Title: SQL Project Overview - Project_info Database

Introduction

This document provides an in-depth walkthrough of the SQL schema, data population, and query logic implemented in the `Project_info` database. The purpose of the database is to simulate a project management system where multiple projects, teams, and associated tasks are managed. The database schema is normalized, employing primary and foreign key constraints to maintain referential integrity and support meaningful relational queries.

---

1. Schema Design and Table Creation

Three main tables were created:

1. **Projects**:
2. Columns: `project_id` (PK), `project_name`, `budget`, `start_date`, `team_id` (FK).
3. Holds general project metadata including allocated team.
4. **Tasks**:
5. Columns: `task_id` (PK), `task_name`, `member_name`, `due_date`, `task_completed`, `project_id` (FK).
6. Stores tasks assigned under each project, completion status, and responsible member.
7. **Teams**:
8. Columns: `member_id` (PK), `member_name`, `role`, `team_id` (FK).
9. Details team members and their association with specific projects.

Additionally, foreign key constraints link:

- `Tasks.project_id` to `Projects.project_id`
- `Teams.team_id` to `Projects.team_id`

This relational linkage enables powerful JOIN operations for comprehensive querying.

---

2. Data Population

Sample data was inserted into each table. Notably:

- `` ` included a project ( `` project_id = 6 `` ) which is not referenced in `` Tasks` for use in left join scenarios.
- `and` included a few `NULL` values to validate queries for missing/optional data fields.

---

3. Query Breakdown and Logic

Query 1: Count of Total vs Completed Tasks

```
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;
```

- Groups tasks by project and evaluates how many tasks were completed.
- `SUM(task_completed = TRUE)` counts `TRUE` as 1 and `FALSE`/`NULL` as 0.

Query 2: Ranking Members by Completion

```
SELECT member_name, project_id, task_completed,
       RANK() OVER (PARTITION BY project_id ORDER BY task_completed DESC) AS
highest_members
FROM Tasks;
```

- Uses the `RANK()` window function to determine top contributors by project.
- Ensures tie-handling in ranking.

Query 3: Projects with Maximum Budget

```
SELECT project_id, project_name
FROM Projects
WHERE budget = (SELECT MAX(budget) FROM Projects);
```

- Identifies the highest-budget project via a subquery.

Query 4: Completion Percentage per Project

```
SELECT project_id,
       (SUM(task_completed = TRUE)/COUNT(*)) * 100 AS
Projectwise_CompletedTasks_Percentage
```

```
FROM Tasks
GROUP BY project_id;
```

- Returns percentage completion for each project based on task status.

Query 5: Task Assigned to Team Lead with Date Logic

```
SELECT task_name, member_name
FROM Tasks
JOIN Teams ON Tasks.member_name = Teams.member_name
WHERE Teams.role = 'Team Lead' AND Tasks.due_date = '2023-11-05' + INTERVAL 15
DAY;
```

- Filters tasks assigned to team leads on a specific due date computed with `INTERVAL`.

Query 6: Projects Without Tasks

```
SELECT Projects.project_id
FROM Projects
LEFT JOIN Tasks ON Projects.project_id = Tasks.project_id
WHERE Tasks.project_id IS NULL;
```

- A `LEFT JOIN` is used to find projects in the `Projects` table that have no corresponding records in `Tasks`.

Query 7: Best AI Model per Project

```
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;
```

- Uses `RANK()` in a CTE to get the best-performing model by project.

Query 8: Recent Large Datasets

```sql
SELECT DISTINCT project_id, dataset_name
FROM Data_Sets
WHERE size_gb > 10 AND last_updated >= CURDATE() - INTERVAL 30 DAY;
```

- Fetches recently updated and large datasets.
- `DISTINCT` removes duplicate `project_id-dataset_name` pairs.

Conclusion

This SQL-based system exemplifies effective use of schema design, data normalization, foreign key constraints, and advanced query techniques including window functions, subqueries, and CTEs. The task-oriented design allows for extensibility into real-world project management systems. Through these queries and design logic, practical understanding of relational databases has been deepened.

All outputs were validated in MySQL Workbench and key result sets were exported for documentation.