Name

UIN

Homework Assignment 2

Due Date: March 14, 2026

CS480 - Database Systems

Please leave this empty!

2.1 2.2

2.3 2.4 2.5 2.6 2.7 2.8 2.9 2.10

2.15 2.16 2.17 2.18 2.19 Sum

Instructions

- Try to answer all the questions using what you have learned in class
- **When writing a query, write the query in a way that it would work over all possible database instances and not just for the given example instance!**
- Some questions are marked as bonus. You do not have to answer these questions to get full points for the assignment. However, you can get bonus points for these questions!
- **Please submit the homework electronically using gradescope**
- **Submissions will be automatically graded. You can submit multiple times.**
- **The due date is 03/14!**

Consider the following database schema and example instance storing information about tourism in different countries:

Country

country_id	name	continent	population_m	gdp_usd_bn
IT	Italy	Europe	59.0	2100
FR	France	Europe	68.0	3000
JP	Japan	Asia	125.0	4200
EG	Egypt	Africa	110.0	400
US	United States	North America	333.0	27000

City

city_id	country_id	name	latitude	longitude	population_m
C1	IT	Rome	41.9028	12.4964	2.9
C2	FR	Paris	48.8566	2.3522	2.1
C3	JP	Kyoto	35.0116	135.7681	1.5
C4	EG	Giza	29.9773	31.1325	4.0
C5	US	New York	40.7128	-74.0060	8.8

FamousSite

site_id	city_id	name	category	unesco	year_est.	latitude	longitude
S1	C1	Colosseum	historical	T	80	41.8902	12.4922
S2	C2	Eiffel Tower	landmark	F	1889	48.8584	2.2945
S3	C3	Kiyomizu-dera	religious	T	778	34.9949	135.7850
S4	C4	Great Pyramid	historical	T	-2560	29.9792	31.1342
S5	C5	Statue of Liberty	landmark	F	1886	40.6892	-74.0445

Traveler

traveler_id	name	home_country_id	birth_date	segment
T1	Alex Kim	US	1998-05-12	student
T2	Camille Dubois	FR	1985-03-21	family
T3	Haru Sato	JP	2001-11-08	solo
T4	Noor Hassan	EG	1992-07-30	business

SiteVisit

trip_id	site_id	visit_date	minutes_spent	group_size	rating
TR1	S1	2025-06-03	180	1	5
TR1	S2	2025-06-07	120	2	4
TR2	S2	2025-07-18	150	4	5
TR3	S3	2025-08-08	200	1	5
TR4	S5	2025-05-12	90	1	4
TR4	S3	2025-07-08	400	1	5
TR4	S3	2025-05-18	100	2	5
TR1	S1	2025-12-08	300	1	5
TR1	S3	2025-07-08	30	1	5
TR1	S4	2025-07-08	700	1	5
TR1	S1	2025-06-15	150	1	5

Trip

trip_id	traveler_id	start_date	end_date	purpose	budget_usd
TR1	T1	2025-06-01	2025-06-10	leisure	1800
TR2	T1	2025-07-15	2025-07-22	family visit	2400
TR3	T3	2025-08-05	2025-08-14	sightseeing	1600
TR4	T4	2025-05-10	2025-05-16	conference	2100

Transaction

transaction_id	trip_id	site_id	date	type	amount_usd	payment
TX1	TR1	S1	2025-06-03	ticket	25.00	card
TX2	TR1	S2	2025-06-07	souvenir	30.00	card
TX3	TR2	S2	2025-07-18	ticket	45.00	cash
TX4	TR3	S3	2025-08-08	food	20.00	card
TX5	TR4	S5	2025-05-12	guided tour	50.00	card
TX6	TR2	S5	2025-05-12	guided tour	10.00	card
TX7	TR1	S5	2025-05-12	guided tour	20.00	card

Hints:

- If you are unsure about the type of an attribute look at the SQL script for the homework. The same applied to foreign key constraints which are defined the DDL script for the homework.
- Attributes with black background form the primary key of an relation.

Part 2.1 SQL DDL (Total: 12 Points)

Question 2.1.1 (6 Points)

Write an SQL statement that adds a column named *month* to relation *Trip*. The value should be the month extracted from start_date. Ensure that the month is between 1 and 12. Backfill the data for the existing instances. The default value should be 0.

Question 2.1.2 (6 Points)

Write an SQL statement that creates a new table *TravelerPreference* recording each traveler's preferences for site categories. For each record store: *traveler_id*, *category*, and *interest_level* (an integer from 1 to 5). A traveler may have preferences for multiple categories, but for each traveler-category pair there is at most one record.

After creating the table insert the following data (you need it to test queries). You can find the SQL code commented out in the SQL script for the homework.

For the autograder submission, we will insert data, just submit the `CREATE TABLE` statement.

```
INSERT INTO TravelerPreference(traveler_id, category, interest_level) VALUES
('T1', 'historical', 5),
('T1', 'landmark', 4),
('T2', 'landmark', 5),
('T3', 'religious', 5),
('T4', 'historical', 4);
```

Part 2.2 SQL Queries (Total: 66 + 15 BONUS Points)

Question 2.2.1 (5 Points)

Write an SQL query that returns the ids and names of countries whose GDP-to-population ratio is less than 35 and whose continent does not contain the string "America". The result schema should be `(country_id, name)`.

Question 2.2.2 (5 Points)

Write an SQL query that returns for each famous site the number of distinct travelers that have visited it. This query should return every famous site, including sites that have not been visited. Order the result by traveler count in descending order. The result schema should be `(site_id, name, numtravelers)`.

Question 2.2.3 (7 Points)

Write an SQL query that returns the UNESCO-protected famous sites with the highest traveler footprint and highest revenue. Traveler footprint is the number of distinct travelers that visited the site. Revenue is the sum of all transaction amounts for that site. Sort the results by traveler footprint (descending) and then revenue (descending). Return the top 3 with this sort order. Multiple rows can be returned in case of ties. The result schema should be `(site_id, name, travelerfootprint, revenue)`.

Question 2.2.4 (7 Points)

Write an SQL query that returns for each city the best season to visit, defined as the season with the highest number of visits to famous sites located in that city. Use the visit date from SiteVisit. Seasons are: Winter (Dec–Feb), Spring (Mar–May), Summer (Jun–Aug), Fall (Sep–Nov). Return one row per city. The result schema should be `(city_id, cityname, season, numvisits)` .

Question 2.2.5 (10 Points)

Write an SQL query that returns the pair(s) of distinct cities that are closest to each other based on latitude/longitude (Haversine Formula). We define closest pairs as the ones within 3000 kilometers of each other. Among the closest city pairs, return the pair with the highest combined revenue, where city revenue is the sum of transaction amounts for all sites located in that city. The result schema should be `(city_name1, city_name2, distance_km, combinedrevenue)`.

Question 2.2.6 (8 Points)

Write an SQL query that returns the travelers that have visited every UNESCO-protected famous site. The result schema should be `(traveler_id, name)`.

Question 2.2.7 (8 Points)

For each country, define each traveler's spend in that country as the total transaction amount for transactions at sites located in that country, across all of the traveler's trips.

Write an SQL query that returns, for each country:

- total_revenue: total revenue in the country,
- top2_share: fraction of total_revenue contributed by the top 2 highest-spending travelers in that country,
- top1_share: fraction contributed by the top 1 highest-spending traveler in that country.

The result schema should be: `(country_id, total_revenue, top2_share, top1_share)`.

Question 2.2.8 (8 Points)

A traveler is said to have covered a country if they have visited every city in that country that contains at least one famous site, and additionally, for each such city, they have visited at least one site in that city with *minutes_spent* >= 60.

Write an SQL query that returns the travelers that have covered every country in Europe. The result schema should be `(traveler_id, name)`.

Question 2.2.9 (8 Points)

Write an SQL query that returns, for each famous site, total revenue at the site, revenue paid by card and by cash, compute the fraction of revenue coming from each payment method (card share and cash share). Additionally, compute the site's most common transaction type (by total_revenue for that type). Order by total_revenue in descending order. Return both rows if there are ties.

The result schema should be: `(site_id, total_revenue, card_revenue, cash_revenue, card_share, cash_share, top_payment_type)`.

Part 2.3 SQL Updates (Total: 22 Points)

Question 2.3.1 (5 Points)

Increase all transaction amounts by 10% for transactions at sites located in countries whose GDP is greater than 3000 (in billions). Round the amount to 2 decimal places.

Question 2.3.2 (5 Points)

Delete all travelers that have not taken any trip.

Question 2.3.3 (6 Points)

For each country, delete the most expensive transaction (by amount_usd) among transactions that occurred at sites in that country. If multiple transactions tie for the maximum amount within a country, delete all such transactions.

Do not delete anything for countries with fewer than 2 transactions.

Question 2.3.4 (6 Points)

Add a boolean column `is_top_site` to *FamousSite* (default `false`). Set `is_top_site` = `true` for the top 2 sites in each country by total revenue (sum of `Transaction.amount_usd`). Set it to `false` for all other sites.

Part 2.4 SQL Bonus (Total: 15 Points)

Question 2.4.1 BONUS (15 Points)

Write an SQL query which computes all legal moves for each player in a game of Go. If you are unfamiliar with the rules of Go you can find an explanation here: https://en.wikipedia.org/wiki/Go_(game). The query should return all legal moves for both players given a board state. The input of the query is a relation go_board with schema (x, y, color) where x and y are integers representing the position on the board (1 to 5), and color is either 'B' (Black) or 'W' (White). Cells not present in the relation are empty. The result schema should be `(player, x, y)` representing all legal placements for each player. A move is legal if:

- The target cell is empty
- Placing a stone there does not result in self-capture i.e., the placed stone's connected group has at least one liberty remaining after placement

A liberty is an empty cell directly adjacent (up, down, left, right) to a stone or group of stones of the same color. You may ignore the ko rule.

- No procedural SQL is allowed.
- **This question is quite challenging. Probably solve all other questions first before attempting to solve this one!**