

2022/2023 FIS Alpine Ski World Cup

Contents

1	Introduction	1
2	Entity-Relationship model and Relational model	2
2.1	Entity-Relationship model	2
2.2	Relational model	3
3	Table creation and entries	5
3.1	Table creation	5
3.2	Entries	6
4	Queries	8
4.1	Simple queries	8
4.2	Multiple table queries	10
4.3	Queries with aggregating functions	12
4.4	Subqueries, nested queries, set operations	13
5	Default values, conditions and comments	16
5.1	Default values	16
5.2	Conditions	16
5.3	Comments	16
6	Indexes	17
7	Procedures	18
8	Triggers	20
9	Conclusion	22

1 Introduction

The topic of this paper will be the creation of a database for the 2022/2023 FIS Alpine Ski World Cup. The database itself will include data exclusively for the Ski World Cup in the previous season. We will start by creating a model based on which we will define tables and make data entry. After that, we'll look at different queries depending on their complexity, and we'll also add some conditions and defaults. Ultimately we will define procedures and triggers. The reason for choosing this topic is that skiing is an interesting and dynamic sport that is often not sufficiently covered by the media. For this reason, it is often not possible to find some information that might be of interest to people who actively follow this sport. For the sake of simplicity all table and attribute names both while modeling tables and in code were left in Croatian but they will be translated, explained and discussed in English.

2 Entity-Relationship model and Relational model

At the beginning of this chapter, we will deal with the entity and relationship model (hereafter ER model). We will explain its structure in detail and state which entities it consists of and what the connections are between them. After that, we will explain how to move from the ER model to the relational model.

2.1 Entity-Relationship model

The aim of this paper is on the study of the 2022/2023 FIS Alpine Ski World Cup, so we need to create appropriate databases accordingly. On Figure 1 we can see the ER model of the database which we will create. The most important entities we have are *Skijas* (Skier) and *Utrka* (Race), which are then linked to some other entities. If we first look at the entity *Skijas*, we can see that it consists of 6 attributes that contain basic information about an individual skier, such as first name, last name, date of birth, etc. Of course, skiers own skis, but they can, but also they don't have to have multiple pairs. For the skis themselves, it is important for us to know who the manufacturer is, and we will store that information in the entity *Skije* (Skis). We apply the same logic when creating the entity *Sponzor* (Sponsor), and it is important to point out that this entity contains the names of individual sponsors of skiers. Namely, the skiers who compete in the FIS Alpine World Cup are members of national teams that themselves have specific sponsors, so for the sake of simplicity, we decided to include different companies in this entity that sponsor only some of the skiers.

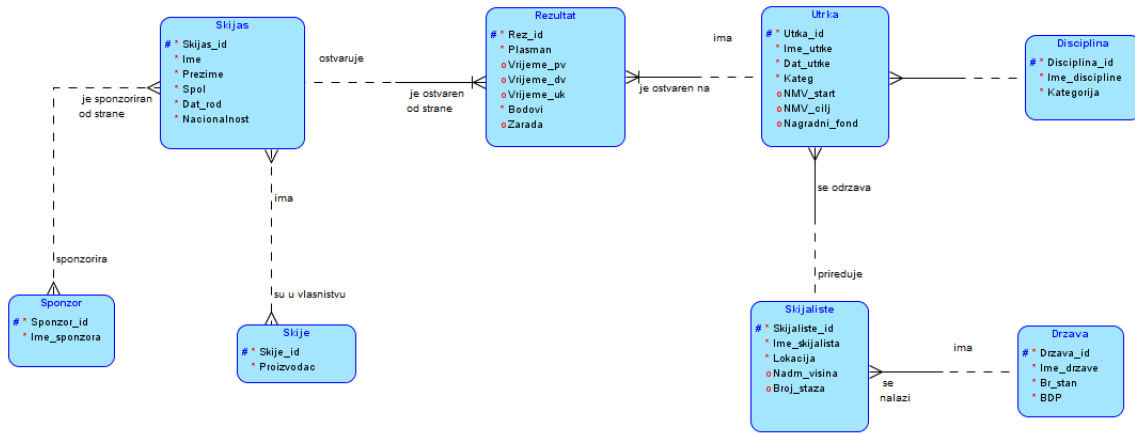


Figure 1: ER model for database used in paper.

As its name suggests, the entity *Utrka* (Race) contains the most important information about each of the races, where it should be emphasized that the attribute *kategorija* (category) contains information about whether the race was held in men's or women's competition. For each race, it is important to know which skiing discipline it belongs to, as well as the ski resort where it was held. For this purpose, we created the entities *Disciplina* (Discipline) and *Skijasliste* (Ski Resort). For the discipline, it is important for us to know its name, but also the category, given that we have speed and technical disciplines. The entity *Skijasliste* (Ski Resort) contains some geographical characteristics of the places where the Alpine Ski World Cup races were held. Note that in this case the *lokacija* (location) attribute provides information on which mountain or mountain range a ski resort is located on. Of course, it is also important to know in which country a ski resort is located, so for that purpose we have created an entity called *Drzava* (Country)

which, in addition to the name, will also contain the number of inhabitants and the GDP of the country hosting the ski races. Let's denote that in the entity *Utrka* (Race) we omitted data about the end time of the race. This information is almost never known because ski races often continue after the end of television broadcasts, and in official records, more attention is given to the start time of the race. Let's now focus on the connections between the entities that are related to the race. Each race is held at one ski resort, and one ski resort may or may not "host" multiple races. Each ski resort is located in one country, while a country may or may not have several ski resorts. If we look at individual races, each one is held in one discipline, but several races can be held in some discipline throughout the season. The only entity we haven't mentioned yet is *Rezultat* (Result). Since skiers compete in multiple races and races are skied by multiple skiers, there would be N:M connection between these entities (Result and Skier). In order to simplify things, but also to find a way to store information about race results, we created the previously mentioned entity, which will convert a many-to-many relationship into two one-to-many relationships. Therefore, the entity *Rezultat* (Result) will contain information about the result of an individual skier in one of the races, such as placement, times of the first and second run, total time, points won and money. Each result belongs to exactly one skier, and skiers may or may not achieve more than one result in a season. Likewise, each result was achieved in one race, but a race can have more results.

2.2 Relational model

Following the conversion rules, we transfer the ER model to the relational model shown in Figure 2. Since one-to-many relationships prevail in the ER model, they are displayed as foreign keys in the relational model. So, the entity *Skijaliste* (Ski Resort) will contain the foreign key *drzava_id* (country_id), and the entity *Utrka* (Race) will contain the foreign keys *skijaliste_id* (skiresort_id) and *disciplina_id* (discipline_id). Likewise, the entity *Rezultat* (Result) will contain foreign keys *utrka_id* (race_id) and *skijas_id* (skier_id).

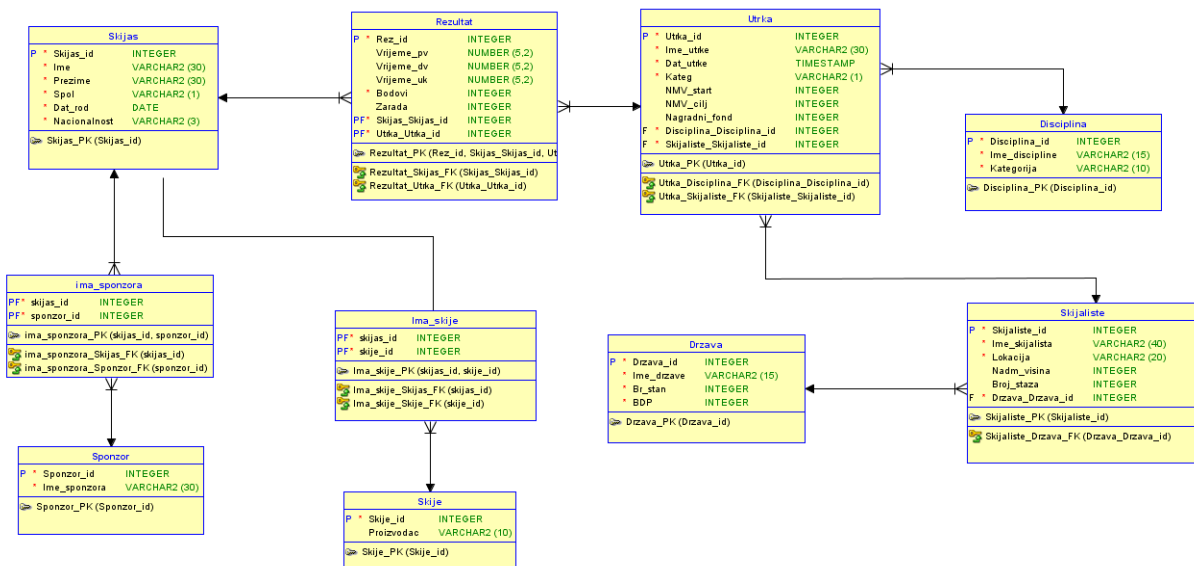


Figure 2: Relational model for the database used in paper.

We have seen that between the entities *Skijas* (Skier) and *Sponzor* (Sponsor) and *Skijas* (Skier) and *Skije* (Skis) we have many-to-many connections. In order to transfer them successfully,

they are converted into two more tables. *Ima_sponzora* (Has_sponsor) takes skijas_id (skier_id) and sponzor_id (sponsor_id) as primary and foreign keys of that table. In the same way skijas_id (skier_id) and skije_id (skis_id) are the primary and foreign keys of the table *Ima_skije* (Has_skis).

3 Table creation and entries

In this chapter, we emphasize the creation of tables that we presented in the ER model, and then in the relational model. After we have created all the tables, paying attention to the keys, we will fill them with data.

3.1 Table creation

Given that we have a large number of tables, we will show how we created only some of them. Below we can see a part of the code which is used to delete the tables.

```
1 drop table SKIJAS CASCADE CONSTRAINT;  
2 drop table SKIJE CASCADE CONSTRAINT;  
3 drop table IMA_SKIJE CASCADE CONSTRAINT;  
4 drop table SPONZOR CASCADE CONSTRAINT;  
5 drop table IMA_SPONZORA CASCADE CONSTRAINT;  
6 drop table DISCIPLINA CASCADE CONSTRAINT;  
7 drop table DRZAVA CASCADE CONSTRAINT;  
8 drop table SKIJALISTE CASCADE CONSTRAINT;  
9 drop table UTRKA CASCADE CONSTRAINT;  
10 drop table REZULTAT CASCADE CONSTRAINT;
```

Listing 1: Primjer koda za brisanje tablica koje ćemo kreirati.

Since *Skijas* (Skier) and *Rezultat* (Result) tables will contain the largest number of entries, let's see how the code for their creation looks like.

```
1 create table SKIJAS (  
2     SKIJAS_ID INTEGER CONSTRAINT SKIJAS_ID_PK PRIMARY KEY,  
3     IME VARCHAR2(30) NOT NULL,  
4     PREZIME VARCHAR2(30) NOT NULL,  
5     SPOL VARCHAR2(1) NOT NULL,  
6     DAT_ROD DATE NOT NULL,  
7     NACIONALNOST VARCHAR2(3) NOT NULL  
8 );
```

Listing 2: Kod korišten za kreiranje tablice Skijas.

```
1 CREATE TABLE REZULTAT (  
2     REZ_ID INTEGER CONSTRAINT REZ_ID_PK PRIMARY KEY,  
3     PLASMAN INTEGER NOT NULL,  
4     VRIJEME_PV NUMBER(5,2),  
5     VRIJEME_DV NUMBER(5,2),  
6     VRIJEME_UK NUMBER(5,2),  
7     BODOVI INTEGER NOT NULL,  
8     ZARADA INTEGER,  
9     SKIJAS_ID INTEGER NOT NULL CONSTRAINT REZULTAT_SKIJAS_FK REFERENCES "  
10    SKIJAS" (SKIJAS_ID),  
11    UTRKA_ID INTEGER NOT NULL CONSTRAINT REZULTAT_UTRKA_FK REFERENCES "  
    UTRKA" (UTRKA_ID)  
);
```

Listing 3: Kod korišten za kreiranje tablice Rezultat.

3.2 Entries

After creating all tables, it is necessary to enter the data. To begin with, we will look at how the entries in the *Utrka* (Race) table look like. Since the ski season in the World Cup consists of more than 70 races in the men's and women's competition, for the sake of simplicity, we entered 10 races in the men's competition and 10 races in the women's competition into the database. We can see some of the entries in the code shown below.

```
1 insert into UTRKA values (001, 'Soelden Ski World Cup', TO_TIMESTAMP('
    23-10-2022 10:00:00', 'dd-MM-yyyy HH24:MI:SS'), 'M', 3040, 2670,
    132000, 4, 1);
2 insert into utrka values (002, 'The Killington Cup', TO_TIMESTAMP('
    26-11-2022 16:00:00', 'dd-MM-yyyy HH24:MI:SS'), 'F', 1066, 782, 132000,
    4, 3);
3 insert into UTRKA values (003, 'Xfinity Birds of Pray', TO_TIMESTAMP('
    04-12-2022 18:00:00', 'dd-MM-yyyy HH24:MI:SS'), 'M', 3337, 2730,
    132000, 2, 4);
4 insert into utrka values (004, 'Lake Louise Alpine World Cup', TO_DATE('
    02-12-2022 20:00:00', 'dd-MM-yyyy HH24:MI:SS'), 'F', 2469, 1680,
    132000, 1, 2);
5 insert into utrka values (005, '55. Saslong Classic', TO_TIMESTAMP('
    17-12-2022 11:45:00', 'dd-MM-yyyy HH24:MI:SS'), 'M', 2249, 1410,
    132000, 1, 5);
```

Listing 4: Prikaz koda korištenog za unos vrijednosti u tablicu Utrka.

Let's now focus on the *Skijas* (Skier) table. As we mentioned before, we entered data for only 20 races from the entire season. As a result, the base will not contain all skiers who competed in the 2022/2023 Alpine Ski World Cup season, but only those who achieved a placement that potentially brings points in selected races. For speed events, this means that the data for first 30 skiers from each race was selected since they earn points. On the other hand, in the technical events, we selected those skiers who entered the second run (first 30 skiers after the first run), which meant potentially winning points, because it can happen that a skier for some reason does not finish the second run or achieves too much of a backlog and thus remains without points. The following code shows us several entries of male skiers and several entries of female skiers.

```
1 insert into SKIJAS values (001, 'Marco', 'Odermatt', 'M', TO_DATE('
    08-10-1997', 'dd-MM-yyyy'), 'SUI');
2 insert into SKIJAS values (002, 'Aleksander', 'Aamodt Kilde', 'M', TO_DATE(
    '21-09-1992', 'dd-MM-yyyy'), 'NOR');
3 insert into SKIJAS values (003, 'Henrik', 'Kristoffersen', 'M', TO_DATE('
    02-07-1994', 'dd-MM-yyyy'), 'NOR');
4 insert into SKIJAS values (004, 'Marco', 'Schwarz', 'M', TO_DATE('
    16-08-1995', 'dd-MM-yyyy'), 'AUT');
5 insert into SKIJAS values (005, 'Alexis', 'Pinturault', 'M', TO_DATE('
    20-03-1991', 'dd-MM-yyyy'), 'FRA');
6
7 insert into SKIJAS values (128, 'Mikaela', 'Shiffrin', 'F', TO_DATE('
    13-03-1995', 'dd-MM-yyyy'), 'USA');
8 insert into SKIJAS values (129, 'Lara', 'Gut-Behrami', 'F', TO_DATE('
    27-04-1991', 'dd-MM-yyyy'), 'SUI');
9 insert into SKIJAS values (130, 'Federica', 'Brignone', 'F', TO_DATE('
    14-07-1990', 'dd-MM-yyyy'), 'ITA');
10 insert into SKIJAS values (131, 'Petra', 'Vlhova', 'F', TO_DATE('13-06-1995
    ', 'dd-MM-yyyy'), 'SVK');
11 insert into SKIJAS values (132, 'Ragnhild', 'Mowinckel', 'F', TO_DATE('
```



```
12-09-1992', 'dd-MM-yyyy'), 'NOR');
```

Listing 5: Prikaz koda korištenog za unos vrijednosti u tablicu Skijas.

The database with the most entries is undoubtedly the one containing the race results. We will fill it in according to the criteria we specified when entering skiers. So, if the race was held in a speed discipline (i.e. downhill or super-G), we take the first 30 of the results, and if it is a technical discipline (i.e. slalom or giant slalom), we will be interested in those who placed within first 30 places in first run. Let's note that in the race that was held as part of the Alpine Ski World Ski Cup finals, we also included data for female skiers who did not manage to finish the first run. We did this solely because the right to perform in that race is achieved by only 25 skiers, which in itself is a smaller number of data than usual. Another note that should be highlighted is that in speed disciplines only one run is held and that time is counted as the total time achieved by the skiers, while the time of the second run is left as a NULL value since it was not recorded. We will do the same thing if the skiers in the technical disciplines did not finish the first/second run.

```
1 -- KVITFJELL (super g)
2 insert into rezultat values (1901, 1, 92.36, NULL, 92.36, 100, 50000, 211,
   019);
3 insert into rezultat values (1902, 2, 92.65, NULL, 92.65, 80, 22000, 134,
   019);
4 insert into rezultat values (1903, 3, 92.77, NULL, 92.77, 60, 11000, 139,
   019);
5
6 -- SOELDEN (veleslalom)
7 insert into REZULTAT values (0101, 1, 59.88, 64.84, 124.72, 100, 50000,
   001, 001);
8 insert into REZULTAT values (0102, 2, 60.57, 64.91, 125.48, 80, 22000,
   011, 001);
9 insert into REZULTAT values (0103, 3, 60.83, 64.86, 125.69, 60, 11000,
   003, 001);
10
11 -- ADELBODEN (veleslalom)
12 insert into rezultat values (1028, 0, 80.43, NULL, NULL, 0, 0, 093, 010);
13 insert into rezultat values (1029, 0, 78.47, NULL, NULL, 0, 0, 082, 010);
14 insert into rezultat values (1030, 0, 80.07, NULL, NULL, 0, 0, 094, 010);
```

Listing 6: Prikaz koda korištenog za unos vrijednosti u tablicu Rezultat.

In the code above, we can see how we entered the results while adhering to all the rules we set when creating the database. In the end, let's point out that 0 in the ranking will indicate that the skier did not achieve a ranking, which we will mostly associate with the fact that they didn't finish that race. The other databases did not contain any additional parameters and notes, so we will not list the codes used to create them.

4 Queries

In the previous chapters, we successfully created databases and then made entries in them. This chapter is dedicated to queries over all created databases. We will start from the simplest queries to the more complex ones.

4.1 Simple queries

To begin with, let's list a few of the simplest queries and the results they will give. Suppose we are interested in the name, last name, date of birth and nationality of all skiers sorted alphabetically by last name. The code with the query and the result can be seen below.

```
1 select IME, PREZIME, DAT_ROD, NACIONALNOST from SKIJAS
2 order by PREZIME asc;
```

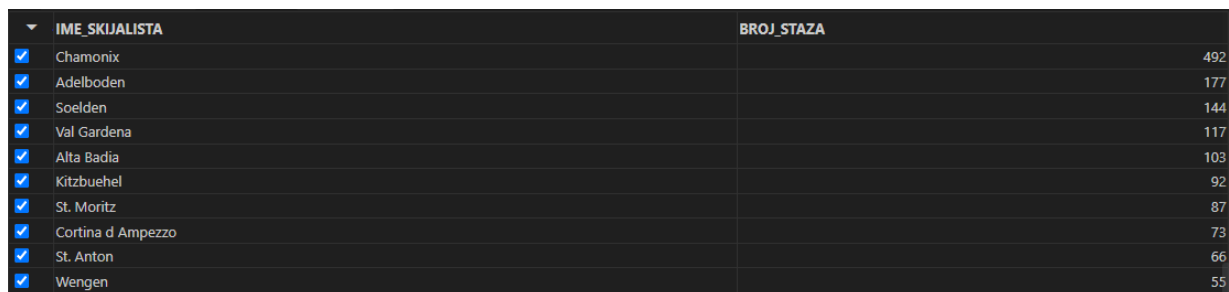


IME	PREZIME	DAT_ROD	NACIONALNOST
Aleksander	Aamodt Kilde	21-SEP-92	NOR
Luca	Aerni	27-MAR-93	SUI
Christina	Ager	11-NOV-95	AUT
Emma	Aicher	13-NOV-03	GER
Cameron	Alexander	31-MAY-97	CAN
Nils	Allegre	02-JAN-94	FRA
Estelle	Alphand	23-APR-95	SWE
Leo	Anguenot	25-AUG-98	FRA
Hanna	Aronsson Elfman	29-DEC-02	SWE
Erik	Arvidsson	03-SEP-96	USA

Figure 3: Output of the first simple query.

Next, let's say that we want to know the names of the ski resorts where the races were held, as well as the number of pistes they have available, and that they are located in the Alps. We will additionally sort them by the number of tracks starting with those with the highest number of tracks.

```
1 select IME_SKIJALISTA, BROJ_STAZA from SKIJALISTE
2 where LOKACIJA = 'Alpe'
3 order by BROJ_STAZA desc;
```



IME_SKIJALISTA	BROJ_STAZA
Chamonix	492
Adelboden	177
Soelden	144
Val Gardena	117
Alta Badia	103
Kitzbuehel	92
St. Moritz	87
Cortina d Ampezzo	73
St. Anton	66
Wengen	55

Figure 4: Output of the second simple query.

Now let's assume that we are interested in a some form of a race calendar, i.e. we are interested in names, dates and times and categories of races arranged chronologically.

```

1 select IME_UTRKE, DAT_UTRKE, KATEG from UTRKA
2 order by DAT_UTRKE asc;

```

IME_UTRKE	DAT_UTRKE	KATEG
<input checked="" type="checkbox"/> Soelden Ski World Cup	23-OCT-22 10.00.00.000000000 AM	M
<input checked="" type="checkbox"/> The Killington Cup	26-NOV-22 04.00.00.000000000 PM	F
<input checked="" type="checkbox"/> Lake Louise Alpine World Cup	02-DEC-22 08.00.00.000000000 PM	F
<input checked="" type="checkbox"/> Xfinity Birds of Prey	04-DEC-22 06.00.00.000000000 PM	M
<input checked="" type="checkbox"/> Alpine Skiing Competitions	17-DEC-22 10.30.00.000000000 AM	F
<input checked="" type="checkbox"/> 55. Saslong Classic	17-DEC-22 11.45.00.000000000 AM	M
<input checked="" type="checkbox"/> Audi FIS Ski World Cup	18-DEC-22 10.00.00.000000000 AM	M
<input checked="" type="checkbox"/> Snow Queen Trophy 2023	04-JAN-23 12.30.00.000000000 PM	F
<input checked="" type="checkbox"/> Ski Welt Cup Garmisch	04-JAN-23 03.40.00.000000000 PM	M
<input checked="" type="checkbox"/> Alpine Skiing Competitions	07-JAN-23 10.30.00.000000000 AM	M

Figure 5: Output of the third simple query.

We might also be interested in a list of all ski manufacturers that we have in our database.

```

1 select PROIZVODAC from SKIJE;

```

PROIZVODAC
<input checked="" type="checkbox"/> Head
<input checked="" type="checkbox"/> Atomic
<input checked="" type="checkbox"/> Rossignol
<input checked="" type="checkbox"/> Stoeckli
<input checked="" type="checkbox"/> Fischer
<input checked="" type="checkbox"/> Voelkl
<input checked="" type="checkbox"/> Van Deer
<input checked="" type="checkbox"/> Dynastar
<input checked="" type="checkbox"/> Nordica
<input checked="" type="checkbox"/> Salomon

Figure 6: Output of the fourth simple query.

Finally, let's assume that we are interested in data on the countries where the races were held, and that their population is greater than 5 million people.

```

1 select IME_DRZAVE, BR_STAN, BDP from DRZAVA
2 where BR_STAN > 5000000;

```

IME_DRZAVE	BR_STAN	BDP
<input checked="" type="checkbox"/> Austrija	9129091	471400000000
<input checked="" type="checkbox"/> Svicarska	8865270	807706000000
<input checked="" type="checkbox"/> Italija	58803163	2010432000000
<input checked="" type="checkbox"/> SAD	333287000	25462700000000
<input checked="" type="checkbox"/> Kanada	40173200	2139840000000
<input checked="" type="checkbox"/> Francuska	68042591	2782905000000
<input checked="" type="checkbox"/> Njemacka	84358845	4072192000000
<input checked="" type="checkbox"/> Ceska	10827529	290924000000
<input checked="" type="checkbox"/> Norveska	5504329	579267000000

Figure 7: Output of the fifth simple query.

4.2 Multiple table queries

The next few queries will be more complex than the previous ones. This means that we will use joins in order to join different tables and at the same time get a more varied amount of information. To begin with, let's say that we want to check how many points were won by Croatian skiers and in which races did it happen. We emphasize that we will order the results from the highest number of points won to the lowest.

```
1 select IME || ' ' || PREZIME "Ime i prezime", BODOVI, IME_UTRKE from
   SKIJAS
2 inner join REZULTAT using (SKIJAS_ID)
3 inner join UTRKA using (UTRKA_ID)
4 where SKIJAS.NACIONALNOST = 'CRO'
5 order by BODOVI desc;
```

Ime i prezime	BODOVI	IME_UTRKE
✓ Leona Popovic		80 World Cup Finals
✓ Zrinka Ljubic		60 Spindl World Cup 2023
✓ Leona Popovic		45 Spindl World Cup 2023
✓ Samuel Kolega		29 Coupe du Monde
✓ Leona Popovic		26 Snow Queen Trophy 2023
✓ Filip Zubcic		26 Audi FIS Ski World Cup
✓ Filip Zubcic		26 Alpine Skiing Competitions
✓ Filip Zubcic		22 Soelden Ski World Cup
✓ Filip Zubcic		18 Coupe du Monde
✓ Filip Zubcic		16 Ski Welt Cup Garmisch

Figure 8: Output of the first multiple tables query.

It is well known that, during 2022/2023 season Mikaela Shiffrin broke the record for the highest total number of victories in career. In addition, she was almost always on the podium, so let's assume that we are interested in the number of podiums she achieved in the observed races and in which discipline.

```
1 select COUNT(REZ_ID) BROJ_POSTOLJA, IME_DISCIPLINE from REZULTAT
2 inner join SKIJAS USING (SKIJAS_ID)
3 inner join UTRKA USING (UTRKA_ID)
4 inner join DISCIPLINA USING (DISCIPLINA_ID)
5 where SKIJAS.IME = 'Mikaela' and SKIJAS.PREZIME = 'Shiffrin' and (PLASMAN
   = 1 or PLASMAN = 2 or PLASMAN = 3)
6 group by IME_DISCIPLINE;
```

BROJ_POSTOLJA	IME_DISCIPLINE
✓ 3	3 Slalom
✓ 1	1 Veleslalom

Figure 9: Output of the second multiple tables query.

One of the interesting pieces of information that we can find out is the ski manufacturer's point leaderboard, which we get using the following query.

```
1 select PROIZVODAC, SUM(BODOVI) as BODOVI from
2 SKIJE inner join IMA_SKIJE using (SKIJE_ID)
3 inner join SKIJAS using (SKIJAS_ID)
4 inner join REZULTAT using (SKIJAS_ID)
5 group by PROIZVODAC
6 order by BODOVI desc;
```

PROIZVODAC	BODOVI
Head	4052
Atomic	3229
Rossignol	2842
Salomon	927
Fischer	762
Stoekli	598
Dynastar	547
Van Deer	366
Voelkl	365
Kaestle	321

Figure 10: Output of the third multiple tables query.

Now let's take a look at the query to find out how many races were held in a particular country.

```

1 select IME_DRZAVE , COUNT(UTRKA_ID) as BR_UTRKA from DRZAVA
2 inner join SKIJALISTE using (DRZAVA_ID)
3 inner join UTRKA using (SKIJALISTE_ID)
4 group by IME_DRZAVE
5 order by BR_UTRKA desc;

```

IME_DRZAVE	BR_UTRKA
Italija	3
Austrija	3
Svicarska	3
SAD	3
Kanada	1
Slovenija	1
Hrvatska	1
Norveska	1
Njemacka	1
Francuska	1

Figure 11: Output of the fourth multiple tables query.

As the last complex query, let's look at the one that will as a result give us the names of speed discipline races and the average race times in them.

```

1 select ROUND(AVG(VRIJEME_UK),2) PROSJ_VR , IME_UTRKE , IME_SKIJALISTA ,
   IME_DISCIPLINE from REZULTAT
2 inner join UTRKA using (UTRKA_ID)
3 inner join SKIJALISTE USING (SKIJALISTE_ID)
4 inner join DISCIPLINA using (DISCIPLINA_ID)
5 where DISCIPLINA.KATEGORIJA = 'Brza'
6 group by IME_UTRKE , IME_SKIJALISTA , IME_DISCIPLINE;

```

PROSJ_VR	IME_UTRKE	IME_SKIJALISTA	IME_DISCIPLINE
84.4	Audi FIS Ski World Cup	Cortina d Ampezzo	Super G
121.94	55. Saslong Classic	Val Gardena	Spust
61.77	Alpine Skiing Competitions	St. Anton	Super G
117.06	Hahnenkamm Races	Kitzbuehel	Spust
67.79	Aspen World Cup	Aspen	Super G
72.02	Xfinity Birds of Prey	Beaver Creek	Super G
109.4	Lake Louise Alpine World Cup	Lake Louise	Spust
116.6	Alpine Skiing Competitions	St. Moritz	Spust
93.71	World Cup Kvitfjell	Kvitfjell	Spust

Figure 12: Output of the fifth multiple tables query.

4.3 Queries with aggregating functions

In this subsection, we will show several queries that contain aggregating functions. Let's note that we have already used some of these functions in examples of queries over multiple tables, but there was no emphasis on them. Let's assume to begin with that we are interested in the average population and median GDP of countries where ski resorts are located.

```
1 select AVG(BR_STAN) PROSJ_BR_STAN , MEDIAN(BDP) MEDIJAN_BDP from DRZAVA ;
```

	PROSJ_BR_STAN	MEDIJAN_BDP
<input checked="" type="checkbox"/>	52090062	693486500000

Figure 13: Output of the first query using aggregating functions.

Next, let's calculate the average age of male and female skiers. Let us note that here it is important to take into account whether the skiers have already celebrated their birthday in that year. In order to ensure that, we will count the number of years in the query as the number of days lived divided by 365.25, where, in order to get the days lived, we will subtract the date of birth from 27.08.2023. (which is the date when this chunk was originally written).

```
1 select ROUND(AVG(FLOOR((TO_DATE('27-AUG-2023') - DAT_ROD)/365.25)),2) "  
   Prosjecna dob", SPOL from SKIJAS  
2 group by SPOL;
```

	Prosječna dob	SPOL
<input checked="" type="checkbox"/>		29.17 M
<input checked="" type="checkbox"/>		27.16 F

Figure 14: Output of the second query using aggregating functions.

Next, we are interested in the total earnings of skiers based on their nationality.

```
1 select SUM(zarada) UK_ZARADA_ZEMLJE , NACIONALNOST from REZULTAT  
2 inner join SKIJAS using(SKIJAS_ID)  
3 group by NACIONALNOST  
4 order by UK_ZARADA_ZEMLJE desc;
```

	UK_ZARADA_ZEMLJE	NACIONALNOST
<input checked="" type="checkbox"/>	573128	SUI
<input checked="" type="checkbox"/>	522972	NOR
<input checked="" type="checkbox"/>	392664	ITA
<input checked="" type="checkbox"/>	358770	AUT
<input checked="" type="checkbox"/>	234303	USA
<input checked="" type="checkbox"/>	145236	GER
<input checked="" type="checkbox"/>	138145	FRA
<input checked="" type="checkbox"/>	119783	SLO
<input checked="" type="checkbox"/>	90100	SVK
<input checked="" type="checkbox"/>	63514	CAN

Figure 15: Output of the third query using aggregating functions.

One of the best skiers today is certainly the Swiss Marco Odermatt. Let's look at the query with which we will check how many points on average he won in the races that we covered in this paper.

```

1 select ROUND(AVG(BODOVI),2) AVG_BODOVI from REZULTAT
2 inner join SKIJAS using (SKIJAS_ID)
3 inner join UTRKA using (UTRKA_ID)
4 inner join DISCIPLINA using (DISCIPLINA_ID)
5 where IME = 'Marco' and PREZIME = 'Odermatt';

```

AVG_BODOVI
79.33

Figure 16: Output of the fourth query using aggregating functions.

Finally, let's look at the average altitude of the starting point and finish point for races.

```

1 select AVG(NMV_START) PROSJ_NMV_START, AVG(NMV_CILJ) PROSJ_NMV_CILJ from
   UTRKA;

```

PROSJ_NMV_START	PROSJ_NMV_CILJ
1832.95	1382.3

Figure 17: Output of the fifth query using aggregating functions.

4.4 Subqueries, nested queries, set operations

In this subchapter, we will list some queries that contain subqueries as well as an example of a query with one of the set operations. Let's first look at the query that will give us information about the nationalities of skiers who, on average, scored more points in races than Croatian skiers.

```

1 select ROUND(AVG(BODOVI),2) AVG_BODOVI, NACIONALNOST from REZULTAT
2 inner join SKIJAS using (SKIJAS_ID)
3 having AVG(BODOVI) > (
4     select AVG(BODOVI) from REZULTAT
5     inner join SKIJAS using (SKIJAS_ID)
6     where NACIONALNOST = 'CRO'
7 )
8 group by NACIONALNOST
9 order by AVG(BODOVI) desc;

```

AVG_BODOVI	NACIONALNOST
80	GRE
44.57	SVK
32.8	NOR
30.43	SLO

Figure 18: Output of the first query with subquery.

As we mentioned earlier in the technical disciplines (slalom and giant slalom) it may happen that skiers do not finish the second run. With the following query, we would like to find out how many points were earned by skiers who did not finish at least one race among those covered in this paper. We will order the results from the highest number of points to the lowest, regardless of whether it is a male or female skier.

```

1 select IME, PREZIME, SUM(BODOVI) UK_BODOVI from REZULTAT
2 inner join SKIJAS using (SKIJAS_ID)
3 where SKIJAS_ID in (
4     select SKIJAS_ID from SKIJAS
5     inner join REZULTAT using (SKIJAS_ID)
6     inner join UTRKA using (UTRKA_ID)
7     inner join DISCIPLINA using (DISCIPLINA_ID)
8     where VRIJEME_DV is NULL and KATEGORIJA = 'Tehnicka'
9 )
10 group by IME, PREZIME
11 order by UK_BODOVI desc;

```

IME	PREZIME	UK_BODOVI
Henrik	Kristoffersen	340
Aleksander	Aamodt Kilde	304
Loic	Meillard	277
Anna	Swenn Larsson	110
Franziska	Gritsch	106
Clement	Noel	100
Linus	Strasser	90
Maria Therese	Tviberg	82
Ana	Bucik	80
Zrinka	Ljutic	72

Figure 19: Output of the second query with subquery.

Since the ski resorts where the races were held are located at different altitudes, we are interested to see which races were held in the ski resorts located at an altitude between 1,500 and 2,100 meters and in which country they are located. With the following query, we find out exactly that information.

```

1 select IME_UTRKE, IME_DRZAVE from UTRKA
2 inner join SKIJALISTE using (SKIJALISTE_ID)
3 inner join DRZAVA using (DRZAVA_ID)
4 where SKIJALISTE_ID in (
5     select SKIJALISTE_ID from SKIJALISTE
6     where NADM_VISINA BETWEEN 1500 and 2200
7 );

```

IME_UTRKE	IME_DRZAVE
Lake Louise Alpine World Cup	Kanada
55. Saslong Classic	Italija
Alpine Skiing Competitions	Svicarska
World Cup Finals	Andora

Figure 20: Output of the third query with subquery.

Suppose we look at ski racing from the perspective of the ski manufacturer. We could certainly be interested to see in which races and disciplines did skiers who use skis from a specific manufacturer compete. With the following query, we will try to find out exactly that for skiers who use Voelkl brand skis.

```

1 select distinct(IME_UTRKE), IME_DISCIPLINE from UTRKA
2 inner join REZULTAT using (UTRKA_ID)
3 inner join DISCIPLINA using (DISCIPLINA_ID)
4 inner join SKIJAS using (SKIJAS_ID)
5 where SKIJAS_ID in (

```



```

6 select SKIJAS_ID from SKIJAS
7 inner join IMA_SKIJE using (SKIJAS_ID)
8 inner join SKIJE using (SKIJE_ID)
9 where PROIZVODAC = 'Voelkl'
10 );

```

IME_UTRKE	IME_DISCIPLINE
<input checked="" type="checkbox"/> Snow Queen Trophy 2023	Slalom
<input checked="" type="checkbox"/> World Cup Finals	Slalom
<input checked="" type="checkbox"/> The Killington Cup	Veleslalom
<input checked="" type="checkbox"/> Coupe du Monde	Slalom
<input checked="" type="checkbox"/> Ski Welt Cup Garmisch	Slalom
<input checked="" type="checkbox"/> Alpine Skiing Competitions	Slalom
<input checked="" type="checkbox"/> 59. Zlata lisica	Veleslalom
<input checked="" type="checkbox"/> Spindl World Cup 2023	Slalom

Figure 21: Output of the fourth query with subquery.

Another thing that might interest us is certainly the list of male and female skiers who competed in all disciplines. In order to get this information, we will use a intersection operation.

```

1 select IME || ' ' || PREZIME "Ime i prezime" from SKIJAS
2 inner join REZULTAT using (SKIJAS_ID)
3 inner join UTRKA using (UTRKA_ID)
4 inner join DISCIPLINA using (DISCIPLINA_ID)
5 where IME_DISCIPLINE = 'Slalom'
6 intersect
7 select IME || ' ' || PREZIME "Ime i prezime" from SKIJAS
8 inner join REZULTAT using (SKIJAS_ID)
9 inner join UTRKA using (UTRKA_ID)
10 inner join DISCIPLINA using (DISCIPLINA_ID)
11 where IME_DISCIPLINE = 'Spust'
12 intersect
13 select IME || ' ' || PREZIME "Ime i prezime" from SKIJAS
14 inner join REZULTAT using (SKIJAS_ID)
15 inner join UTRKA using (UTRKA_ID)
16 inner join DISCIPLINA using (DISCIPLINA_ID)
17 where IME_DISCIPLINE = 'Veleslalom'
18 intersect
19 select IME || ' ' || PREZIME "Ime i prezime" from SKIJAS
20 inner join REZULTAT using (SKIJAS_ID)
21 inner join UTRKA using (UTRKA_ID)
22 inner join DISCIPLINA using (DISCIPLINA_ID)
23 where IME_DISCIPLINE = 'Super G';

```

	Ime i prezime
<input checked="" type="checkbox"/>	Michelle Gisin
<input checked="" type="checkbox"/>	Mikaela Shiffrin

Figure 22: Output of the query with set operation.

5 Default values, conditions and comments

In this chapter we will focus on conditions and default values. We will also add comments to some tables and columns so that we know what information they contain.

5.1 Default values

As the first default value we will set the prize pools in the races. According to FIS regulations, race organizers are obliged to provide at least 120,000 Swiss francs for prizes. That amount is then divided between the first 30 placed skiers. Of course, the exception is those races, mostly in technical disciplines, in which the skiers did not finish the second run, so they could not earn money. For this reason, we will default to that amount.

```
1 alter table UTRKA
2 modify NAGRADNI_FOND default 120000;
```

The second default value considers the entry of new results, more precisely placement. When entering the data, we mentioned that 0 means that the skier did not place. In the case of new entries, unless otherwise stated, we will consider that placement has not been achieved.

```
1 alter table REZULTAT
2 MODIFY PLASMAN default 0;
```

5.2 Conditions

We check the conditions during entry, and the first one we will define is related to the discipline.

```
1 alter table REZULTAT
2 alter table DISCIPLINA add CONSTRAINT PROV_DISC check (IME_DISCIPLINE in (
    'Slalom', 'Veleslalom', 'Spust', 'Super G'));
```

It also makes sense to set the prize pool check condition which, we saw a little while ago, defaults to 120000 Swiss francs.

```
1 alter table UTRKA add CONSTRAINT PROM_FONDA check (NAGRADNI_FOND >=
    120000);
```

5.3 Comments

Now let's look at some of the comments that we will place on tables and rows.

```
1 -- komentari na tablice
2 comment on table SPONZOR is 'Tablica individualnih sponzora skijasa';
3 comment on table REZULTAT is 'Tablica svih rezultata u svim utrkama';
4 comment on table SKIJAS is 'Tablica svih skijasa';
5
6 -- komentari na retke
7 comment on column UTRKA.KATEG is 'Je li odrzana utrka u muskoj ili u
    zenskoj konkurenciji';
8 comment on column DISCIPLINA.KATEGORIJA is 'Za lakse razlikovanje brzih i
    tehnickih disciplina';
```

6 Indexes

In this chapter we will define several indexes on our data. They are defined in order to speed up our database search if we have a large amount of data. We distinguish between B-tree index and Bitmap index. B-tree indexes are suitable for columns that contain a large number of different values. For example, in our work up to now, in many queries, we have used attributes related to the names and surnames of skiers, which contain a large number of different values. For this reason, it makes sense to define an index over these attributes.

```
1 create index SKIJAS_INDEX on SKIJAS(IME, PREZIME);
```

Likewise, we often search for skiers by nationality, so it makes sense to create an index that could speed this up. If we look at ski resorts, we usually search them, in addition to the country in which they are located, by location, i.e. by the mountain/mountain range in which it is located, so it also makes sense to create a B-tree index for that.

```
1 create index NAC_INDEX on SKIJAS(NACIONALNOST);  
2  
3 create index PLANINA_INDEX on SKIJALISTE(LOKACIJA);
```

Bitmap index is suitable for columns that have a large number of rows, but contain a small number of different values. If we take that into account, it makes sense for us to define a Bitmap index for the gender of skiers, given that we only have men and women.

```
1 create bitmap index SPOL_SKIJAS_INDEX on SKIJAS(SPOL);
```

7 Procedures

In this chapter we will deal with the procedures. They are introduced to simplify and speed up some frequently used queries. We will look at two procedures: the first, in which the emphasis is on entering and changing data about skiers, and the second, whose goal is to print out the ten best by number of points, based on the discipline and gender of the skier. Let's focus on the first procedure. As we said at the beginning, for the sake of simplicity, we limited ourselves to a certain number of races that we entered into the database, so we do not have a list of all the skiers who competed in the last season of the World Ski Cup. If you were to add new skiers to database using the procedure given below, it would speed up the new entry and facilitate data exchange.

```
1 create or replace procedure SKIJASI_IN_UP(
2     SK_ID in SKIJAS.SKIJAS_ID%TYPE,
3     SK_IME in SKIJAS.IME%TYPE,
4     SK_PREZ in SKIJAS.PREZIME%TYPE,
5     SK_SPOL in SKIJAS.SPOL%TYPE,
6     SK_D_R in SKIJAS.DAT_ROD%TYPE,
7     SK_NAC in SKIJAS.NACIONALNOST%TYPE
8 )
9 as
10     BROJ_SKIJASA integer;
11 begin
12     select COUNT(*) into BROJ_SKIJASA from SKIJAS where SKIJAS_ID = SK_ID;
13     IF BROJ_SKIJASA = 0 THEN
14         insert into SKIJAS values (SK_ID, SK_IME, SK_PREZ, SK_SPOL, SK_D_R
15 , SK_NAC);
16         DBMS_OUTPUT.PUT_LINE('Dodan je skijas s id-jem ' || SK_ID);
17         commit;
18     ELSE
19         update SKIJAS set IME = SK_IME, PREZIME = SK_PREZ, SPOL = SK_SPOL,
20         DAT_ROD = SK_D_R, NACIONALNOST = SK_NAC where SK_ID = SKIJAS_ID;
21         DBMS_OUTPUT.PUT_LINE('Napravljena je izmjena kod skijasa s id-jem
22 ' || SK_ID);
23         commit;
24     END IF;
25 end;
26 /
27
28 -- Napravimo unos skijasa kojeg nemamo u bazi... Pietro Zazzi, 22-07-1994,
29     ITA te unesimo namjerno krivi datum rodenja
30 call SKIJASI_IN_UP(223, 'Pietro', 'Zazzi', 'M', TO_DATE('21-07-1994', 'dd-
31     MM-yyyy'), 'ITA');
32
33 -- ako sada opet pozovemo proceduru samo s tocnim datumom
34 call SKIJASI_IN_UP(223, 'Pietro', 'Zazzi', 'M', TO_DATE('22-07-1994', 'dd-
35     MM-yyyy'), 'ITA');
```

```

SQL> call SKIJASI_IN_UP(223, 'Pietro', 'Zazzi', 'M', TO_DATE('21-07-1994', 'dd-MM-yyyy'), 'ITA');

Dodan je skijas s id-jem 223

Call completed.

SQL> call SKIJASI_IN_UP(223, 'Pietro', 'Zazzi', 'M', TO_DATE('22-07-1994', 'dd-MM-yyyy'), 'ITA');

Napravljena je izmjena kod skijasa s id-jem 223

Call completed.

```

Figure 23: Output of the first procedure.

Everyone who follows any type of sport is always interested in knowing which athletes are in the top ten, apart from the first place winners. This was the motive for the creation of the second procedure. Given that there are four disciplines in skiing that are held in the men's and women's categories, in order to simplify the search for the top 10, we created a procedure to which we pass the discipline and the gender of the skiers, and it returns the order of the top ten. In order to retrieve data on the top 10 skiers, i.e. female skiers, we used a cursor.

```

1 create or replace procedure TOP_TEN(
2     DISC in DISCIPLINA.IME_DISCIPLINE%type,
3     SP in SKIJAS.SPOL%type
4 )
5 as
6     cursor T_TEN is
7         select IME, PREZIME, SUM(BODOVI) UK_BODOVI from SKIJAS
8         inner join REZULTAT using (SKIJAS_ID)
9         inner join UTRKA using (UTRKA_ID)
10        inner join DISCIPLINA using (DISCIPLINA_ID)
11        where IME_DISCIPLINE = DISC and SPOL = SP
12        group by IME, PREZIME
13        order by UK_BODOVI desc
14        fetch first 10 rows only;
15    T_TEN_DVA T_TEN%rowtype;
16 BEGIN
17     open T_TEN;
18     LOOP
19         fetch T_TEN into T_TEN_DVA;
20         EXIT WHEN T_TEN%NOTFOUND;
21         DBMS_OUTPUT.PUT_LINE(T_TEN_DVA.IME || ' ' || T_TEN_DVA.PREZIME ||
22         ' je ostvario/la ' || T_TEN_DVA.UK_BODOVI || ' bodova. ');
23     END LOOP;
24     close T_TEN;
25 end;
26 /
27
28 call top_ten('Slalom', 'F');

```

```
SQL> call TOP_TEN('Slalom', 'F');

Mikaela Shiffrin je ostvario/la 240 bodova.
Petra Vlhova je ostvario/la 200 bodova.
Leona Popovic je ostvario/la 151 bodova.
Lena Duerr je ostvario/la 118 bodova.
Anna Swenn Larsson je ostvario/la 110 bodova.
Wendy Holdener je ostvario/la 95 bodova.
Franziska Gritsch je ostvario/la 76 bodova.
Sara Hector je ostvario/la 72 bodova.
Michelle Gisin je ostvario/la 71 bodova.
Amelia Smart je ostvario/la 67 bodova.

Call completed.
```

Figure 24: Output of the second procedure.

Let's also mention that we could have created similar procedures for entering or changing data in any database, as well as, for example, the order of skiers depending on their earnings.

8 Triggers

In this chapter we will define two triggers. Given that there are no negative points in skiing, and with 0 we marked the points if the skier did not finish the race, we will create a trigger that will be activated if someone tries to enter negative points. Also we can see how trigger works in action.

```
1 create or replace TRIGGER NEG_BOD_TRIGGER
2 before update or insert of BODOVI on REZULTAT
3 for each row
4 when (new.BODOVI < 0)
5 begin
6     raise_application_error(-20111, 'Broj bodova ne moze biti negativan');
7 end;
8 /
9
10 -- PR
11 -- pokusajmo promijeniti bodove Filipu Zubcicu u veleslalomu u Adelbodenu,
12     to je rezultat s id-jem 0112
13 select ime, prezime, bodovi from SKIJAS
14 inner join rezultat using (skijas_id)
15 where rez_id = 0112;
16 -- ima 22 boda
17 update rezultat set bodovi = -10 where rez_id = 0112;
```

```
SQL> update rezultat set bodovi = -10 where rez_id = 0112;

update rezultat set bodovi = -10 where rez_id = 0112
*
ERROR at line 403:
ORA-20111: Broj bodova ne moze biti negativan
ORA-06512: at "IBISKUP.NEG_BODOVI_TRIGGER", line 2
ORA-04088: error during execution of trigger 'IBISKUP.NEG_BODOVI_TRIGGER'
```

Figure 25: Output of result of first trigger activation.

The second trigger that we will make is related to the skier's earnings in a particular race. The insured fund for the races depends on the organizers, but from the data available on the In-

ternet, we can find out that the largest fund in the season 2022/2023 was 333,200 Swiss francs, and the earnings of the first-placed skier then amounted to 98,000 Swiss francs. If all 30 skiers finished the race, the minimum possible earnings is 450 Swiss francs. Accordingly, we create a trigger that will be activated if we try to enter values outside the previously specified interval.

```
1 create or replace TRIGGER ZARADA_TRIGGER
2 before update or insert of ZARADA on REZULTAT
3 for each row
4 when (new.ZARADA < 550 or new.ZARADA > 98000)
5 begin
6     raise_application_error(-20110, 'Nije moguće zaraditi toliko novca');
7 end;
8 /
9
10 -- pokusajmo promijeniti zaradu za isti primjer kao i malo prije. Dakle,
    pokusat cemo promijeniti zaradu Filipu Zubcicu na utrci u Adelbodenu,
    id rezultata je 0112, zarada 1600CHF
11
12 update rezultat set zarada = 400 where rez_id = 0112;
```

```
SQL> update rezultat set zarada = 400 where rez_id = 0112;

update rezultat set zarada = 400 where rez_id = 0112
*
ERROR at line 419:
ORA-20110: Nije moguće zaraditi toliko novca
ORA-06512: at "IBISKUP.ZARADA_TRIGGER", line 2
ORA-04088: error during execution of trigger 'IBISKUP.ZARADA_TRIGGER'
```

Figure 26: Output of result of second trigger activation.

9 Conclusion

In this paper, we modeled part of the databases used for the purpose of data storage for 2022/2023 FIS Alpine Ski World Cup. Since this is a simplified variant, there are many possibilities to expand and add new data. However, for the purposes of this work, we decided to stay within the limits of a somewhat simpler model from which we could also find out a lot of information that may not be available to people who follow skiing.