Josh Lehman (336545876)
Binyamin Feinmesser (537445675)
Raphi Jacobson (522964823)

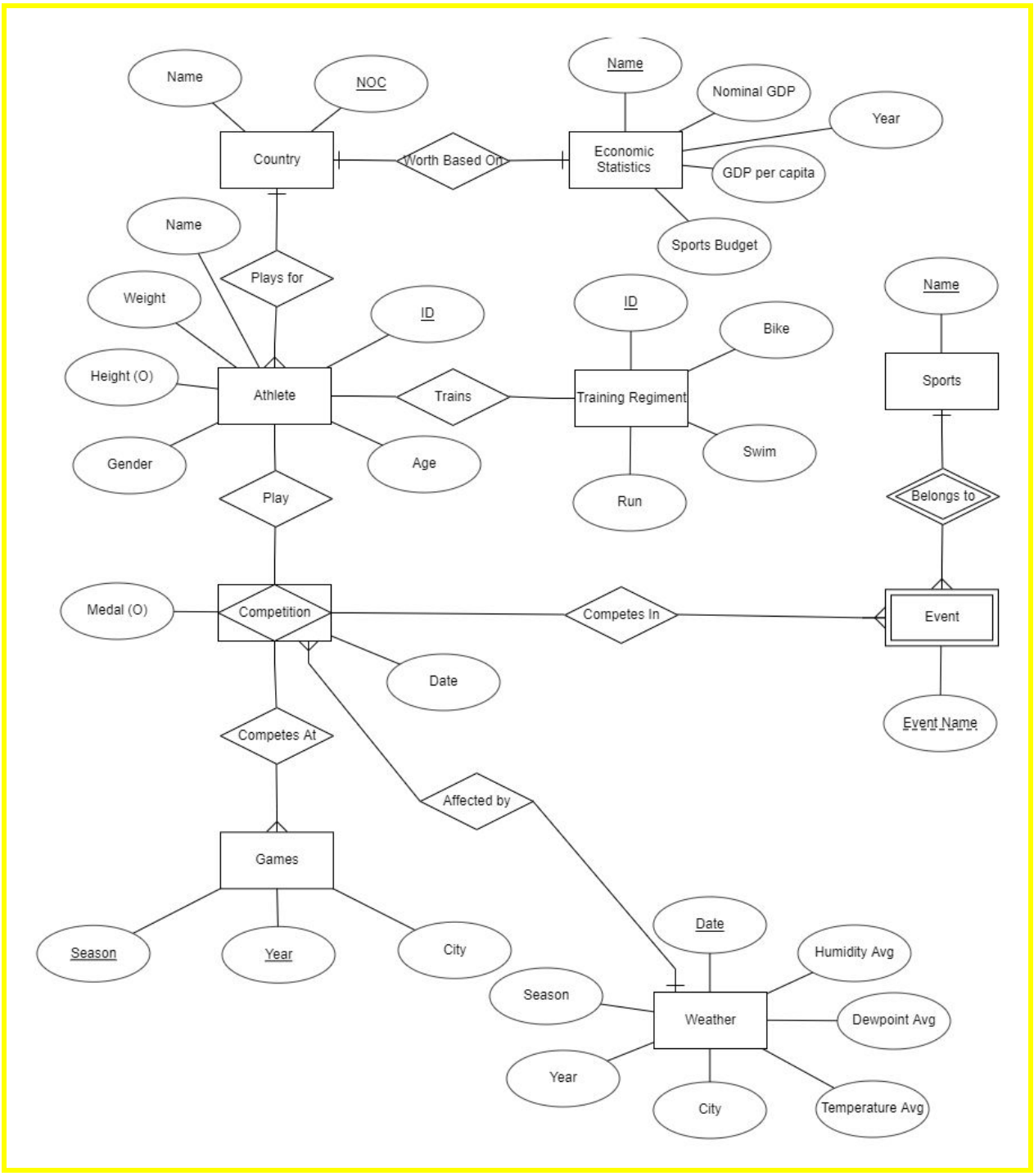
בס"ד

Database Project Report:

THE 1st STAGE

Sunday - 8/23
We chose our dataset of an Olympics record.
Link:
(https://www.kaggle.com/heesoo37/120-years-of-olympic-history-athletes-and-results).
We wanted to make sure that there were enough entities to meet the project requirements. We reached out to the professor who told us that in our case 5 would be enough due to the complex many-to-many relationships in the dataset.
Entities: *Country, Athlete, Games, Event, Sport*.
Event derives from *Sport* as a weak entity.
Added Entities: weather, Economic Status, Training Regiment
Sources:
https://www.kaggle.com/markliversedge/goldencheetah-opendata-athlete-activity-and-mmp
https://www.kaggle.com/vin725k/olympics-data?select=dataset_homemade.xlsx
https://www.kaggle.com/grubenm/austin-weather

Wednesday - 8/23
We created our ERD on a program called ERDPlus.



Thursday - 8/27

- We presented our ERD to the professor who gave us positive feedback. He pointed out to us that there is a 3NF violation in the entity *Games* and that we could consider splitting the two keys of *Season* and *Year* into different tables. After discussing with him and amongst ourselves we decided to intentionally violate the 3NF in this specific incident. We decided this as we thought that it would unnecessarily complicate the structure of our ERD to have the Year and Season in different entities as they are both very relevant to the *Games* entity. In addition to would cause excess joins and increase the calculation time of our queries.
- We began to brainstorm different ideas for our web-app and the types of questions it could answer. We came up with a few interesting ideas
    - Which countries are most/least successful in the Olympics
    - Which Athlete are most/least successful in the Olympics
    - Are countries of colder/warmer regions more successful at the winter/summer olympics respectively.
    - How many of each Medal has been won
    - How many entires has each country had.
    - After Rafi's Addition to the ERD, Some new questions were posed to include the added entities
        - Is an Athlete who runs more, more likely to win a medal?
        - Is a country with a higher gdp more likely to win a medal?
        - Does Higher Humidity affect the amount of the most medals won? (SInce the best athlete would get tired more easily)

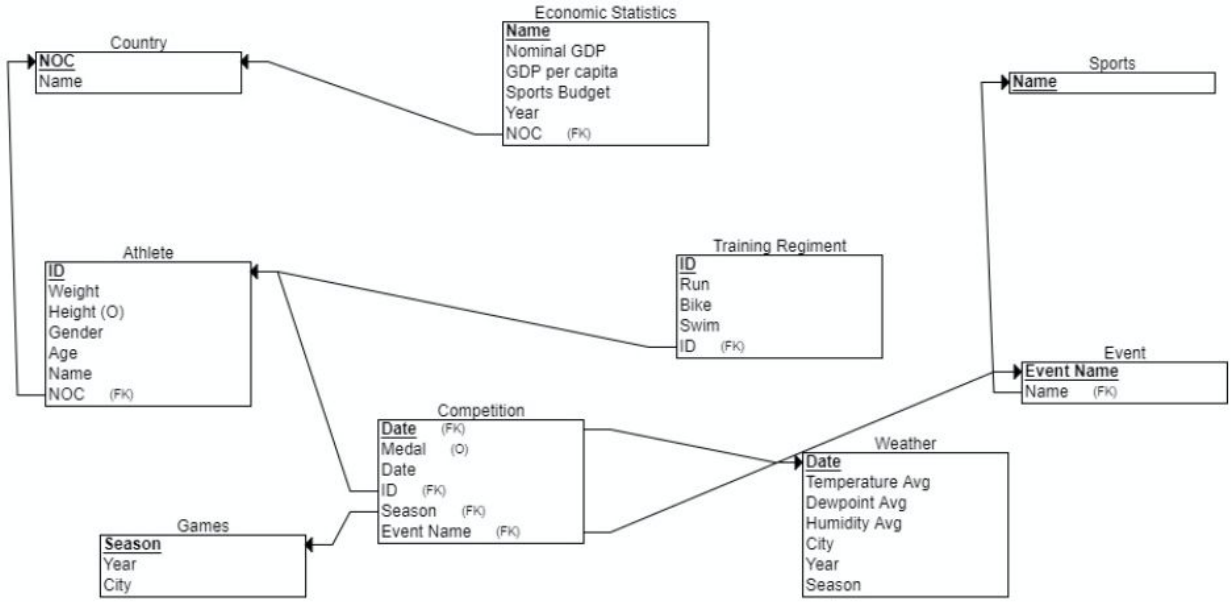
Friday - 8/28

- We began converting our ERD into a Schema
    - Country(<u>noc</u>, name)
    - Athlete(<u>id</u>, weight, height (O), gender, name, age)
    - Games(<u>season</u>, <u>year</u>, city)
    - Sport(<u>name</u>)
    - Event(<u>event_name</u>)
    - Competition(athlete_id, season, year, event_name, medal (O))
    - Training Regiment(<u>ID,</u> Bike, Run, Swim)
    - Economic Statistics(<u>Country,</u> Nominal GDP, GDP per Capita, Sports Budget)
    - Weather(<u>Date,</u> Humidity Avg, Dewpoing Avg, Temperature Avg, Season, City)
- The program we were using, ERDPlus, converted our ERD into an SQL code. Here is a short example of how it looks for two of our entities:

```
CREATE TABLE Country
(
 Name STRING NOT NULL,
 NOC STRING NOT NULL,
 PRIMARY KEY (NOC)
);

CREATE TABLE Games
(
 Year INT NOT NULL,
 Season STRING NOT NULL,
 City STRING NOT NULL,
 PRIMARY KEY (Year),
 UNIQUE (Season)
);
```

- WE built a DSD out of our ERD. Here we can clearly see the relationship between the entities in terms of the Keys and Foreign Keys.



Sunday - 8/30

- Began working on uploading our database into the PL/SQL editor software.
- Our database has 271,117 records, meeting the size requirements.
- We uploaded one of the tables, *Athlete*, with INSERT commands to demonstrate the functionality of it. This is the **physical SCHEMA**. These are attached at Physical Data Games.sql etc...

- The rest of the data was imported using the text importer.
- We generated random data for *Athlete* although did not use it as we have real world data. We did this merely to demonstrate the functionality of the *Data Generator.* It is saved as DataGenerator.sql. Attached is a screenshot of the randomly generated data:

Tuesday - 9/1

- We validated the integrity of our data by running various queries on the imported data. We compared the results to the external source files. Attached are the query scripts referred to as *Validation1.sql, Validation2.sql*... I am also showing a screenshot of such a query:

Wednesday - 9/2

- We ran reports on our data. This includes Compilation Error reports as well as reports for each individual schema. Those reports are attached in the zip under the names OBJECT_REPORTS.rep and SCEME_REPORTS.rep and CompilationErrors.rep

- We exported our Tables from Oracle to our computer, deleted the data, and re-imported our tables. We wrote a script that deletes all of the tables in our system. It is attached under the name DeleteTableScript.sql

THE 2nd STAGE

QUERIES

Sunday - 9/6

- We wrote queries for our data with attempts to answer relevant questions
- The Queries answer the following questions
    - Which athlete has won the most medals (8.216 seconds)
    - Which country has won the most medals (.117 seconds)
    - How many medals were won in total (.058 seconds)
    - Given any country as input, who competes for that country (2.095 seconds)
    - How many entries has a country had (.208 seconds)

    - Is there a correlation between an athlete who runs more and wins more medals(42.682 seconds)
    - Find All Teams that are located in both the countries and economics statistics tables (.392 seconds)
    - How many times was the average temperature over 60 in each year(.078)
- Attached in the ZIP are the sql files for the queries as well as the csv files for the results.

INDEX STRUCTURES

Monday - 9/7

- We added 3 index structures to our tables Athlete, Competition, training, ECONOMIC_STATUS,
- We re-ran the queries that use the tables Athlete, Competition, training, ECONOMIC_STATUS Tables and compared the times of each query before and after the index structures were added.

	QUERY	BEFORE INDEX	AFTER INDEX
1	Which athlete has won the most medals?	8.216 seconds	.191 seconds
2	Which country has won the most medals?	Database Project Report with Rafi Additions.docx - Google Docs	.069 seconds
3	How many medals were won in total?	.058 seconds	.011 seconds
4	Given any country as input, who competes for that country?	~2.095 seconds (May depends on which country was chosen. We used AUS here)	~.044 seconds (Using same input of AUS as previously)
5	How many entries has a country had?	.208 seconds	.125 seconds
6	Is there a correlation between an athlete who runs more and wins more medals?	42.682 seconds	4.612 seconds
7	Find All Teams that are located in both the countries and economics statistics tables	.392 seconds	.038 seconds
8	How many times was the average temperature over 60 in each year	.078 seconds	.055 seconds

- Analysis of Index structures. We noticed a drop in execution time in every single query we ran. There were significant drops in queries 1, 4 and 6. We think the reason is that these are more complicated queries which heavily rely on the shared keys between Athlete and Competition. By making these keys indexes, we have heavily decreased the execution time.

Update

Wednesday - 9/9

- We updated the records of an athlete named Ryan Bailey. We wrote a query that would update the weight of all athletes with this name to 3000 pounds. This ultimately updated 6 records as he competed 6 times. Attached are the screenshots before and after the update. The SQL file for the update is zipped.

Delete

- We deleted two files from any entry where the country name is 'Brigantia'. Two records were deleted. The delete SQL file is zipped.

Rollbacks

I. We wrote a script called *Rollback.sql* that inserts into our Athlete table the entry *(123456789, 'Josh Lehman', 'M', 25, 190, 90, 'USA')*.
We then show the result of selecting this entry from athlete.
Then we commit.
Next we update the name from Josh Lehman to Guy Kelman
We then do a Rollback and the entry reverts back to Josh Lehman from Guy Kelman (rollback takes us back to the last commit).

II. We compare the results of select in (I1) and select in (I4) which are before and after the *rollback* respectively. As you can see they produce equivalent results. Even though we updated the Athlete entry, after the *rollback* we are left with our initial entry.
We would like to suggest that the DBMS committed changes and then started a new transaction. Then we updated the entry within the new transaction. By calling *rollback* we have essentially told the DBMS to undo any changes done since the transaction started.

III. We wrote a new script that uses *commit* instead of *rollback* is (I3). This new script will tell the DBMS to fully lock-in the update and not allow for any future *rollback* to change it. We believe that the DBMS is essentially 'ending' the transaction by calling a *commit*.
We have attached the sql file in the name commit.sql. As you can see here, the update is not undone and the entry name stays changed to Guy Kelman.

IV. We take advantage of the SQL GRANT command to allow other users access to tables in the user's table space.

V. For unknown reasons, despite doing the Grant command from our user, oracle00 still was not able to access our Athlete table, preventing us from initially succeeding in this section.We were able to fix this error by naming the table jlehman.athlete so that oraclee00 could find it. We ran the same script, rollback.sql as we did in the user jlehman and we received the exact same results.

Thursday - 9/10

Constraints

We implemented the constraints by writing ALTER TABLE commands. They are attached in a file called CONSTRAINTS.sql. We then tested them by trying to insert entries that violate our constraints. We saved that scripts in a file called CONSTRAINT_TEST.sql. We only needed to use insert commands to demonstrate the errors resulting from our constraints.

- The first error is a result of having the name of the country we inserted be in lowercase despite the uppercase constraint.
- The second error is a result of inserting the same data twice into training despite the unique constraint.
- The Third error is a result of inserting a negative value for the average temperature of a certain day despite the positive constraint.

Views

View_1.sql - We made a view of all of the countries names, weight, and heights.
View_1_Query_1.sql - We found the countries with the tallest average athletes.
View_1_Query_2.sql - We found the countries with the heaviest average athletes.
View_1_INSERT,UPDATE,DELETE.sql - We wrote a script with two of each of these statements and received a very interesting error. It says that because our view is made of the joins of a few tables it is erroneous to attempt to add to such our tables through a view. We corrected this error in our second view by only creating the view from one single table

View_2.sql - We made a view of all the athletes and their biking training level
View_2_Query_1.sql - We found the most trained bikers
View_2_Query_2.sql - We found the average training of the bikers
View_2_INSERT,UPDATE,DELETE.sql - Unlike in View_1, we were successfully able to insert, update, and delete with our view.

Because the view was only taken from one table, we are no longer receiving an error when trying to insert and update the view.

THE 3rd STAGE

Sunday - 9/13
We built a program that runs a cursor loop. The goal of the program is to take the average temperature of all games ever played. This can be useful to know as it can help athletes prepare for the conditions they can expect at an olympics event.
The cursor loop goes through every single event date and takes the average temperature for that day. Then it sums up these averages and takes the total average. We print it using the DBMS_OUTPUT. Attached is the file containing the loop by the name CursorLoop.tst

Tuesday - 9/15
We build a program called trigger.tst which is set to automatically update the ID of a newly inserted athlete. We use the before trigger to update the ID before it is added. (we did not implement the 'after' trigger in the example as it did not make so much sense to. Here we demonstrating adding new athletes and each new athlete has the correct subsequent ID number as they increment numerically:

Thursday - 9/17
The last part of the stage asked us to create a package and include with in it the following:

- A variable declaration
- An sql query
- An update or delete
- And a cursor loop

We created a program that is meant to take in an athlete (by ID number) and convert his/her weight into US metrics. We declare many variables in the program, run queries to find the weight, use update to update the athlete's file, and use a cursor loop to convert the weights of all of the athletes.

Attached are the files for the package named myPackage.spc and myPackage.bdy.We ran a simple query beforehand and saw that all of the weights are in kg. After running the procedure from our package we can clearly see here that all of the weight for the chosen country (Cuba in

this case) was converted to lbs. We did this for 10 files just to demonstrate that it works. Here are screenshots of the test file where we call our procedure as well as the results in lbs:

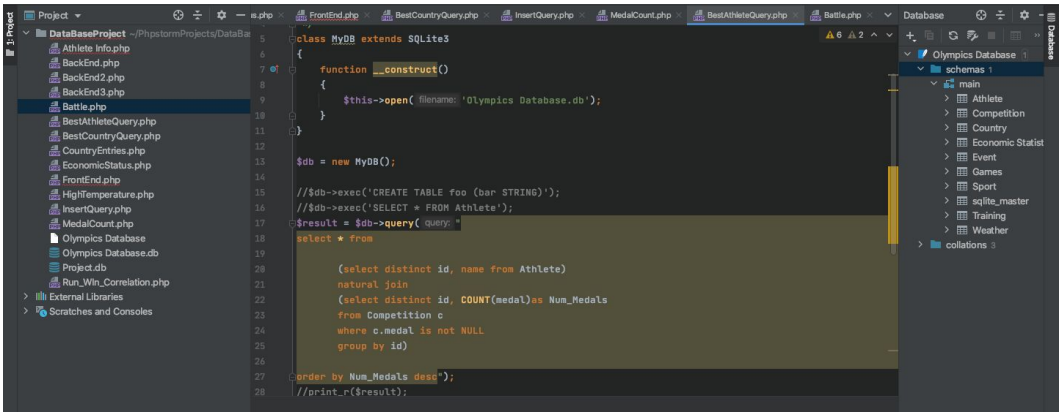
THE 4th STAGE

We intended to build a web-app that would answer our initial queries as well as adding a few additional *cooler* functionalities. We created the app using HTML and PHP

We included 3 sections in our Web-app

1. Queries - To answer our initial questions from Stage 1 and 2
2. Add Athlete - To add a new Athlete to tha Athlete table
3. Battle - Allow the user to choose two countries to “battle” or compare which is the more winning country.

DATABASE CONNECTION:



HTML FRONTEND: