Note: The solutions in this document are only indicative. They may be incomplete or even contain errors.

Practicum Preparation 04-25 SQL

Consider the following schema on soccer teams and matches between the “home” and “visiting” teams:

```
TEAM (NAME, YEARLY_BUDGET),
MATCH (HOME_T → TEAM.NAME, VISIT_T → TEAM.NAME, DATE, HOME_GOALS,
VISIT_GOALS)
```

as SQL schema:

```
CREATE TABLE team (
      name STRING,
      yearly_budget INTEGER,
      PRIMARY KEY (name));

CREATE TABLE match (
      home_t STRING,
      visit_t STRING,
      date DATE,
      home_goals INTEGER,
      visit_goals INTEGER,
      PRIMARY KEY (home_t, visit_t, date),
      FOREIGN KEY (home_t) REFERENCES team(name),
      FOREIGN KEY (visit_t) REFERENCES TEAM(name));
```

Some sample data

```
INSERT INTO team VALUES ('Manchester',500),('Barcelona',400),('Bayern',
300);
INSERT INTO match VALUES ('Barcelona','Manchester','2017-03-07',2,0),
                         ('Bayern','Barcelona','2017-04-24',1,1),
                          ('Bayern','Manchester','2017-05-15',0,1),
                          (‘Barcelona’,'Bayern','2017-06-15',3,1);
```

Practicum task 1

Write a SQL query that computes the average money spent for each goal for all teams.

Solution:

```
SELECT name, (yearly_budget / ngoals ) AS avg FROM (
  SELECT team_name, SUM(goals) AS ngoals FROM (
    SELECT home_t AS team_name, home_goals AS goals FROM match
    UNION ALL
    SELECT visit_t AS team_name, visit_goals AS goals FROM match) g
  GROUP BY team_name) ng
```

```
JOIN team t ON team_name = name
ORDER BY avg DESC;
```

Practicum task 2

1) Write a SQL query that computes a “winners table”, include the date of the match, the home, visiting and winning team while filtering out draws (same number of goals) and order by date
2) Write a SQL query that computes a table of the top teams including the team name, budget and the number of matches won. The top teams are the ones that have won the most matches.

Solutions:

```
SELECT date, home_t, visit_t,
       CASE WHEN home_goals > visit_goals THEN home_t ELSE visit_t END AS
winning_team
FROM   match
WHERE  home_goals <> visit_goals
ORDER  BY date;

-- alternatively, without CASE
SELECT date, home_t, visit_t, home_t as winning_team
FROM   match
WHERE  home_goals > visit_goals
UNION ALL
SELECT date, home_t, visit_t, visit_t as winning_team
FROM   match
WHERE  home_goals < visit_goals
ORDER  BY date;

-- top teams, can also re-use previous result as subquery
SELECT t, COUNT(*) AS n, yearly_budget
FROM   (SELECT home_t AS t
  FROM   match
  WHERE  home_goals > visit_goals
  UNION ALL
  SELECT visit_t AS t
  FROM   match
  WHERE  visit_goals > home_goals) ts
JOIN team ON (ts.t = team.name)
GROUP  BY t
ORDER  BY n DESC;
```