now be as follows. Assume this is all the data accessible to you.

order_dt	main_dish	qty	price
2025-08-25	CHEESEBURGER	3	9.99
2025-08-25	CHICKEN TENDERS	8	8.99
2025-08-26	CHICKEN SANDWICH	2	7.00
2025-08-26	CHEESEBURGER	1	10.01
2025-08-27	SALAD BOWL	3	11.49
2025-08-27	VEGGIE BURGER	6	8.50
2025-08-28	SALMON SANDWICH	2	11.99
2025-08-28	BULGOGI FRIES	2	12.00

Suppose we were to convert this relation into a **matrix** for analytical purposes. Answer the following questions:

i. [2 pt] How many columns would the resulting matrix have? Use one-hot encoding for all categorical attributes.

____

Solution: 13 columns

ii. [2 pt] What happens to the number of columns in the matrix if the café adds 10 new and unique menu items next semester? Provide a 1 line explanation for your answer.

○ A. No Change

○ **B. Increases**

○ C. Decreases

____

____

____

Solution: One hot Encoding would result in 10 additional columns

Chapter 6: Data Preparation [10 pt]

Assume the following table named `orders` for the next set of questions:

```
CREATE TABLE orders (
id INT,
customer VARCHAR(10),
amount INT);
```

id	customer	amount
1	A	100
2	A	100
3	B	50
4	B	50

6.1. [2 pt] We are given the following window function query:

```
SELECT id,customer,amount,
  SUM(amount) OVER (PARTITION BY customer) AS total_by_customer
FROM orders;
```

Which of the following queries would produce the same result as the above window function query on the table shown above, without using any window functions?

○ A.

```
SELECT o.id, o.customer, o.amount,
SUM(o.amount) AS total_by_customer
FROM orders o
GROUP BY o.id, o.customer, o.amount;
```

○ B.

```
SELECT o.id, o.customer, o.amount,
    (SELECT SUM(o2.amount)
    FROM orders o2
    WHERE o2.customer = o.customer
    AND o2.id <= o.id) AS total_by_customer
FROM orders o;
```

○ C.

```
WITH totals AS (
  SELECT customer, id, SUM(amount) AS total_by_customer
  FROM orders
  GROUP BY customer, id
)
SELECT o.id, o.customer, o.amount, t.total_by_customer
FROM orders o
JOIN totals t
  ON t.customer = o.customer AND t.id = o.id;
```

○ **D.**

```
WITH totals AS (
  SELECT customer, SUM(amount) AS total_by_customer
  FROM orders
  GROUP BY customer
)
SELECT o.id, o.customer, o.amount, t.total_by_customer
FROM orders o
JOIN totals t
ON t.customer = o.customer;
```

○ E. None of the above

6.2. [2 pt] For the same table above write a query to add a new column `normalized_amount` that divides each Amount by the sum of the previous 5 Amount values (including the current row) for the same Customer, ordered by id. Assume that id corresponds to the time of transaction (higher id means a later transaction). Which of the following query correctly solves the given task?

○ A.

```
SELECT id, customer, amount,
amount * 1.0 / SUM(amount) OVER (
    PARTITION BY customer
    ROWS BETWEEN 5 PRECEDING AND CURRENT ROW
) AS normalized_amount
FROM orders;
```

○ B.

```
SELECT id, customer, amount,
Amount * 1.0 / SUM(amount) OVER (
    PARTITION BY customer
    ORDER BY id
    RANGE BETWEEN 5 PRECEDING AND CURRENT ROW
) AS normalized_amount
FROM orders;
```

○ C.

```
SELECT id, customer, amount,
amount * 1.0 / SUM(amount) OVER (
    ORDER BY id
    ROWS BETWEEN 4 PRECEDING AND CURRENT ROW
) AS normalized_amount
FROM orders;
```

○ **D.**

```
SELECT id, customer, amount,
amount * 1.0 / SUM(amount) OVER (
    PARTITION BY customer
    ORDER BY id
    ROWS BETWEEN 4 PRECEDING AND CURRENT ROW
) AS normalized_amount
FROM orders;
```

○ E. None of the above

6.3. Assume the following schema for the next set of questions:

```
CREATE TABLE state_lookup (
    state_numeric INT PRIMARY KEY,
    state_name VARCHAR(20));


CREATE TABLE national (
    school_id INT PRIMARY KEY,
    state_numeric INT,
    county_numeric INT,
    school_type VARCHAR(50),
    enrollment INT,
    academic_year INT,
    FOREIGN KEY (state_numeric) REFERENCES state_lookup(state_numeric));
```

i. [2 pt] Which of the following are examples of **rolling up** data based on the schema provided? **Select all that apply**.

□ **A. Summarizing enrollment numbers by state**

□ **B. Aggregating enrollment by county and school type**

□ **C. Aggregating total enrollment across all schools for each academic year**

□ D. Listing all the individual schools located in a specific state.

ii. [2 pt] Which of the following are examples of **drilling down** in the context of the schema? **Select all that apply**.

□ **A. Breaking down a state's total enrollment to see the totals for each county within that state**

□ B. Aggregating individual school enrollment data to calculate the total enrollment for each academic year.

□ C. Converting continuous enrollment counts into bins like [0–100], [101–500], [501+]

□ D. Filtering enrollment data to only show public schools

□ **E. Viewing the individual school records that make up a county's total enrollment**

iii. [2 pt] Our goal is to compute state-level total enrollment for the most recent `academic_year`, excluding rows with `school_type = 'Public'`. Output should have one row per state.

```
SELECT s.state_name, n.county_numeric,
SUM(n.enrollment) AS total_enrollment
FROM national n
JOIN state_lookup s
ON n.state_numeric = s.state_numeric
WHERE n.school_type <> 'Public'
GROUP BY s.state_name, n.county_numeric;
```

Identify the errors in the given query? **Select all that apply**.

□ **A. Grouping by `n.county_numeric` produces county-level results, not state totals.**

□ **B. The filter for the most recent `academic_year` is missing.**

□ **C. The query should group only by `s.state_name` to produce one row per state.**

□ D. The filter `school_type <> 'Public'` must be in a HAVING clause instead of WHERE.

□ E. SUM must be used with DISTINCT to avoid double counting.

Chapter 7: Congratulations! [0 pt]

Congratulations! You have completed this exam.

- Make sure that you have written your Student ID number on every other page of the exam. You may lose points on pages where you have not done so.
- Also ensure that you have signed the Honor Code on the cover page of the exam.
- If more than 10 minutes remain in the exam period, you may hand in the exam **and** the reference packet and leave.
- If ≤ 10 minutes remain, please sit quietly until the exam concludes.

[Optional, 0 pts] Use this page to draw your favorite Data 101/CS 187 moment!