Data 101/CS 187: Data Engineering
Final Exam Solutions

UC Berkeley, Fall 2025

Dec 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 **117** points spread over **5 questions** (including the Honor Code), and must be completed in the 170-minute time period on Dec 16, 2025, 8:00am – 11:00am 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: The SQL Squeakquel [22 pt]

You are a data engineer at a company MOOQL that builds software for tracking students' progress in online courses. The company stores information in the following relations.

```
CREATE TABLE students (
    id          INT PRIMARY KEY,
    name        VARCHAR(50) NOT NULL,
    join_date   DATE,
    major       VARCHAR(50));

CREATE TABLE courses (
    id              INT PRIMARY KEY,
    title           VARCHAR(100) NOT NULL,
    category        VARCHAR(40),
    difficulty_lvl  INT);

CREATE TABLE enrollments (
    id              INT PRIMARY KEY,
    student_id      INT REFERENCES students(id),
    course_id       INT REFERENCES courses(id),
    enroll_date     DATE,
    progress_pct    INT,        -- 0-100
    is_completed    BOOLEAN);
```

1.1. [3 pt] MOOQL wants to classify each student's engagement status based on their overall progress in a course. We define:

- If `progress_pct` = 100, the student's status is 'Completed'
- If `progress_pct` >= 50 but less than 100, status is 'In Progress'
- Otherwise, status is 'Low Engagement'

Write a SQL query to return the following columns: `student_id`, `course_id`, `progress_pct`, `engagement_status`, by filling the blanks below.

```
SELECT e.student_id, e.course_id, e.progress_pct
    _____
    _____
    _____
    _____
FROM enrollments e;
```

Solution:

CASE
WHEN e.progress_pct = 100 THEN 'Completed'

```
WHEN e.progress_pct >= 50 THEN 'In Progress'
ELSE 'Low Engagement'
END AS engagement_status
```

1.2. [5 pt] MOOQL wants to analyze how engagement in discussion forums changes over time. Each time a student makes a post, the system records its length (in characters). You want to identify all cases where a student's next post is longer than their current post, which may indicate growing understanding of the material.

A new table `discussion_posts` is available:

```
CREATE TABLE discussion_posts (
    id              INT PRIMARY KEY,
    student_id      INT REFERENCES students(id),
    course_id       INT REFERENCES courses(id),
    post_date       DATE,
    content_length  INT    -- number of characters in the post
);
```

Write a SQL query to find all posts where the next post by the same student has a longer `content_length` than the current post. Make sure your output includes the columns `student_id`, `post_id`, `content_length`,`next_content_length`

Available relations:

```
students(id, name, join_date, major)
courses(id, title, category, difficulty_lvl)
discussion_posts(id, student_id, course_id, post_date, content_length)
```

```
SELECT student_id, id  AS post_id, content_length

 _____( _____, _____),

OVER ( PARTITION BY _____

ORDER BY _____

)AS next_content_length

FROM discussion_posts

WHERE _____> _____
```

Solution:

```
SELECT student_id, id AS post_id, content_length,
LEAD(content_length, 1) – (or just LEAD(content_length))
OVER ( PARTITION BY student_id
ORDER BY post_date ASC
) AS next_content_length
FROM discussion_posts
WHERE next_content_length > content_length;
```

1.3. [3 pt] You are now given the following relations:

```
students(id, name, join_date, major)
courses(id, title, category, difficulty_lvl)
enrollments(id, student_id, course_id, enroll_date, progress_pct, is_completed)
```

Which of the following relational algebra expressions returns all students who have completed at least one course with difficulty greater than the average course difficulty of all courses they are enrolled in? For the expressions below, assume `s`, `e`, and `c` are aliases for the `students`, `enrollments`, and `courses` relations, respectively.

For the following expressions, you can assume

- R corresponds to $e \bowtie_{e.course_id=c.id} c$
- G corresponds to $\gamma_{e.student_id;\text{AVG}(c.difficulty_lvl)\rightarrow avg_diff}(e \bowtie_{e.course_id=c.id} c)$

◯ **A.**

$$\pi_{s.id,s.name}\left(s \bowtie_{s.id=e.student_id} \sigma_{e.is_completed=\text{TRUE}\wedge c.difficulty_lvl>\text{AVG}(c.difficulty_lvl)}(R)\right)$$

◯ **B.**

$$\pi_{s.id,s.name}\left(s \bowtie_{s.id=e.student_id} \sigma_{e.is_completed=\textbf{TRUE}\wedge c.difficulty_lvl>avg_diff}\left(R \bowtie_{e.student_id} G\right)\right)$$

◯ **C.**

$$\pi_{s.id,s.name}\left(s \bowtie \sigma_{is_completed=TRUE}(R \bowtie G)\right)$$

◯ **D.**

$$\pi_{s.id,s.name}\left(s \bowtie_{s.id=e.student_id} \sigma_{e.is_completed=\text{TRUE}\wedge c.difficulty_lvl>G}(R)\right)$$

1.4. [2 pt] A data analyst is investigating enrollments for a massively popular introductory course. They run the following query and forced the database to use B+ tree index on the `course_id` column, so the query planner is choosing an Index Scan.

```
SELECT * FROM enrollments WHERE course_id = 101;
```

The enrollments table contains 1,000,000 tuples of which 50% are for 101. The index on `course_id` is unclustered, and each page contains hundreds of tuples.

Given this scenario, select all that apply.

☐ **A. Using an unclustered index is inefficient because it leads to a large number of random I/O operations, as the required tuples are scattered across disk pages.**

☐ **B. Having a clustered index on `course_id` would improve the performance of the query**

☐ **C. A sequential scan of the disk pages is likely to be faster than the index scan.**

☐ **D. The query would run faster if the condition was `course_id = 202` which has 20% of the total tuples.**

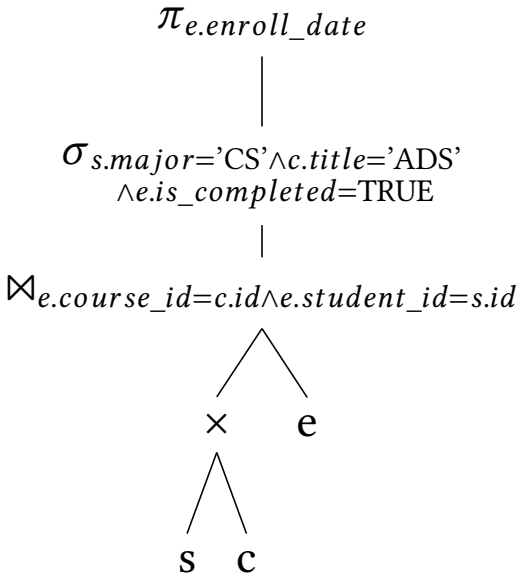
1.5. The data engineering team is running a query to find the `enroll_date` for all computer science (CS) majors who have completed the highly specialized course, Advance Database Systems (aka ADS).

```
SELECT e.enroll_date
FROM students s
JOIN enrollments e ON s.id = e.student_id
JOIN courses c ON e.course_id = c.id
WHERE s.major = 'CS'
  AND c.title = 'ADS'
  AND e.is_completed = TRUE;
```

Assume the following statistics:

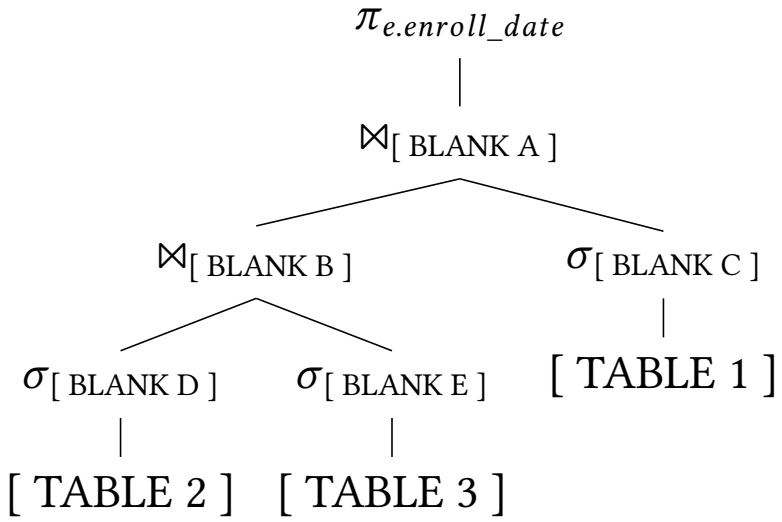
- Enrollments: 1 million rows
- Students: 500,000 rows, with only 50 students with a CS major
- Courses: 2000 rows.

The database query optimizer due to a bug generates the following highly inefficient logical query plan.



i. [1 pt] Briefly explain in 1-2 sentences why the following query plan is inefficient.

Solution: Late filtering and the resulting large intermediate data size from cross product/join



ii. The query optimizer is trying to fix the query plan. Consider the above template and fill in the blanks for an optimized query plan:

a) [1 pt] Blank A:

_____

Solution: e.course_id = c.id

b) [1 pt] Blank B:

_____

Solution: s.id = e.student_id

c) [1 pt] Blank C:

_____

Solution: c.title = 'ADS'

d) [1 pt] Blank D:

_____

Solution: s.major = 'CS'

e) [1 pt] Blank E:

_____

Solution: e.is_completed = TRUE

f) [1 pt] Table 1:

_____

Solution: courses

g) [1 pt] Table 2:

_____

Solution: students

h) [1 pt] Table 3:

_____

Solution: enrollments

Chapter 2: Mixing MapReduce, Graphs, and Samples [22 pt]

2.1. Let's create your "Spotify Wrapped" based on your listening history. For the following questions, assume each record is a tuple of the form `user_id, timestamp, song_id, artist_id, milliseconds_played`. We want to compute how many milliseconds a user has played per artist using one Mapreduce pipeline (aka one map and one reduce).

i. [1 pt] Suppose we choose `user_id` as the Map output key. As part of the Map output value, which of the following must be included.

- ☐ **A. `artist_id`**
- ☐ **B. `milliseconds_played`**
- ☐ C. `song_id`
- ☐ D. `user_id`

ii. [1 pt] What Reduce function best fits this strategy? Describe what it does in 1-2 sentences.

Solution: The reduce function receives a user_id and a list of (artist_id, milliseconds_played) values. It aggregates these by summing milliseconds_played for each artist_id, outputting the total milliseconds played per artist for that user.

iii. [1 pt] Suppose we choose `(user_id, artist_id)` as the Map Output Key. As part of the Map output value, which of the following must be included?

- ☐ A. `artist_id`
- ☐ **B. `milliseconds_played`**
- ☐ C. `song_id`
- ☐ D. `user_id`

iv. [1 pt] What Reduce function best fits this strategy? Describe what it does in 1-2 sentences.

Solution: The reducer function receives a list of milliseconds_played for each (user_id, artist_id) key, and it should sum them and output the total milliseconds_played per artist.

v. [2 pt] Suppose we have a power user with 100,000 records, across 10,000 different artists but other users have only listened to small number of songs and artists. Which of the above two strategies, one with key: `user_id` or the one with key: `(user_id, artist_id)` will have better performance? Justify your answer in 1-2 lines.

◯ `A.` Both are the same

◯ `B.` Map with key: `user_id` is better

◯ **`C.` Map with key: `(user_id, artist_id)` is better**

Solution: Using (user_id, artist_id) spreads the power user's 100,000 records across many reducer keys, avoiding a single reducer becoming a bottleneck. In contrast, using user_id alone sends all of that user's data to one reducer, causing load imbalance and slower performance.

vi. [2 pt] Now, suppose the power user's 100,000 sessions were just for one artist. Now, which of the above two strategies, one with key: `user_id` or the one with key: `(user_id, artist_id)` will have better performance? Justify your answer in 1-2 lines.

◯ **`A.` Both are the same**

◯ `B.` Map with key: `user_id` is better

◯ `C.` Map with key: `(user_id, artist_id)` is better

Solution: If the power user listens to only one artist, both key choices send all 100,000 records to a single reducer, so neither strategy improves parallelism or load balancing.

2.2. [3 pt] Now, let's assume a user ran the following SQL query.

```
SELECT user_id, COUNT(*)
FROM spotify_records
GROUP BY user_id;
```

The user wants to run the corresponding query in mapreduce. The `spotify_records` relation contains tuples of the format `user_id, timestamp, song_id, artist_id, milliseconds_played` as defined at the beginning of the section. Fill in the following blanks:

```
def map(_____, _____)

return { _____ : _____ }


def reduce(_____ , _____):

return {
    _____ : _____(_____)
}
```

Solution:

```
def map(user_id,song_id)
    return {user_id : 1}

def reduce (mapOutputKey, mapOutputValue):
    return {mapOutputKey: sum(mapOutputValue)}
```

2.3. [2 pt] Consider the following SQL query:

```
SELECT artist_id, SUM(milliseconds_played)
FROM Plays
WHERE timestamp >= '2024-01-01'
GROUP BY artist_id
ORDER BY SUM(milliseconds_played) DESC;
```

Which specific step in this query execution plan is a blocking operation that breaks the pipeline? Select all that apply.

☐ A. Reading the raw tuples from the disk partitions.

☐ `B.` Discarding rows where the timestamp is before 2024.

☐ `C.` Extracting the `artist_id` and `milliseconds_played` columns from the raw data.

☐ **`D.` Ordering the final list of artists from highest playing time to lowest.**

☐ **`E.` Grouping the artists.**

2.4. [2 pt] You are trying to analyze the social graph of Spotify; assume all nodes correspond to users and edges of type "follows" reference following behavior between users. Also, let the node variable `n` represent you.

You want to find users that you follow directly, but who are also followed by at least one other person that you follow. Which Cypher query pattern correctly identifies these users:

◯ A.
```
MATCH (n)-[:Follows] → (friend) -[:Follows]→ (other)
WHERE (n)-[:Follows]→(another)
RETURN other
```

◯ B.
```
MATCH (n)-[:Follows]→(friend)
MATCH (friend)-[:Follows]→(other)
WHERE (n)-[:Follows]→ (other)
RETURN other
```

◯ **C.**
```
MATCH (n)-[:Follows]→(friend)-[:Follows]→(other)
WHERE (n)-[:Follows]→(other)
RETURN other
```

◯ D.
```
MATCH (n)-[:Follows]→(friend)-[:Follows]→(other)-[:Follows]→(n)
RETURN other
```

2.5. [5 pt] Assume a relational database with table, `Follows(follower_id, followee_id)` to represent a social graph. We want to find all the follow chains starting from user id "007" that are up to 3 hops long. Complete the recursive SQL query below.

```
WITH RECURSIVE FollowPaths(start_node, end_node, depth) AS (
    SELECT _____
    _____
    FROM Follows
    _____
    _____
    UNION ALL
    SELECT _____
    _____
    FROM FollowsPath p
    JOIN _____
    _____
    WHERE _____)
SELECT * FROM FollowPaths;
```

Solution:

```
WITH RECURSIVE FollowPaths(start_node, end_node, depth) AS (
    SELECT
        follower_id,
        followee_id,
        1
    FROM Follows
    WHERE follower_id = '007'

    UNION ALL

    SELECT
        p.start_node,
        f.followee_id,
        p.depth + 1
```

```
    FROM FollowPaths p
    JOIN Follows f
        ON p.end_node = f.follower_id
    WHERE p.depth < 3
)
SELECT * FROM FollowPaths;
```

2.6. Spotify is building a personalized "Discovery" playlist of 3 songs for you using reservoir sampling. The algorithm considers songs from your listening history in chronological order. The listening history is: "Blinding Lights", "As It Was", "Good 4 U", "Anti-Hero", "Levitating", "Stay", "Heat Waves", "Shivers", "Peaches", "Industry Baby", "Drivers License", "Montero".

i. [1 pt] The reservoir (your playlist, with k=3) currently holds "Blinding Lights", "As It Was", "Good 4 U". When the next song ("Anti-Hero") is considered, what is the probability that it replaces "Good 4 U"?

◯ **1/4** ◯ 1/3 ◯ 3/4 ◯ 1/12

ii. [1 pt] When the fifth song in the history ("Levitating") is considered, what is the probability that the playlist is still exactly "Blinding Lights", "As It Was", "Good 4 U" after the algorithm processes it?

◯ 3/5 ◯ 1/5 ◯ **1/10** ◯ 2/5

Chapter 3: The Entity-Relationship Ecosystem [23 pt]

To highlight its incredible nature, the state of California is gathering information about its public parks. The `park_visits` table has been created to store information about every visit to a California park within the past month. A sample of data from the `park_visits` table is shown below.

visitor_id	visitor_name	visit_date	park_id	park_name	park_size_acres	managing_org	operation_level
2	Sahil	2025-11-07	005	Yosemite	748,542	National Parks Service	federal
1	Joshua	2025-11-02	109	Tilden	740	East Bay Regional Parks	regional
2	Sahil	2025-11-25	732	Bushrod	10	City of Oakland	local
4	Jiajun	2025-11-16	649	Willard	3	City of Berkeley	local
5	Thanh	2025-11-16	649	Willard	3	City of Berkeley	local
3	Sofia	2025-11-07	005	Yosemite	748,542	National Parks Service	federal

3.1. [2 pt] What functional dependencies defenitely *do not* hold based on the portion of the `park_visits` table shown above. Select all that apply.

- ☐ A. park_id → park_size_acres
- ☐ **B. operation_level → visit_date**
- ☐ **C. (visit_date, park_id) → visitor_name**
- ☐ D. managing_org → operation_level
- ☐ **E. park_name → visitor_name**
- ☐ **F. visitor_id → (park_name, managing_org)**

3.2. [4 pt] The data engineers in the state of California want to perform normalization on `park_visits` to reduce data redundancy. What columns should be in each of the tables below? Select all that apply.

i. `visitors`
- ☐ **A. visitor_id**
- ☐ **B. visitor_name**
- ☐ C. visit_date
- ☐ D. park_id
- ☐ E. park_name
- ☐ F. park_size_acres
- ☐ G. managing_org
- ☐ H. operation_level

ii. `parks`
- ☐ A. visitor_id
- ☐ B. visitor_name
- ☐ C. visit_date
- ☐ **D. park_id**
- ☐ **E. park_name**
- ☐ **F. park_size_acres**
- ☐ **G. managing_org**
- ☐ H. operation_level

iii. `organizations`

- ☐ `A.` visitor_id
- ☐ `B.` visitor_name
- ☐ `C.` visit_date
- ☐ `D.` park_id
- ☐ `E.` park_name
- ☐ `F.` park_size_acres
- ☐ **`G.` managing_org**
- ☐ **`H.` operation_level**

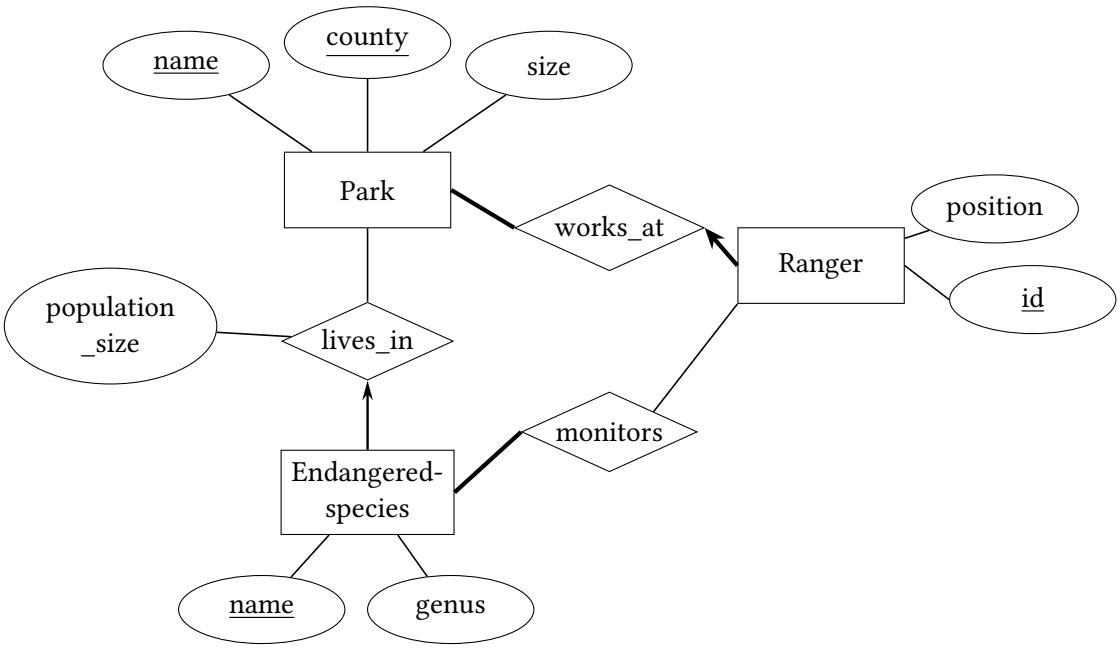
iv. `visits`

- ☐ **`A.` visitor_id**
- ☐ `B.` visitor_name
- ☐ **`C.` visit_date**
- ☐ **`D.` park_id**
- ☐ `E.` park_name
- ☐ `F.` park_size_acres
- ☐ `G.` managing_org
- ☐ `H.` operation_level

3.3. [2 pt] The data engineers are considering converting the `park_visits` relation into a JSON to accommodate a more flexible schema. They decide to add the attributes of the orgs into the parks table, but they want to nest the rest of the relations to maintain foreign key relationships between parks, visits, and visitors. Which of the following JSON nesting orders are valid ways of representing the `park_visits` while preserving all foreign key relationships.

- ☐ **`A.` parks → visits → visitors**
- ☐ `B.` visits → parks → visitors
- ☐ **`C.` visitors → visits → parks**
- ☐ `D.` visitors → parks → visits
- ☐ `E.` parks → visitors → visits

3.4. [2 pt] The California parks organizations have created the following entity relationship diagram to model the relationships between the parks, the rangers who work at the parks, and the endangered species that live in the parks.



Identify whether the following statements are implied by the ER diagram above. Select all that apply.

☐ A. Every ranger monitors an endangered species

☐ B. Two parks cannot have the same name

☐ **C. A ranger must work at only one park**

☐ **D. An endangered species cannot live in two different parks**

3.5. [3 pt] Now the California parks organizations want to translate their ER diagram into a relational schema. During the relational design phase, the team introduces a new attribute, `park_id`, to serve as a unique identifier for each park. They propose the following relations:

```
parks(park_id, name, county, size)
endangered_species(name, genus)
rangers(id, position)
lives_in(park_id, species_name, population_size)
monitors(species_name, ranger_id)
works_at(park_id, ranger_id)
```

They are not entirely satisfied with this result. If possible, they want to reduce the number of relations to save space while still avoiding data redundancy. In the same format as the proposed schema above, write a relational schema with the minimum number of relations that still maintains normalization. Please refer to the ER diagram in Question 4 for constraints on the relationships between entities. You may not need all provided lines for your answer.

Solution:

```
parks(park_id, name, county, size)
endangered_species(name, genus, park_id, population_size)
rangers(id, position, park_id)
monitors(species_name, ranger_id)
```

Alternative answers are possible that reduce the schema to the minimum number of relations (4) while maintaining normalization.

3.6. [5 pt] The state of California also cares about its restaurants. Data about California restaurants is recorded in the MongoDB collection `db.restaurants`. An example document from restaurants is shown below.

```
{
  '_id': ObjectId('692220383e31a82fa00d7825'),
  'name': 'Mezzo',
  'address': '2442 Telegraph Ave',
  'city': 'Berkeley',
  'stars': 4.2,
  'cuisine': 'American',
  'takeout': True,
  'attributes': {
    'BikeParking': True,
    'GoodForGroups': True,
    'CompostableTogo': True
  },
  'info': [
    {'food': ['Sandwiches', 'Salad', 'Coffee & Tea']},
    {'recommendations': ['Park Approved']}
  ]
}
```

For each type of cuisine, find the restaurant that has the most stars and also offers take out (i.e. the **'takeout'** field is True). In the case that two restaurants serving the same cuisine have equal stars, break ties by sorting restaurant name in alphabetical order.

The documents returned by this query should have the following structure:

```
[
  { '_id': 'Thai', 'top_restaurant': 'Neecha Thai', 'stars': 5 },
  { '_id': 'Ethiopian', 'top_restaurant': 'Cafe Colucci', 'stars': 4.9 },
  { '_id': 'Japanese', 'top_restaurant': 'Otafuku', 'stars': 4.9 },
  ...
]
```

Complete the query by filling in the blanks. You may not need all provided lines, and include necessary syntax (ex: , "") if needed. Both MongoDB and PyMongo syntax are acceptable.

```
db.restaurants.aggregate([ ____________________________________

$sort: { ____________________________________________________},

$group: { ____________________________________________________

  ____________________________________________________

  ____________________________________________________

  ____________________________________________________

}
])
```

Solution:

```
db.restaurants.aggregate([
  {$match: {takeout: True }},
  {$sort: {stars: -1, name: 1}},
  {$group: {
     _id: "$cuisine",
     top_restaurant: {$first: "$name"},
     stars: {$first: "$stars"}
  }}
])
```

3.7. [5 pt] California parks organizations are highlighting sustainable restaurants by adding 'Park Approved' to the recommendations list within the info field. A restaurant should only be 'Park Approved' if it uses compostable to-go containers (i.e., the 'CompostableTogo' attribute exists and is True). The organizations are worried they may have approved restaurants that do not meet this criteria. Write a query to check if there are any restaurants that have been marked as 'Park Approved' but do not offer Compostable to-go.

If documents exist that satisfy this condition, the documents returned by this query should have the following structure:

```
[
  { 'name': 'Share Tea', 'info.recommendations': 'Park Approved',
   ↪  'CompostableTogo': False },
  { 'name': 'Corso', 'info.recommendations': 'Park Approved' },
  { 'name': 'Roast & Toast', 'info.recommendations': 'Park Approved',
   ↪  'CompostableTogo': False }
]
```

Note: The 'takeout' field exists for every document in restaurants, but the 'Compostable-Togo' field only exists for documents where 'takeout' is True.

Fill in the blanks below. You may not need all of the lines provided, and include necessary syntax (ex: , "") if needed. Both MongoDB and PyMongo syntax are acceptable.

```
db.restaurants.aggregate([

{ _____},

$match: {

 _____

 _____
},

$project: {

 _____

 _____

 _____}
])
```

Solution:

```
db.restaurants.aggregate([
    {$unwind: info},
    {$match:
        {info.recommendations: 'Park Approved',
        attributes.CompostableTogo : {$ne, 'True'}}
    },
    {$project:
        {_id: 0, name: 1, info.recommendations: 1,
        attributes.CompostableTogo: 1}
    }
])
```

Chapter 4: The Data Preparation Odyssey [23 pt]

4.1. A data scientist begins working with the following dataset:

Initial data = $[10, 15, 17, 60, 100]$

They create two new datasets, Result A and Result B, by applying two different outlier handling strategies.

Result A = $[10, 15, 17, 60, 60]$
Result B = $[10, 15, 17]$

i. [2 pt] Identify which process could have led to Result A. Select all that apply.

- ☐ A. Data was winsorized at the 20th and 80th percentile.
- ☐ **B. Values more than 2 standard deviations from the mean are replaced by the next lowest value.**
- ☐ C. Values were clipped at the interquartile range.

ii. [2 pt] Identify which process could have led to Result B. Select all that apply.

- ☐ A. Outliers were identified as any value more than 3 standard deviations from the mean and then dropped.
- ☐ **B. Outliers were identified using Hampel x84 with $K = 3$ and then dropped.**
- ☐ C. The data was trimmed by dropping all values below the 20th percentile and above the 80th percentile.

iii. [1 pt] Which of the following metrics gets changed most dramatically after transformation to Result A?

◯ Mean ◯ Median ◯ **Standard Deviation** ◯ Interquartile Range

4.2. The data scientist is now working with a time-ordered dataset of sensor readings that contains some missing values (NaN).

Index	Value
1	10
2	20
3	NaN
4	NaN
5	50

Three different imputation methods are used to fill the missing data, creating three new versions of the value column.

Result A = $[10, 20, 26.7, 26.7, 50]$
Result B = $[10, 20, 30, 40, 50]$
Result C = $[10, 20, 20, 20, 50]$

The four possible imputation processes are listed below:

A. Each NaN is replaced by the last observed non NaN value.

B. Each NaN is replaced by the mean of the available, non-missing values.

C. Linear Interpolation: Missing values are filled based on a straight line connecting the preceding and succeeding non NaN values.

D. A pre-trained model, with value = 5*index + 5 is used fill the NaN values.

i. [1 pt] Result A was created by:

◯ A. A

◯ **B. B**

◯ C. C

◯ D. D

ii. [1 pt] Result B was created by:

◯ A. A

◯ B. B

◯ **C. C**

◯ D. D

iii. [1 pt] Result C was created by:

◯ **A. A**

◯ B. B

◯ C. C

◯ D. D

4.3. You are tasked with finding matching author entities in a small database. You know from a ground-truth source that the following pairs represent the same entity: "Data", "DataC", "Bear", "Beears" , "Cal", "Berkeley". You decide to use q-gram blocking to generate candidate pairs for comparison.

i. [2 pt] For Blocking with 2-grams ($q = 2$), which of the specific pairs below are generated as candidates? (select all that apply)

☐ **A. "Data" and "Datac"**

☐ **B. "Bear" and "Beears"**

☐ **C. "Bear" and "Berkeley"**

☐ D. "Cal" and "Berkeley"

ii. [2 pt] For Blocking with 4-grams ($q = 4$), Which of the specific pairs below are generated as candidates? (select all that apply)

☐ **A. "Data" and "Datac"**

☐ B. "Bear" and "Beears"

☐ C. "Bear" and "Berkeley"

☐ D. "Cal" and "Berkeley"

iii. [1 pt] Which statement describes the effect of increasing q from 2 to 4?

◯ **A. Precision increased, but Recall decreased.**

◯ B. Recall increased, but Precision decreased.

◯ C. Both Precision and Recall increased.

◯ D. No Change

4.4. The data scientist is now working with data from two sensors ('A' and 'B'). The sensors report readings every 15 minutes, but the data is imperfect. The data is stored in a table named `sensor_readings`. It contains both an outlier (denoted by *) and a gap (denoted by †) in the readings.

Sensor_id	Timestamp	Reading
A	2025-09-14 10:00:00	20
A	2025-09-14 10:15:00	500*
A	2025-09-14 10:30:00	25
B	2025-09-14 14:00:00	75
B	2025-09-14 14:30:00	85†

i. [5 pt] First, you must handle the outliers. You decide to use winsorization at the 25th and 75th percentiles. Write a SQL query that returns all columns from the original table, plus an additional new column named `cleaned_reading` which contains the winsorized values.

```
WITH SensorBounds AS (
    SELECT sensor_id, reading,
  _____
  _____
  _____
   FROM sensor_readings )
SELECT sensor_id,  timestamp, reading,
CASE
  _____
  _____
  _____
  _____
END AS cleaned_reading
FROM sensor_readings,SensorBounds;
```

Solution:

For SensorBounds:
PERCENTILE_DISC(0.25) AS lower_bound
PERCENTILE_DISC(0.75) AS upper_bound

For cleaned_reading:
WHEN reading < lower_bound THEN lower_bound
WHEN reading > upper_bound THEN upper_bound
ELSE reading

Note that PERCENTILE_CONT can be used instead of PERCENTILE_DISC, and any column names for upper_bound and lower_bound can be used as long as they are consistent.

ii. [5 pt] Assume your query in previous part returned the table named `clean_sensor_data`. Your task is to monitor if the sensor readings are drifting. Use the `cleaned_reading` column from the previous part. The drift, called as `drift_values` in the output table is calculated as:

`(running_average_so_far) - (overall_average_for_that_sensor)`

```
SELECT sensor_id, timestamp, cleaned_reading,

  _____

  _____

  _____

  _____

  _____

  _____

AS drift_values

FROM clean_sensor_data;
```

Solution:

```
AVG(cleaned_reading) OVER (
    PARTITION BY sensor_id
    ORDER BY timestamp
) - AVG(cleaned_reading) OVER (
    PARTITION BY sensor_id)
```

Chapter 5: Ultimate Data Systems Potpourri Endgame [26 pt]

5.1. [1 pt] For the following schedule, write a possible valid serialization (i.e., an equivalent serial schedule), if any. If no serialization exists, then write "None". If a serialization exists, write it as a permutation of the three transactions separated by commas (e.g., T1, T2, T3)

T1	R(A)				R(B)	W(B)
T2		R(A)	W(A)			
T3				W(B)		

Solution: T3, T1, T2

5.2. Consider the following non-serializable schedule.

T1	R(A)			R(B)		W(C)		
T2		R(B)			W(A)			W(C)
T3			R(C)				W(C)	

i. [2 pt] Select all the edges present in corresponding conflict graph.

- ☐ **A. T1 → T2**
- ☐ **B. T1 → T3**
- ☐ C. T2 → T1
- ☐ D. T2 → T3
- ☐ **E. T3 → T1**
- ☐ **F. T3 → T2**

ii. [2 pt] If you had the ability to remove one action to make the schedule given above serializable, which of the following would you remove?

- ◯ A. T2: R(B)
- ◯ B. T1: R(B)
- ◯ C. T2: W(C)
- ◯ **D. T1: W(C)**

5.3. The DATA 101 course staff is collaboratively building a "Grading Marathon" playlist for grading final exams. The songs are stored in the following table:

```
CREATE TABLE playlist (
    position INTEGER PRIMARY KEY,
    song_title TEXT);
```

The staff are all editing the playlist at the same time. Pranav runs a transaction; he wants to move his favorite song to the top of the list.

```
BEGIN TRANSACTION;
SELECT * FROM playlist;
UPDATE playlist SET position = 1 WHERE song_title = 'Blinding Lights';
END TRANSACTION;
```

Josh has a lot of recommendations and runs a transaction to bulk-add three new songs.

```
BEGIN TRANSACTION;
INSERT INTO playlist VALUES (1, 'The Hills');
INSERT INTO playlist VALUES (2, 'Starboy');
INSERT INTO playlist VALUES (3, 'After Hours');
SELECT * FROM playlist;
END TRANSACTION;
```

Due to an implementation error, the database does not guarantee **ACID** properties!

i. [2 pt] Both transactions complete successfully. However, Pranav reports that he sees rows for “Blinding Lights", “The Hills”, and "Starboy", but not "After Hours". Which property(ies) are violated? Select all that apply.

- ☐ **A. Atomicity**
- ☐ B. Consistency
- ☐ **C. Isolation**
- ☐ D. Durability
- ☐ E. None of the above

ii. [2 pt] Josh reports that he see the rows (1, "Blinding Lights"), (1, "The Hills"), (2, ’Starboy’), (3, "After Hours"). Which property(ies) are violated? Select all that apply.

- ☐ A. Atomicity
- ☐ **B. Consistency**
- ☐ **C. Isolation**
- ☐ D. Durability
- ☐ E. None of the above

5.4. [2 pt] Which of the following statements correctly distinguish Workflow Systems (e.g., Airflow) from Dataflow Systems (e.g., Spark, dbt)? Select all that apply.

- ☐ **A. Workflow systems are primarily designed to manage dependencies between heterogeneous tasks (processes), which may include API calls or bash scripts.**
- ☐ B. Workflow systems generally do not support "human-in-the-loop" steps or approvals.
- ☐ **C. Dataflow systems are specialized to manage the movement and transformation of data through operations, handling data dependencies like inputs and outputs.**
- ☐ **D. Workflow systems facilitate failure recovery by tracking the state of individual tasks, allowing for retries without restarting the entire process.**

5.5. [2 pt] Which of the following is true of DBT? Select all that apply:

- ☐ **A. DBT uses a mix of SQL + Jinja to create dataflow graphs.**
- ☐ **B. DBT can allow users to implement incremental maintenance of views if the underlying databases don't support it.**
- ☐ **C. DBT allows users to add tests to be run as part of a pipeline.**
- ☐ D. DBT uses its own internal compute engine to execute transformations, extracting data from the warehouse to process it.

5.6. You are the lead data engineer for a retail analytics platform. You have designed a 3-dimensional data cube to track sales performance. The cube dimensions are:

```
Product (Hierarchy: Category → SubCategory → ItemName)
Location (Hierarchy: Country → State → City)
Time (Hierarchy: Year → Quarter → Month → Day)
```

The measure being tracked in each cell is `Total_Sales`.

i. [1 pt] If you want to view the `Total_Sales` per product per location, which OLAP operation are you effectively performing on the cube?

- ◯ A. Roll-up
- ◯ B. Drill-down
- ◯ C. Slice
- ◯ D. Dice
- ◯ **E. Pivot**

ii. [1 pt] Currently, your cube is aggregated at the Quarter level for the Time dimension. A business analyst asks to see sales data at the Month level. This operation is impossible with the current pre-computed cube because the data was stored at too coarse a granularity. To support this request in the future, which operation you must perform on the Time dimension during the cube design phase to store data at a finer granularity.

Solution: Drill-down

iii. [2 pt] Your cube currently contains the following counts at the levels being analyzed:

- Product: 3 Categories
- Location: 4 States
- Time: 2 Years

If an analyst performs a slice on the Product dimension selecting only the "Electronics" Category, how many cube cells remain? Just provide the numeric value.

Solution: Only 1 Product category remains. $1 \times 4 \times 2 = 8$ cells

iv. [2 pt] You are deciding how to physically store this data cube. You are considering MOLAP (Multidimensional OLAP) versus ROLAP (Relational OLAP). Which of the following statements accurately describe the trade-offs? Select all that apply.

- ☐ **A. MOLAP stores data in a multidimensional array, which is optimized for fast retrieval of aggregates but can be inefficient for sparse data.**
- ☐ **B. ROLAP stores data in a relational star or snowflake schema, requiring joins to aggregate data at runtime.**
- ☐ C. MOLAP typically requires no pre-computation and calculates all slices on-the-fly.
- ☐ **D. ROLAP generally scales better for data with very high cardinality dimensions compared to MOLAP**

5.7. Consider a spreadsheet which has the following two tabular regions:

	A	B	C	D
1	Order_ID	Zone	Weight	Shipping_Cost
2	1001	Zone 2	Medium	
3	1002	Zone 3	Light	
4	1003	Zone 1	Heavy	

Table 1: Orders (Cells A-C)

	F	G	H	I
1	Class/Zone	Zone 1	Zone 2	Zone 3
2	Light	$10	$12	$15
3	Medium	$20	$25	$30
4	Heavy	$50	$60	$80

Table 2: Rate_Matrix (The Lookup Table) (Cells F-I)

i. Write a single formula in cell `D2` that calculates the shipping cost. Fill in the blanks to complete the formula. For reference, here is the syntax for

```
XLOOKUP:XLOOKUP(search_key, lookup_range, result_range)
```

```
= XLOOKUP(  _____ A_____,  $F$2:$F$4 ,  _____ B_____  ( B2 ,
_____ C_____, $G$2:$I$4 ) )
```

a) [1 pt] Blank A

_____

b) [1 pt] Blank B

_____

c) [1 pt] Blank C

_____

Solution: C2 (We first look up the "Medium" weight class)

XLOOKUP (We start the nested function to handle the column/zone match)

$G$1:$I$1 (We look for "Zone 2" inside the header row)

ii. [2 pt] We are computing a weighted count of orders so far, defined as : count of order × shipping cost. Cell E2 contains the following formula:

```
= COUNT($A$2:A2) * D2
```

You drag the formula down to `E4`. Fill in the formula which would appear in cell `E4`:

= _____

Solution: COUNT($A$2:A4) * D4

iii. [2 pt] You enter the following formula in cell `G5`:

```
= MAX($G4:G4) + G4
```

You copy the formula to cell `I6`. Fill in the formula which would appear in cell `I6`:

= _____

Solution: MAX($G5:I5) + I5

Chapter 6: 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 $\leq$ 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!