Contents

1 Basic Test Results

```
Extracting Archive:
Archive:  /tmp/bodek.etw3dk3v/db/ex1/oren503/presubmission/submission
  inflating: create.sql
  inflating: drop.sql
  inflating: ex1.pdf
  inflating: ex1.py
 extracting: README

************************************************************
** Testing that all necessary files were submitted:
README:
    SUBMITTED
create.sql:
    SUBMITTED
drop.sql:
    SUBMITTED
ex1.py:
    SUBMITTED
ex1.pdf:
    SUBMITTED

************************************************************
** Checking for correct README format:
OK!

************************************************************
** Testing table creation:
Output:
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE

Number of tables created: 4

************************************************************
** Processing file:
Inserting country.csv
Output:
COPY 187

Inserting university.csv
Output:
COPY 15842

Inserting closed.csv
Output:
COPY 5

Inserting acceptance_data.csv
Output:
COPY 136596


************************************************************
** Testing dropping of tables:
Output:
DROP TABLE
DROP TABLE
```

```
DROP TABLE
DROP TABLE

Number of tables dropped: 4
```

2 README

```
oren503,nir.levi4
```

3 create.sql

5.1

```sql
CREATE TABLE country(
    countrycode char(3) PRIMARY KEY,
    countryname varchar(200) NOT NULL,
    region varchar(200) CHECK (region='South Asia' or region='Europe and Central Asia' or region='Middle East and North Afri
    or region='Sub-Saharan Africa' or region='Latin America and Caribbean' or region='East Asia and Pacific' or region='Nort
    incomegroup varchar(200) CHECK (incomegroup='Low income' or incomegroup='Upper middle income'
    or incomegroup='Lower middle income' or incomegroup='High income')
);

CREATE TABLE university(
    iau_id1 varchar(200) PRIMARY KEY,
    eng_name varchar(200) NOT NULL,
    orig_name varchar(200) NOT NULL,
    foundedyr integer CHECK (foundedyr > 0),
    private01 boolean NOT NULL,
    divisions integer CHECK (divisions >= 0),
    phd_granting boolean NOT NULL,
    specialized boolean NOT NULL,
    latitude numeric CHECK (latitude >= -90 and latitude <= 90),
    longitude numeric CHECK (longitude >= -180 and longitude <= 180),
    countrycode char(3),
    FOREIGN KEY(countrycode) REFERENCES country(countrycode)
);

CREATE TABLE closed(
    iau_id1 varchar(200) PRIMARY KEY,
    yrclosed integer CHECK (yrclosed > 0),
    FOREIGN KEY(iau_id1) REFERENCES university(iau_id1)
);

CREATE TABLE acceptance_data(
    year integer CHECK (year > 0),
    student5_estimated integer CHECK (student5_estimated >= 0),
    iau_id1 varchar(200),
    PRIMARY KEY (year, iau_id1),
    FOREIGN KEY(iau_id1) REFERENCES university(iau_id1)
);
```

4 drop.sql

```sql
DROP TABLE IF EXISTS acceptance_data CASCADE;

DROP TABLE IF EXISTS closed CASCADE;

DROP TABLE IF EXISTS university CASCADE;

DROP TABLE IF EXISTS country CASCADE;
```

תרגיל 1

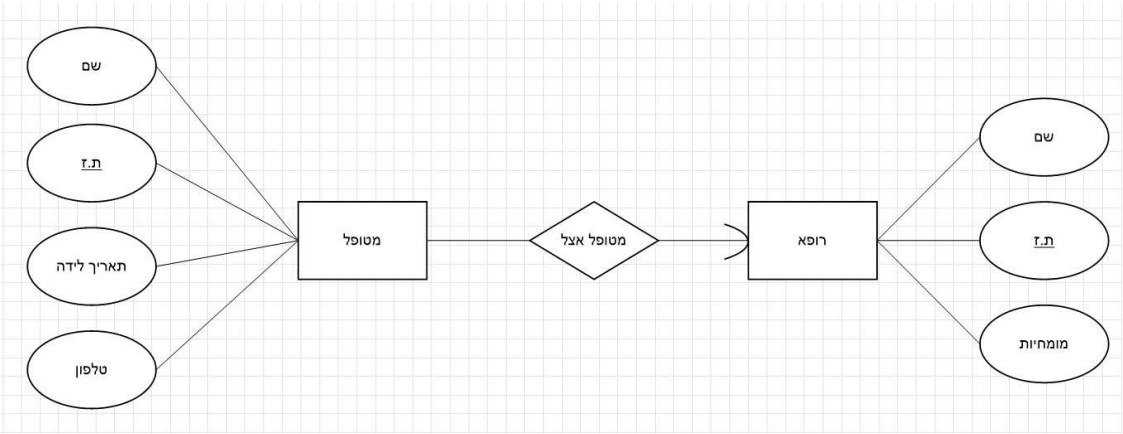
מגישים :

אורן מוטיעי, 321174591

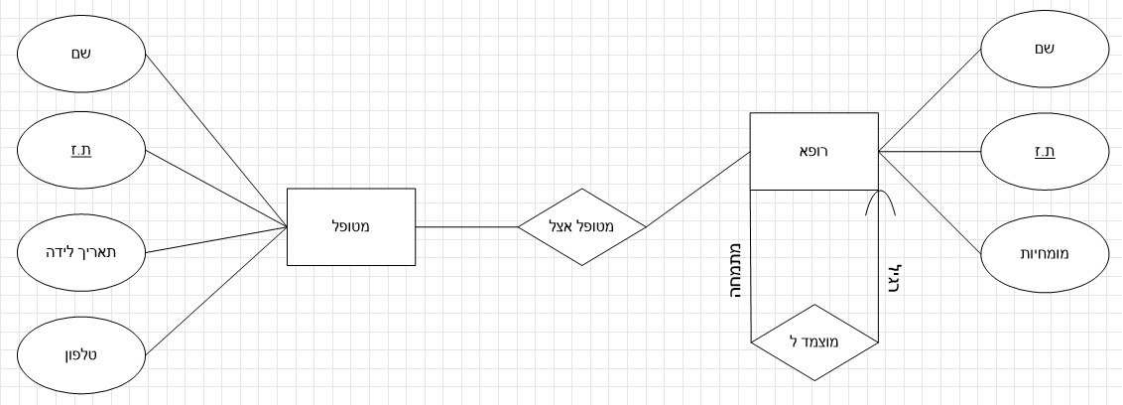
ניר לוי, 206067738

שאלה 1

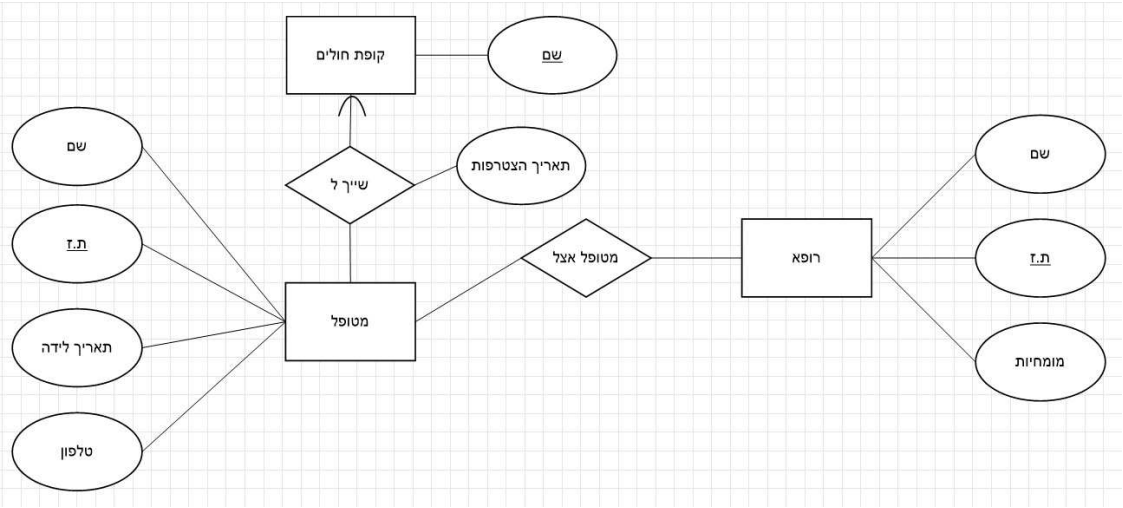
א.



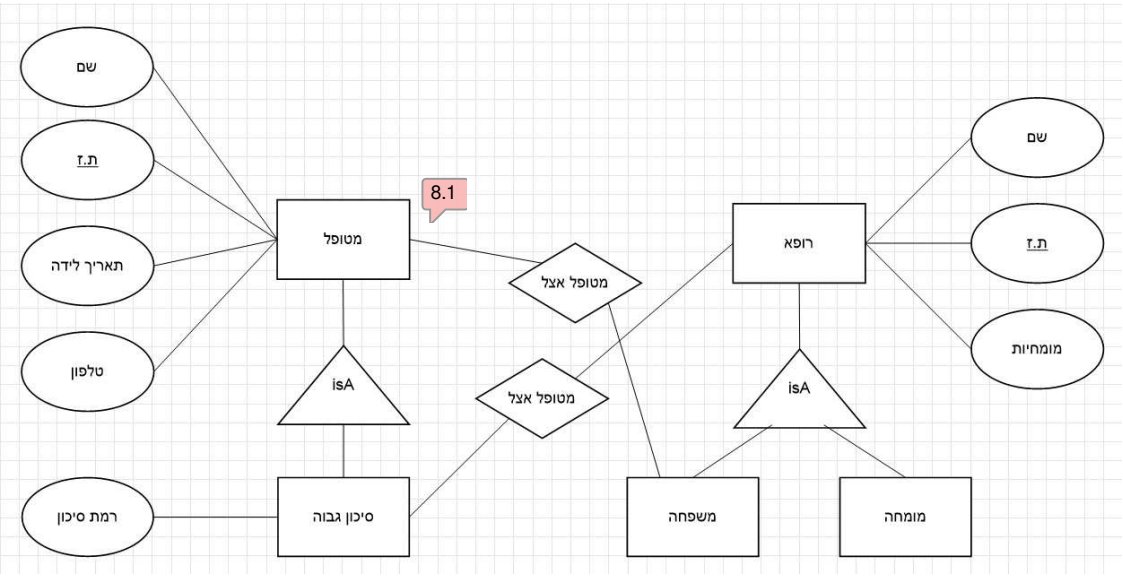
ב.



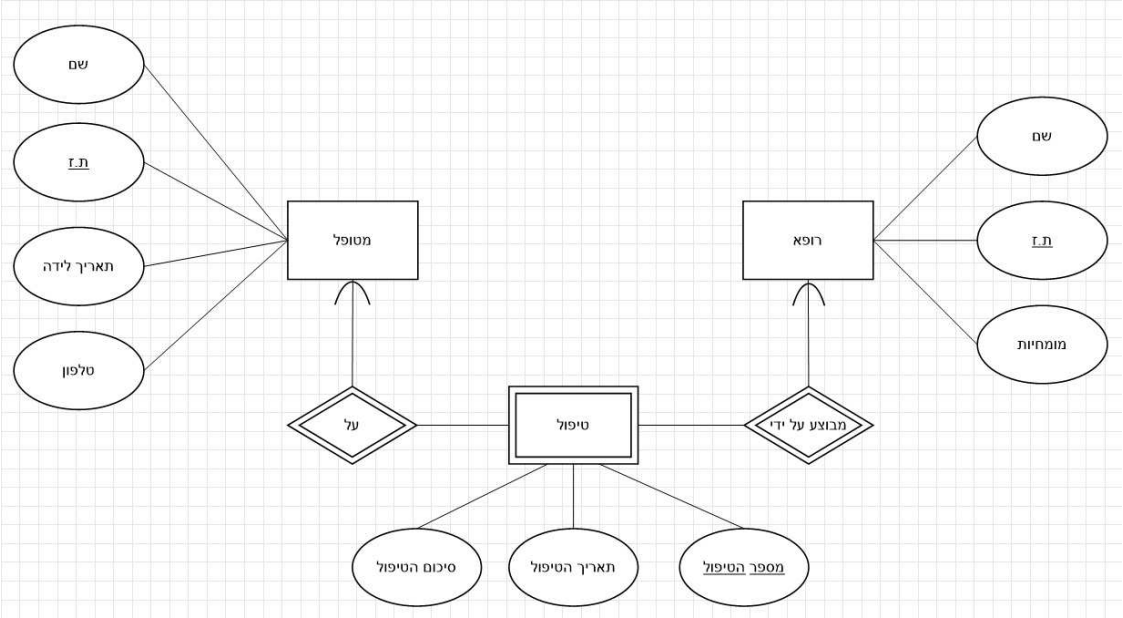
ג.


קופת חולים
שם
שם
ת.ז
תאריך לידה
טלפון
שייך ל
תאריך הצטרפות
מטופל
מטופל אצל
רופא
שם
ת.ז
מומחיות

ד.


שם
ת.ז
תאריך לידה
טלפון
מטופל
8.1
isA
רמת סיכון
סיכון גבוה
מטופל אצל
מטופל אצל
רופא
isA
משפחה
מומחה
שם
ת.ז
מומחיות

ה.



שאלה 2

א. i.

B (e, f)
A (a, b, c, d, e)

ii. לא ניתן לקבוע.

ב. i.

B (e, f) 9.1
A (a, b, c, d, e)

ii. |B| >= |A|.

ג. i.

A (a, c, d)
B (b, e)
F (f)
G (g)
R (c, b, f, g) or R (c, b, f, g)

ii. |A| >= |B|.

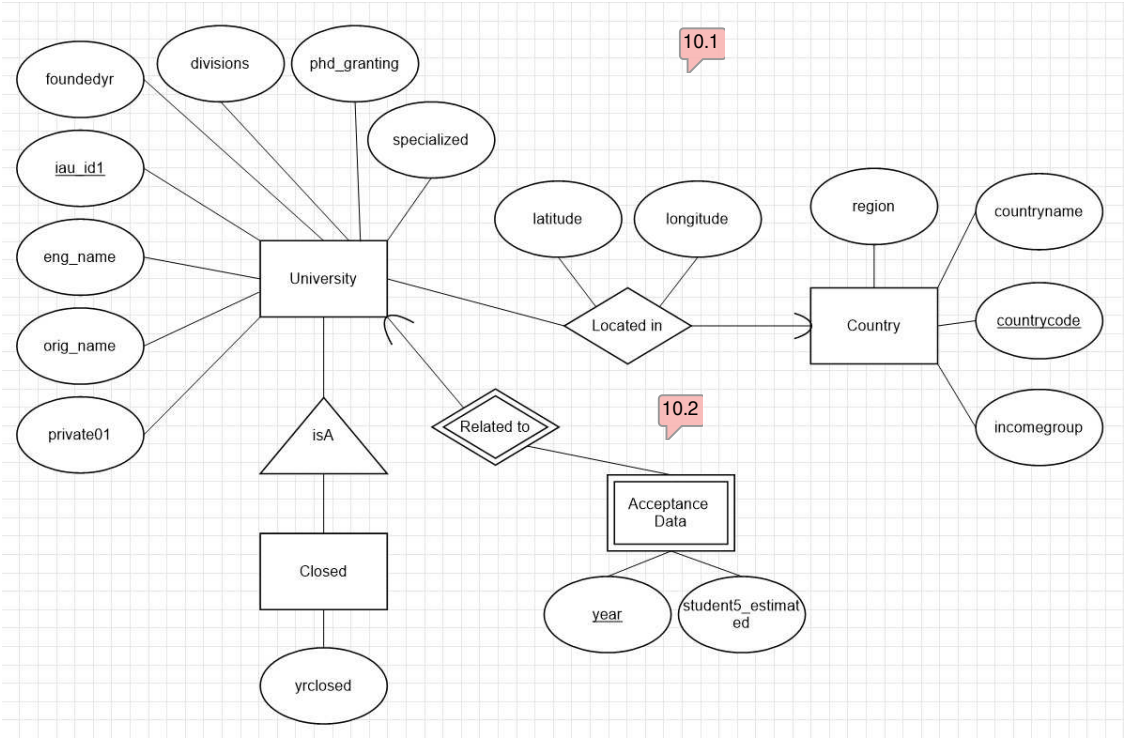
ד. i.

A (a, d)
B (a, b)
C (c, e, A.a, B.a)

ii. |A| >= |B|.

שאלה 3

א.



הסברים:

- נשים לב ששינינו את האטריביוט "country" ל-"countryname" לשם הבהירות.
- כל אוניברסיטה ממוקמת בדיוק במדינה אחת, לכן הוספנו חץ עגול לתוך Country. נשים לב שאוניברסיטה מסוימת יכולה להיות ממוקמת בכמה מדינות, אבל ה-iau_id1 שלהן בכל מדינה שונה ולכן הדיאגרמה עדיין תקפה. לדוגמא, אוניברסיטת Webster באוסטריה מזוהה באמצעות IAU-021164-1 ואוניברסיטת Webster בהולנד מזוהה באמצעות IAU-021162-1.
- מידע על מספר סטודנטים שהתקבלו בשנה מסוימת רלוונטי עבור אוניברסיטה מסוימת, לכן Acceptance Data היא ישות חלשה שניתן לזהות אותה רק בצירוף אוניברסיטה.

ב. נתרגם את הדיאגרמה ליחסים רלציונים:

Country (countrycode, countryname, region, incomegroup)
University (iau_id1, eng_name, orig_name, foundedyr, private01, divisions, phd_granting, specialized, latitude, longitude, countrycode)
Closed (iau_id1, yrclosed)
Acceptance Data (year, student5_estimated, iau_id1)

6 ex1.py

```python
import csv
from io import TextIOWrapper
from zipfile import ZipFile

RELATIONS = ["country", "university", "closed", "acceptance_data"]

# opens file.
country_file = open(f"{RELATIONS[0]}.csv", 'w', encoding='UTF8')
country_writer = csv.writer(country_file, delimiter=",", quoting=csv.QUOTE_MINIMAL)

university_file = open(f"{RELATIONS[1]}.csv", 'w', encoding='UTF8')
university_writer = csv.writer(university_file, delimiter=",", quoting=csv.QUOTE_MINIMAL)

closed_file = open(f"{RELATIONS[2]}.csv", 'w', encoding='UTF8')
closed_writer = csv.writer(closed_file, delimiter=",", quoting=csv.QUOTE_MINIMAL)

acceptance_file = open(f"{RELATIONS[3]}.csv", 'w', encoding='UTF8')
acceptance_writer = csv.writer(acceptance_file, delimiter=",", quoting=csv.QUOTE_MINIMAL)


# process_file goes over all rows in original csv file, and sends each row to process_row()
def process_file():
    with ZipFile('enrollment.zip') as zf:
        with zf.open('enrollment.csv', 'r') as infile:
            countries = set()
            reader = csv.reader(TextIOWrapper(infile, 'utf-8'))
            cur_row = next(reader)
            while cur_row:
                try:
                    next_row = next(reader)
                    is_last_row = True if (cur_row[4] != next_row[4]) else False
                    process_row(cur_row, is_last_row, countries)
                    cur_row = next_row
                except StopIteration:
                    process_row(cur_row, True, countries)
                    break
    country_file.close()
    university_file.close()
    closed_file.close()
    acceptance_file.close()

# processes a single row
def process_row(row, is_last_row, countries):
    country_name = row[0]
    country_code = row[1]
    region = row[2]
    income_group = row[3]
    iau_id1 = row[4]
    eng_name = row[5]
    orig_name = row[6]
    founded_yr = row[7]
    yr_closed = row[8]
    private01 = row[9]
    latitude = row[10]
    longitude = row[11]
    phd_granting = row[12]
    divisions = row[13]
    specialized = row[14]
    year = row[15]
```

```python
    students5_estimated = row[16]

    acceptance_writer.writerow([year, students5_estimated, iau_id1])
    if is_last_row:
        university_writer.writerow([iau_id1, eng_name, orig_name, founded_yr, private01, divisions,
                                    phd_granting, specialized, latitude, longitude, country_code])
        if yr_closed != "":
            closed_writer.writerow([iau_id1, yr_closed])
        if country_code not in countries:
            country_writer.writerow([country_code, country_name, region, income_group])
            countries.add(country_code)


# return the list of all tables
def get_names():
    return RELATIONS


if __name__ == "__main__":
    process_file()
```

Index of comments

5.1 -1/-1 missing check constraints (code='q3.4.missing_check') country should be unique

8.1 0/0 (code='q1.4.missing_link_expert_doctor')

9.1 -0.5/-0.5 missing second option to translate according to the diagnal arrow (code='q2.2.1.only_first_option')

10.1 -1/-1 incomgroup and region are both a finite set of values and should get an entety set (code='q3.1.missing_enteties_for_incomgroup_region')

10.2 -5/-5 your modeling of the enrollment data is wrong (code='q3.1.wrong_modeling_enrollment_data')