FINAL PROJECT

COMP41090 SQL PROGRAMMING

ANDREW DOYLE
STUDENT NUMBER: 12252388
MSc COMPUTER SCIENCE (CONVERSION)

04th MAY 2013

Table of Contents

List of Figures

INTRODUCTION

This introduction serves sets out the format of the author`s submission for this project. For each exercise, the SQL code is provided in this document accompanied by explanatory notes which serve to demonstrate and reinforce knowledge of the topics covered. The notes may prove useful to the author as a reference point for future projects, interviews, etc. This serves to aid the learning process, however in industry, concise comments within the sql code is more common.

In each section screenshots are also provided which provide evidence of the SQL code being implemented in the database together with the actual output to the screen. Additionally, separate .sql files are included with the submission (one for each exercise), which the Examiner may wish to utilize to test the code.

Only one change was made to the original ER diagram, and that was to rename the 'Meal' table as the 'Meal Plan' table. The ER diagram is included in the Appendix Section along with an excel spread sheet showing the table design and initial data. Queries carried out throughout the project may have added or removed data.

This submission contains:

1. This PDF.
2. SQL Files
3. Database Backup.

SECTION A - INNER JOINS

1. What training sessions involve the skill of defending?

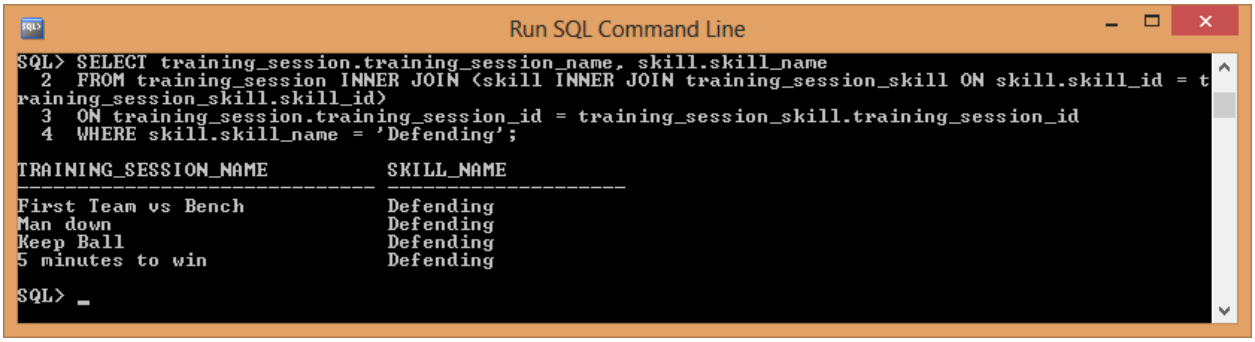
2. What meal plans are included in the regular player diet?

3. Display all fitness plans that include the workout ‘Circuit Training 1’.

4. What defenders complete the workout ‘Swimming’, and how many times per week?

INNER JOIN 1

Figure A.1 – Inner Join 1



EXPLANATORY NOTES

This query demonstrates an inner join involving three tables; Training Session, Skill, and Training Session Skill. The aim was to output a list of training session names that involve the skill of defending.

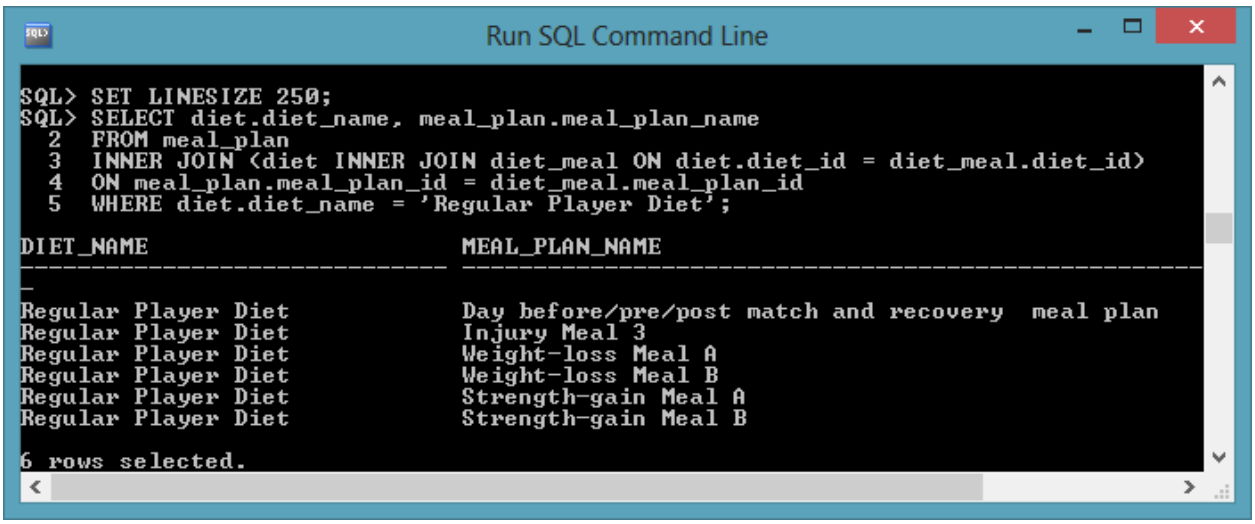
In order for this information to be retrieved, the junction table **training_session_skill** had to be utilised. Therefore a nested **INNER JOIN** was required. The **skill** table was joined to the **training_session_skill** table and this join was itself joined to the **training_session** table.

SQL USED:

```
SELECT training_session.training_session_name, skill.skill_name
FROM training_session INNER JOIN (skill INNER JOIN training_session_skill
ON skill.skill_id = training_session_skill.skill_id)
ON training_session.training_session_id =
training_session_skill.training_session_id
WHERE skill.skill_name = 'Defending';
```

INNER JOIN 2

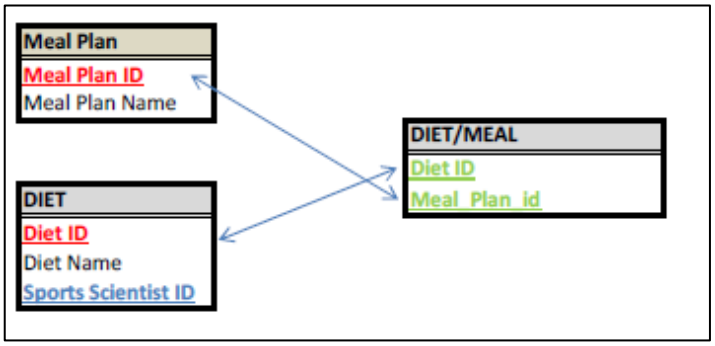
Figure A2 – Inner Join 2



EXPLANATORY NOTES

This query demonstrates an inner join involving three tables; Meal Plan, Diet_Meal, and Diet. The aim was to output a list of meal plans included in the Regular Player Diet plan.

In order for this information to be retrieved, the junction table **diet_meal** had to be utilised. Therefore a nested **INNER JOIN** was required. The **diet** table was joined to the **diet_meal** table and this join was itself joined to the **meal_plan** table.

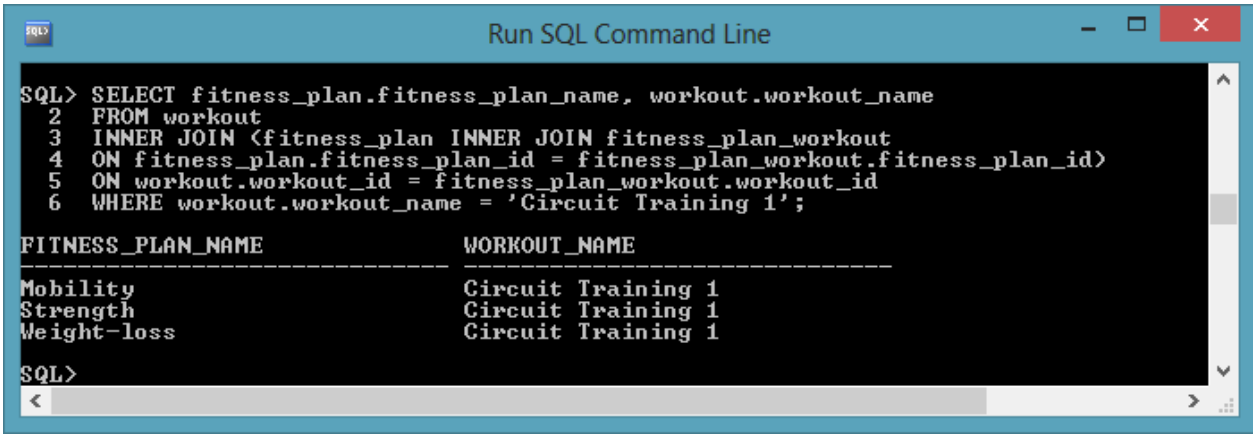


SQL USED:

```
-- INNER JOIN QUERY 2
SELECT diet.diet_name, meal_plan.meal_plan_name
FROM meal_plan
INNER JOIN (diet INNER JOIN diet_meal ON diet.diet_id = diet_meal.diet_id)
ON meal_plan.meal_plan_id = diet_meal.meal_plan_id
WHERE diet.diet_name = 'Regular Player Diet';
```

INNER JOIN 3

Figure A3 – Inner Join 2



SQL USED:

```
SELECT fitness_plan.fitness_plan_name, workout.workout_name
FROM workout
INNER JOIN (fitness_plan INNER JOIN fitness_plan_workout
ON fitness_plan.fitness_plan_id = fitness_plan_workout.fitness_plan_id)
ON workout.workout_id = fitness_plan_workout.workout_id
WHERE workout.workout_name = 'Circuit Training 1';
```

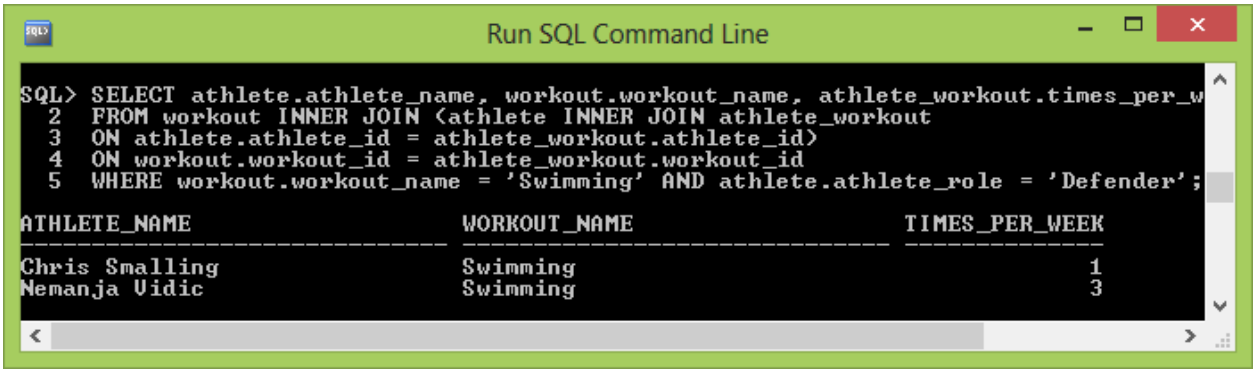
EXPLANATORY NOTES

This query demonstrates an inner join involving three tables; Workout, Fitness Plan, and Fitness_Plan_Workout. The aim was to output a list of fitness plans that include the workout 'Circuit Training 1'.

In order for this information to be retrieved, the junction table **fitness_plan_workout** had to be utilised. Therefore a nested **INNER JOIN** was required. Within the parenthesis, the **fitness_plan** and **fitness_plan_workout** tables are joined. Outside the parenthesis, the previous join is joined to the **workout** table.

INNER JOIN 4

Figure A4 – Inner Join 4



EXPLANATORY NOTES

This query demonstrates an inner join involving three tables; Athlete, Workout, and Athlete_Workout. The aim was to output a list of athletes that swim and the number of swims they complete a week.

In order for this information to be retrieved, the junction table **athlete_workout** had to be utilised. Therefore a nested **INNER JOIN** was required. Within the parenthesis, the **athlete** and **athlete_workout** tables are joined. Outside the parenthesis, the previous join is joined to the **workout** table.

SQL USED:

```
SELECT athlete.athlete_name, workout.workout_name, athlete_workout.times_per_week
FROM workout INNER JOIN (athlete INNER JOIN athlete_workout
ON athlete.athlete_id = athlete_workout.athlete_id)
ON workout.workout_id = athlete_workout.workout_id
WHERE workout.workout_name = 'Swimming' AND athlete.athlete_role = 'Defender';
```

SECTION B - OUTER JOINS

1. Full Outer Join 1: Display a list of midfielders, their athlete role, and their team name.

2. Full Outer Join 2: Display a list of training session names, along with details of the manager assigned (manager name and description).

3. Left Outer Join 1: List all coaches, with the Training Session ID`s that they are assigned to.

4. Left Outer Join 2: List all meal plans with their associated Diet ID`s.

5. Right Outer Join 1: List all diet names, along with the details regarding the Sports Scientist whom prescribed the diet (name and role).

6. Right Outer Join 2: List all fitness plan names, along with the details regarding the Sports Scientist whom the recommended the fitness plan (name and role).

FULL OUTER JOIN 1

Figure B1 – Full Outer Join 1

Run SQL Command Line

```
SQL> SELECT athlete.athlete_name, athlete.athlete_role, team.team_name
  2  FROM team FULL OUTER JOIN athlete ON team.team_id = athlete.team_id;

ATHLETE_NAME                   ATHLETE_ROLE         TEAM_NAME
------------------------------ -------------------- --------------------
David De Gea                   Goalkeeper           First_Team
Rafael Da Silva                Right_back           First_Team
Patrice Evra                   Left_back            First_Team
Philip Jones                   Defender             First_Team
Rio Ferdinand                  Defender             First_Team
Johnny Evans                   Defender             First_Team
Antonio Valencia               Right_Midfielder     First_Team
Anderson Da Silva              Midfielder           First_Team
Wayne Rooney                   Striker              First_Team
Ryan Giggs                     Midfielder           First_Team
Chris Smalling                 Defender             First_Team

ATHLETE_NAME                   ATHLETE_ROLE         TEAM_NAME
------------------------------ -------------------- --------------------
Anders Lindegaard              Goalkeeper           First_Team
Javier Hernandez               Striker              First_Team
Nemanja Vidic                  Defender             First_Team
Michael Carrick                Midfielder           First_Team
Luis Nani                      Left_Midfielder      First_Team
Ashley Young                   Left_Midfielder      First_Team
Danny Welbeck                  Striker              First_Team
Robin Van Persie               Striker              First_Team
Paul Scholes                   Midfielder           First_Team
Thomas Cleverley               Midfielder           First_Team
Darren Fletcher                Midfielder

ATHLETE_NAME                   ATHLETE_ROLE         TEAM_NAME
------------------------------ -------------------- --------------------
Chris Powell                   Midfielder           First_Team
Shinji Kagawa                  Midfielder           First_Team
Alexander Buttner              Left_back            First_Team
Ben Amos                       Goalkeeper
                                                    Reserve_Team
                                                    Under_18s
                                                    Under_21s

29 rows selected.
```

SQL USED:

```
SELECT athlete.athlete_name,
athlete.athlete_role,
team.team_name

FROM team FULL OUTER JOIN
athlete ON team.team_id =
athlete.team_id;
```

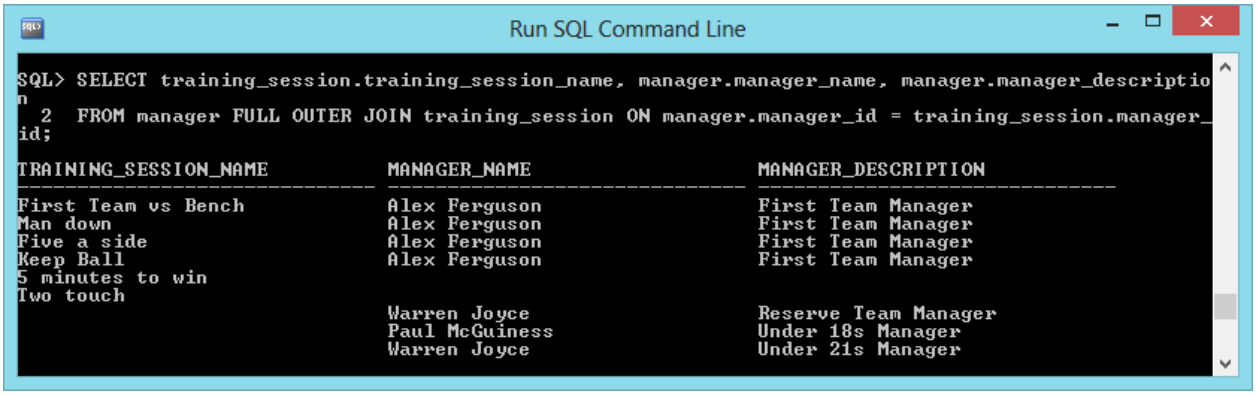
EXPLANATORY NOTES

This query demonstrates a full outer join involving the **athlete** table and the **team** table. The athletes name, role and corresponding team name are selected from both tables (joined by **team_id**).

Because a **FULL OUTER JOIN** is used, the **team_name**`s with no corresponding athlete details are also output. Also, the athlete who is not assigned to a team is also shown (Ben Amos). If an **INNER JOIN** was used instead, the reserve_team, under_18s and under_21s information would not be output to the screen.

FULL OUTER JOIN 2

Figure B2 – Full Outer Join 2



```
SQL> SELECT training_session.training_session_name, manager.manager_name, manager.manager_descriptio
n
  2  FROM manager FULL OUTER JOIN training_session ON manager.manager_id = training_session.manager_
id;

TRAINING_SESSION_NAME          MANAGER_NAME                   MANAGER_DESCRIPTION
------------------------------ ------------------------------ ------------------------------
First Team vs Bench            Alex Ferguson                  First Team Manager
Man down                       Alex Ferguson                  First Team Manager
Five a side                    Alex Ferguson                  First Team Manager
Keep Ball                      Alex Ferguson                  First Team Manager
5 minutes to win
Two touch
                               Warren Joyce                   Reserve Team Manager
                               Paul McGuiness                 Under 18s Manager
                               Warren Joyce                   Under 21s Manager
```

SQL USED:

```
SELECT training_session.training_session_name, manager.manager_name,
manager.manager_description
FROM manager FULL OUTER JOIN training_session ON manager.manager_id =
training_session.manager_id;
```

EXPLANATORY NOTES

This query demonstrates a **FULL OUTER JOIN** involving the **training session** table and the **manager** table. The training session name and corresponding manager name and manager description are selected from both tables (joined by **manager_id**).

Because a **FULL OUTER JOIN** is used, the **manager name**`s and **manager descriptions**`s with no corresponding training session details are also output. Also, the training sessions with no manager assigned are returned. If an **INNER JOIN** was used instead, the reserve_team, under_18s and under_21s manager information would not be output to the screen, nor would the training sessions '5 minutes to win' or 'two touch' .

LEFT OUTER JOIN 1

Figure B3 – Left Outer Join 1

```
Run SQL Command Line
SQL> SELECT coaches.coach_name, coaches.role, training_session_coaches.training_session_id
  2  FROM coaches LEFT OUTER JOIN training_session_coaches
  3  ON coaches.coach_id = training_session_coaches.coach_id;

COACH_NAME                     ROLE                           TRAINING_SESSION_ID
------------------------------ ------------------------------ -------------------
Eric Steele                    Goalkeeping Coach                                1
Andrew Doyle                   Assistant Coach                                  1
Mick Clegg                     Attack Coach                                     1
Richard Hawkins                Defense Coach                                    1
Eric Steele                    Goalkeeping Coach                                2
Andrew Doyle                   Assistant Coach                                  2
Richard Hawkins                Defense Coach                                    2
Andrew Doyle                   Assistant Coach                                  3
Mick Clegg                     Attack Coach                                     3
Richard Hawkins                Defense Coach                                    3
Andrew Doyle                   Assistant Coach                                  5

COACH_NAME                     ROLE                           TRAINING_SESSION_ID
------------------------------ ------------------------------ -------------------
Andrew Doyle                   Assistant Coach                                  6
Rene Meulensteen               First Team Coach

13 rows selected.

SQL> _
```

SQL USED:

```sql
SELECT coaches.coach_name, coaches.role, training_session_coaches.training_session_id
FROM coaches LEFT OUTER JOIN training_session_coaches
ON coaches.coach_id = training_session_coaches.coach_id;
```

EXPLANATORY NOTES

This query demonstrates a **LEFT OUTER JOIN** involving the **coaches** table and the **training_session_coaches** table. The coach name, role and corresponding training session id are selected from both tables (joined by **coach_id**).

Because a **LEFT OUTER JOIN** is used, the First Team coach, Rene Meulensteen with no corresponding training session details is also output. Rene is assigned to planning training sessions but does not attend.

If an **INNER JOIN** was used instead, Rene`s details would not be output to the screen.

LEFT OUTER JOIN 2

Figure B4 – Left Outer Join 2

```
Run SQL Command Line
SQL> SELECT meal_plan.meal_plan_name, diet_meal.diet_id
  2  FROM meal_plan LEFT OUTER JOIN diet_meal
  3  ON meal_plan.meal_plan_id = diet_meal.meal_plan_id
  4  ORDER BY meal_plan.meal_plan_name;

MEAL_PLAN_NAME                                                   DIET_ID
---------------------------------------------------------------- ----------
Day before/pre/post match and recovery  meal plan                     2
Day before/pre/post match and recovery  meal plan                     3
Fat-gain Chinese Meal
Fat-gain Sweets Meal
Injury Meal 1                                                          1
Injury Meal 2                                                          1
Injury Meal 3                                                          1
Injury Meal 3                                                          3
Injury Meal 3                                                          2
Strength-gain Meal A                                                   2
Strength-gain Meal A                                                   5

MEAL_PLAN_NAME                                                   DIET_ID
---------------------------------------------------------------- ----------
Strength-gain Meal A                                                   3
Strength-gain Meal B                                                   3
Strength-gain Meal B                                                   2
Strength-gain Meal B                                                   5
Weight-loss Meal A                                                     2
Weight-loss Meal A                                                     3
Weight-loss Meal A                                                     4
Weight-loss Meal B                                                     2
Weight-loss Meal B                                                     3
Weight-loss Meal B                                                     4

21 rows selected.
```

SQL USED:

```sql
SELECT meal_plan.meal_plan_name, diet_meal.diet_id
FROM meal_plan LEFT OUTER JOIN diet_meal
ON meal_plan.meal_plan_id = diet_meal.meal_plan_id
ORDER BY meal_plan.meal_plan_name;
```

EXPLANATORY NOTES

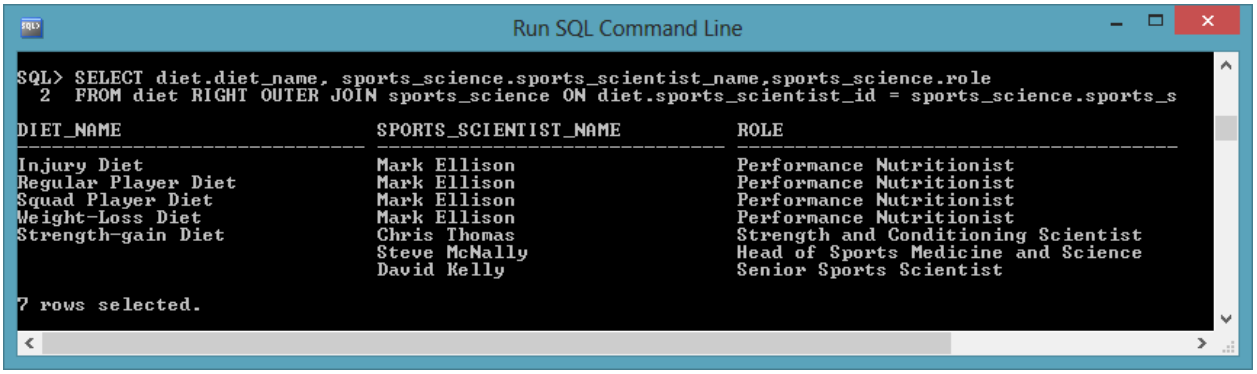
This query demonstrates a **LEFT OUTER JOIN** involving the **meal plan** table and the **diet_meal** table. The meal plan name and corresponding diet id are selected from both tables (joined by **meal_plan_id**).

Because a **LEFT OUTER JOIN** is used, two meals (Fat-gain Chinese/sweets meals) with no corresponding diet details is also returned. This meals have yet to be assigned to a diet plan.

If an **INNER JOIN** was used instead, these fat-gain meals would not be displayed .

RIGHT OUTER JOIN 1

Figure B5 – Right Outer Join 1



```
Run SQL Command Line

SQL> SELECT diet.diet_name, sports_science.sports_scientist_name,sports_science.role
  2  FROM diet RIGHT OUTER JOIN sports_science ON diet.sports_scientist_id = sports_science.sports_s

DIET_NAME                      SPORTS_SCIENTIST_NAME          ROLE
------------------------------ ------------------------------ ------------------------------------
Injury Diet                    Mark Ellison                   Performance Nutritionist
Regular Player Diet            Mark Ellison                   Performance Nutritionist
Squad Player Diet              Mark Ellison                   Performance Nutritionist
Weight-Loss Diet               Mark Ellison                   Performance Nutritionist
Strength-gain Diet             Chris Thomas                   Strength and Conditioning Scientist
                               Steve McNally                  Head of Sports Medicine and Science
                               David Kelly                    Senior Sports Scientist

7 rows selected.
```

SQL USED:

```sql
SELECT diet.diet_name,
sports_science.sports_scientist_name,sports_science.role
FROM diet RIGHT OUTER JOIN sports_science ON diet.sports_scientist_id =
sports_science.sports_scientist_id;
```

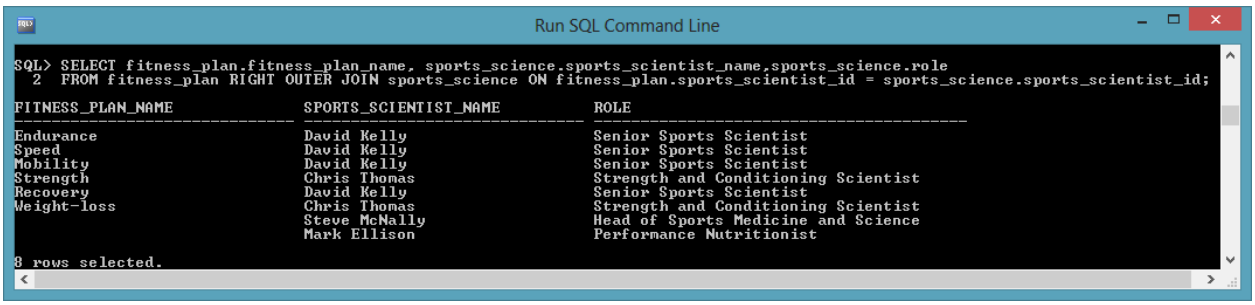
EXPLANATORY NOTES

This query demonstrates a **RIGHT OUTER JOIN** using the **diet** table and the **sports science** table. The diet name, sports scientist name and role are selected from both tables which are joined by the sports scientist id.

Steve McNally and David Kelly have been output to the screen, even though they did not themselves prescribe any diet plans. This is because a **RIGHT OUTER JOIN** is specified, so all rows have been output from the relevant columns in the sports science table.

RIGHT OUTER JOIN 2

Figure B6 – Right Outer Join 2



```
Run SQL Command Line
SQL> SELECT fitness_plan.fitness_plan_name, sports_science.sports_scientist_name,sports_science.role
  2  FROM fitness_plan RIGHT OUTER JOIN sports_science ON fitness_plan.sports_scientist_id = sports_science.sports_scientist_id;

FITNESS_PLAN_NAME              SPORTS_SCIENTIST_NAME          ROLE
------------------------------ ------------------------------ ------------------------------------
Endurance                      David Kelly                    Senior Sports Scientist
Speed                          David Kelly                    Senior Sports Scientist
Mobility                       David Kelly                    Senior Sports Scientist
Strength                       Chris Thomas                   Strength and Conditioning Scientist
Recovery                       David Kelly                    Senior Sports Scientist
Weight-loss                    Chris Thomas                   Strength and Conditioning Scientist
                               Steve McNally                  Head of Sports Medicine and Science
                               Mark Ellison                   Performance Nutritionist

8 rows selected.
```

SQL USED:

```sql
SELECT fitness_plan.fitness_plan_name,
sports_science.sports_scientist_name,sports_science.role
FROM fitness_plan RIGHT OUTER JOIN sports_science ON
fitness_plan.sports_scientist_id = sports_science.sports_scientist_id;
```

EXPLANATORY NOTES

This query demonstrates a **RIGHT OUTER JOIN** using the **fitness plan** table and the **sports science** table. The fitness plan name, sports scientist name and role are selected from both tables which are joined by the sports scientist id.

The performance nutrionist, Mark Ellison, and the head of the department, Steve McNally, have both been output to the screen; even though they did not themselves prescribe any fitness plans. This is because a **RIGHT OUTER JOIN** is specified, so all rows have been output from the relevant columns in the sports science table.

SECTION C - CUBE QUERY

Figure C1 – Cube Query

Run SQL Command Line

```
SQL> SELECT athlete_role, athlete_name,count(*) AS nbr_role, SUM(salary)
  2  FROM  athlete
  3  GROUP BY CUBE(athlete_role,athlete_name)
  4  ORDER BY athlete_role;

ATHLETE_ROLE         ATHLETE_NAME                     NBR_ROLE SUM(SALARY)
-------------------- ------------------------------ ---------- -----------
Defender             Chris Smalling                          1       40000
Defender             Johnny Evans                            1       45000
Defender             Nemanja Vidic                           1       90000
Defender             Philip Jones                            1       40000
Defender             Rio Ferdinand                           1      110000
Defender                                                     5      325000
Goalkeeper           Anders Lindegaard                       1       45000
Goalkeeper           Ben Amos                                1        7500
Goalkeeper           David De Gea                            1       50000
Goalkeeper                                                   3      102500
Left_Midfielder      Ashley Young                            1       90000

ATHLETE_ROLE         ATHLETE_NAME                     NBR_ROLE SUM(SALARY)
-------------------- ------------------------------ ---------- -----------
Left_Midfielder      Luis Nani                               1       90000
Left_Midfielder                                              2      180000
Left_back            Alexander Buttner                       1       25000
Left_back            Patrice Evra                            1       75000
Left_back                                                    2      100000
Midfielder           Anderson Da Silva                       1       40000
Midfielder           Chris Powell                            1        5000
Midfielder           Darren Fletcher                         1       50000
Midfielder           Michael Carrick                         1       55000
Midfielder           Paul Scholes                            1       30000
Midfielder           Ryan Giggs                              1       70000

ATHLETE_ROLE         ATHLETE_NAME                     NBR_ROLE SUM(SALARY)
-------------------- ------------------------------ ---------- -----------
Midfielder           Shinji Kagawa                           1       60000
Midfielder           Thomas Cleverley                        1       75000
Midfielder                                                   8      385000
Right_Midfielder     Antonio Valencia                        1       60000
Right_Midfielder                                             1       60000
Right_back           Rafael Da Silva                         1       40000
Right_back                                                   1       40000
Striker              Danny Welbeck                           1       50000
Striker              Javier Hernandez                        1       60000
Striker              Robin Van Persie                        1      180000
Striker              Wayne Rooney                            1      180000

ATHLETE_ROLE         ATHLETE_NAME                     NBR_ROLE SUM(SALARY)
-------------------- ------------------------------ ---------- -----------
Striker                                                      4      470000
                     Alexander Buttner                       1       25000
                     Anders Lindegaard                       1       45000
                     Anderson Da Silva                       1       40000
                     Antonio Valencia                        1       60000
                     Ashley Young                            1       90000
                     Ben Amos                                1        7500
                     Chris Powell                            1        5000
                     Chris Smalling                          1       40000
                     Danny Welbeck                           1       50000
                     Darren Fletcher                         1       50000

ATHLETE_ROLE         ATHLETE_NAME                     NBR_ROLE SUM(SALARY)
-------------------- ------------------------------ ---------- -----------
                     David De Gea                            1       50000
                     Javier Hernandez                        1       60000
                     Johnny Evans                            1       45000
                     Luis Nani                               1       90000
                     Michael Carrick                         1       55000
                     Nemanja Vidic                           1       90000
                     Patrice Evra                            1       75000
                     Paul Scholes                            1       30000
                     Philip Jones                            1       40000
                     Rafael Da Silva                         1       40000
                     Rio Ferdinand                           1      110000

ATHLETE_ROLE         ATHLETE_NAME                     NBR_ROLE SUM(SALARY)
-------------------- ------------------------------ ---------- -----------
                     Robin Van Persie                        1      180000
                     Ryan Giggs                              1       70000
                     Shinji Kagawa                           1       60000
                     Thomas Cleverley                        1       75000
                     Wayne Rooney                            1      180000
                                                            26     1662500
```

SQL USED:

```
SELECT athlete_role, athlete_name,count(*) AS nbr_role, SUM(salary)
FROM  athlete
GROUP BY CUBE(athlete_role,athlete_name)
ORDER BY athlete_role;
```

EXPLANATORY NOTES

This query demonstrates a **CUBE QUERY**. 4 columns are returned in the query: the athletes role, name, salary, and a count of the role.

The **GROUP BY CUBE** keyword has **athlete role** as its first parameter, this means the query results are categorised based on this column. In Figure C1, you can see that there are rows with no athlete names.. In these rows, a count of the defenders, midfielders etc is displayed along with the sum of the salaries from that role.

At the end, a regular list of athletes is displayed along with a count of athletes and a sum of their salaries.

SECTION D - SUBQUERIES

1. List the athlete(s) with the highest salary.

2. List all fitness plans with the workout 'Yoga 1'.

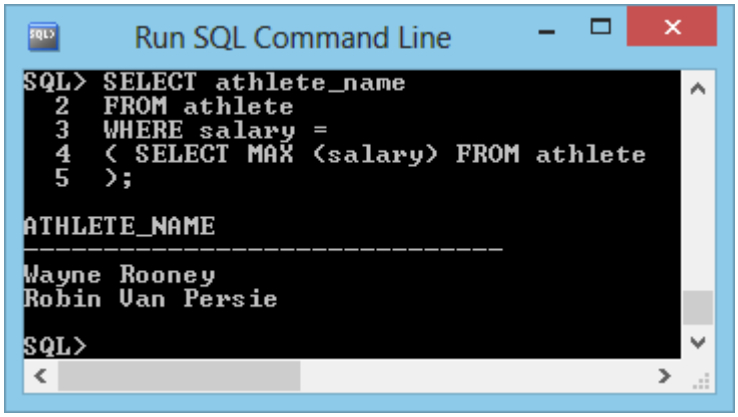
3. Select all athletes with a salary which is half the value of the average salary.

4. Select all the meal plans with a diet_id of 4.

5. Select all athletes with a greater than average market value.

SUB QUERY 1

Figure D1 – Sub Query 1



SQL USED:

```
SELECT athlete_name
FROM athlete
WHERE salary =
( SELECT MAX (salary) FROM athlete
);
```

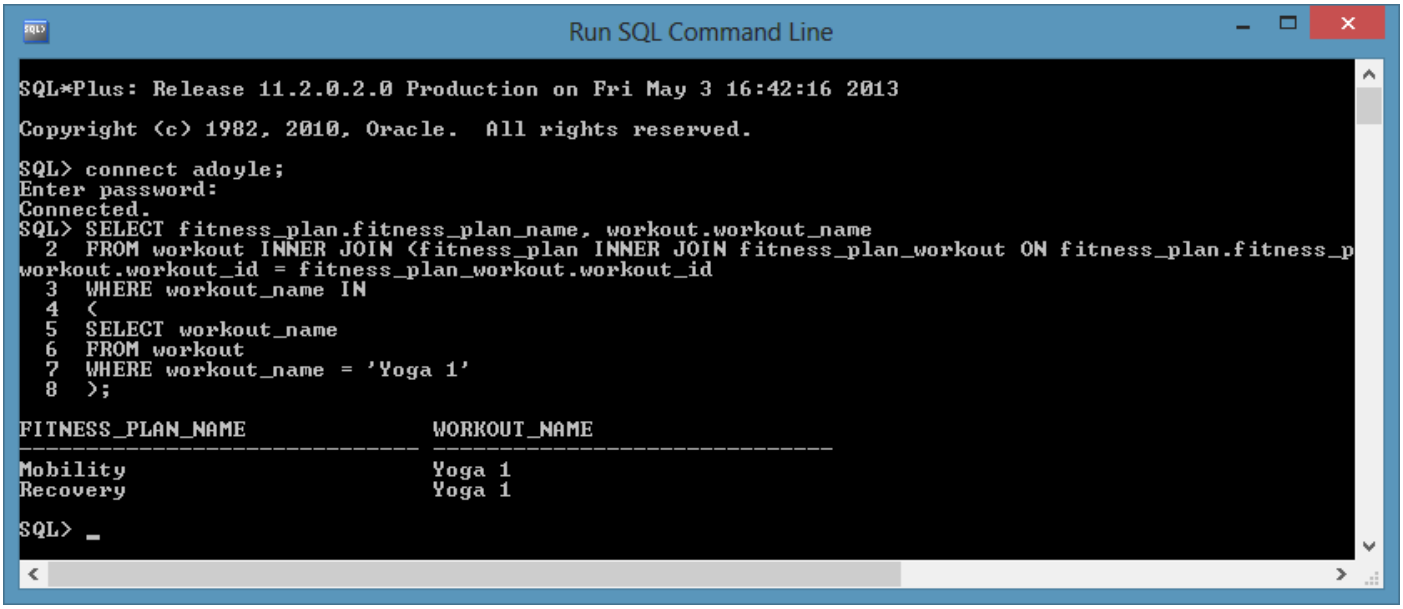
EXPLANATORY NOTES

This query demonstrates a simple sub-query, whereby the athletes are listed with the highest salaries. Two athletes (shown in the query result) share the highest salary amount, Wayne Rooney and Robin Van Persie.

In the subquery, the max salary is selected from the athlete table. In the outer query the athlete names are selected where the salary is equal to the sub query (max salary).

SUB QUERY 2

Figure D2 – Subquery 2



SQL USED:

```
SELECT fitness_plan.fitness_plan_name, workout.workout_name
FROM workout INNER JOIN (fitness_plan INNER JOIN fitness_plan_workout ON
fitness_plan.fitness_plan_id = fitness_plan_workout.fitness_plan_id) ON
workout.workout_id = fitness_plan_workout.workout_id
WHERE workout_name IN
(
SELECT workout_name
FROM workout
WHERE workout_name = 'Yoga 1'
);
```

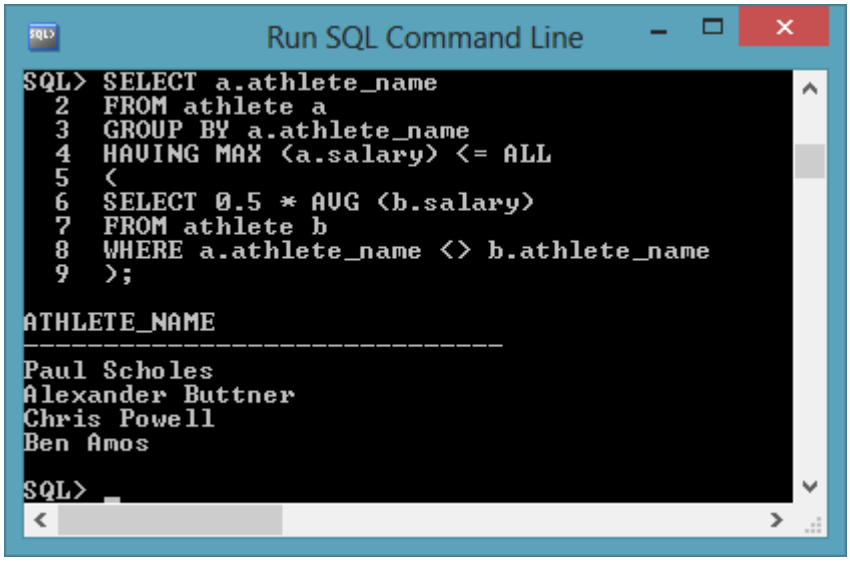
EXPLANATORY NOTES

This query lists all fitness plans with the workout 'Yoga 1'. The workout table is joined to the fitness_plan and fitness_plan_workout tables.

The subquery, within the where clause, is where it is specified to only return values with a workout_name of 'Yoga 1'.

SUB QUERY 3

Figure D3 – Subquery 3



SQL USED:

```
SELECT a.athlete_name
FROM athlete a
GROUP BY a.athlete_name
HAVING MAX (a.salary) <= ALL
(
SELECT 0.5 * AVG (b.salary)
FROM athlete b
WHERE a.athlete_name <> b.athlete_name
);
```

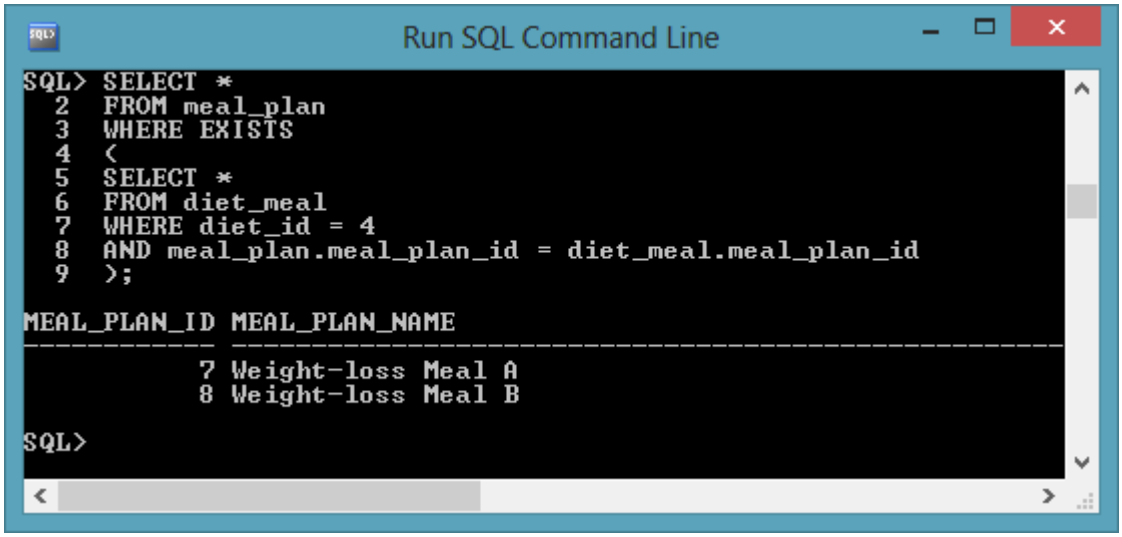
EXPLANATORY NOTES

This query selects all athletes with a salary which is half the value of the average salary.
In order to successfully complete this table, table aliases are required i.e. **athlete a** and **athlete b**.

The subquery selects a value which is half the average salary. In the main query, the **HAVING MAX** command specfies to return athlete names with salary values less than that returned by the subquery.

SUB QUERY 4

Figure D4 – Subquery 4



SQL USED:

```
SELECT *
FROM meal_plan
WHERE EXISTS
(
SELECT *
FROM diet_meal
WHERE diet_id = 4
AND meal_plan.meal_plan_id = diet_meal.meal_plan_id
);
```

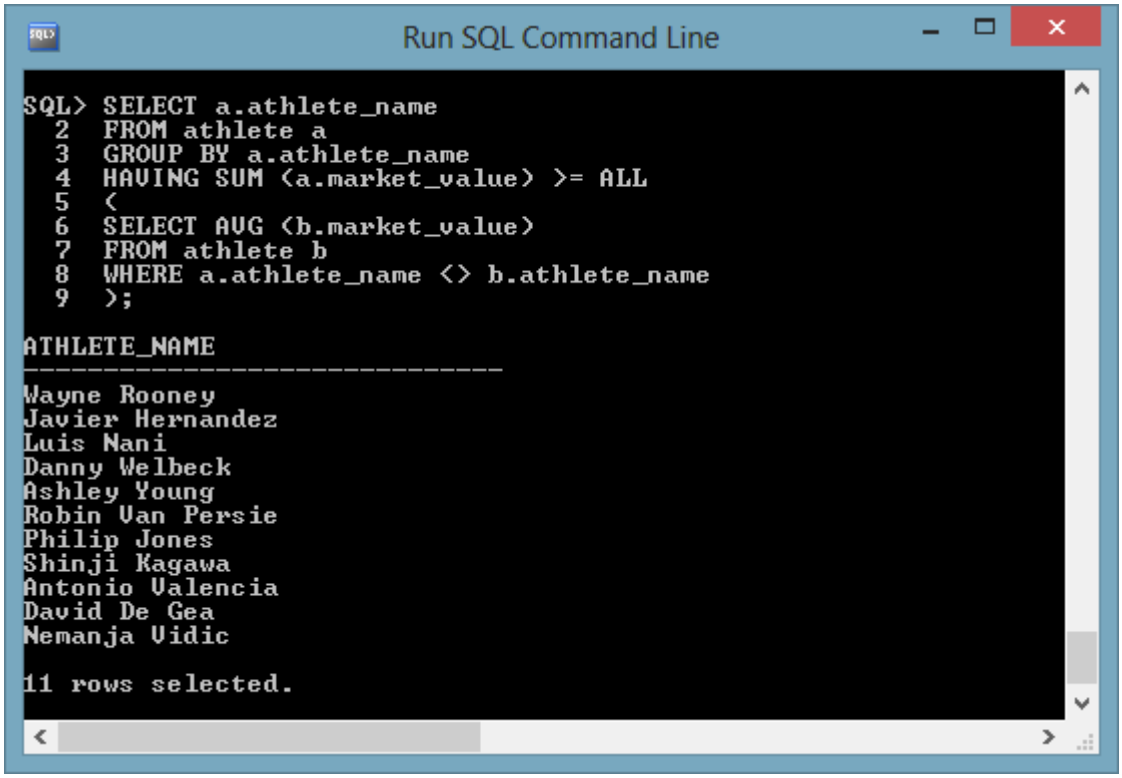
EXPLANATORY NOTES

This query selects all the meal plans with a diet_id of 4, this is possible using the **WHERE EXISTS** command. In the subquery, the diet_id is specified from the diet_meal table.

The outer query selects meal_plan details which relate to the diet_id specified in the subquery.

SUB QUERY 5

Figure D5 – Subquery 5



SQL USED:

```
SELECT a.athlete_name
FROM athlete a
GROUP BY a.athlete_name
HAVING SUM (a.market_value) >= ALL
(
SELECT AVG (b.market_value)
FROM athlete b
WHERE a.athlete_name <> b.athlete_name
);
```

EXPLANATORY NOTES

This query selects all athletes with a greater than average market value, again using table aliases.

The subquery calculates the average salary and the outer query selects all athlete names who have a market_value less than that calculated in the subquery.

SECTION E - PL/SQL PROCEDURES

1. Procedure 1 inserts a new meal plan id and name into the meal plan table.

2. Procedure 2 finds a particular athlete with their ID and outputs their name, role, transfer cost, and market value.

3. Procedure 3 uses a cursor and selects diet names from the diet table when executed.

4. Procedure 4 counts how many defenders are in the first team squad.

5. Procedure 5 deletes a meal plan from the meal plan table.

PROCEDURE 1

Figure E1 – Procedure 1

```
Run SQL Command Line
SQL> SELECT * FROM meal_plan;

MEAL_PLAN_ID MEAL_PLAN_NAME
------------ ----------------------------------------------------------------
           1 Day before/pre/post match and recovery  meal plan
           3 Injury Meal 1
           4 Injury Meal 2
           5 Injury Meal 3
           7 Weight-loss Meal A
           8 Weight-loss Meal B
           9 Strength-gain Meal A
          10 Strength-gain Meal B
          11 Fat-gain Chinese Meal
          12 Fat-gain Sweets Meal

10 rows selected.

SQL> CREATE OR REPLACE PROCEDURE new_meal_plan (p_meal_plan_id IN NUMBER, p_meal_plan_name IN VARCHA
  2  IS
  3  BEGIN
  4      INSERT INTO meal_plan (meal_plan_id, meal_plan_name) VALUES (p_meal_plan_id, p_meal_plan_nam
  5      EXCEPTION
  6      WHEN value_error THEN DBMS_OUTPUT.PUT_LINE('TRY AGAIN! That is the wrong type of input!');
  7  END;
  8  /

Procedure created.

SQL> BEGIN
  2  new_meal_plan(13,'Full Irish Breakfast');
  3  END;
  4  /

PL/SQL procedure successfully completed.

SQL> SELECT * FROM meal_plan;

MEAL_PLAN_ID MEAL_PLAN_NAME
------------ ----------------------------------------------------------------
           1 Day before/pre/post match and recovery  meal plan
           3 Injury Meal 1
           4 Injury Meal 2
           5 Injury Meal 3
           7 Weight-loss Meal A
           8 Weight-loss Meal B
           9 Strength-gain Meal A
          10 Strength-gain Meal B
          11 Fat-gain Chinese Meal
          12 Fat-gain Sweets Meal
          13 Full Irish Breakfast

11 rows selected.
```

SQL USED:

```sql
CREATE OR REPLACE PROCEDURE
new_meal_plan (p_meal_plan_id IN NUMBER,
p_meal_plan_name IN VARCHAR2 )
IS
BEGIN
    INSERT INTO meal_plan (meal_plan_id,
meal_plan_name) VALUES (p_meal_plan_id,
p_meal_plan_name);
    EXCEPTION
    WHEN value_error THEN
DBMS_OUTPUT.PUT_LINE('TRY AGAIN! That is
the wrong type of input!');
END;
/

BEGIN
new_meal_plan(13,'Full Irish
Breakfast');
END;
/
```

EXPLANATORY NOTES

This procedure inserts a new meal plan id and name into the meal plan table

Figure E1 demonstrates the **meal_plan** table before and after the creation and implementation of the procedure **new_meal_plan**.

The procedure creation statement specifies parameters in be input. In the body, the input parameters (VALUES) are inserted into the meal_plan table.

After creation, to implement the procedure, the parameters are specified in parameters after the procedure name.

PROCEDURE 2

Figure E2 – Procedure 2

```
Run SQL Command Line

PL/SQL procedure successfully completed.

SQL> CREATE OR REPLACE PROCEDURE find_athlete
  2  (p_athlete_id IN VARCHAR2,
  3  o_athlete_name OUT VARCHAR2,
  4  o_athlete_role OUT VARCHAR2,
  5  o_transfer_cost OUT VARCHAR2,
  6  o_market_value OUT VARCHAR2
  7  )
  8  AS
  9  BEGIN
 10     SELECT athlete_name, athlete_role, transfer_cost, market_value
 11     INTO o_athlete_name,o_athlete_role, o_transfer_cost, o_market_value
 12     FROM athlete
 13     WHERE athlete_id = p_athlete_id;
 14     EXCEPTION
 15     WHEN OTHERS
 16     THEN
 17     DBMS_OUTPUT.PUT_LINE('Error in finding athlete id:
 18     '||p_athlete_id);
 19  END find_athlete;
 20  /

Procedure created.

SQL>
SQL> DECLARE
  2     temp_out_athlete_name athlete.athlete_name%TYPE;
  3     temp_out_athlete_role athlete.athlete_role%TYPE;
  4     temp_out_transfer_cost athlete.transfer_cost%TYPE;
  5     temp_out_market_value athlete.market_value%TYPE;
  6  BEGIN
  7     find_athlete
  8     (20, temp_out_athlete_name, temp_out_athlete_role, temp_out_transfer_cost, temp_out_market_value);
  9     DBMS_OUTPUT.PUT_LINE
 10     ('Athlete 20 is: '||temp_out_athlete_name||
 11     ' | '|| temp_out_athlete_role||' | ' || temp_out_transfer_cost ||' | ' || temp_out_market_value ||' | '
 12     );
 13  END;
 14  /
Athlete 20 is: Robin Van Persie | Striker | 24000000 | 45000000 |

PL/SQL procedure successfully completed.

SQL> _
```

EXPLANATORY NOTES

This procedure finds a particular athlete with their ID and outputs their name, role, transfer cost, and market value. This demonstrates the use of input **and** output parameters with procedures.

The athlete id is input to the procedure, and athlete details are output. In the body, the required fields are selected into the output parameters on the condition that they match the p_athlete_id input to the procedure.

After creation, the procedure is implemented by first of all declaring temporary variables for output to the screen. The athlete id is then specified and the required temporary variables are **DBMS_OUTPUT** to the screen.

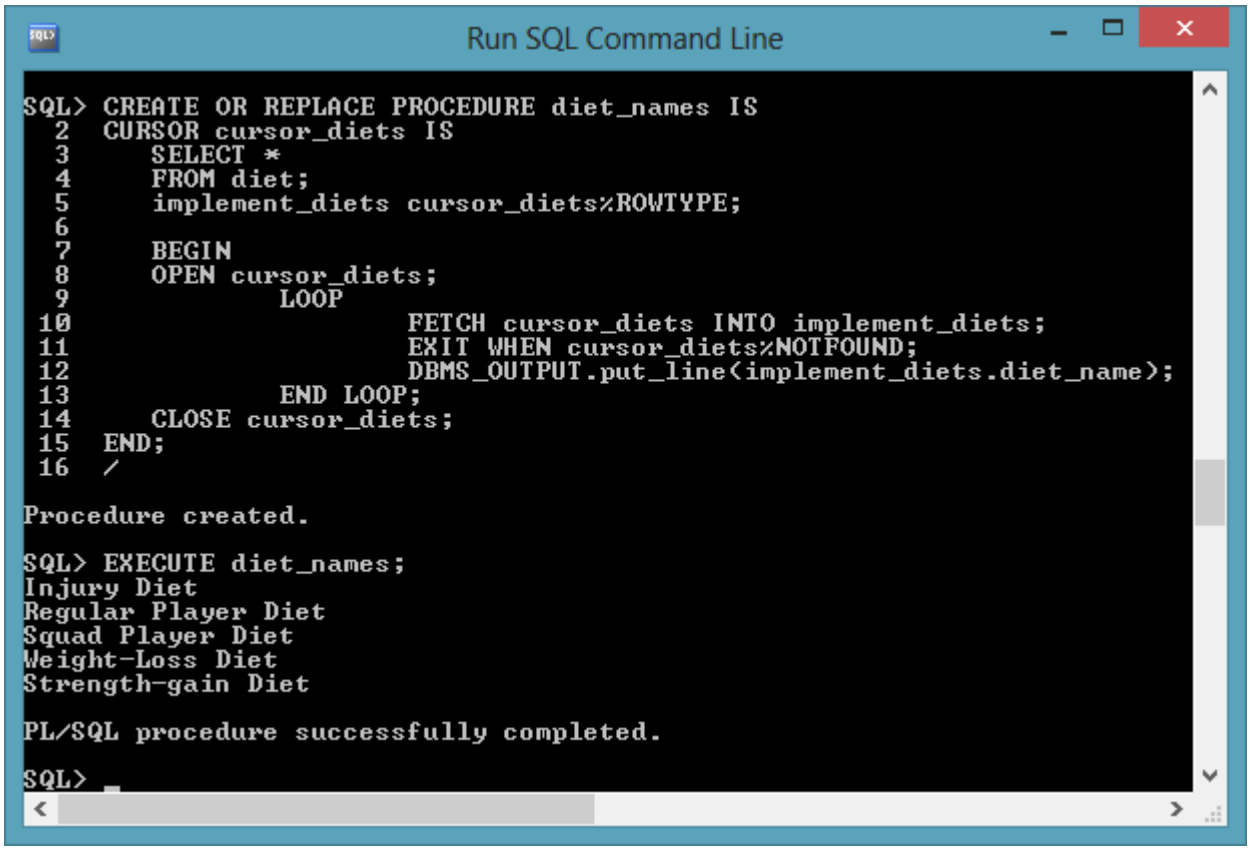
SQL USED:

```
CREATE OR REPLACE PROCEDURE find_athlete
(p_athlete_id IN VARCHAR2,
o_athlete_name OUT VARCHAR2,
o_athlete_role OUT VARCHAR2,
o_transfer_cost OUT VARCHAR2,
o_market_value OUT VARCHAR2
)
AS
BEGIN
    SELECT athlete_name, athlete_role, transfer_cost, market_value
    INTO o_athlete_name,o_athlete_role, o_transfer_cost, o_market_value
    FROM athlete
    WHERE athlete_id = p_athlete_id;
    EXCEPTION
    WHEN OTHERS
    THEN
    DBMS_OUTPUT.PUT_LINE('Error in finding athlete id:
    '||p_athlete_id);
END find_athlete;
/

DECLARE
    temp_out_athlete_name athlete.athlete_name%TYPE;
    temp_out_athlete_role athlete.athlete_role%TYPE;
    temp_out_transfer_cost athlete.transfer_cost%TYPE;
    temp_out_market_value athlete.market_value%TYPE;
BEGIN
    find_athlete
    (20, temp_out_athlete_name, temp_out_athlete_role,
temp_out_transfer_cost, temp_out_market_value);
    DBMS_OUTPUT.PUT_LINE
    ('Athlete 20 is: '||temp_out_athlete_name||
    ' | '|| temp_out_athlete_role||' | ' || temp_out_transfer_cost ||' | '
|| temp_out_market_value ||' | '
    );
END;
/
```

PROCEDURE 3

Figure E3 – Procedure 3



SQL USED:

```
CREATE OR REPLACE PROCEDURE diet_names IS
CURSOR cursor_diets IS
    SELECT *
    FROM diet;
    implement_diets cursor_diets%ROWTYPE;

    BEGIN
    OPEN cursor_diets;
        LOOP
            FETCH cursor_diets INTO
implement_diets;
            EXIT WHEN cursor_diets%NOTFOUND;

DBMS_OUTPUT.put_line(implement_diets.diet_name);
        END LOOP;
    CLOSE cursor_diets;
END;
/

EXECUTE diet_names;
```

EXPLANATORY NOTES

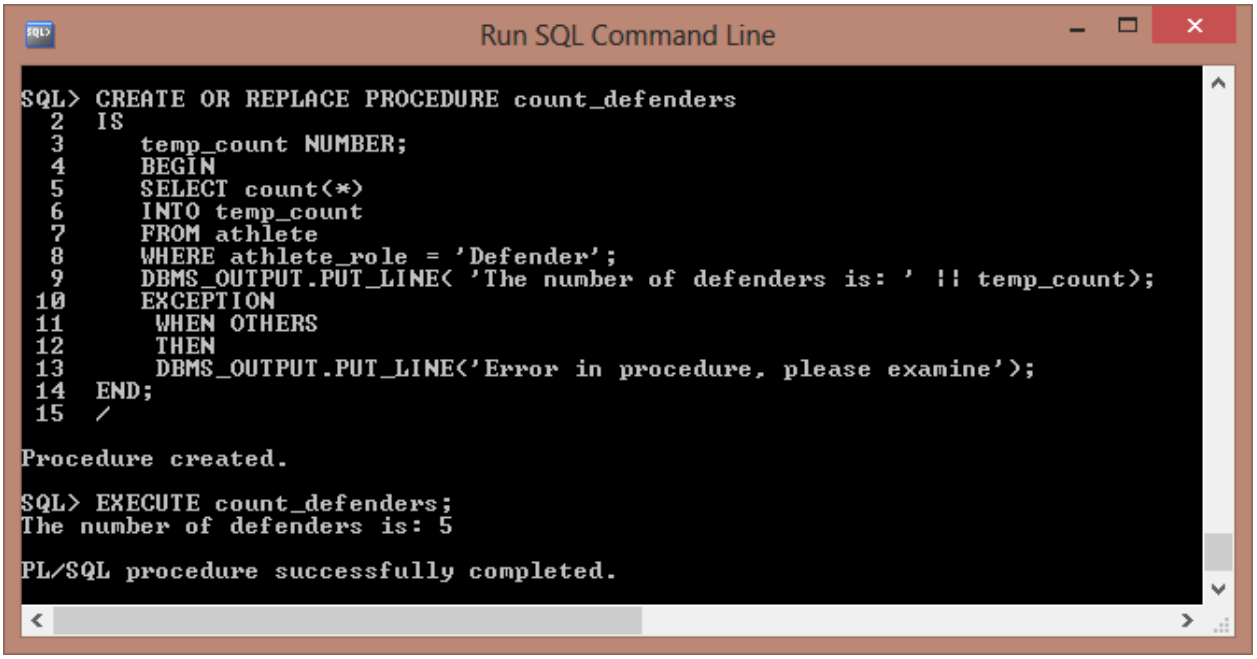
This query uses a cursor and selects diet names from the diet table when executed.

First of all, all information is **SELECT *** from the table. In the loop within the body, the contents of the cursor(**cursor_diets**) are fetched into **implement_diets**.

Within the **DBMS.OUTPUT.PUT_LINE** command, the field **diet_name** is specified; this is why this column is the only one output to the screen.

PROCEDURE 4

Figure E4 – Procedure 4



SQL USED:

```
CREATE OR REPLACE PROCEDURE count_defenders
IS
    temp_count NUMBER;
    BEGIN
    SELECT count(*)
    INTO temp_count
    FROM athlete
    WHERE athlete_role = 'Defender';
          DBMS_OUTPUT.PUT_LINE( 'The number of defenders   is: ' || temp_count);
    EXCEPTION
    WHEN OTHERS
    THEN
    DBMS_OUTPUT.PUT_LINE('Error in procedure, please examine');
END;
/

EXECUTE count_defenders;
```

EXPLANATORY NOTES

This procedure counts how many defenders are in the first team squad. A count of the defenders is select into the variable **temp_count**, and this is **DBMS_OUTPUT** to the screen.

The procedure is implemented using the **EXECUTE** command.

PROCEDURE 5

Figure E5 – Procedure 5

```
Run SQL Command Line

SQL> SELECT * FROM meal_plan;

MEAL_PLAN_ID MEAL_PLAN_NAME
------------ ----------------------------------------------------------------
           1 Day before/pre/post match and recovery  meal plan
           3 Injury Meal 1
           4 Injury Meal 2
           5 Injury Meal 3
           7 Weight-loss Meal A
           8 Weight-loss Meal B
           9 Strength-gain Meal A
          10 Strength-gain Meal B
          11 Fat-gain Chinese Meal
          12 Fat-gain Sweets Meal
          13 Full Irish Breakfast

11 rows selected.

SQL> CREATE OR REPLACE PROCEDURE delete_meal_plan (p_meal_plan_id IN NUMBER )
  2  IS
  3  BEGIN
  4      DELETE FROM meal_plan
  5     WHERE meal_plan_id = p_meal_plan_id;
  6      EXCEPTION
  7      WHEN value_error THEN DBMS_OUTPUT.PUT_LINE('TRY AGAIN! That is the wrong type of input!');
  8  END;
  9  /

Procedure created.

SQL> BEGIN
  2  delete_meal_plan(13);
  3  END;
  4  /

PL/SQL procedure successfully completed.

SQL> SELECT * FROM meal_plan;

MEAL_PLAN_ID MEAL_PLAN_NAME
------------ ----------------------------------------------------------------
           1 Day before/pre/post match and recovery  meal plan
           3 Injury Meal 1
           4 Injury Meal 2
           5 Injury Meal 3
           7 Weight-loss Meal A
           8 Weight-loss Meal B
           9 Strength-gain Meal A
          10 Strength-gain Meal B
          11 Fat-gain Chinese Meal
          12 Fat-gain Sweets Meal

10 rows selected.

SQL>
```

SQL USED:

```oracle
CREATE OR REPLACE PROCEDURE
delete_meal_plan (p_meal_plan_id IN
NUMBER )
IS
BEGIN
    DELETE FROM meal_plan
    WHERE meal_plan_id = p_meal_plan_id;
    EXCEPTION
    WHEN value_error THEN
DBMS_OUTPUT.PUT_LINE('TRY AGAIN! That is
the wrong type of input!');
END;
/

BEGIN
delete_meal_plan(13);
END;
/
```

EXPLANATORY NOTES

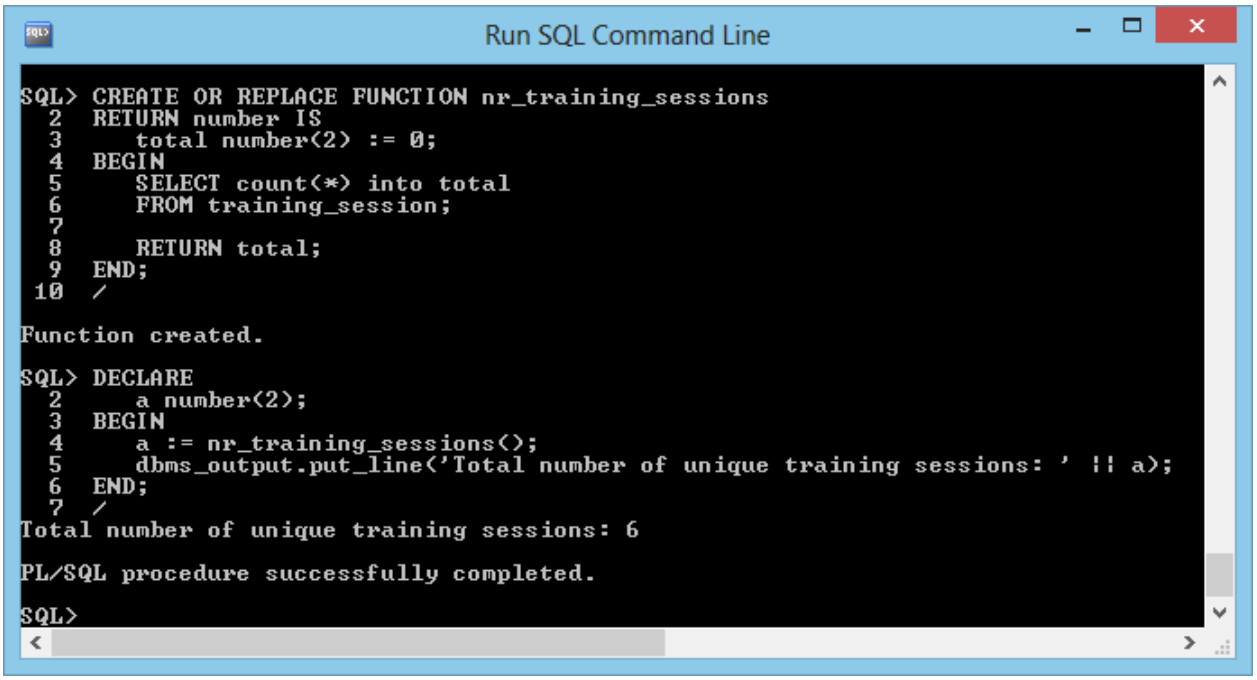
This procedure deletes a meal plan from the meal plan table.

The **DELETE FROM** command is carried out on the condition that the meal_plan_id matches the p_meal_plan_id passed to the procedure.

SECTION F - PL/SQL FUNCTION

This function counts the number of training sessions.

Figure F1 – Pl/SQL Function



SQL USED:

```
CREATE OR REPLACE FUNCTION nr_training_sessions
RETURN number IS
   total number(2) := 0;
BEGIN
   SELECT count(*) into total
   FROM training_session;

   RETURN total;
END;
/

DECLARE
   a number(2);
BEGIN
   a := nr_training_sessions();
   DBMS_OUTPUT.PUT_LINE('Total number of unique training sessions: ' || a);
END;
/
```

EXPLANATORY NOTES

This function counts the number of training sessions. The variable **total** is declared. In the body, a count of the rows is selected into **total** which is returned from the function.

a is assigned the returned value from the function, and is **DBMS_OUTPUT** to the screen.

SECTION G - TRIGGERS

TRIGGER 1 – Before Insert Trigger

SQL USED:

```
CREATE SEQUENCE a2
START WITH 41
INCREMENT BY 1;

CREATE OR REPLACE TRIGGER athlete_insert
BEFORE INSERT ON athlete
FOR EACH ROW
    BEGIN
    :NEW.athlete_id:=a2.NEXTVAL;
END;
/
BEGIN
INSERT INTO
athlete (athlete_name, athlete_role, salary, transfer_cost, market_value, team_id,
manager_id)
VALUES('The New Guy', 'All Positions', 5000, 5000000, 4000000, 1, 1);
END;
/
```

EXPLANATORY NOTES

This **BEFORE INSERT TRIGGER** in conjunction with the **SEQUENCE** a2 allow the automatic generation of an athlete ID upon inserting values into the table.

The sequence is set to start at 41 and increment by 1. A trigger is created which specifies that the new value of athlete_id is equal to the next value from the sequence (using the **a2.NEXTVAL** syntax).

To test that the trigger works, values are inserted into the tables, except for **athlete_id**, which is automatically inserted, as shown in Figure G3.

Figure G1 – Trigger 1 before implementation

Run SQL Command Line

```
SQL> SELECT * FROM ATHLETE;

ATHLETE_ID ATHLETE_NAME                   ATHLETE_ROLE           SALARY TRANSFER_COST MARKET_VALUE TEAM_ID MANAGER_ID
---------- ------------------------------ -------------------- ------- ------------- ------------ ------- ----------
         1 David De Gea                   Goalkeeper             50000      18900000     17000000       1          1
         2 Rafael Da Silva                Right_back             40000             0     14500000       1          1
         3 Patrice Evra                   Left_back              75000      11000000     13000000       1          1
         4 Philip Jones                   Defender               40000      16984000     17000000       1          1
         5 Rio Ferdinand                  Defender              110000      29300000      3500000       1          1
         6 Johnny Evans                   Defender               45000             0     13000000       1          1
         7 Antonio Valencia               Right_Midfielder       60000      16632000     24000000       1          1
         8 Anderson Da Silva              Midfielder             40000      20400000     12000000       1          1
        10 Wayne Rooney                   Striker               180000      27000000     65000000       1          1
        11 Ryan Giggs                     Midfielder             70000             0      1000000       1          1
        12 Chris Smalling                 Defender               40000      12000000     14000000       1          1

ATHLETE_ID ATHLETE_NAME                   ATHLETE_ROLE           SALARY TRANSFER_COST MARKET_VALUE TEAM_ID MANAGER_ID
---------- ------------------------------ -------------------- ------- ------------- ------------ ------- ----------
        13 Anders Lindegaard              Goalkeeper             45000       7000000      6500000       1          1
        14 Javier Hernandez               Striker                60000       8000000     22000000       1          1
        15 Nemanja Vidic                  Defender               90000      12000000     21000000       1          1
        16 Michael Carrick                Midfielder             55000      18600000     11000000       1          1
        17 Luis Nani                      Left_Midfielder        90000      17300000     27000000       1          1
        18 Ashley Young                   Left_Midfielder        90000      18000000     25000000       1          1
        19 Danny Welbeck                  Striker                50000             0     17000000       1          1
        20 Robin Van Persie               Striker               180000      24000000     45000000       1          1
        22 Paul Scholes                   Midfielder             30000             0      1000000       1          1
        23 Thomas Cleverley               Midfielder             75000             0     12500000       1          1
        24 Darren Fletcher                Midfielder             50000             0      7500000             1

ATHLETE_ID ATHLETE_NAME                   ATHLETE_ROLE           SALARY TRANSFER_COST MARKET_VALUE TEAM_ID MANAGER_ID
---------- ------------------------------ -------------------- ------- ------------- ------------ ------- ----------
        25 Chris Powell                   Midfielder              5000       5000000      5000000       1          1
        26 Shinji Kagawa                  Midfielder             60000      21000000     21000000       1          1
        28 Alexander Buttner              Left_back              25000       4400000      5000000       1          1
        40 Ben Amos                       Goalkeeper              7500        500000       750000             1
```

Figure G2 – Trigger Implementation

Run SQL Command Line

```
SQL> CREATE SEQUENCE a2
  2  START WITH 41
  3  INCREMENT BY 1;

Sequence created.

SQL>
SQL> CREATE OR REPLACE TRIGGER athlete_insert
  2  BEFORE INSERT ON athlete
  3  FOR EACH ROW
  4     BEGIN
  5     :NEW.athlete_id:=a2.NEXTVAL;
  6  END;
  7  /

Trigger created.

SQL> BEGIN
  2  INSERT INTO
  3  athlete (athlete_name, athlete_role, salary, transfer_cost, market_value, team_id, manager_id)
  4  VALUES('The New Guy', 'All Positions', 5000, 5000000, 4000000, 1, 1);
  5  END;
  6  /

PL/SQL procedure successfully completed.
```

Figure G3 – Result of Trigger Implementation

Run SQL Command Line

```
SQL> SELECT * FROM athlete;

ATHLETE_ID ATHLETE_NAME                   ATHLETE_ROLE           SALARY TRANSFER_COST MARKET_VALUE TEAM_ID MANAGER_ID
---------- ------------------------------ -------------------- ------- ------------- ------------ ------- ----------
         1 David De Gea                   Goalkeeper             50000      18900000     17000000       1          1
         2 Rafael Da Silva                Right_back             40000             0     14500000       1          1
         3 Patrice Evra                   Left_back              75000      11000000     13000000       1          1
         4 Philip Jones                   Defender               40000      16984000     17000000       1          1
         5 Rio Ferdinand                  Defender              110000      29300000      3500000       1          1
         6 Johnny Evans                   Defender               45000             0     13000000       1          1
         7 Antonio Valencia               Right_Midfielder       60000      16632000     24000000       1          1
         8 Anderson Da Silva              Midfielder             40000      20400000     12000000       1          1
        10 Wayne Rooney                   Striker               180000      27000000     65000000       1          1
        11 Ryan Giggs                     Midfielder             70000             0      1000000       1          1
        12 Chris Smalling                 Defender               40000      12000000     14000000       1          1

ATHLETE_ID ATHLETE_NAME                   ATHLETE_ROLE           SALARY TRANSFER_COST MARKET_VALUE TEAM_ID MANAGER_ID
---------- ------------------------------ -------------------- ------- ------------- ------------ ------- ----------
        13 Anders Lindegaard              Goalkeeper             45000       7000000      6500000       1          1
        14 Javier Hernandez               Striker                60000       8000000     22000000       1          1
        15 Nemanja Vidic                  Defender               90000      12000000     21000000       1          1
        16 Michael Carrick                Midfielder             55000      18600000     11000000       1          1
        17 Luis Nani                      Left_Midfielder        90000      17300000     27000000       1          1
        18 Ashley Young                   Left_Midfielder        90000      18000000     25000000       1          1
        19 Danny Welbeck                  Striker                50000             0     17000000       1          1
        20 Robin Van Persie               Striker               180000      24000000     45000000       1          1
        22 Paul Scholes                   Midfielder             30000             0      1000000       1          1
        23 Thomas Cleverley               Midfielder             75000             0     12500000       1          1
        24 Darren Fletcher                Midfielder             50000             0      7500000              1

ATHLETE_ID ATHLETE_NAME                   ATHLETE_ROLE           SALARY TRANSFER_COST MARKET_VALUE TEAM_ID MANAGER_ID
---------- ------------------------------ -------------------- ------- ------------- ------------ ------- ----------
        25 Chris Powell                   Midfielder              5000       5000000      5000000       1          1
        26 Shinji Kagawa                  Midfielder             60000      21000000     21000000       1          1
        28 Alexander Buttner              Left_back              25000       4400000      5000000       1          1
        40 Ben Amos                       Goalkeeper              7500        500000       750000              1
        41 The New Guy                    All Positions           5000       5000000      4000000       1          1
```

TRIGGER 2 – after insert trigger

Figure G4 – Trigger 2 (After Insert Trigger)

```
Run SQL Command Line

SQL> CREATE OR REPLACE TRIGGER trigger2
  2  AFTER INSERT OR UPDATE ON athlete
  3  FOR EACH ROW
  4  BEGIN
  5      IF :NEW.salary > 250000 THEN
  6      dbms_output.put_line('This is a message from the club owners, the new employee, with an ID of: '
  7      || :NEW.athlete_id || ' has a ridicolously high salary: ?' || :NEW.salary || ' It better be worthwhile or your getting fired!');
  8      END IF;
  9  END;
 10  /

Trigger created.

SQL> INSERT INTO
  2  athlete (athlete_name, athlete_role, salary, transfer_cost, market_value, team_id, manager_id)
  3  VALUES('Lionel Messi', 'Magician', 275000, 50000000, 40000000, 1, 1);
This is a message from the club owners, the new employee, with an ID of: 42 has a ridicolously high salary: ?275000 It better be worthwhile or your ge
tting fired!

1 row created.

SQL> SELECT * FROM athlete;

ATHLETE_ID ATHLETE_NAME                   ATHLETE_ROLE            SALARY TRANSFER_COST MARKET_VALUE TEAM_ID MANAGER_ID
---------- ------------------------------ --------------------- -------- ------------- ------------ ------- ----------
         1 David De Gea                   Goalkeeper              50000      18900000     17000000       1          1
         2 Rafael Da Silva                Right_back              40000             0     14500000       1          1
         3 Patrice Evra                   Left_back               75000      11000000     13000000       1          1
         4 Philip Jones                   Defender                40000      16984000     17000000       1          1
         5 Rio Ferdinand                  Defender               110000      29300000      3500000       1          1
         6 Johnny Evans                   Defender                45000             0     13000000       1          1
         7 Antonio Valencia               Right_Midfielder        60000      16632000     24000000       1          1
         8 Anderson Da Silva              Midfielder              40000      20400000     12000000       1          1
        10 Wayne Rooney                   Striker                180000      27000000     65000000       1          1
        11 Ryan Giggs                     Midfielder              70000             0      1000000       1          1
        12 Chris Smalling                 Defender                40000      12000000     14000000       1          1

ATHLETE_ID ATHLETE_NAME                   ATHLETE_ROLE            SALARY TRANSFER_COST MARKET_VALUE TEAM_ID MANAGER_ID
---------- ------------------------------ --------------------- -------- ------------- ------------ ------- ----------
        13 Anders Lindegaard              Goalkeeper              45000       7000000      6500000       1          1
        14 Javier Hernandez               Striker                 60000       8000000     22000000       1          1
        15 Nemanja Vidic                  Defender                90000      12000000     21000000       1          1
        16 Michael Carrick                Midfielder              55000      18600000     11000000       1          1
        17 Luis Nani                      Left_Midfielder         90000      17300000     27000000       1          1
        18 Ashley Young                   Left_Midfielder         90000      18000000     25000000       1          1
        19 Danny Welbeck                  Striker                 50000             0     17000000       1          1
        20 Robin Van Persie               Striker                180000      24000000     45000000       1          1
        22 Paul Scholes                   Midfielder              30000             0      1000000       1          1
        23 Thomas Cleverley               Midfielder              75000             0     12500000       1          1
        24 Darren Fletcher                Midfielder              50000             0      7500000             1

ATHLETE_ID ATHLETE_NAME                   ATHLETE_ROLE            SALARY TRANSFER_COST MARKET_VALUE TEAM_ID MANAGER_ID
---------- ------------------------------ --------------------- -------- ------------- ------------ ------- ----------
        25 Chris Powell                   Midfielder               5000       5000000      5000000       1          1
        26 Shinji Kagawa                  Midfielder              60000      21000000     21000000       1          1
        28 Alexander Buttner              Left_back               25000       4400000      5000000       1          1
        40 Ben Amos                       Goalkeeper               7500        500000       750000             1
        42 Lionel Messi                   Magician               275000      50000000     40000000       1          1

27 rows selected.
```

SQL USED:

```
CREATE OR REPLACE TRIGGER trigger2
AFTER INSERT OR UPDATE ON athlete
FOR EACH ROW
BEGIN
    IF :NEW.salary > 250000 THEN
    dbms_output.put_line('This is a message from the club owners, the new employee,
with an ID of: '
    || :NEW.athlete_id || ' has a ridicolously high salary: €' || :NEW.salary || ' It
better be worthwhile or your getting fired!');
    END IF;
END;
/
INSERT INTO
athlete (athlete_name, athlete_role, salary, transfer_cost, market_value, team_id,
manager_id)
VALUES('Lionel Messi', 'Magician', 275000, 50000000, 40000000, 1, 1);
```

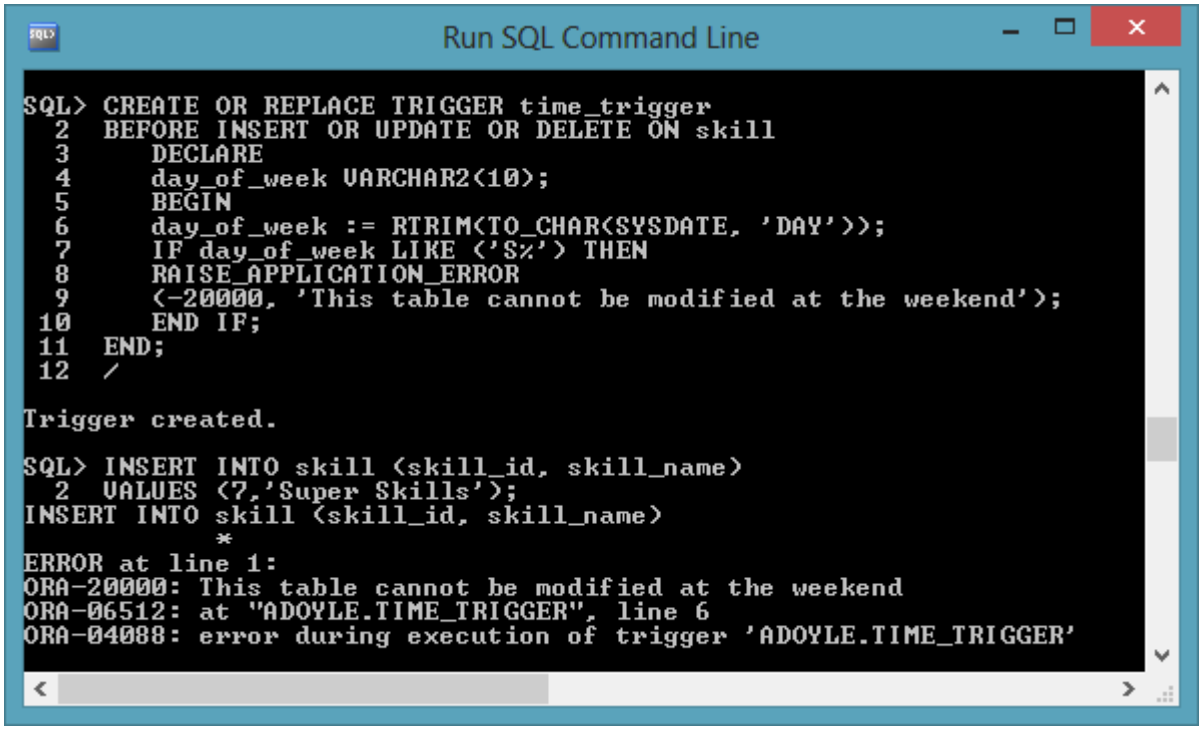
EXPLANATORY NOTES

This **AFTER INSERT TRIGGER** tests that if an inserted salary value over 250,000, a message is output to the screen, as shown in Figure G4. The **:NEW** keyword is required for this trigger to be possible.

The athlete_id is automatically added as it uses the **BEFORE INSERT** trigger previously created.

TRIGGER 3 – day of week condition

Figure G5 – Trigger 3



SQL USED:

```
CREATE OR REPLACE TRIGGER time_trigger
BEFORE INSERT OR UPDATE OR DELETE ON skill
    DECLARE
    day_of_week VARCHAR2(10);
    BEGIN
    day_of_week := RTRIM(TO_CHAR(SYSDATE, 'DAY'));
    IF day_of_week LIKE ('S%') THEN
    RAISE_APPLICATION_ERROR
    (-20000, 'This table cannot be modified at the weekend');
    END IF;
END;
/

INSERT INTO skill (skill_id, skill_name)
VALUES (7,'Super Skills');
```

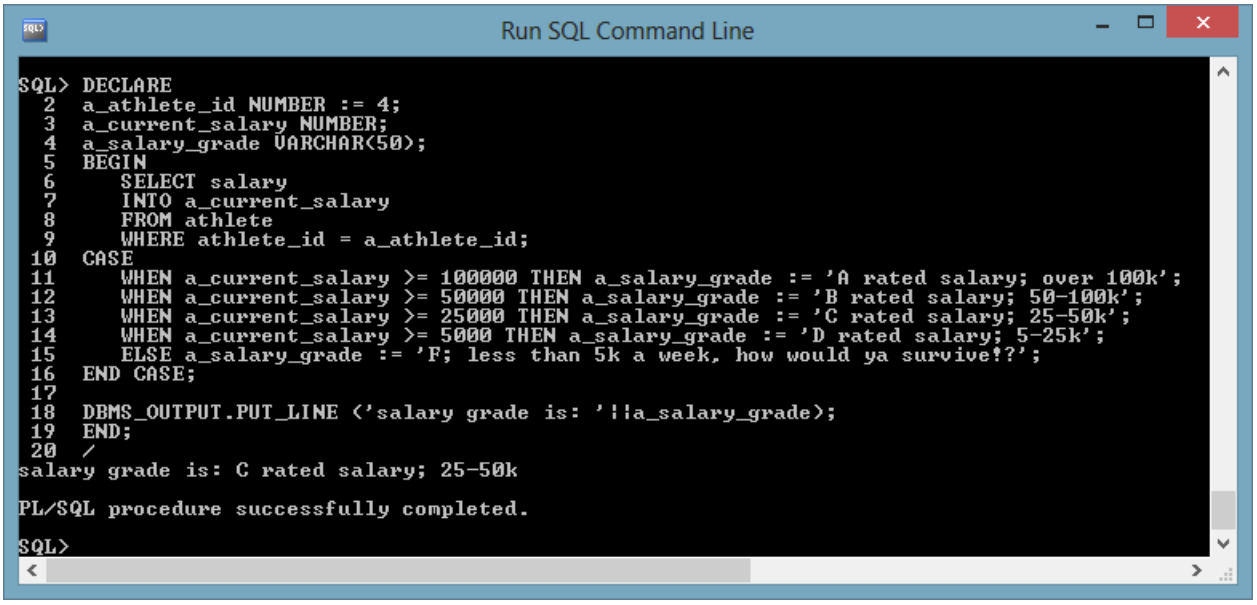
EXPLANATORY NOTES

This trigger checks the day of the week, and if it is a Saturday or Sunday, an alteration fails on the skill table and a warning is displayed to the user that the table cannot be modified at the weekend. .

The query was run by the author and the results are shown in Figure G5.

EXTRA FEATURE

Figure H – Extra Feature: Case Statement




```
DECLARE
a_athlete_id NUMBER := 4;
a_current_salary NUMBER;
a_salary_grade VARCHAR(50);
BEGIN
    SELECT salary
    INTO a_current_salary
    FROM athlete
    WHERE athlete_id = a_athlete_id;
CASE
    WHEN a_current_salary >= 100000 THEN a_salary_grade := 'A rated salary; over
100k';
    WHEN a_current_salary >= 50000 THEN a_salary_grade := 'B rated salary; 50-100k';
    WHEN a_current_salary >= 25000 THEN a_salary_grade := 'C rated salary; 25-50k';
    WHEN a_current_salary >= 5000 THEN a_salary_grade := 'D rated salary; 5-25k';
    ELSE a_salary_grade := 'F; less than 5k a week, how would ya survive!?';
END CASE;

DBMS_OUTPUT.PUT_LINE ('salary grade is: '||a_salary_grade);
END;
```

EXPLANATORY NOTES

This extra feature demonstrates a case-statement. Based on the value declared for **a_athlete_id**, a salary grade is output to the user.

Salaries are categorised into certain ranges (**WHEN** command) and a certain message is output based on that value (**THEN** command). The athlete_id '4' is used to test this case statement. This athlete is Phil Jones, with a salary of 40000, so the output is the 'C rated salary' i.e. >=25000 as specified in the **WHEN** command.

REFLECTION

The author learned a lot from this project, and may utilize the knowledge gained in future endeavours. This project provided a sneak preview into the world of database administration. Doing this project made the author realise that there is ten times as much yet to learn as has been learned already.

The author was intrigued by the ability of Oracle Database systems to communicate with other systems such as Microsoft Excel and Access. The author utilised Microsoft Excel in conjunction with Oracle SQL developer to import data created in Excel to the Oracle database by saving Excel Spread sheets as .csv files (comma separated).

This is only scratching the surface of the capabilities of database integration and the author is aware that this database could be made significantly more powerful and complex – the author may progress this project further to show potential employers at interviews etc.

BIBLIOGRAPHY

1. Rosenzweig,B., Rakhimove,E (2010). *Oracle PL/SQL by Example*. Fourth Edition, Boston: Addison-Wesley.

2. Rosenblum, M. , Dorsey, P.(2006). *Oracle PL/SQL for Dummies.* Indianapolis: Wiley Publishing.

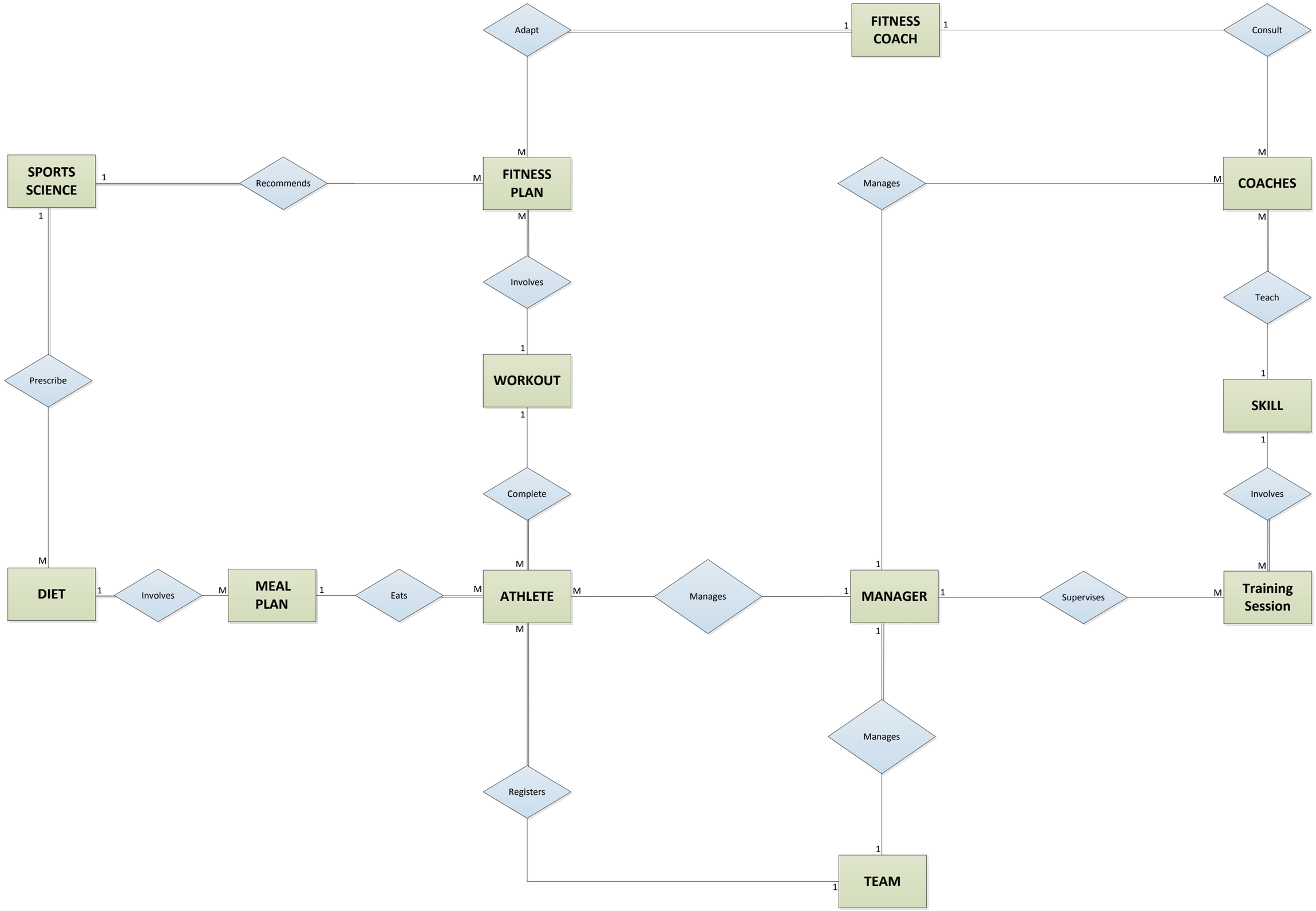
3. Taylor, A.(2010). *SQL for Dummies.* Indianapolis: Wiley Publishing.

APPENDIX

TABLE DATA AND ER DIAGRAM

PROJECT SPECIFICATION 1

ER DIAGRAM (FINAL)



Denotes a Primary Key
Denotes a Primary Key (which is also a Foreign key)
Denotes a Foreign Key

ATHLETE	TYPE
Athlete ID	NUMBER(3)
Athlete Name	VARCHAR(30)
Athlete Role	VARCHAR(20)
Salary	NUMBER (6)
Transfer Cost	NUMBER (8)
Market Value	NUMBER (8)
Team ID	Number(1)
Manager ID	Number(2)

TEAM	TYPE
Team ID	INT(1)
Team Name	VARCHAR(20)

MANAGER	TYPE
Manager ID	Number(2)
Manager Name	VARCHAR(30)
Manager Description	VARCHAR(30)
Salary	NUMBER (6)

TRAINING SESSION	TYPE
Training Session ID	Number(2)
Training Session Name	VARCHAR(30)

SKILL	TYPE
Skill ID	Number(2)
Skill Name	VARCHAR(20)

COACHES	TYPE
Coach ID	Number(2)
Coach Name	VARCHAR(30)
Role	VARCHAR(30)
Salary	NUMBER (6)

FITNESS COACH	
Fitness Coach ID	INT(1)
Fitness Coach Name	VARCHAR(30)
Education	VARCHAR(30)
Salary	NUMBER (6)

FITNESS PLAN	
Fitness Plan ID	Number(2)
Fitness Plan Name	VARCHAR(30)
Fitness Coach ID	Number(2)
Sports Scientist ID	Number(2)

WORKOUT	
Workout ID	Number(2)
Workout Name	VARCHAR(30)

Meal Plan	
Meal Plan ID	Number(2)
Meal Plan Name	VARCHAR(30)

DIET	
Diet ID	Number(2)
Diet Name	VARCHAR(30)
Sports Scientist ID	INT(1)

SPORTS SCIENCE	
Sports Scientist ID	Number(2)
Sports Scientist Name	VARCHAR(30)
Role	VARCHAR(40)
Salary	NUMBER (6)

ATHLETE/WORKOUT	
Athlete ID	Number(3)
Workout ID	Number(2)
Times per week	INT(1)

FITNESS PLAN/WORKOUT	
Fitness Plan ID	Number(2)
Workout ID	Number(2)

ATHLETE/MEAL	
Athlete ID	Number(3)
Meal ID	Number(2)
Schedule Time	Number(2)

DIET/MEAL	
Diet ID	Number(2)
Meal ID	Number(2)

TRAINING SESSION/SKILL	
Training Session ID	Number(2)
Skill ID	Number(2)

COACHES/SKILL	
Coach ID	Number(2)
Skill ID	Number(2)

ATHLETE							
Athlete ID	Athlete Name	Athlete Role	Salary	Transfer Cost	Market Value	Team ID	Manager ID
1	David De Gea	Goalkeeper	50000	18900000	17000000	1	1
2	Rafael Da Silva	Right_back	40000	0	14500000	1	1
3	Patrice Evra	Left_back	75000	11000000	13000000	1	1
4	Philip Jones	Defender	40000	16984000	17000000	1	1
5	Rio Ferdinand	Defender	110000	29300000	3500000	1	1
6	Johnny Evans	Defender	45000	0	13000000	1	1
7	Antonio Valencia	Right_Midfielder	60000	16632000	24000000	1	1
8	Anderson Da Silva	Midfielder	40000	20400000	12000000	1	1
10	Wayne Rooney	Striker	180000	27000000	65000000	1	1
11	Ryan Giggs	Midfielder	70000	0	1000000	1	1
12	Chris Smalling	Defender	40000	12000000	14000000	1	1
13	Anders Lindegaard	Goalkeeper	45000	7000000	6500000	1	1
14	Javier Hernandez	Striker	60000	8000000	22000000	1	1
15	Nemanja Vidic	Defender	90000	12000000	21000000	1	1
16	Michael Carrick	Midfielder	55000	18600000	11000000	1	1
17	Luis Nani	Left_Midfielder	90000	17300000	27000000	1	1
18	Ashley Young	Left_Midfielder	90000	18000000	25000000	1	1
19	Danny Welbeck	Striker	50000	0	17000000	1	1
20	Robin Van Persie	Striker	180000	24000000	45000000	1	1
22	Paul Scholes	Midfielder	30000	0	1000000	1	1
23	Thomas Cleverley	Midfielder	75000	0	12500000	1	1
24	Darren Fletcher	Midfielder	50000	0	7500000	1	1
25	Nick Powell	Midfielder	5000	5000000	5000000	1	1
26	Shinji Kagawa	Midfielder	60000	21000000	21000000	1	1
28	Alexander Buttner	Left_back	25000	4400000	5000000	1	1
40	Ben Amos	Goalkeeper	7500	500000	750000	1	1

DIET		
Diet ID	Diet Name	Sports Scientist ID
1	Injury Diet	1
2	Regular Player Diet	1
3	Squad Player Diet	1
4	Weight-Loss Diet	1
5	Strength-gain Diet	3

FITNESS PLAN			
Fitness Plan ID	Fitness Plan Name	Fitness Coach ID	Sports Scientist ID
1	Endurance	1	2
2	Speed	1	2
3	Mobility	1	2
4	Strength	1	3
5	Recovery	1	2
6	Weight-loss	1	3

TEAM	
Team ID	Team Name
1	First_Team
2	Under_21s
3	Under_18s
4	Reserve_Team

MANAGER			
Manager ID	Manager Name	Manager Description	Salary
1	Alex Ferguson	First Team Manager	170000
2	Warren Joyce	Under 21s Manager	10000
3	Paul McGuiness	Under 18s Manager	7500
4	Warren Joyce	Reserve Team Manager	20000

WORKOUT	
Workout ID	Workout Name
1	Nine Mile Run
2	Swimming
3	Cycling
4	Light running
5	Walk
6	Prowler
7	Sprints
8	Circuit Training 1
9	Weight Training 1
10	Circuit Training 2
11	Weight Training 2
12	Yoga 1
13	Yoga 2
14	Burpees
15	Kettlebells
16	Gymnastics

..

SKILL	
Skill ID	Skill Name
1	Defending
2	Attacking
3	Long-Ball
4	Set-Pieces
5	Dribbling
6	Ten vs Eleven

MEAL PLAN	
Meal Plan ID	Meal Plan Name
1	Day before/pre/post match and recovery meal plan
3	Injury Meal 1
4	Injury Meal 2
5	Injury Meal 3
7	Weight-loss Meal A
8	Weight-loss Meal B
9	Strength-gain Meal A
10	Strength-gain Meal B
11	Fat-gain Chinese Meal
12	Fat-gain Sweets Meal

TRAINING SESSION		
Training Session ID	Training Session Name	Manager ID
1	First Team vs Bench	1
2	Man down	1
3	Five a side	1
4	Keep Ball	1
5	5 minutes to win	
6	Two touch	

COACHES			
Coach ID	Coach Name	Role	Salary
1	Eric Steele	Goalkeeping Coach	6000
2	Rene Meulensteen	First Team Coach	15000
3	Mick Clegg	Attack Coach	9000
4	Richard Hawkins	Defense Coach	8000
5	Andrew Doyle	Assistant Coach	12000

FITNESS COACH			
Fitness Coach ID	Fitness Coach Name	Education	Salary
1	Tony Strudwick	Sports Science	6000

SPORTS SCIENCE			
Sports Scientist ID	Sports Scientist Name	Role	Salary
1	Mark Ellison	Performance Nutritionist	1900
2	David Kelly	Senior Sports Scientist	3000
3	Chris Thomas	Strength and Conditioning Scientist	2000
4	Steve McNally	Head of Sports Medicine and Science	4000

ATHLETE/WORKOUT

Athlete ID	Workout ID	Times / week
1	1	1
1	6	2
1	8	1
1	9	3
1	2	1
2	1	1
2	7	3
2	13	1
2	10	2
2	3	1
3	1	1
3	6	3
3	15	1
3	9	2
3	5	1
4	1	1
4	7	2
4	10	1
4	11	3
4	8	1
5	1	1
5	7	2
5	14	1
5	9	3
5	8	1
6	1	1
6	6	2
6	16	1
6	11	3
6	10	1
7	1	1
7	7	3
7	10	1
7	9	2
7	6	1
8	1	2
8	6	2
8	8	1
8	11	2
8	7	1
10	1	2
10	6	2
10	10	1
10	11	2
10	7	1

Athlete ID	Workout ID	Times per week
11	1	1
11	7	2
11	14	1
11	11	2
11	3	1
12	2	1
12	3	2
12	4	4
12	9	0
12	13	0
13	1	1
13	7	2
13	15	1
13	9	3
13	2	1
14	1	2
14	7	2
14	12	1
14	9	2
14	2	1
15	2	3
15	3	0
15	4	1
15	10	0
15	13	0
16	1	1
16	6	2
16	12	1
16	9	2
16	7	1
17	1	1
17	6	3
17	8	1
17	11	2
17	3	1
18	2	4
18	3	0
18	4	1
18	10	1
18	13	0
19	1	1
19	6	2
19	14	1
19	9	2
19	3	1

Athlete ID	Workout ID	Times / week
20	1	1
20	7	2
20	15	1
20	9	2
20	3	1
22	2	2
22	3	2
22	4	0
22	8	0
22	13	1
23	1	1
23	7	2
23	12	1
23	9	2
23	2	1
24	2	1
24	3	1
24	4	1
24	5	1
24	13	0
25	1	1
25	6	2
25	10	1
25	11	2
25	2	1
26	1	1
26	7	2
26	8	1
26	9	2
26	15	1
28	1	1
28	6	3
28	13	1
28	9	2
28	3	1
40	1	1
40	6	2
40	10	1
40	11	3
40	16	1

FITNESS PLAN/WORKOUT

Fitness Plan ID	Workout ID
1	1
1	2
1	3
2	6
2	7
3	8
3	10
3	12
3	13
3	14
3	15
3	16
4	8
4	9
4	10
4	11
4	15
5	4
5	5
5	2
5	3
5	12
5	13
6	1
6	2
6	3
6	6
6	7
6	8
6	9
6	10
6	11
6	14
6	15

ATHLETE/MEAL

Athlete ID	Meal Plan ID	Scheduled Time
1	1	Pre and post match
1	9	Mid week
2	1	Pre and post match
2	9	Mid week
3	1	Pre and post match
3	10	Mid week
4	1	Pre and post match
4	9	Mid week
5	1	Pre and post match
5	9	Mid week
6	1	Pre and post match
6	9	Mid week
7	1	Pre and post match
7	10	Mid week
8	1	Pre and post match
8	9	Mid week
10	1	Pre and post match
10	10	Mid week
11	1	Pre and post match
11	9	Mid week
12	4	Once a week
12	5	Three times a week
12	1	Twice a week
12	3	Once a week
13	1	Pre and post match
13	9	Mid week
14	1	Pre and post match
14	9	Mid week
15	5	Twice a week
15	3	Twice a week
15	3	Twice a week
15	9	Once a week
16	1	Pre and post match
16	9	Mid week
17	1	Pre and post match
17	10	Mid week
18	3	Twice a week
18	3	Twice a week
18	4	Twice a week
18	5	Once a week
19	1	Pre and post match
19	9	Mid week
20	1	Pre and post match
20	9	Mid week
22	3	Twice a week
22	4	Twice a week
22	2	Twice a week
22	10	Once a week
23	1	Pre and post match
23	9	Mid week
24	3	Twice a week
24	3	Twice a week
24	2	Twice a week
24	4	Once a week
25	1	Pre and post match
25	9	Mid week
26	1	Pre and post match
26	9	Mid week
28	1	Pre and post match
28	10	Mid week
40	1	Pre and post match
40	9	Mid week

DIET/MEAL

Diet ID	Meal Plan ID
1	3
1	4
1	5
2	1
2	5
2	7
2	8
2	9
2	10
3	1
3	5
3	7
3	8
3	9
3	10
4	7
4	8
5	9
5	10

Training Session / Coach

Training Session	Coach ID
1	1
1	2
1	3
1	4
2	1
2	2
2	4
3	2
3	3
3	4
5	
6	

Training Session / Skill

Training Session ID	Skill ID
1	1
1	2
1	4
2	1
2	2
2	6
3	5
4	1
4	5
5	1
5	2
6	2

COACHES/SKILL

Coach ID	Skill ID
1	1
1	2
1	3
1	4
2	1
2	2
2	3
2	4
2	5
2	6
3	2
3	3
3	4
3	5
3	6
4	1
4	3
4	4
4	6