SQL Queries Explained Simply

Query 1:

```
SELECT * FROM (SELECT u.USERNAME, COUNT(upt.TASKID) AS COMPLETED_TASKS
    FROM USERPROJECTTASK upt
    JOIN TASK t ON upt.TASKID = t.TASKID
    JOIN PROJECTDETAILS pd ON upt.USERPROJECT_ID = pd.USERPROJECT_ID
    JOIN USERACCOUNT u ON pd.USERID = u.USERID
    WHERE t.TASKSTATUS = 'Completed' AND pd.PROJECTID = :PROJECTID
    GROUP BY u.USERNAME ORDER BY COMPLETED_TASKS DESC)
    WHERE ROWNUM <= 3
```

Explanation: Lists the top 3 users who completed the most tasks in a specific project.

Query 2:

```
SELECT p.PROJECTID AS PROJECT_ID, p.PROJECTNAME AS PROJECT_NAME,
    p.PROJECTSTARTDATE AS PROJECT_START_DATE, p.PROJECTDUEDATE AS PROJECT_DUE_DATE,
    p.PROJECTSTATUS AS PROJECT_STATUS, m.MILESTONEID AS MILESTONE_ID,
    m.MILESTONENAME AS MILESTONE_NAME, m.MILESTONEIDDUEDATE AS MILESTONE_DUE_DATE,
    m.MILESTONESTATUS AS MILESTONE_STATUS
    FROM project p JOIN milestone m ON p.PROJECTID = m.PROJECTID
    WHERE p.PROJECTID = :PROJECTID
```

Explanation: Retrieves detailed information about milestones associated with a given project.

Query 3:

```
SELECT ua.USERID, ua.USERNAME, ua.USEREMAIL, ua.USERCONTACT, ua.USERROLE,
    p.PROJECTID, p.PROJECTNAME, p.PROJECTSTARTDATE, p.PROJECTDUEDATE, p.PROJECTSTATUS
    FROM useraccount ua JOIN projectdetails pd ON ua.USERID = pd.USERID
    JOIN project p ON pd.PROJECTID = p.PROJECTID
    WHERE ua.USERID = :USERID
```

Explanation: Fetches project details assigned to a specific user.

Query 4:

```
SELECT p.PROJECTNAME, COUNT(t.TASKID) AS OVERDUE_TASKS
    FROM PROJECT p LEFT JOIN PROJECTDETAILS pd ON p.PROJECTID = pd.PROJECTID
    LEFT JOIN USERPROJECTTASK upt ON pd.USERPROJECT_ID = upt.USERPROJECT_ID
      LEFT JOIN TASK t ON upt.TASKID = t.TASKID AND t.TASKENDDATE < SYSDATE AND t.TASKSTATUS !=
'Completed'
    GROUP BY p.PROJECTNAME ORDER BY OVERDUE_TASKS DESC
```

Explanation: Counts overdue tasks grouped by each project name, sorted by the highest overdue tasks.

Query 5:

```
SELECT COUNT(*) AS Total_Project FROM PROJECT
```

Explanation: Counts the total number of projects.

Query 6:

```
SELECT COUNT(*) AS TOTAL_TASK FROM TASK
```

Explanation: Counts the total number of tasks.

Query 7:

```
SELECT COUNT(*) AS TOTAL_SUBTASK FROM SUBTASK
```

Explanation: Counts the total number of subtasks.

Query 8:

```
SELECT COUNT(*) AS TOTAL_USERS FROM USERACCOUNT
```

Explanation: Counts the total number of users.

Query 9:

```
SELECT ua.USERNAME, COUNT(t.TASKID) AS TOTAL_TASKS_COMPLETED
    FROM TASK t JOIN USERPROJECTTASK upt ON t.TASKID = upt.TASKID
    JOIN PROJECTDETAILS pd ON upt.USERPROJECT_ID = pd.USERPROJECT_ID
    JOIN USERACCOUNT ua ON pd.USERID = ua.USERID
    WHERE t.TASKSTATUS = 'Completed' GROUP BY ua.USERNAME
    ORDER BY TOTAL_TASKS_COMPLETED DESC
```

Explanation: Lists users and counts tasks they've completed, ordered by the most tasks.

Query 10:

```
SELECT PROJECTSTATUS, COUNT(PROJECTID) AS TOTAL_PROJECTS
    FROM PROJECT GROUP BY PROJECTSTATUS ORDER BY TOTAL_PROJECTS DESC
```

Explanation: Counts projects based on their status, like completed or ongoing.

Query 11:

```
SELECT T.TASKNAME, COUNT(S.SUBTASKID) AS SUBTASKCOUNT
    FROM SUBTASK S JOIN TASK T ON S.TASKID = T.TASKID
    GROUP BY T.TASKNAME ORDER BY SUBTASKCOUNT
```

Explanation: Counts the number of subtasks for each task and sorts them by subtask count.