Assignment #2

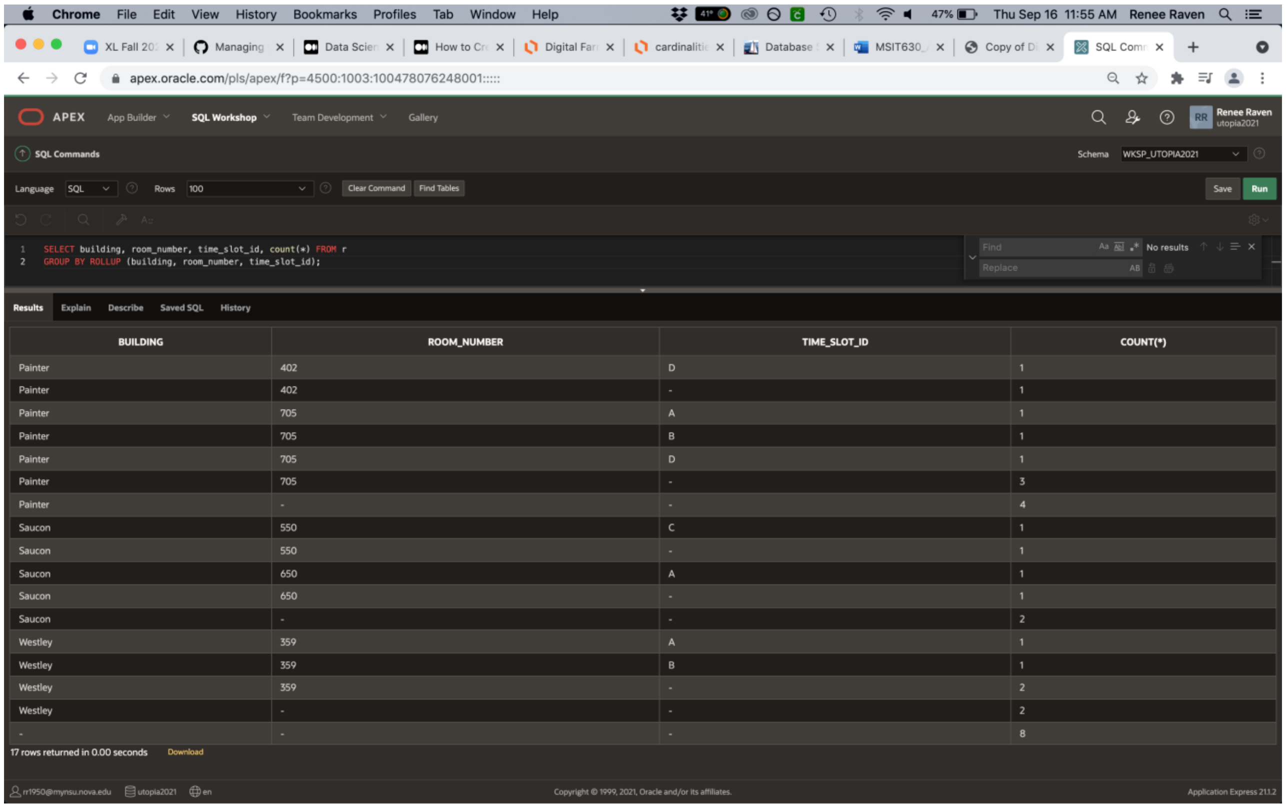
MSIT 630 Database Systems
Total: 50 points
Renee Raven

1. Consider the relation, r, shown below. Give the result of the following query: (**5 points**)

building	**room_number**	**time_slot_id**	**course_id**	**sec_id**
Westley	359	A	BIO-101	1
Westley	359	B	BIO-101	2
Saucon	650	A	CS-101	2
Saucon	550	C	CS-319	1
Painter	705	A	BIO-305	1
Painter	705	D	MU-199	1
Painter	705	B	CS-101	1
Painter	402	D	FIN-201	1

SELECT building, room_number, time_slot_id, count(*)
FROM r
GROUP BY ROLLUP (building, room_number, time_slot_id)

In MySQL, use:
SELECT building, room_number, time_slot_id, count(*)
FROM r
GROUP BY building, room_number, time_slot_id with rollup;



```
SELECT building, room_number, time_slot_id, count(*) FROM r
GROUP BY ROLLUP (building, room_number, time_slot_id);
```

BUILDING	ROOM_NUMBER	TIME_SLOT_ID	COUNT(*)
Painter	402	D	1
Painter	402	-	1
Painter	705	A	1
Painter	705	B	1
Painter	705	D	1
Painter	705	-	3
Painter	-	-	4
Saucon	550	C	1
Saucon	550	-	1
Saucon	650	A	1
Saucon	650	-	1
Saucon	-	-	2
Westley	359	A	1
Westley	359	B	1
Westley	359	-	2
Westley	-	-	2
-	-	-	8

17 rows returned in 0.00 seconds Download

rr1950@mynsu.nova.edu utopia2021 en Copyright © 1999, 2021, Oracle and/or its affiliates. Application Express 21.1.2

Process:

Determine primary key of table and create r.sql file to create table and rData.sql to populate table.

Run r.sql to create table:

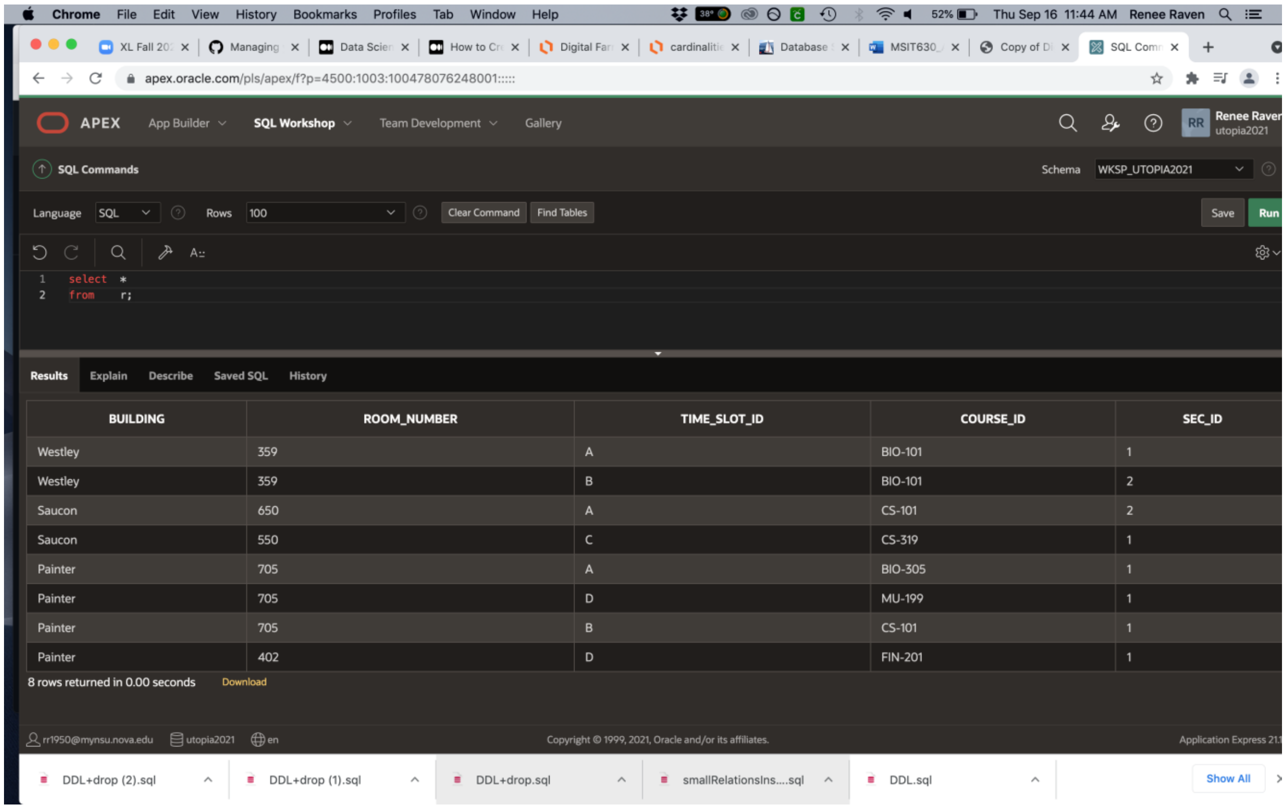
```
delete from building;
delete from room_number;
delete from time_slot_id;
delete from course_id;
delete from sec_id;
insert into r values ('Westley', '359', 'A', 'BIO-101', '1');
insert into r values ('Westley', '359', 'B', 'BIO-101', '2');
insert into r values ('Saucon', '650', 'A', 'CS-101', '2');
insert into r values ('Saucon', '550', 'C', 'CS-319', '1');
insert into r values ('Painter', '705', 'A', 'BIO-305', '1');
insert into r values ('Painter', '705', 'D', 'MU-199', '1');
insert into r values ('Painter', '705', 'B', 'CS-101', '1');
insert into r values ('Painter', '402', 'D', 'FIN-201', '1');
```

Then run rData.sql to populate file:

```
create table r
(building               varchar(15),
 room_number                    varchar(7),
 time_slot_id           varchar(2),
 course_id              varchar(8),
 sec_id                         varchar(8),
 primary key (building, room_number, time_slot_id)
);
```

Run query to check table:

```
select *
from   r;
```



```
select *
from   r;
```

BUILDING	ROOM_NUMBER	TIME_SLOT_ID	COURSE_ID	SEC_ID
Westley	359	A	BIO-101	1
Westley	359	B	BIO-101	2
Saucon	650	A	CS-101	2
Saucon	550	C	CS-319	1
Painter	705	A	BIO-305	1
Painter	705	D	MU-199	1
Painter	705	B	CS-101	1
Painter	402	D	FIN-201	1

8 rows returned in 0.00 seconds

Verified created table matched with question.

2. Explain the distinction between the terms *primary key*, *candidate key*, and *superkey*. **(5 points)**

A superkey is a set of one or more attributes that, taken collectively, allow us to uniquely identify a tuple in the relations. For instance, one's driver's license number is a superkey. Additionally, one's driver's license number and last name combined is a superkey because superkeys do allow extraneous attributes. Last name and birthday are not a super key as there may be more than one person with those attributes.

Candidate keys minimal superkeys, In essence, candidate keys are superkeys in which none of the subset could act as a superkey independently. Unlike super keys, candidate keys do not allow extraneous attributes. Therefore, a driver's license number may be a candidate key but a combination of driver's license number and last name would not be a candidate key.

A primary key is selected from all potential candidate keys as the primary identifier of the tuple in a relation. The primary key is unique for every tuple and cannot be null.

3. Explain *integrity constraints* and how they ensure data integrity within a database. **(5 points)**

Integrity constraints prevent accidental damage to the database. Even if a user is authorized and they make authorized changes, there is a risk that those changes could result in a loss of data consistency. However, integrity constraints should be applied judiciously as they do take up resources and can be costly to test. The text discusses three main integrity constraints: not null, unique, and check.

The not null constraint specifies attributes that must be present for each tuple. For example, a primary key must satisfy the non null constraint. The unique constraint specifies each attribute in the tuple must be unique. However, the unique constraint allows null values unless the non null constraint is applied. An example might be a relation of Florida residents where every driver's

license number must be unique, but not every resident requires a driver's license number. The check clause constraint ensures a specific requirement, set by the DLL for the table, is met. For example, a relation of songs for exercise might require a song to have a bpm >= 120:
check (bmp >= 120);

4. Explain what an *index* is in SQL, how used, and their importance in processing transactions. **(5 points)**

Searching through relation(s)/database takes memory, energy, and time. An index on an attribute of a relation is a data structure that allows the database system to find those tuples in the relation that have a specified value for that attribute efficiently, without scanning through all the tuples of the relation. In theory, a database system could automatically create indices, but it is far more efficient for a programmer to decide which item to index. It is important to remember that a SQL index adds another data structure to the relation, increasing space demands and potentially slowing down updates.

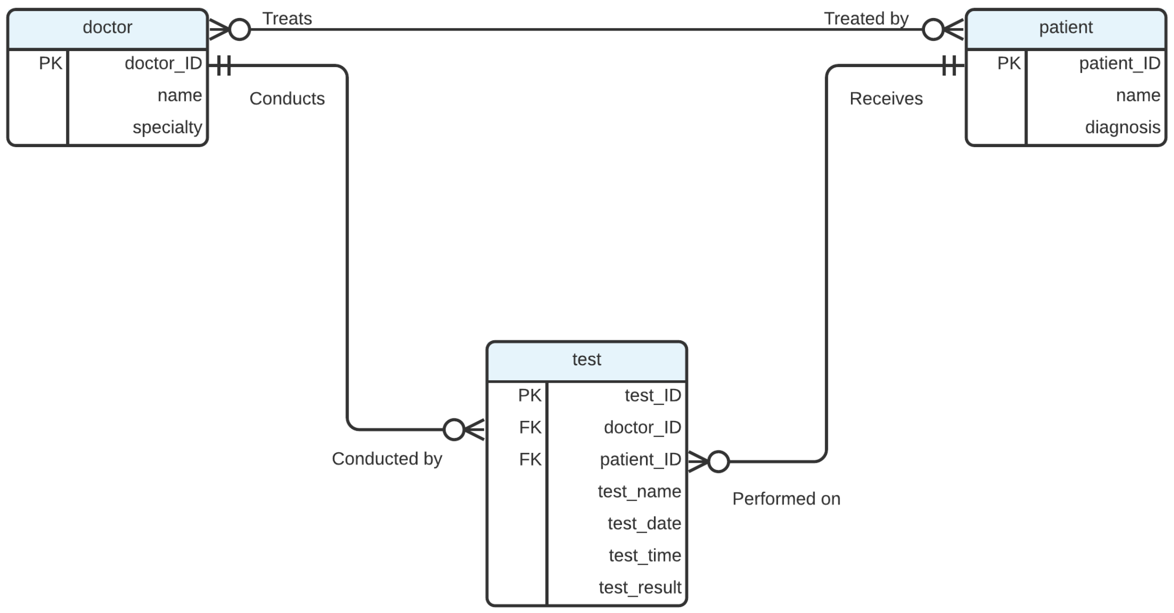
Consider a relation that logs the test name, test result, ordering doctor, and patient for any number of test occurrences. If we create an index on the attribute test_name of relation test, the database system can find the record with any specified test_name value, such as 'CBC' or 'Chest_X_Ray,' directly by checking the query against the created (see below) test_index without reading all the tuples of the test relation. To create this index, the programmer could enter a DLL command:
Create index test_index on test (test_name);

While they are redundant data structures and not required, SQL indices improve performance and efficiency and are an important consideration in database design. The SQL query processor automatically uses the index whenever applicable to the query. Searching through an index takes less memory, time, and resources than searching through the entire database (every tuple).

5. Construct an E-R diagram for a hospital with a set of patients and a set of medical doctors. Associate with each patient a log of the various tests and examinations conducted by the doctors. (**8 points**)

Hospital Entity Relationship Diagram

Doctors, Patient, Log



6. Explain the distinction between disjoint and overlapping constraints. Provide an example for each constraint. (**4 points**)

In a disjoint constraint, an entity may belong to, at most, one specialized entity set while in an overlapping constraint, an entity may belong to multiple specialized entity sets.

Referring to the above ERD, in an overlapping constraint both doctor and patient would fall under specializations of person. Conversely, in a disjoint constraint the result could be positive or negative for specializations of test_result, but not both. The book uses the example of student and employee as specializations of person for overlapping. For disjoint, the book uses instructor and secretary as specializations of employee, pg. 272.

7. Explain the distinction between total and partial constraints. Provide an example for each constraint. (**4 points**)

The participation of an entity set E in a relationship set R is said to be total if every entity in E must participate in at least one relationship in R. If it is possible that some entities in E do not participate in relationships in R, the participation of entity set E in relationship R is said to be partial.

Referring to the above ERD, every test must be conducted by a doctor, therefore the participation of doctor in the conduct relationship to a test is total. In contrast, a doctor doesn’t need to conduct any tests. It is possible that only some of the doctor entities are related to the test entity set through the conduct relationship. Therefore, the participation of the doctor in the conduct relationship is partial.

8. Explain each of the mapping cardinalities, what they are used for, and how they contribute to describing relationships on E-R diagrams. **(4 points)**

Cardinality ratios (also known as mapping cardinalities) indicate the number of entities to which another entity can be associated via a relationship set. Mapping cardinalities are especially useful in describing relationships between two entity sets (binary relationships), but it can also be help describe relationships among more than two entity sets.

When describing a binary relationship, there are four types of mapping cardinalities: one-to-one, one-to-many, many-to-one, and many-to-many. It is useful to note that some elements in either set may not be mapped to any element in the other set.

Given two sets, A and Z:

In a one-to-one cardinality, an entity in A is associated with, at most, one entity in Z, and an entity in Z is associated with, at most, one entity in A.

In a one-to-many cardinality, an entity in A is associated with any number (0, 1, or many) of entities in Z. But an entity in Z can be associated with, at most, one entity in A.

In a many-to-one cardinality, an entity in a is associated with, at most, one entity in Z. An entity in Z, however, may be associated with 0, 1, or many entities in A.

In a many-to-many cardinality, an entity in A is associated with 0, 1, or many entities in Z. And an entity in Z is associated with any number (0, 1, or many) entities in A.

9. What are the components of a data warehouse? What are the issues to be addressed in building a warehouse? (**5 points**)

The components of a data warehouse include the data loaders to gather the data from multiple data sources, the DBMS and the query and analysis tools used to extract data from the DBMS. Extract, transform, and load (ETL) are the tasks used to get the data from sources, transform the data as needed, and load the data into the data warehouse.

Some issues to address when building a data warehouse include:

1. When and how to gather data (source-driven or destination-driven architectures, how to synchronize data).
2. What schema to use and how to integrate data sources using different schemas.
3. Data transformation and cleansing involves correcting and preprocessing data to compensate for common errors.
4. How to propagate data and relation updates across the data warehouse.
5. What data to summarize when raw data is prohibitively large.

10. The large volume of data generated on the Internet such as social-media data requires a very high degree of parallelism on both data storage and processing. Please explain the MapReduce paradigm for parallel processing. (**5 points**)

Parallel databases are databases run on many, often hundreds, of machines referred to as a cluster. The parallel processing and storage allow for vast volumes of data, such as social-media generated files. The MapReduce paradigm allows processing to be broken into sections and processed in aggregation. As the text states "some processing, identified by the map()

function, is applied to each of a large number of input records, and then some form of aggregation, identified by the reduce() function, is applied to the result of the map function.” p. 483. The MapReduce function is run in parallel across a cluster of machines. Often, each map() function is run on part of the data (files, parts of files) and then the reduce() function runs on many machines simultaneously. Then the results of the reduce() function are merged and sorted, eventually producing parallel file output.