I have created the database 'Project\_info' in *MySQL Workbench* -> Query Tab and visualized in Schemas and Tables in the Navigation bar.

CREATE DATABASE Project\_info;

USE Project\_info;

According to the problem statement, I implemented a Schema that must have interlinked tables. Therefore, we will need primary keys and foreign keys referenced to the linked table.

I have created 3 tables:

* Projects
* Tasks
* Teams

**Table\_name:** Projects

**Column\_name:** (1) project\_id (2) project\_name (3) budget (4) start\_date (5) team\_id

CREATE TABLE Projects (

project\_id INT PRIMARY KEY,

project\_name VARCHAR(255),

budget DECIMAL(10,2),

start\_date DATE,

team\_id INT UNIQUE

);

**Table\_name:** Tasks

**Column\_name:** task\_id, task\_name, member\_name, due\_date, task\_completed, project\_id (Projects table)

Projects.project\_id is the PRIMARY KEY and will act as a FOREIGN KEY in the Tasks table.

CREATE TABLE Tasks (

task\_id INT PRIMARY KEY,

task\_name VARCHAR(255),

member\_name VARCHAR(100),

due\_date DATE,

task\_completed BOOLEAN,

project\_id INT,

FOREIGN KEY (project\_id) REFERENCES Projects(project\_id)

);

**Table\_name:** Teams

**Column\_name:** member\_id, member\_name, team\_id, role. Projects.team\_id is UNIQUE and will act as the FOREIGN KEY in Teams table

CREATE TABLE Teams (

member\_id INT,

member\_name VARCHAR(100),

team\_id INT PRIMARY KEY,

role VARCHAR(100),

FOREIGN KEY (team\_id) REFERENCES Projects(team\_id)

);

I populated all the three tables with values with the following MySQL command, keeping in mind that there must not be a record in the Tasks table for one of the Projects entries.

INSERT INTO table\_name (column1, column2, column3, ...)

VALUES (value1, value2, value3, ...);

INSERT INTO Projects (project\_id, project\_name, budget, start\_date, team\_id) VALUES

(1, 'AI Chatbot', 50000.00, '2023-10-01', 101),

(2, 'Data Science', 30000.00, '2023-11-15', 102),

(3, 'Web Design', 45000.00, '2023-12-05', 103),

(4, 'Mobile App', 60000.00, '2024-01-10', 104),

(5, 'Market Research', 25000.00, '2024-02-20', 105),

(6, 'Inventory Analysis', 40000.00, '2024-03-15', 106);

INSERT INTO Teams (member\_id, member\_name, role, team\_id) VALUES

(1, 'Swati Mishra', 'Team Lead', 105),

(2, 'Rahul Pandey', 'Developer', 101),

(3, 'Kritika Awasthi', 'Team Lead', 102),

(4, 'Neha Shukla', 'Analyst', 103),

(5, 'Dimpal Tripathi', 'Team Lead', 104),

(6, 'Rahul Mishra', NULL, 106);

INSERT INTO Tasks (task\_id, task\_name, member\_name, due\_date, task\_completed, project\_id) VALUES

(201, 'UI Design', 'Swati Mishra', '2023-11-01', TRUE, 1),

(202, 'Backend Setup', 'Rahul Mishra', '2023-11-10', FALSE, 1),

(203, 'Data Cleanup', 'Kritika Awasthi', '2023-12-01', TRUE, 2),

(204, 'User Testing', 'Neha Shukla', '2024-01-05', FALSE, 3),

(205, 'App Deployment', 'Rahul Pandey', '2024-02-01', TRUE, 4),

(206, 'Inventory Review', NULL, '2024-02-15', NULL, 5);

**Answer #1:** I created a Common Table Expression (CTE) with the name 'ProjectsTasks' where I have GROUP BY project\_id to COUNT the number of tasks and SUM(IF( task\_completed = TRUE)) FROM the 'Tasks' table and later output the 'ProjectsTasks' CTE.

WITH ProjectsTasks AS (

SELECT

project\_id,

COUNT(\*) AS total\_tasks\_by\_project,

SUM(task\_completed = TRUE) AS total\_completed\_tasks\_by\_project

FROM Tasks

GROUP BY project\_id

)

SELECT \* FROM ProjectsTasks;

**Answer #2:** I created a CTE with the name 'RankedMember' as CTE makes the code clean, readable and understandable where I have used ROW\_NUMBER function which is a scaler function to identify unique rows and can be used for ranking. It has been used over project\_id arranged (ORDER BY) in descending order (DESC) by task\_completed FROM the 'Tasks' table and later output the 'RankedMembers' CTE limiting the (LIMIT) output to top two entries.

WITH RankedMembers AS (

SELECT

member\_name,

project\_id,

task\_completed,

ROW\_NUMBER() OVER (PARTITION BY project\_id ORDER BY task\_completed DESC) AS highest\_members

FROM Tasks

)

SELECT \*

FROM RankedMembers

LIMIT 2;

**Answer #3:** I have create a correlated subquery. The outer parent query is checking for task\_id(s) and task\_names(s) that have due\_date(s) earlier than the child query where everytime the average due\_date is calculated from the Tasks table.

SELECT task\_id, task\_name, due\_date

FROM Tasks

WHERE due\_date < (SELECT AVG(due\_date) FROM Tasks);

**Answer #4:** I have created a subquery where the outer query is selecting the project\_id, project\_name and budget from the Projects table and the inner query is filtering with the MAX(budget). Since the question asked for a subquery, I performed it otherwise depending on the schema of my database, it can be performed without a subquery using ORDER BY budget DESC and outputing only the first row with LIMIT 1.

SELECT project\_id, project\_name, budget FROM Projects

WHERE budget = (SELECT MAX(budget) FROM Projects);

**Answer #5:** This is a derivation from the first answer where I have just converted the sum of task\_completed = TRUE with respect to COUNT(\*) from Tasks table into a percentage using a CTE 'Percent\_CompletedTasks'. The Group By command was used for aggregation within the CTE.

WITH Percent\_CompletedTasks AS (

SELECT

project\_id,

(SUM(task\_completed = TRUE)/COUNT(\*)) \* 100 AS Projectwise\_CompletedTasks\_Percentage

FROM Tasks

GROUP BY project\_id

)

SELECT \* FROM Percent\_CompletedTasks;

**Answer #6:** Similar to the scaler function above PARTITION window function was used to ORDER BY member\_name and COUNT the number of tasks\_assigned.

SELECT

member\_name AS assigned\_to,

task\_name,

COUNT(\*) OVER (PARTITION BY member\_name) AS task\_count

FROM Tasks

ORDER BY assigned\_to;

**Answer #7:** Since I wanted that two conditions from two different tables be satisfied, therefore, I performed a JOIN where I joined the Tasks and the Teams tables at member\_name and only if the Teams.role is 'Team Lead' AND the due\_date is within the next 15 days, then give me the member\_name from Teams table and task\_name from Tasks table. Since my dates were in the past therefore I did not use the NOW() or the CURDATE() method. I had to use DATE\_ADD() method to an assumed date '2023-10-17' with an interval of 15 days. The output gave 'UI Design Swati Mishra', which was expected.

SELECT

Tasks.task\_name,

Teams.member\_name

FROM

Tasks

JOIN

Teams ON Tasks.member\_name = Teams.member\_name

WHERE

Teams.role = 'Team Lead'

AND Tasks.due\_date = DATE\_ADD('2023-10-17', INTERVAL 15 DAY);

**Answer #8:** To solve this question, LEFT JOIN was performed. Since we want all the project details from the Projects table for which there is an entry in the Tasks table with the foreign key Projects.project\_id. Since the Projects table appears first, MySQL will treat it as the left table and Tasks appears later therefore it will be treated as the right table. Therefore, the LEFT JOIN will give all the values in the Projects Table that have an entry in the Tasks table. Since there is no entry for Projects.project\_id = 6, therefore, the same output was returned.

SELECT

Projects.project\_id

FROM

Projects

LEFT JOIN

Tasks ON Projects.project\_id = Tasks.project\_id

WHERE

Tasks.project\_id IS NULL;

**Answer #9:** I have created and populated the Model\_Training table as asked and used the Projects.project\_id for referencing it to the parent table. I found out the best-performing models for a project using the RANK() function in a CTE named 'ProjectAccuracyAI'.

CREATE TABLE Model\_Training (

training\_id INT PRIMARY KEY,

project\_id INT,

model\_name VARCHAR(100),

accuracy DECIMAL(5,2),

training\_date DATE,

FOREIGN KEY (project\_id) REFERENCES Projects(project\_id)

);

INSERT INTO Model\_Training (training\_id, project\_id, model\_name, accuracy, training\_date)

VALUES

(1, 1, 'SVM Classifier', 88.75, '2023-10-10'),

(2, 1, 'Transformers', 91.50, '2023-10-12'),

(3, 2, 'Decision Tree', 75.00, '2023-11-20'),

(4, 2, 'Random Forest', 82.30, '2023-11-22'),

(5, 3, 'KNN', 68.90, '2023-12-10'),

(6, 3, 'XGBoost', 85.25, '2023-12-11'),

(7, 4, 'Bayesian', 89.50, '2024-01-12'),

(8, 4, 'Polynomial Fitting', 92.60, '2024-01-15'),

(9, 5, 'Linear Regression', 72.10, '2024-02-22');

WITH ProjectAccuracyAI AS (

SELECT

project\_id,

model\_name,

accuracy,

RANK() OVER (PARTITION BY project\_id ORDER BY accuracy DESC) AS rank\_acc

FROM Model\_Training

)

SELECT \*

FROM ProjectAccuracyAI

WHERE rank\_acc = 1;

**Answer #10:** I created the table Data\_Sets as was asked and then populated. Then I performed JOIN as earlier as I wanted information from two different table wherein each of them, two different condtions must be satisified. size\_gb > 10: This ensures we only get large datasets and last\_updated >= CURDATE() - INTERVAL 30 DAY filters datasets updated in the last 30 days.

CREATE TABLE Data\_Sets (

dataset\_id INT PRIMARY KEY,

project\_id INT,

dataset\_name VARCHAR(255),

size\_gb DECIMAL(5,2),

last\_updated DATE,

FOREIGN KEY (project\_id) REFERENCES Projects(project\_id)

);

INSERT INTO Data\_Sets (dataset\_id, project\_id, dataset\_name, size\_gb, last\_updated) VALUES

(1, 1, 'Customer Behavior', 12.5, '2025-07-10'),

(2, 2, 'Sales Trends', 8.0, '2025-07-15'),

(3, 3, 'Web Analytics', 15.0, '2025-06-15'),

(4, 4, 'App Feedback', 20.0, '2025-07-25'),

(5, 5, 'Survey Responses', 9.5, '2025-07-28'),

(6, 1, 'Clickstream Data', 13.2, '2025-07-20');

SELECT DISTINCT

Projects.project\_id,

Projects.project\_name,

Data\_Sets.dataset\_name,

Data\_Sets.size\_gb,

Data\_Sets.last\_updated

FROM

Projects

JOIN

Data\_Sets ON Projects.project\_id = Data\_Sets.project\_id

WHERE

Data\_Sets.size\_gb > 10

AND Data\_Sets.last\_updated >= CURDATE() - INTERVAL 30 DAY;