Football Team Management

Databases project

Sabrina Lupsan

Faculty of Cybernetics, Statistics and Economic Informatics

Series G

Group 1064

Description of the theme

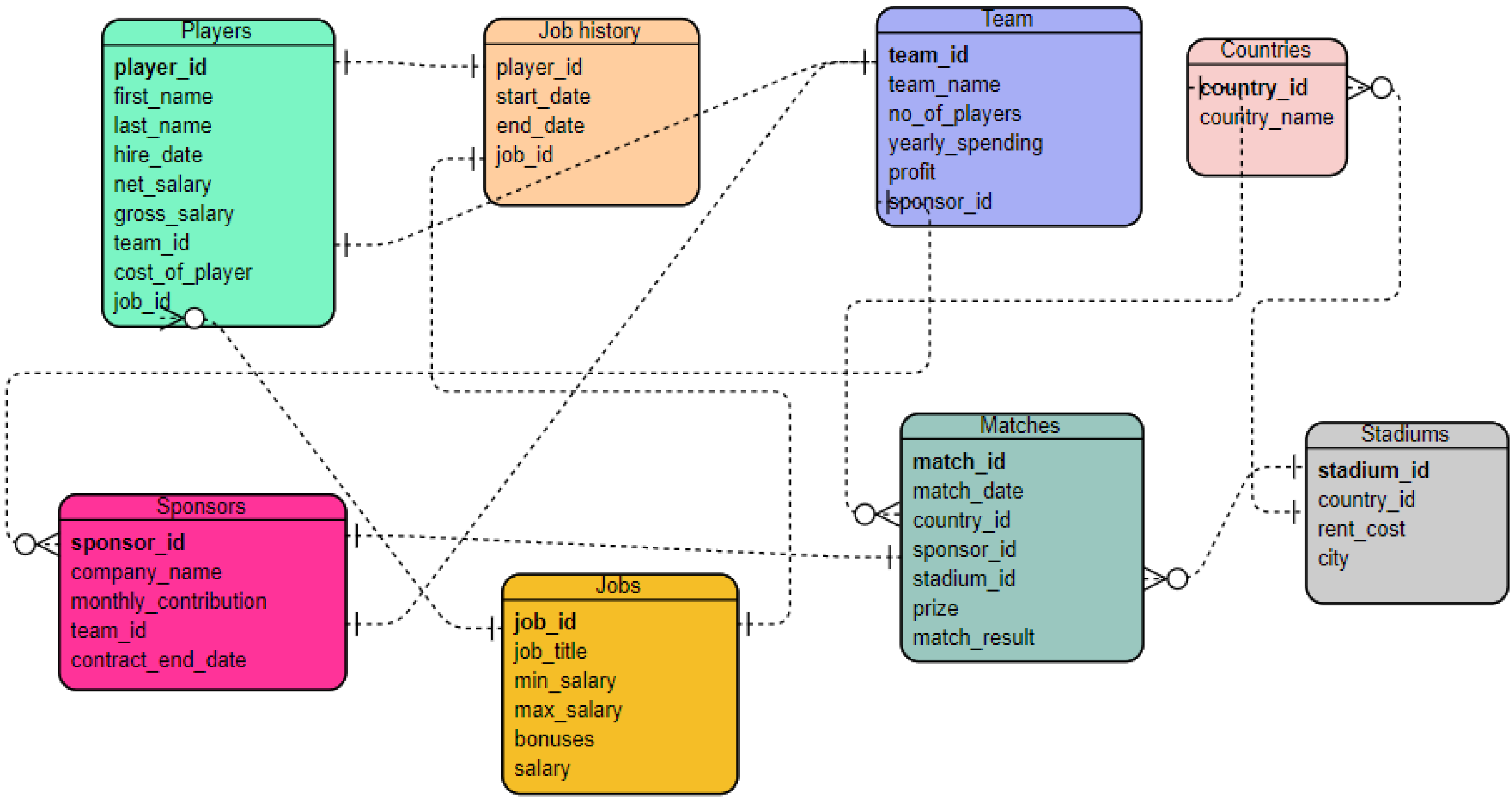
I chose to make a database of a Football Team and learned how to manage it. The players have four different kind of jobs, based on real-life evaluation and strategies – goalkeeper, midfielder, defender and attacker. The minimum and maximum salary of each job depend on the job and the net salaries were calculated using the gross salary and the taxes applied. Each team has players, a sponsor, matches played. The matches were played in different stadiums, and those stadiums are in different countries. This is how it all ties together, bringing the management of a Football Team. I believe that such a database is useful for storing data for official competitions like Champions League or The World Championship, which need official data, and for media news, who need statistics, such as the ones displayed in the exercises given.

For the economic part, rather than the management part, I used economic indicators such as: calculation of net salary based on gross salary and the imposed taxes, computation of the most profitable career based on monthly earning (considering salary, bonuses etc.), amount of taxes payed by each team, restricted to taxes on salaries.

All of the teams and names were fictious and the salaries chosen for this database are not official or true.

The conceptual scheme of the database

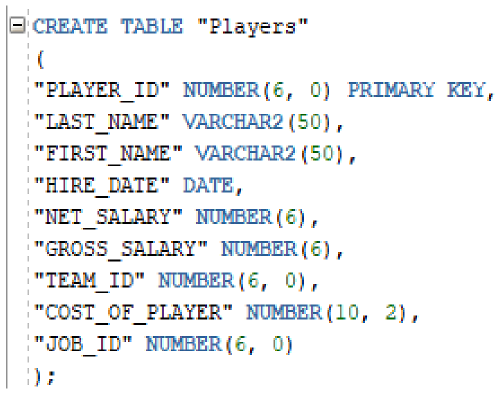
A database always begins with a conceptual scheme, which helps design and organize the coding part. This also helps identify the unique keys, foreign keys, connections to other tables in the database and, eventually, errors.



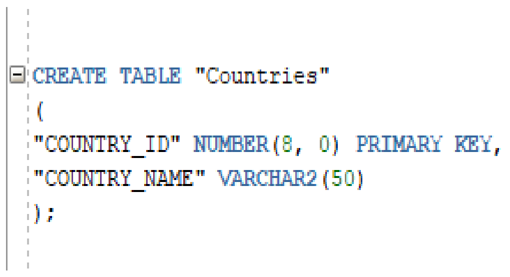
Database construction

1. Creating the database

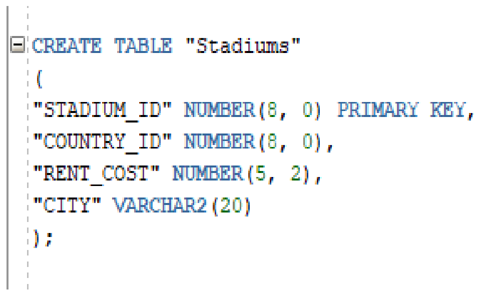
The first step is to create the tables using the command CREATE.



```
CREATE TABLE "Players"
(
"PLAYER_ID" NUMBER(6, 0) PRIMARY KEY,
"LAST_NAME" VARCHAR2(50),
"FIRST_NAME" VARCHAR2(50),
"HIRE_DATE" DATE,
"NET_SALARY" NUMBER(6),
"GROSS_SALARY" NUMBER(6),
"TEAM_ID" NUMBER(6, 0),
"COST_OF_PLAYER" NUMBER(10, 2),
"JOB_ID" NUMBER(6, 0)
);
```



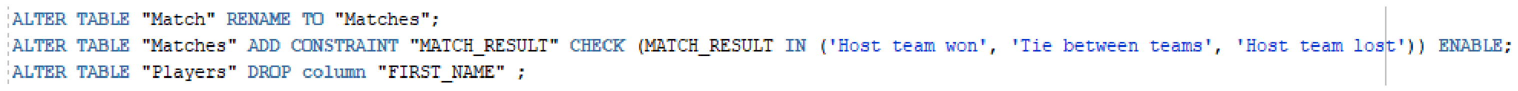
```
CREATE TABLE "Countries"
(
"COUNTRY_ID" NUMBER(8, 0) PRIMARY KEY,
"COUNTRY_NAME" VARCHAR2(50)
);
```



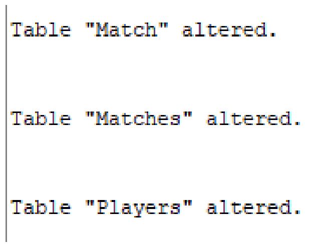
```
CREATE TABLE "Stadiums"
(
"STADIUM_ID" NUMBER(8, 0) PRIMARY KEY,
"COUNTRY_ID" NUMBER(8, 0),
"RENT_COST" NUMBER(5, 2),
"CITY" VARCHAR2(20)
);
```

2. Altering tables

After I created all the tables according to the conceptual schema, I used ALTER commands: I renamed the table Match, I added a constraint to the match result because it has only 3 possible outcomes and I removed the column first_name from players because I considered there were too many fields and the player_id was already differentiating one from another (as it is the primary key). I also added other constraints but didn't show all of them here.

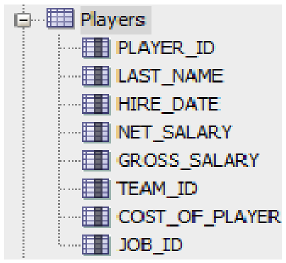


```
ALTER TABLE "Match" RENAME TO "Matches";
ALTER TABLE "Matches" ADD CONSTRAINT "MATCH_RESULT" CHECK (MATCH_RESULT IN ('Host team won', 'Tie between teams', 'Host team lost')) ENABLE;
ALTER TABLE "Players" DROP column "FIRST_NAME" ;
```



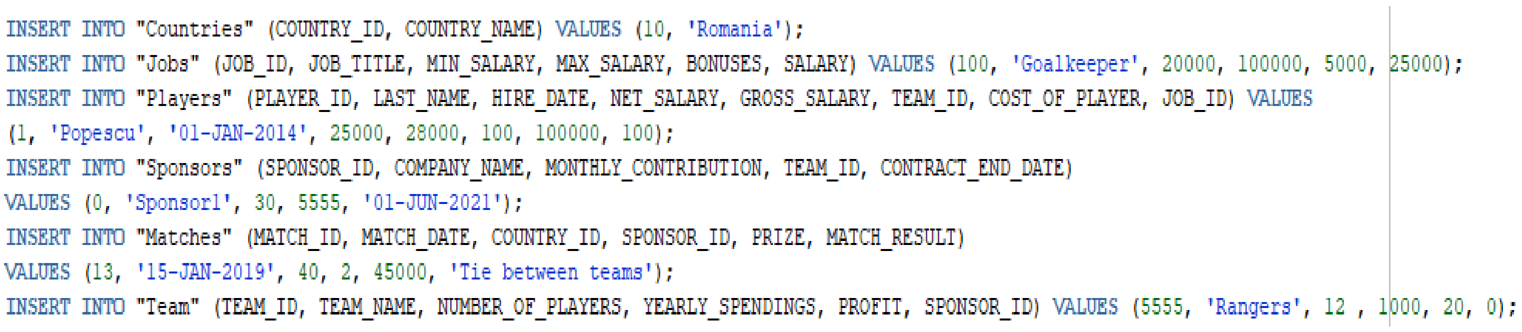
```
Table "Match" altered.


Table "Matches" altered.


Table "Players" altered.
```

The result can be seen on the right, in the "tables" section.

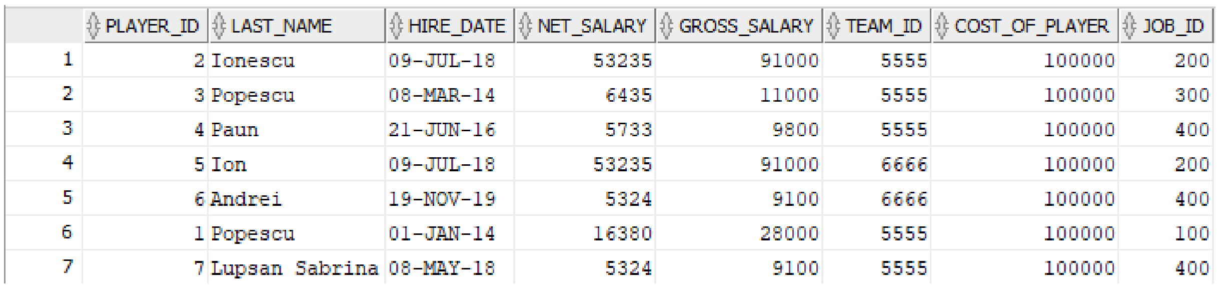

```
Players
    PLAYER_ID
    LAST_NAME
    HIRE_DATE
    NET_SALARY
    GROSS_SALARY
    TEAM_ID
    COST_OF_PLAYER
    JOB_ID
```

Data update operations

1. Insert values in the tables. This sample of code only contains some of the insert commands introduced.


```
INSERT INTO "Countries" (COUNTRY_ID, COUNTRY_NAME) VALUES (10, 'Romania');
INSERT INTO "Jobs" (JOB_ID, JOB_TITLE, MIN_SALARY, MAX_SALARY, BONUSES, SALARY) VALUES (100, 'Goalkeeper', 20000, 100000, 5000, 25000);
INSERT INTO "Players" (PLAYER_ID, LAST_NAME, HIRE_DATE, NET_SALARY, GROSS_SALARY, TEAM_ID, COST_OF_PLAYER, JOB_ID) VALUES
(1, 'Popescu', '01-JAN-2014', 25000, 28000, 100, 100000, 100);
INSERT INTO "Sponsors" (SPONSOR_ID, COMPANY_NAME, MONTHLY_CONTRIBUTION, TEAM_ID, CONTRACT_END_DATE)
VALUES (0, 'Sponsor1', 30, 5555, '01-JUN-2021');
INSERT INTO "Matches" (MATCH_ID, MATCH_DATE, COUNTRY_ID, SPONSOR_ID, PRIZE, MATCH_RESULT)
VALUES (13, '15-JAN-2019', 40, 2, 45000, 'Tie between teams');
INSERT INTO "Team" (TEAM_ID, TEAM_NAME, NUMBER_OF_PLAYERS, YEARLY_SPENDINGS, PROFIT, SPONSOR_ID) VALUES (5555, 'Rangers', 12 , 1000, 20, 0);
```

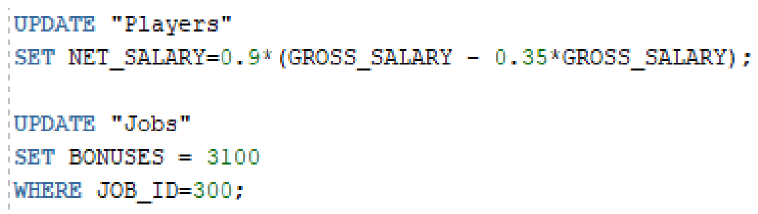
This is how the table Players looks after finishing this section of the project. The other tables were populated too.



	PLAYER_ID	LAST_NAME	HIRE_DATE	NET_SALARY	GROSS_SALARY	TEAM_ID	COST_OF_PLAYER	JOB_ID
1	2	Ionescu	09-JUL-18	53235	91000	5555	100000	200
2	3	Popescu	08-MAR-14	6435	11000	5555	100000	300
3	4	Paun	21-JUN-16	5733	9800	5555	100000	400
4	5	Ion	09-JUL-18	53235	91000	6666	100000	200
5	6	Andrei	19-NOV-19	5324	9100	6666	100000	400
6	1	Popescu	01-JAN-14	16380	28000	5555	100000	100
7	7	Lupsan Sabrina	08-MAY-18	5324	9100	5555	100000	400

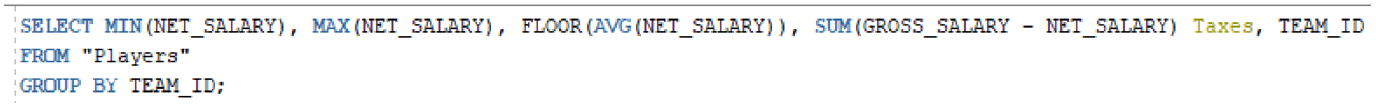
2. Update the net salary of each player by deducting social (25%) and health (10%) insurance and the income tax (10%) from the gross salary.

3. Set the bonuses field to 3100 for the job with the ID 300.

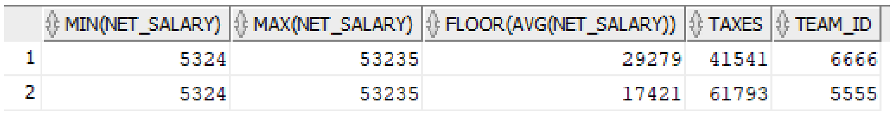

```
UPDATE "Players"
SET NET_SALARY=0.9*(GROSS_SALARY - 0.35*GROSS_SALARY);

UPDATE "Jobs"
SET BONUSES = 3100
WHERE JOB_ID=300;
```

Requirements for the database

1. Display the minimum, maximum and the average salary for all the players in each team and compute the amount of taxes that is being payed from their salaries for each team. For average, round up to the biggest integer smaller than the selected number.

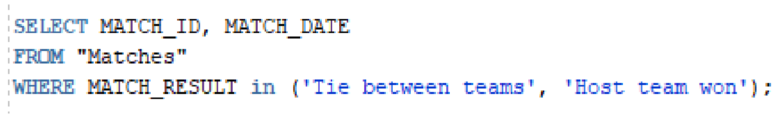


```
SELECT MIN(NET_SALARY), MAX(NET_SALARY), FLOOR(AVG(NET_SALARY)), SUM(GROSS_SALARY - NET_SALARY) Taxes, TEAM_ID
FROM "Players"
GROUP BY TEAM_ID;
```

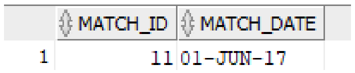


	MIN(NET_SALARY)	MAX(NET_SALARY)	FLOOR(AVG(NET_SALARY))	TAXES	TEAM_ID
1	5324	53235	29279	41541	6666
2	5324	53235	17421	61793	5555

2. Display the match details where the host team got at least 1 point (=>the host team won or there was a tie).

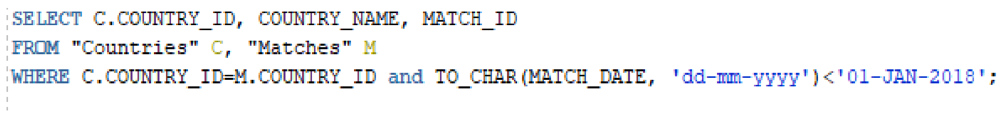


```
SELECT MATCH_ID, MATCH_DATE
FROM "Matches"
WHERE MATCH_RESULT in ('Tie between teams', 'Host team won');
```

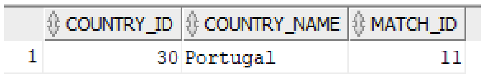


	MATCH_ID	MATCH_DATE
1	11	01-JUN-17

3. Display the country details and the match ID for all the matches that were played before the start of the year 2018.

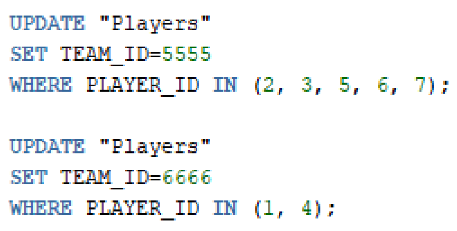


```
SELECT C.COUNTRY_ID, COUNTRY_NAME, MATCH_ID
FROM "Countries" C, "Matches" M
WHERE C.COUNTRY_ID=M.COUNTRY_ID and TO_CHAR(MATCH_DATE, 'dd-mm-yyyy')<'01-JAN-2018';
```

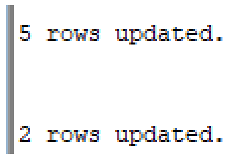


	COUNTRY_ID	COUNTRY_NAME	MATCH_ID
1	30	Portugal	11

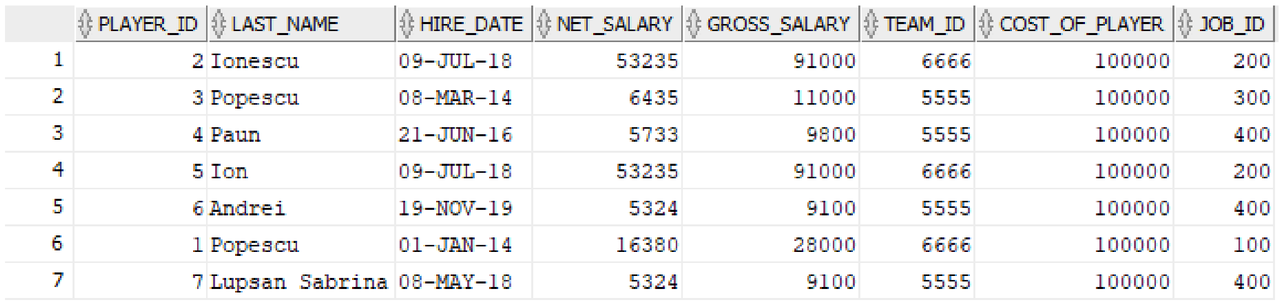
4. Make transfers. Players with IDs 2, 3, 5, 6 and 7 should be in the team with the ID 5555 and the ones with the IDs 1 and 4 in the team with the ID 6666.



```
UPDATE "Players"
SET TEAM_ID=5555
WHERE PLAYER_ID IN (2, 3, 5, 6, 7);

UPDATE "Players"
SET TEAM_ID=6666
WHERE PLAYER_ID IN (1, 4);
```

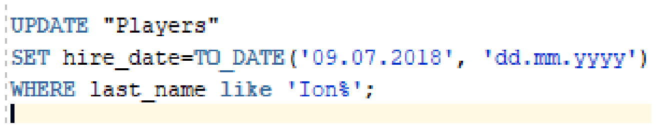


```
5 rows updated.


2 rows updated.
```

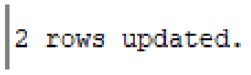


	PLAYER_ID	LAST_NAME	HIRE_DATE	NET_SALARY	GROSS_SALARY	TEAM_ID	COST_OF_PLAYER	JOB_ID
1	2	Ionescu	09-JUL-18	53235	91000	6666	100000	200
2	3	Popescu	08-MAR-14	6435	11000	5555	100000	300
3	4	Paun	21-JUN-16	5733	9800	5555	100000	400
4	5	Ion	09-JUL-18	53235	91000	6666	100000	200
5	6	Andrei	19-NOV-19	5324	9100	5555	100000	400
6	1	Popescu	01-JAN-14	16380	28000	6666	100000	100
7	7	Lupsan Sabrina	08-MAY-18	5324	9100	5555	100000	400

5. Update the table Players by setting the hire date of all the players whose name starts with "Ion" to be the 9th of July 2018.

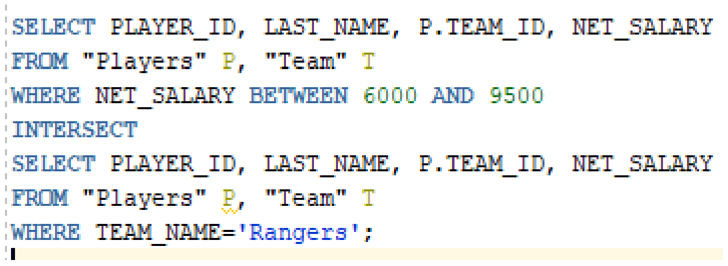


```
UPDATE "Players"
SET hire_date=TO_DATE('09.07.2018', 'dd.mm.yyyy')
WHERE last_name like 'Ion%';
```

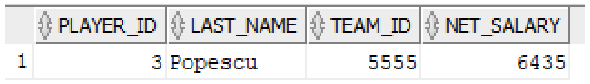


```
2 rows updated.
```

6. Display all the players from the team "Rangers" that have the net salaries no more than 9500 and at least 6000.

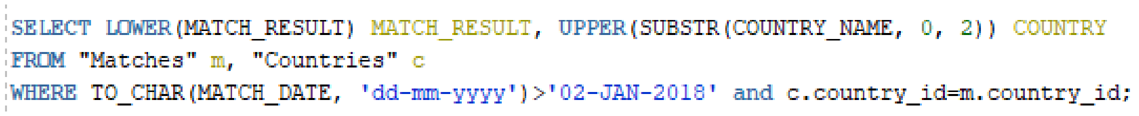


```
SELECT PLAYER_ID, LAST_NAME, P.TEAM_ID, NET_SALARY
FROM "Players" P, "Team" T
WHERE NET_SALARY BETWEEN 6000 AND 9500
INTERSECT
SELECT PLAYER_ID, LAST_NAME, P.TEAM_ID, NET_SALARY
FROM "Players" P, "Team" T
WHERE TEAM_NAME='Rangers';
```

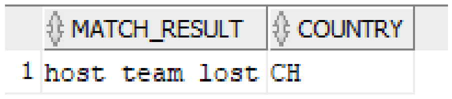


	PLAYER_ID	LAST_NAME	TEAM_ID	NET_SALARY
1	3	Popescu	5555	6435

7. Display the match result and the country's code for the matches played after the 2nd of January 2018.

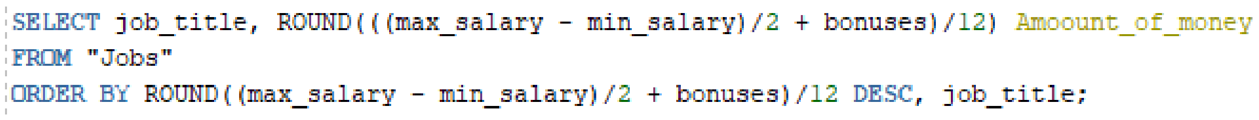


```
SELECT LOWER(MATCH_RESULT) MATCH_RESULT, UPPER(SUBSTR(COUNTRY_NAME, 0, 2)) COUNTRY
FROM "Matches" m, "Countries" c
WHERE TO_CHAR(MATCH_DATE, 'dd-mm-yyyy')>'02-JAN-2018' and c.country_id=m.country_id;
```

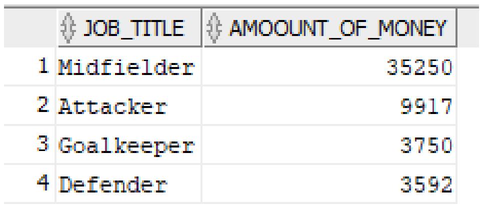


	MATCH_RESULT	COUNTRY
1	host team lost	CH

8. Estimate the average amount of money a future player can win per month and order descending based on that to see which the most profitable career is.

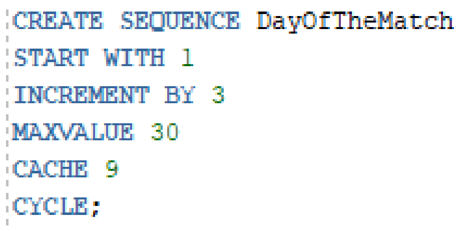


```
SELECT job_title, ROUND(((max_salary - min_salary)/2 + bonuses)/12) Amoount_of_money
FROM "Jobs"
ORDER BY ROUND((max_salary - min_salary)/2 + bonuses)/12 DESC, job_title;
```

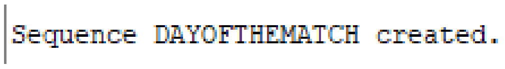


	JOB_TITLE	AMOOUNT_OF_MONEY
1	Midfielder	35250
2	Attacker	9917
3	Goalkeeper	3750
4	Defender	3592

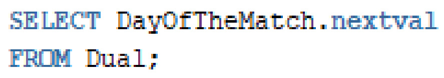
9. Create a sequence that generates the day of the month in which the next matches will be played. Every match is played once every 3 days. Suppose every month's length is 30 days. Then, display the day of the first game of the year.



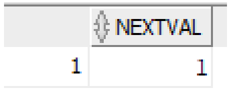
```
CREATE SEQUENCE DayOfTheMatch
START WITH 1
INCREMENT BY 3
MAXVALUE 30
CACHE 9
CYCLE;
```



```
Sequence DAYOFTHEMATCH created.
```

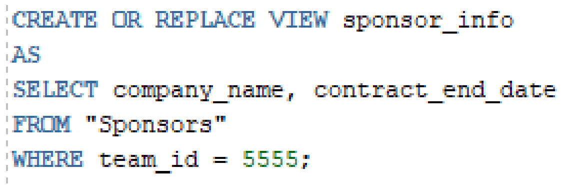


```
SELECT DayOfTheMatch.nextval
FROM Dual;
```

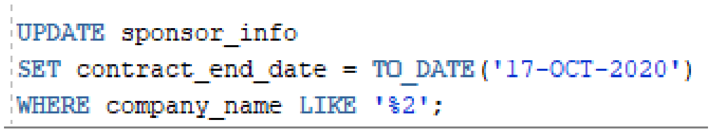


	NEXTVAL
1	1

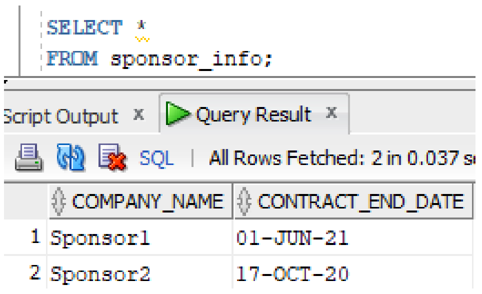
10. Create a view of the sponsors of the team with the ID = 5555. Then, change the contract of 'Sponsor2' to end on the 17th of October 2020. Display the final view.



```
CREATE OR REPLACE VIEW sponsor_info
AS
SELECT company_name, contract_end_date
FROM "Sponsors"
WHERE team_id = 5555;
```



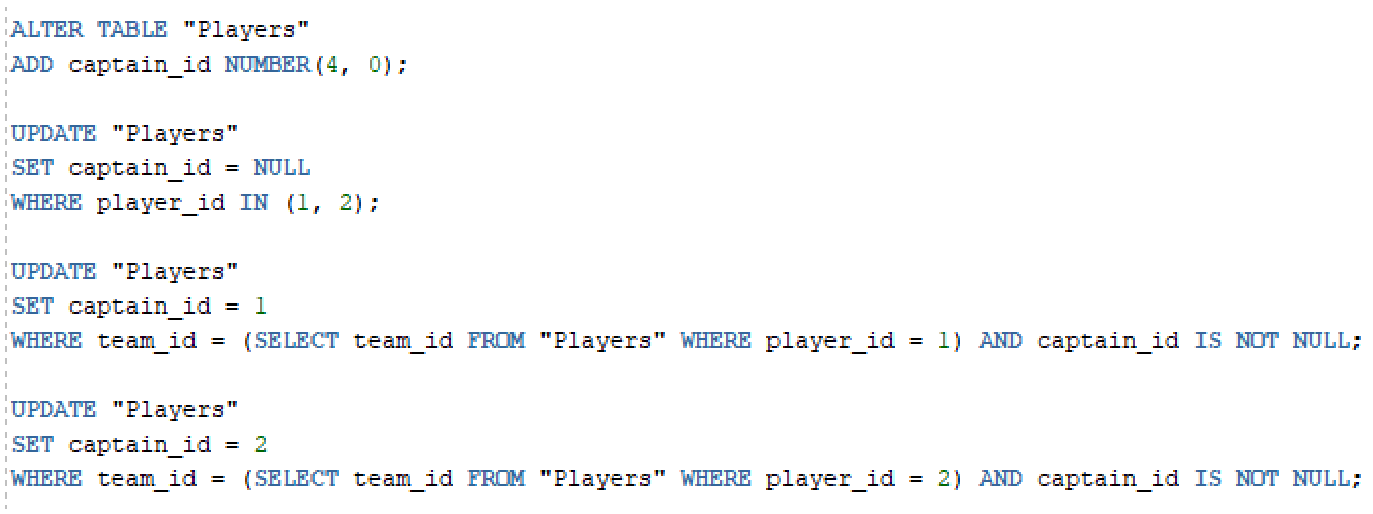
```
UPDATE sponsor_info
SET contract_end_date = TO_DATE('17-OCT-2020')
WHERE company_name LIKE '%2';
```



```
SELECT *
FROM sponsor_info;
```

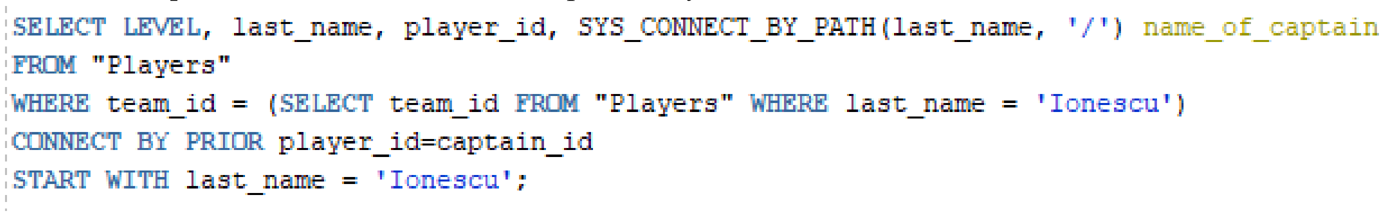
	COMPANY_NAME	CONTRACT_END_DATE
1	Sponsor1	01-JUN-21
2	Sponsor2	17-OCT-20

11. Add a new column in the table Players with the captain_id. There is only one captain for every team. The captain for the team with the ID 6666 is the player with the ID 1 and the captain for the team with the ID 5555 is the player with the ID 2.

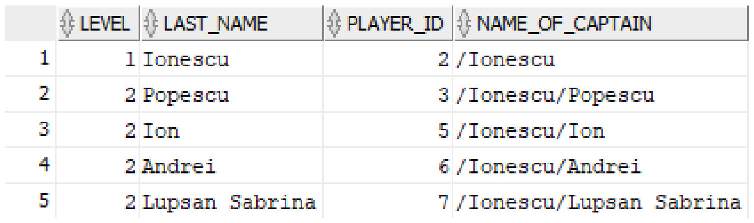


```
ALTER TABLE "Players"
ADD captain_id NUMBER(4, 0);

UPDATE "Players"
SET captain_id = NULL
WHERE player_id IN (1, 2);

UPDATE "Players"
SET captain_id = 1
WHERE team_id = (SELECT team_id FROM "Players" WHERE player_id = 1) AND captain_id IS NOT NULL;

UPDATE "Players"
SET captain_id = 2
WHERE team_id = (SELECT team_id FROM "Players" WHERE player_id = 2) AND captain_id IS NOT NULL;
```

12. Display the level, last name, player ID of all the players (inferiors) with the captain whose last name is 'Ionescu', captain's name and their name separated by a '/'.

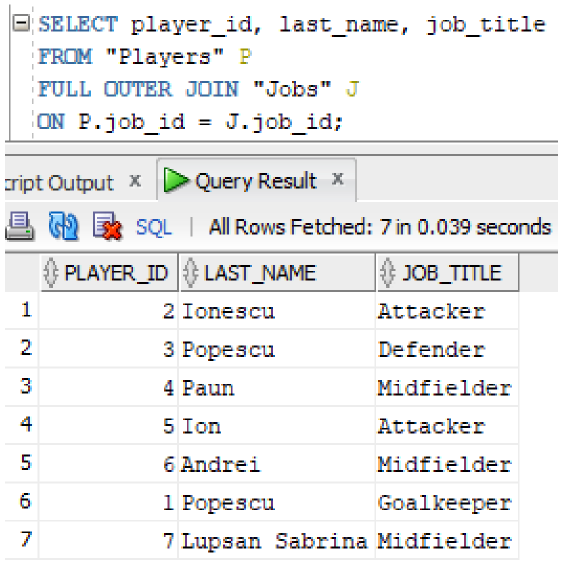


```
SELECT LEVEL, last_name, player_id, SYS_CONNECT_BY_PATH(last_name, '/') name_of_captain
FROM "Players"
WHERE team_id = (SELECT team_id FROM "Players" WHERE last_name = 'Ionescu')
CONNECT BY PRIOR player_id=captain_id
START WITH last_name = 'Ionescu';
```



	LEVEL	LAST_NAME	PLAYER_ID	NAME_OF_CAPTAIN
1	1	Ionescu	2	/Ionescu
2	2	Popescu	3	/Ionescu/Popescu
3	2	Ion	5	/Ionescu/Ion
4	2	Andrei	6	/Ionescu/Andrei
5	2	Lupsan Sabrina	7	/Ionescu/Lupsan Sabrina

13. Display all the players who have an existent job ID (=> all the players, because that column is not nullable, since it is the primary key for the Jobs table).

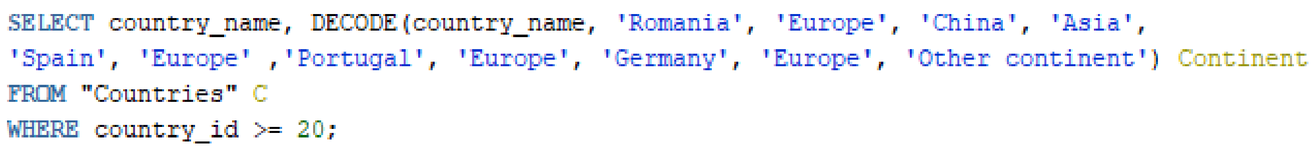


```
SELECT player_id, last_name, job_title
FROM "Players" P
FULL OUTER JOIN "Jobs" J
ON P.job_id = J.job_id;
```

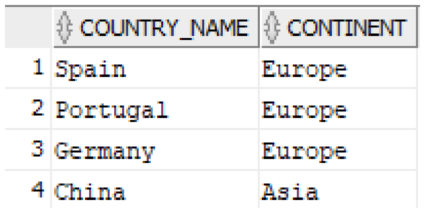
All Rows Fetched: 7 in 0.039 seconds

	PLAYER_ID	LAST_NAME	JOB_TITLE
1	2	Ionescu	Attacker
2	3	Popescu	Defender
3	4	Paun	Midfielder
4	5	Ion	Attacker
5	6	Andrei	Midfielder
6	1	Popescu	Goalkeeper
7	7	Lupsan Sabrina	Midfielder

14. Display the countries and their continents for the countries with the ID bigger or equal than 20.

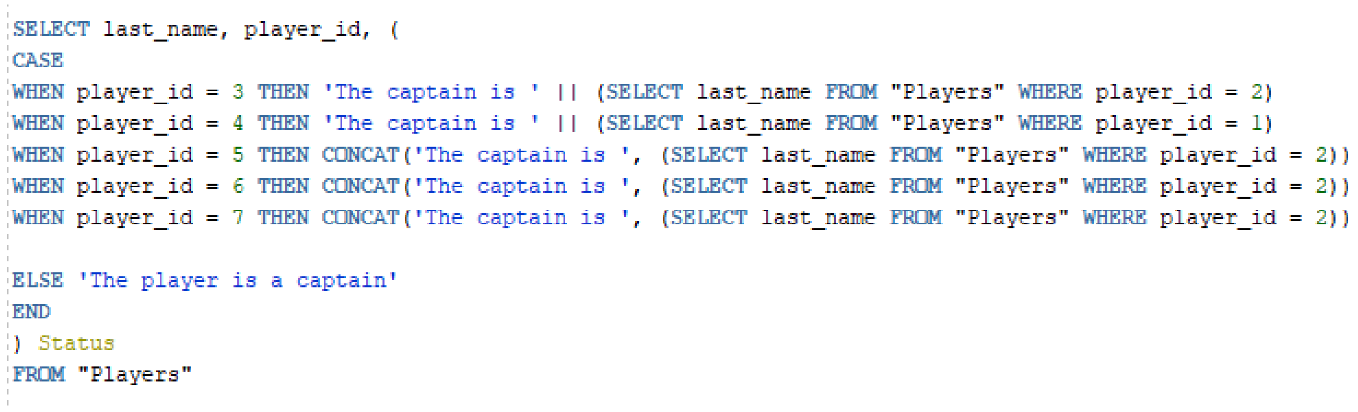


```
SELECT country_name, DECODE(country_name, 'Romania', 'Europe', 'China', 'Asia',
'Spain', 'Europe' ,'Portugal', 'Europe', 'Germany', 'Europe', 'Other continent') Continent
FROM "Countries" C
WHERE country_id >= 20;
```

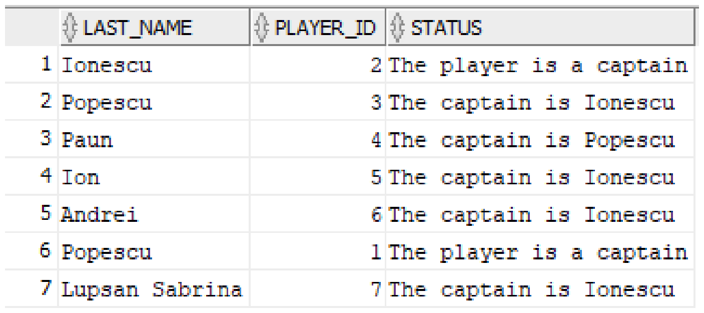


	COUNTRY_NAME	CONTINENT
1	Spain	Europe
2	Portugal	Europe
3	Germany	Europe
4	China	Asia

15. Display all of the players and their captain using the CASE function. If they don't have a captain, their status is that they are a captain.

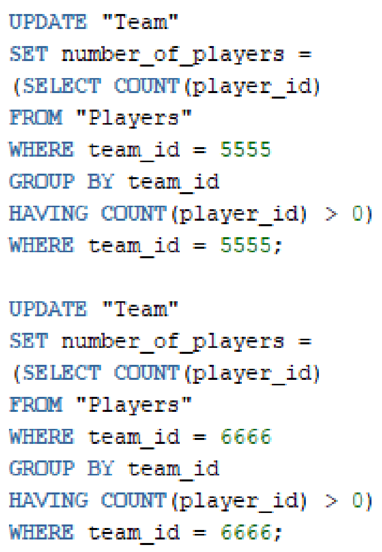


```
SELECT last_name, player_id, (
CASE
WHEN player_id = 3 THEN 'The captain is ' || (SELECT last_name FROM "Players" WHERE player_id = 2)
WHEN player_id = 4 THEN 'The captain is ' || (SELECT last_name FROM "Players" WHERE player_id = 1)
WHEN player_id = 5 THEN CONCAT('The captain is ', (SELECT last_name FROM "Players" WHERE player_id = 2))
WHEN player_id = 6 THEN CONCAT('The captain is ', (SELECT last_name FROM "Players" WHERE player_id = 2))
WHEN player_id = 7 THEN CONCAT('The captain is ', (SELECT last_name FROM "Players" WHERE player_id = 2))

ELSE 'The player is a captain'
END
) Status
FROM "Players"
```

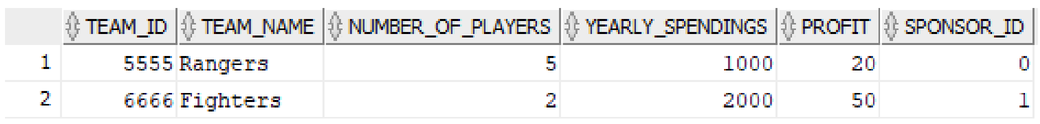


	LAST_NAME	PLAYER_ID	STATUS
1	Ionescu	2	The player is a captain
2	Popescu	3	The captain is Ionescu
3	Paun	4	The captain is Popescu
4	Ion	5	The captain is Ionescu
5	Andrei	6	The captain is Ionescu
6	Popescu	1	The player is a captain
7	Lupsan Sabrina	7	The captain is Ionescu

16. Update the table Team with the real number of players currently in the database.

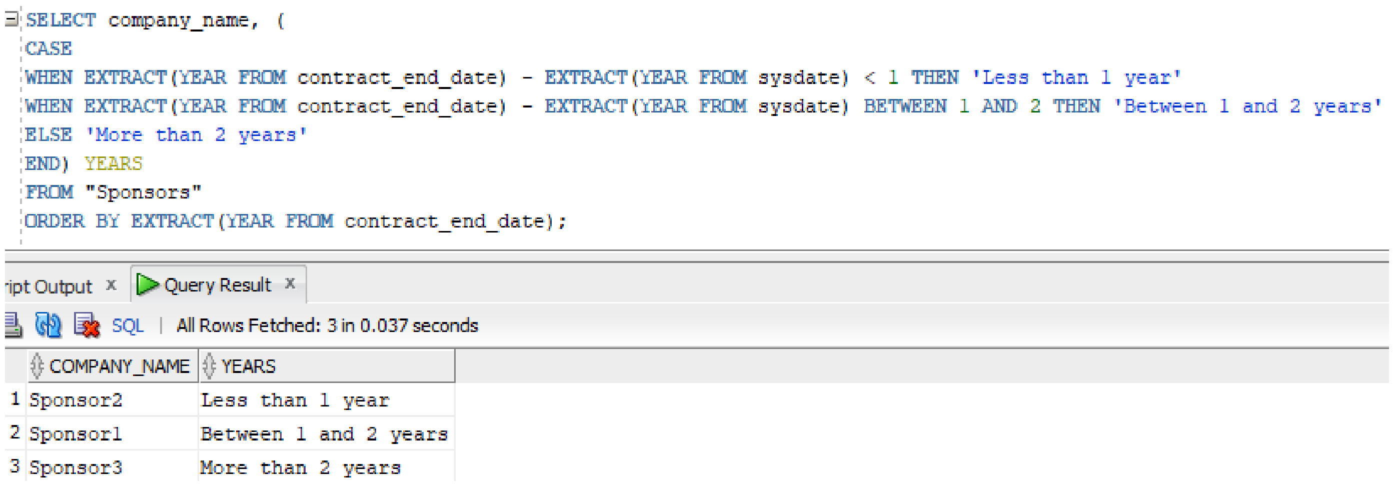


```
UPDATE "Team"
SET number_of_players =
(SELECT COUNT(player_id)
FROM "Players"
WHERE team_id = 5555
GROUP BY team_id
HAVING COUNT(player_id) > 0)
WHERE team_id = 5555;

UPDATE "Team"
SET number_of_players =
(SELECT COUNT(player_id)
FROM "Players"
WHERE team_id = 6666
GROUP BY team_id
HAVING COUNT(player_id) > 0)
WHERE team_id = 6666;
```



	TEAM_ID	TEAM_NAME	NUMBER_OF_PLAYERS	YEARLY_SPENDINGS	PROFIT	SPONSOR_ID
1	5555	Rangers	5	1000	20	0
2	6666	Fighters	2	2000	50	1

17. Display in how much time the contract of every sponsor will expire, in intervals of: <1 year, 1-2 years, > 2 years.



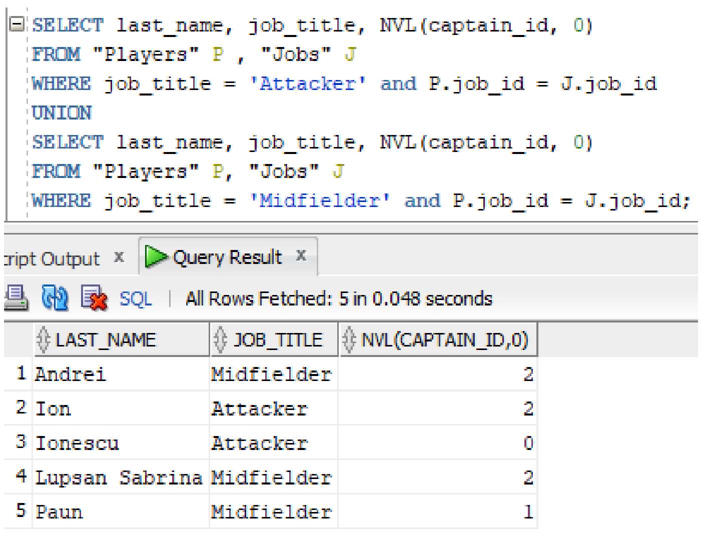
```
SELECT company_name, (
CASE
WHEN EXTRACT(YEAR FROM contract_end_date) - EXTRACT(YEAR FROM sysdate) < 1 THEN 'Less than 1 year'
WHEN EXTRACT(YEAR FROM contract_end_date) - EXTRACT(YEAR FROM sysdate) BETWEEN 1 AND 2 THEN 'Between 1 and 2 years'
ELSE 'More than 2 years'
END) YEARS
FROM "Sponsors"
ORDER BY EXTRACT(YEAR FROM contract_end_date);
```

ipt Output × Query Result ×

SQL | All Rows Fetched: 3 in 0.037 seconds

	COMPANY_NAME	YEARS
1	Sponsor2	Less than 1 year
2	Sponsor1	Between 1 and 2 years
3	Sponsor3	More than 2 years

18. Display the last name of all the players that have the job title Attacker or Midfielder, their job title and their captain's ID (if they are a captain, the captain_id = 0).

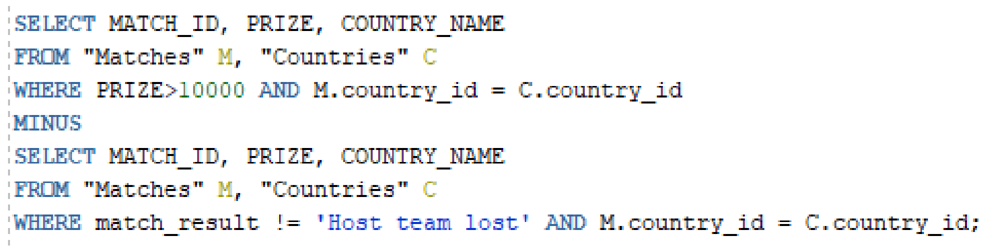


```sql
SELECT last_name, job_title, NVL(captain_id, 0)
FROM "Players" P , "Jobs" J
WHERE job_title = 'Attacker' and P.job_id = J.job_id
UNION
SELECT last_name, job_title, NVL(captain_id, 0)
FROM "Players" P, "Jobs" J
WHERE job_title = 'Midfielder' and P.job_id = J.job_id;
```

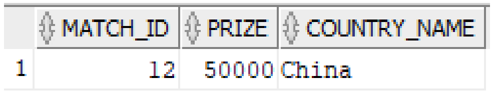
All Rows Fetched: 5 in 0.048 seconds

	LAST_NAME	JOB_TITLE	NVL(CAPTAIN_ID,0)
1	Andrei	Midfielder	2
2	Ion	Attacker	2
3	Ionescu	Attacker	0
4	Lupsan Sabrina	Midfielder	2
5	Paun	Midfielder	1

19. Display the matches with the prize bigger than 10.000 dollars but exclude the ones in which the guest team did not win or they tied.

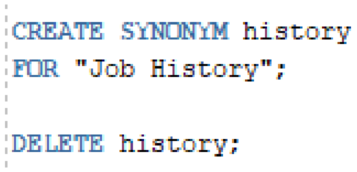


```sql
SELECT MATCH_ID, PRIZE, COUNTRY_NAME
FROM "Matches" M, "Countries" C
WHERE PRIZE>10000 AND M.country_id = C.country_id
MINUS
SELECT MATCH_ID, PRIZE, COUNTRY_NAME
FROM "Matches" M, "Countries" C
WHERE match_result != 'Host team lost' AND M.country_id = C.country_id;
```

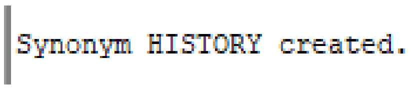


	MATCH_ID	PRIZE	COUNTRY_NAME
1	12	50000	China

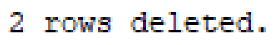
20. Create a synonym for the table “Job history” and then delete all the rows without deleting the table.



```sql
CREATE SYNONYM history
FOR "Job History";

DELETE history;
```

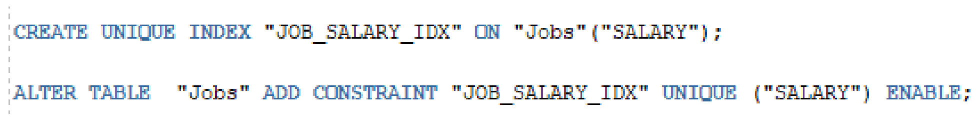


```
Synonym HISTORY created.
```

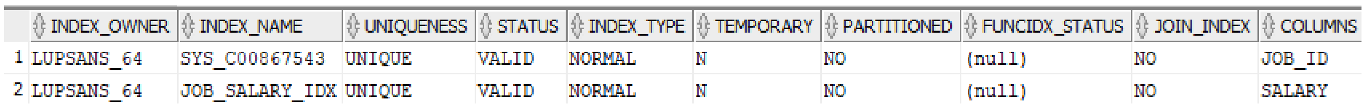


```
2 rows deleted.
```

21. Create a unique index for the column “Salary” from the table “Jobs”.



```sql
CREATE UNIQUE INDEX "JOB_SALARY_IDX" ON "Jobs"("SALARY");

ALTER TABLE  "Jobs" ADD CONSTRAINT "JOB_SALARY_IDX" UNIQUE ("SALARY") ENABLE;
```



	INDEX_OWNER	INDEX_NAME	UNIQUENESS	STATUS	INDEX_TYPE	TEMPORARY	PARTITIONED	FUNCIDX_STATUS	JOIN_INDEX	COLUMNS
1	LUPSANS_64	SYS_C00867543	UNIQUE	VALID	NORMAL	N	NO	(null)	NO	JOB_ID
2	LUPSANS_64	JOB_SALARY_IDX	UNIQUE	VALID	NORMAL	N	NO	(null)	NO	SALARY