Children’s House Montessori

Student Information System (SIS)

December 2020 // TEAM LCCB

TABLE OF CONTENTS ——————————————————————————

1 — ABOUT TEAM LCCB AND THE CMS SIS PROJECT ————————

Team LCCB was established in September 2020 to design and propose an information system suited to handle CMS's operating objectives, agreeing to a tentative schedule and contract[1] before commencing work. LCCB is composed of the following members: Lucas **L**ower, Jennifer **C**arr, ZhiYong **C**hen, and Randall **B**ecker.

The CMS student information system, or SIS, is built on a relational database system, Oracle 10g, and is accessed using the SQL querying language through Oracle's SQL Developer program. A user manual[2] detailing the use of this program in the context of the CMS database has been created, as well as a set of frequently used queries pulling commonly tabulated information.

[1] Team LCCB contract included as *Appendix D*
[2] Database user manual included as *Appendix A*

2 — CLIENT INTERVIEW REPORT ————————————————————————

Group members (all present) met with the client on November 12th to discuss a preliminary version of the database design, and to receive feedback on how it could further meet the needs of CMS. Eliminating some complexities of the program data was discussed, as well as the associated fee and tuition pricing information. Student evaluations were discussed and simplified to become a component of the enrollment file.

3 — FINAL ENTITY RELATIONSHIP DIAGRAM ———————————————

An ERD is an *entity relationship diagram*. It is a blueprint to display where the data will be stored and how the data will be related to each other in the entities a.k.a. tables. An *entity* is some collection of closely related data (each individual piece of data being an *attribute*), such as a child's registration information, a teacher evaluation file, or a child's attendance record. The *relationships* between them capture the various business processes currently in place at CMS, and serves to codify them for future efficiency.

On the next page, the final revision of the ERD is shown in *Figure 1*. There are some changes to be noted from the initial revision of the ERD[3]:

We had planned to have many more tables than what was necessary. We planned to have a separate table to store the alumni information and decided to move this information into the table with the student information, and store a graduation date there. We also decided to store whether or not the student was on a waiting list in the enrollment table instead of using a separate table. An enrolled table was also removed since the information is stored in the enrollments with the newer attribute of the waiting list status. There was a plan of having separate tables for each program type instead of the newer concept of using information from added rows in the table. The fees are now stored in the program table instead of a fees table. There is now no payroll table since that wasn't requested for this project. Student evaluations are now attributes in the enrollment table. There is only one evaluation table, and that is for the evaluation of the teachers. The final ERD is less complex and a cleaner looking setup.

[3] Initial ERD revision included as Appendix E

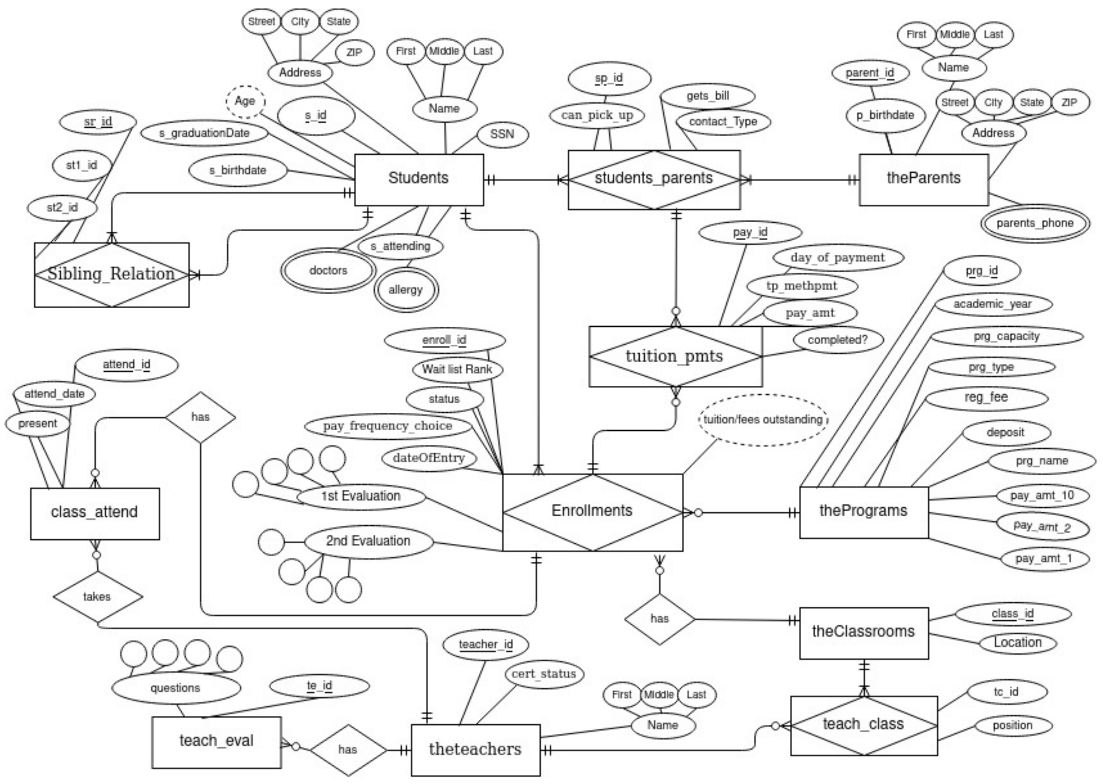


Figure 1 — Final revision of the CMS database ERD, a blueprint of the data's structure and relationships

4 — LOGICAL DATABASE MODEL ——————————————————————

The *logical database model* provides a means to visualize the functional relations and constraints of the database's tabular structure (entities from the previous section's ERD). The logical model produced by LCCB is shown below in *Figure 2*. It represents the "physical" database structure that stores CMS data, and can be used to aid in the construction of new queries when necessary.

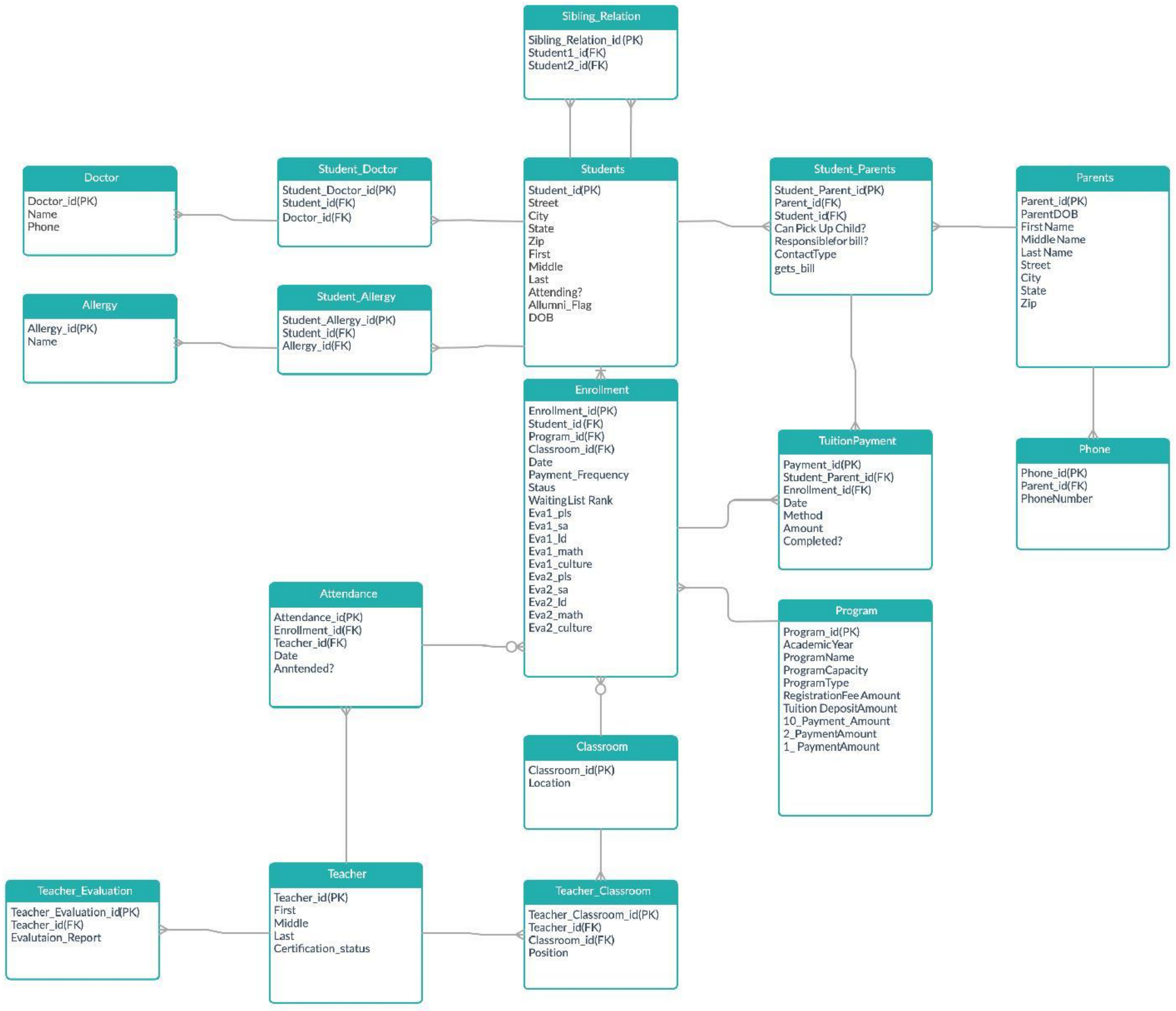


Figure 2 — Logical model of the CMS student database

5 — NOTE ON REFERENTIAL INTEGRITY CONSTRAINTS ————————

Referential integrity is a property of data stating that all of a dataset's contained references must be valid. In the context of relational databases, it requires that if a value of one attribute (column) of a relation (table) references a value of another attribute (either in the same or a different relation), then the referenced value must exist. We need referential integrity constraints because they keep users from introducing errors into the database. We can also merge tables based on the referential integrity built into the database.

In the CMS project there are many referential integrity constraints [note that entity/table names are in italic, and that *PK* and *FK* refer to *primary key* (the main identifier of the entity) and *foreign key* (an identifier referring to another entity), respectively].

- In *Sibling_Relation*, Student1_id and Student2_id both are FK, and refer to Student_id which is the PK in *Student*.
- In *Student_Doctor*, Student_id as FK refer to Student_id PK in *Students* and Doctor_id as FK refer to Doctor_id in *Doctor*.
- In *Student_Allergy*, Student_id as FK refer to Student_id PK in *Students* and Allergy_id as FK refer to Allergy_id in *Allergy*.
- In *Student_Parent*, Student_id as FK refer to Student_id PK in *Students* and Parent_id as FK refer to Parent_id in *Parent*.
- In *Phone*, Parent_id as FK refers to Parent_id PK in *Parent*.
- In *Enrollment*, Student_id as FK refer to Student_id PK in *Students*, Program_id as FK refer to Program_id PK in *Program* and Classroom_id as FK refer to Classroom_id PK in *Classroom*.
- In *TuitionPayment*, Student_Parent_id as FK refers to Student_Parent_id PK in *Student_Parents* and Enrollment_id as FK refers to Enrollment_id PK in *Enrollment*.
- In *Attendance*, Enrollment_id as FK refers to Enrollment_id PK in *Enrollment* and Teacher_id as FK refers to Teacher_id PK in *Teacher*.
- In *Teacher_Classroom*, Teacher_id as FK refers to Teacher_id PK in *Teacher* and Classroom_id as FK refers to Classroom_id PK in *Classroom*.
- In *Teacher_Evaluation*, Teacher_id as FK refers to Teacher_id PK in *Teacher*.

6 — IMPLEMENTATION DETAILS ————————————————————

The implementation of the database was straightforward once an ERD design had been created and agreed upon. ERD discussions took the majority of group meetings early on, simply because there were many different ways to achieve the requested querying capabilities. Our discussions on the design to handle a given problem centered on which approach would 1) store enough data to allow a query to return the correct results, and 2) store said data in such a way that the queries written could be simple and concise.

To design the ERD, a Google Drive-connected diagramming app, draw.io, was used. It included preset shapes for common ERD components like entities, attributes, relationships, and cardinality indicators. This allowed easier brainstorming of the ERD during group meetings, as changes could be made as they were discussed.

The logical model was also created with draw.io and was kept updated alongside the development of the SQL syntax to create the database's table structure (as name changes due to length requirements were made).

7 — DATABASE SETUP & TESTING ———————————————————

The database is built using Oracle 10g, a dependable SQL relational database system. Once the table creation syntax[4] was created, it was committed to the Oracle database system and test data[5] was populated. Basic test data was inserted to each table first, with further sample data being added as needed to test various requested queries. As there were multiple ways to organize each query, each team member was assigned 6 of the 9 requested queries. Each member then independently wrote their queries, and results and implementations were discussed and the best query was found as a result.

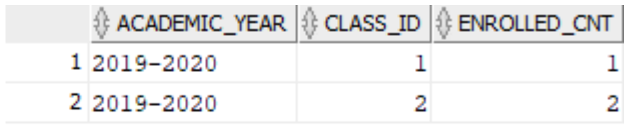
8 — DATA ACCESS & FILTERING CAPABILITIES ————————————

CMS has asked us to show some of the querying capabilities of SQL Developer and our database by providing a few common queries they can run in the database. Please see these frequently asked questions, their associated SQL statements, and example output on the next page.

[4] Oracle SQL syntax for creating the database on a fresh install is included as *Appendix B*
[5] Test data INSERT statements included as *Appendix C*

1. How many children are enrolled in each classroom?

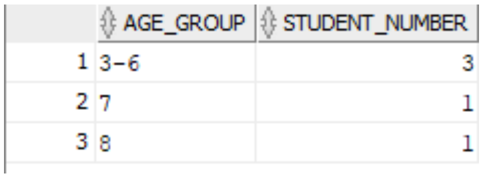
```
SELECT
    academic_year, class_id, count(s_id) as enrolled_cnt
FROM
    enrollments, theprograms
WHERE
    enrollments.prg_id = theprograms.prg_id
AND
    class_id IS NOT NULL
GROUP BY
    academic_year, class_id
ORDER BY
    academic_year;
```



	ACADEMIC_YEAR	CLASS_ID	ENROLLED_CNT
1	2019-2020	1	1
2	2019-2020	2	2

2A. How many children are on the waiting list for each age group (3-6, 7, and 8)? (We assume that only in the current year a waitlist will exist for students and in previous years, only enrolled and not enrolled status are considered).

```
SELECT
    age_group, count(s_id) as student_number
FROM (
    SELECT
        s_id,
        (CASE age
            WHEN 3 THEN '3-6'
            WHEN 4 THEN '3-6'
            WHEN 5 THEN '3-6'
            WHEN 6 THEN '3-6'
            WHEN 7 THEN '7'
            WHEN 8 THEN '8'
            ELSE 'OTHER'
        END) as age_group
    FROM (
        SELECT
            DISTINCT(s_id), trunc((sysdate-s_birthdate)/365.25) as age
        FROM
            students
        NATURAL JOIN
            enrollments
        WHERE
            status = 'waitlist'
    )
)
GROUP BY age_group
ORDER BY age_group;
```

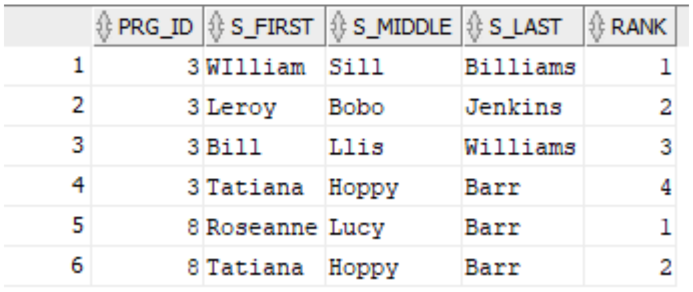


	AGE_GROUP	STUDENT_NUMBER
1	3-6	3
2	7	1
3	8	1

2B. What is the rank order of waitlist students for each program?

```
SELECT
    prg_id, s_first, s_middle, s_last,
    rank() OVER(
        PARTITION BY prg_id
        ORDER BY prg_id, s_attending desc, s1.sib_attending desc,
            dateofentry, s_birthdate) as rank
FROM (
    SELECT
        s_id, s_first, s_middle, s_last, s_attending, s_birthdate,
        (CASE sib_indicator
            WHEN 1 THEN  'Y'
            WHEN 2 THEN  'Y'
            WHEN 3 THEN  'Y'
            WHEN 4 THEN  'Y'
            WHEN 5 THEN  'Y'
            ELSE 'N'
        END) as sib_attending
    FROM students, (
        SELECT
            st, sum(sibling_indicator) as sib_indicator
        FROM (
            SELECT
                s1.s_id st,
                (CASE s2.s_attending
                     WHEN 'Y' THEN 1
                     WHEN 'N' THEN 0
                END) as sibling_indicator
            FROM
                Sibling_relation, students s1, students s2
            WHERE
                sibling_relation.st1_id = s1.s_id
            AND
```

```
                sibling_relation.st2_id = s2.s_id
        )
        GROUP BY st
        HAVING SUM(sibling_indicator) > 0
    )
    WHERE st (+)= s_id
) s1
JOIN (
    SELECT
        s_id, enrollments.prg_id, dateofentry
    FROM
        theprograms, enrollments
    WHERE
        enrollments.prg_id = theprograms.prg_id
    AND
        enrollments.status = 'waitlist'
    AND
        theprograms.academic_year = '2019-2020'
) e
ON
    e.s_id =s1.s_id;
```



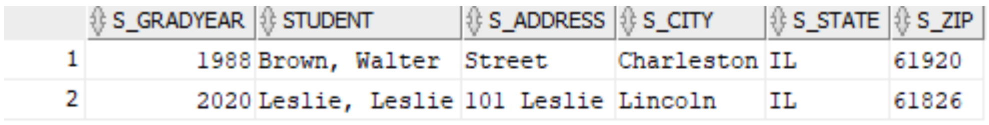
	PRG_ID	S_FIRST	S_MIDDLE	S_LAST	RANK
1	3	WIlliam	Sill	Billiams	1
2	3	Leroy	Bobo	Jenkins	2
3	3	Bill	Llis	Williams	3
4	3	Tatiana	Hoppy	Barr	4
5	8	Roseanne	Lucy	Barr	1
6	8	Tatiana	Hoppy	Barr	2

As this query is quite long, some explanation is in order:

1. Self Join the Student Table as s1, s2 and get the table for sibling attending using sibling indicator 0 and 1, 0 being not attending and 1 being attending. For this table, we group by by the s1.id. If sum(sibling_indicator)>0 there exists a sibling who is attending the school. If sum = 0 then there exists siblings but all of them are not attending the school.
2. Outer Join step 1 table with Students and get the label for sibling attending. This table takes over the s1 alias from step 1.
3. Inner Join enrollment table and program table, set the academic year to this year, and select the student on the waitlist. This table is set to alias ‘e’.
4. Join s1 and e, partitioning by different programs, then sort the data by the 4 conditions CMS uses to rank order their waitlist.

3. Who are the students that graduated from CMS? What are their current addresses?

```
SELECT
    EXTRACT(year FROM s_graduationdate) as s_gradyear,
    s_last || ', ' || s_first as student,
    s_address,
    s_city,
    s_state,
    s_zip
FROM
    students
WHERE
    s_attending = 'N'
AND
    s_graduationdate IS NOT NULL
ORDER BY
    s_gradyear, s_last;
```



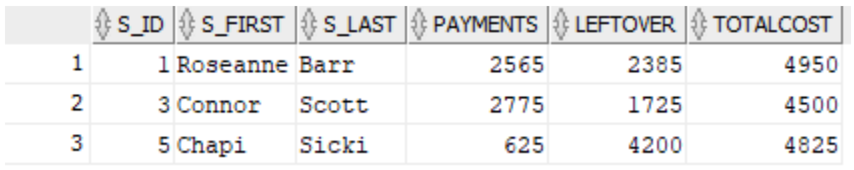
	S_GRADYEAR	STUDENT	S_ADDRESS	S_CITY	S_STATE	S_ZIP
1	1988	Brown, Walter	Street	Charleston	IL	61920
2	2020	Leslie, Leslie	101 Leslie	Lincoln	IL	61826

4. How many parents use a particular payment method (checks, money orders, cash, monthly autopay, and bank checks)?

```
SELECT
    COUNT(DISTINCT p.parent_id) total, tp.tp_methpmt the_method
FROM
    student_parents sp, tuition_pmts tp, theparents p
WHERE
    tp.sp_id = sp.sp_id And sp.parent_id = p.parent_id
GROUP BY
    tp.tp_methpmt;
```

5. What are the outstanding tuition amounts for each child?

```
SELECT
    s.s_id, s.s_first, s.s_last, results.payments,
    (CASE enrl.pay_frequency_choice
        WHEN 10 THEN pay_amt_10*10
        WHEN 2 THEN pay_amt_2*2
        ELSE prg.pay_amt_1
    END + prg.reg_fee) - results.payments as leftover,
    (CASE enrl.pay_frequency_choice
        WHEN 10 THEN pay_amt_10*10
        WHEN 2 THEN pay_amt_2*2
        ELSE prg.pay_amt_1
    END) + prg.reg_fee as totalCost
FROM
    enrollments enrl, theprograms prg, students s, (
        SELECT
            tp.enroll_id, SUM(tp.pay_amt) as payments
        FROM
            tuition_pmts tp
        GROUP BY tp.enroll_id
    ) results
WHERE
    results.enroll_id = enrl.enroll_id
AND
    enrl.prg_id = prg.prg_id
AND
    enrl.s_id = s.s_id;
```



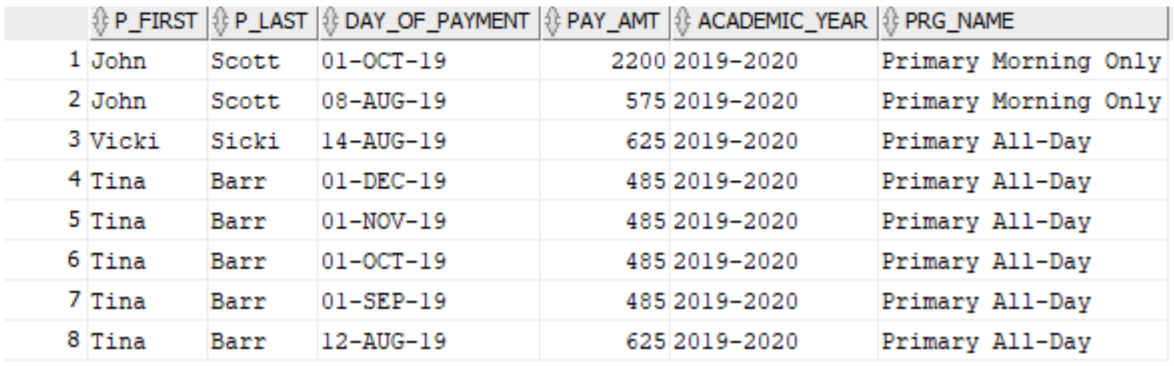
	S_ID	S_FIRST	S_LAST	PAYMENTS	LEFTOVER	TOTALCOST
1	1	Roseanne	Barr	2565	2385	4950
2	3	Connor	Scott	2775	1725	4500
3	5	Chapi	Sicki	625	4200	4825

6. What is the amount that was already paid by a parent (all dates and amounts should be listed) for a particular school year or program?

The below query lists the overall payments made to CMS over all time, merged with enrollment and parent information (i.e. a dump of all payments received).

```
SELECT
    prnt.p_first, prnt.p_last, tp.day_of_payment, tp.pay_amt, p.academic_year,
    p.prg_name
FROM
    tuition_pmts tp, student_parents sp, enrollments e, theparents prnt,
    theprograms p
WHERE
    tp.sp_id = sp.sp_id
AND
    tp.enroll_id = e.enroll_id
```

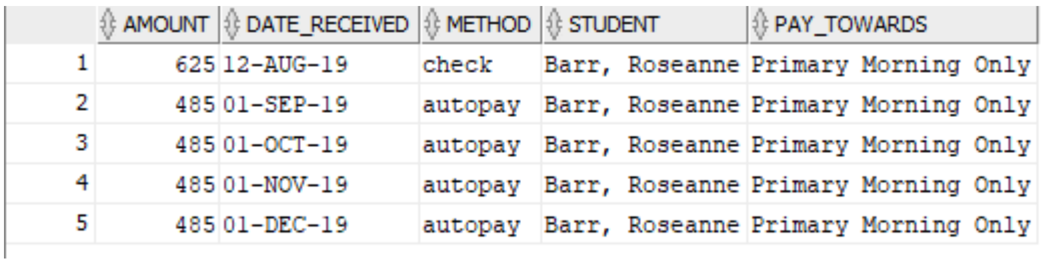
```
AND
    e.prg_id = p.prg_id
AND
    sp.parent_id = prnt.parent_id;
```



	P_FIRST	P_LAST	DAY_OF_PAYMENT	PAY_AMT	ACADEMIC_YEAR	PRG_NAME
1	John	Scott	01-OCT-19	2200	2019-2020	Primary Morning Only
2	John	Scott	08-AUG-19	575	2019-2020	Primary Morning Only
3	Vicki	Sicki	14-AUG-19	625	2019-2020	Primary All-Day
4	Tina	Barr	01-DEC-19	485	2019-2020	Primary All-Day
5	Tina	Barr	01-NOV-19	485	2019-2020	Primary All-Day
6	Tina	Barr	01-OCT-19	485	2019-2020	Primary All-Day
7	Tina	Barr	01-SEP-19	485	2019-2020	Primary All-Day
8	Tina	Barr	12-AUG-19	625	2019-2020	Primary All-Day

The below query lists the payments received from a given parent during a given school year (when running this query, SQL developer prompts the user to replace the named parameters, which start with a colon, with the parent and year information).

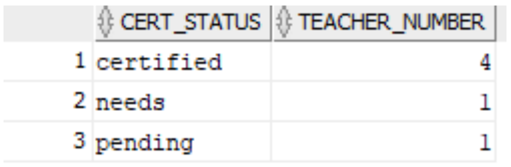
```
SELECT
    tp.pay_amt as amount,
    tp.day_of_payment as date_received,
    tp.tp_methpmt as method,
    s_last || ', ' || s_first as student,
    pr.prg_name as pay_towards
FROM
    tuition_pmts tp
JOIN
    student_parents sp ON sp.sp_id = tp.sp_id
JOIN
    students s ON s.s_id = sp.s_id
JOIN
    enrollments e ON e.enroll_id = tp.enroll_id
JOIN
    theprograms pr ON pr.prg_id = e.enroll_id
WHERE
    sp.parent_id = :parent_id
AND
    pr.academic_year = :acad_year
ORDER BY
    tp.DAY_OF_PAYMENT;
```



	AMOUNT	DATE_RECEIVED	METHOD	STUDENT	PAY_TOWARDS
1	625	12-AUG-19	check	Barr, Roseanne	Primary Morning Only
2	485	01-SEP-19	autopay	Barr, Roseanne	Primary Morning Only
3	485	01-OCT-19	autopay	Barr, Roseanne	Primary Morning Only
4	485	01-NOV-19	autopay	Barr, Roseanne	Primary Morning Only
5	485	01-DEC-19	autopay	Barr, Roseanne	Primary Morning Only

7. How many teachers are certified, going through certification, and need certification?

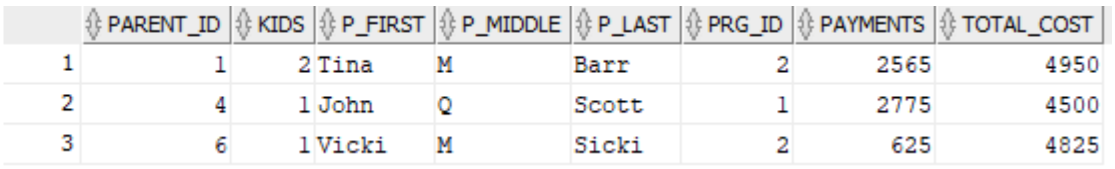
```
SELECT
    cert_status, COUNT(teacher_id) as teacher_number
FROM
    theteachers
GROUP BY
    cert_status;
```



	CERT_STATUS	TEACHER_NUMBER
1	certified	4
2	needs	1
3	pending	1

8. Which parents have the top 5 highest outstanding tuition amounts?

```
SELECT
    results.parent_id, prnt.kids, results.p_first, results.p_middle,
    results.p_last, results.prg_id, results.payments,
    results.total_cost
FROM (
    SELECT *
    FROM (
        SELECT
            prnt.parent_id, prnt.p_first, prnt.p_middle, prnt.p_last,
            prg.prg_id, results.payments,
            (CASE e.pay_frequency_choice
                WHEN 10 THEN prg.pay_amt_10*10
                WHEN 2 THEN prg.pay_amt_2*2
                ELSE prg.pay_amt_1
            END) + prg.reg_fee as total_cost
        FROM
            theparents prnt, student_parents sp, students s,
            enrollments e, theprograms prg, (
                SELECT
                    tp.enroll_id, sp.sp_id, SUM(tp.pay_amt) as
                    payments
             FROM
                    student_parents sp, tuition_pmts tp
                WHERE
                    tp.sp_id = sp.sp_id
                GROUP BY
                    tp.enroll_id, sp.sp_id
            ) results
        WHERE sp.parent_id = prnt.parent_id
        AND results.sp_id = sp.sp_id
        AND results.enroll_id = e.enroll_id
        AND sp.s_id = s.s_id
        AND e.s_id = s.s_id
        AND e.prg_id = prg.prg_id
    )
    WHERE ROWNUM <= 5
) results, (
    SELECT
        COUNT(sp.sp_id) as kids, p.parent_id, p.p_first, p.p_middle,
        P.p_last
    FROM
        theparents p, student_parents sp
    WHERE sp.parent_id = p.parent_id
    GROUP BY
        p.parent_id, p.p_first, p.p_middle, p.p_last
) prnt
WHERE
    results.parent_id = prnt.parent_id;
```

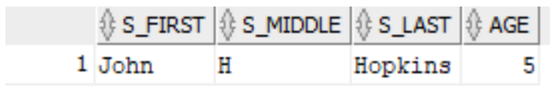


	PARENT_ID	KIDS	P_FIRST	P_MIDDLE	P_LAST	PRG_ID	PAYMENTS	TOTAL_COST
1	1	2	Tina	M	Barr	2	2565	4950
2	4	1	John	Q	Scott	1	2775	4500
3	6	1	Vicki	M	Sicki	2	625	4825

9. What children have birthdays this month?

```
SELECT
    s_first, s_middle, s_last,
    TRUNC((SYSDATE - s_birthdate)/365.25) as age
FROM
    students
WHERE
```

```
    s_attending = 'Y'
AND
    EXTRACT(month FROM s_birthdate) = EXTRACT(month FROM sysdate);
```



	S_FIRST	S_MIDDLE	S_LAST	AGE
1	John	H	Hopkins	5

9 — LESSONS & CONCLUSIONS ————————————————————————

Along the way, we learned various lessons about working on a database information system as a group. Here are some lessons and conclusions from each group member:

Lucas Lower

I realized that having a solid plan and timeline is key to working efficiently as a group. In some group settings in the past, we've decided to "wing it" so to speak, and discuss meeting agendas and action items at the time of the meeting. In this group, we were more organized and had a clear topic/agenda to discuss in each meeting. This made better use of time and made the project much less stressful and easier to manage. In the future, I will definitely not forget the benefits of clear agendas, definitive work assignments, and frequent meetings!

Jennifer Carr

This project has been a fun one. I am grateful that I have been able to work with this team. Everyone has been able to bring a perspective that I wouldn't have thought of. I am not an organized person. I learned that planning out a goal schedule like the one that we have had for this project helps break down tasks. I am not trying to figure out what I should work on next. Also, I tend to make things more complicated than it needs to be. Others in my group came up with ideas that were less complex. For example, the student evaluations didn't need to have a separate table. Our contract stated that we would work on separate parts in a volunteer-like manner. We all chose parts that we felt comfortable with. This project was a great experience for me.

ZhiYong Chen

In the past, I didn't particularly like group projects, but after joining LCCB, I found that many people have different ideas, which made me feel very open, and comparing these ideas made me less stubborn. In addition, I realized that good works come from good ideas, if you can complete a great design from the beginning. So when the application is finally designed, it will be a piece of cake. Generally speaking, my team members work hard and are very punctual. Although the time schedule for this semester is very messy, I am very satisfied with everyone and I totally believe that the project will be very impressive.

Randall Becker

Over the course of this project, I have learned how to better manage my time and set appropriate expectations before pen hits paper. Sadly, I was an exception to the group because I was unable to attend the first day of class and joined the group after what could be considered an afterthought. For the most part however our team has persevered in getting the work done when we needed with plenty of healthy discussion along the way. To each of the team members of LCCB I hope a long a prosperous life in all their future endeavors. Thank you for the opportunity to let us work together in-class during this very weird semester.

A Brief Explanation of SQL Developer

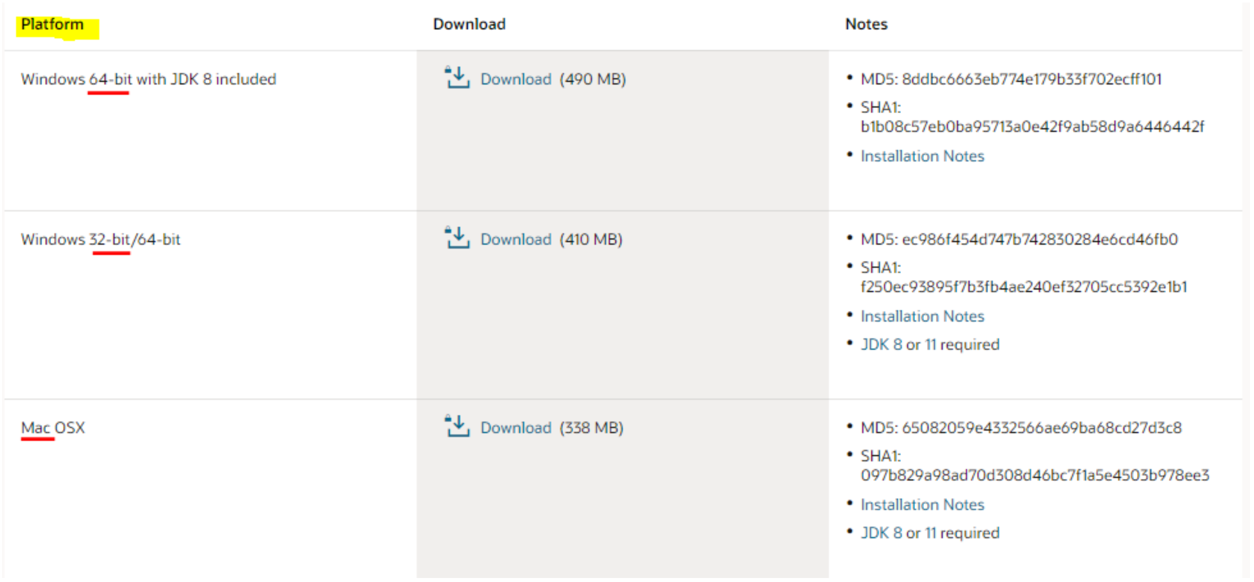
SQL Developer is a tool developed by oracle that uses a simplified coding language meant to help sort data. Known as SQL (Structured Query Language), it is way of communicating with the database we helped build for you. Though it sounds daunting, the language is relatively simple. To aid in the questions posed by your firm LCCB has developed a few queries that should answer a few of your immediate questions about CMS's data.

Step 1. Download SQL Developer

Though it is likely Oracle's SQL developer is already installed on a managing workstation, there is modality in using this program. In the sense that Oracle databases, like the one set up for the school, are remotely accessible with the SQL Developer tool.

Most up-to-date download: https://www.oracle.com/tools/downloads/sqldev-v192-downloads.html

Each download is the same software and has the same syntax and functionality across platforms.



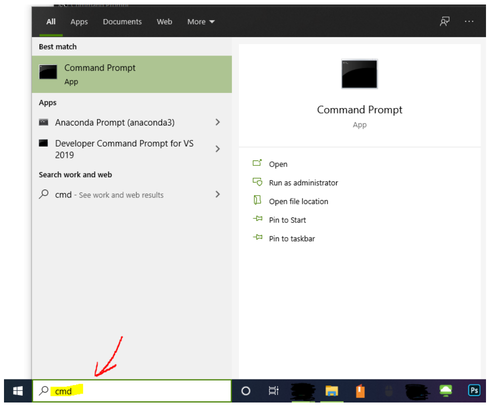
Platform	Download	Notes
Windows 64-bit with JDK 8 included	Download (490 MB)	• MD5: 8ddbc6663eb774e179b33f702ecff101 • SHA1: b1b08c57eb0ba95713a0e42f9ab58d9a6446442f • Installation Notes
Windows 32-bit/64-bit	Download (410 MB)	• MD5: ec986f454d747b742830284e6cd46fb0 • SHA1: f250ec93895f7b3fb4ae240ef32705cc5392e1b1 • Installation Notes • JDK 8 or 11 required
Mac OSX	Download (338 MB)	• MD5: 65082059e4332566ae69ba68cd27d3c8 • SHA1: 097b829a98ad70d308d46bc7f1a5e4503b978ee3 • Installation Notes • JDK 8 or 11 required

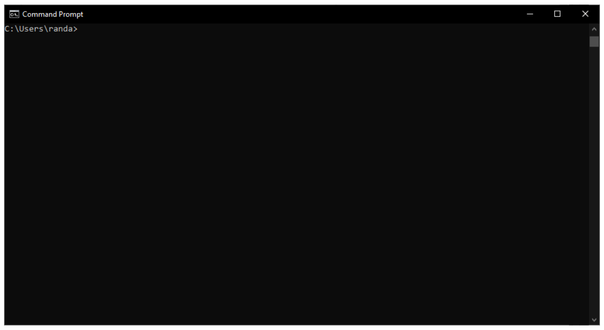
If you do not know your operating system you can find out what platform you are running by opening the command prompt and using the "systeminfo" command.

Please double-check it is okay to install SQL Developer for remote access before installing something on a work computer from management.

Step 1a. Determine the Operating System (OPTIONAL)

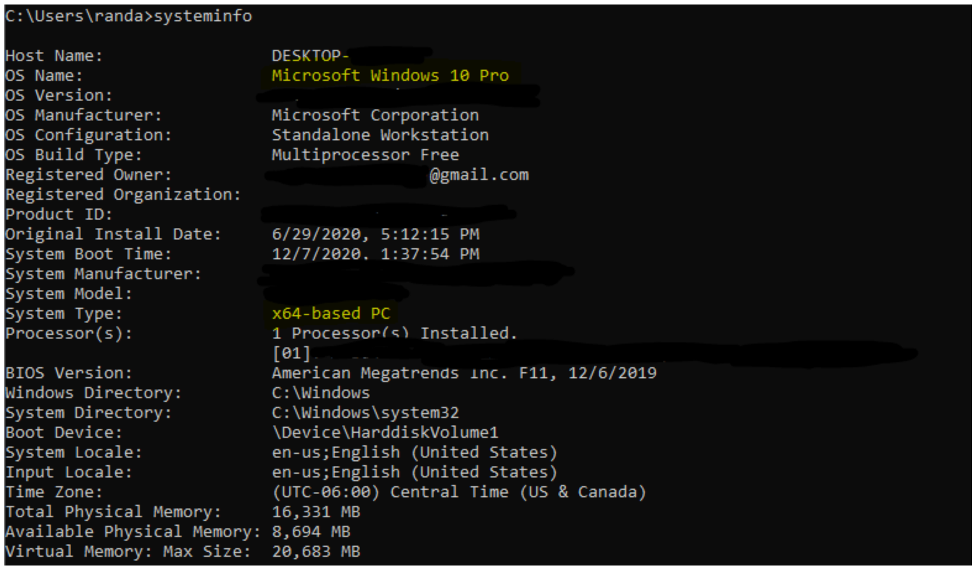
This set of instructions is for a Windows computer. Select the home search bar and type 'cmd' which will bring up command prompt. Open the app. It will look something like the image on the right.





```
C:\Users\randa>
```

Type "systeminfo" into the command prompt so it looks like this:
C:\Users\(user)>systeminfo
Press enter and wait a few seconds for the system specifications to appear.



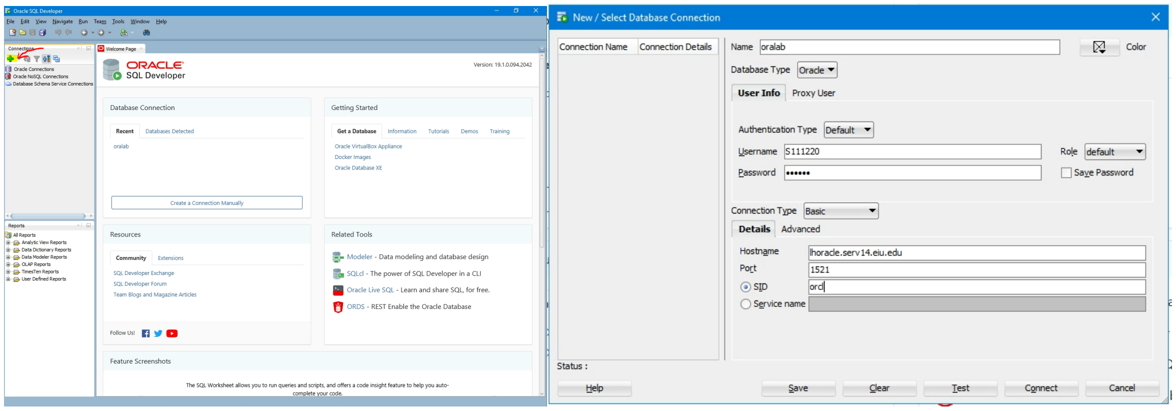
```
C:\Users\randa>systeminfo

Host Name:                 DESKTOP-[redacted]
OS Name:                   Microsoft Windows 10 Pro
OS Version:                [redacted]
OS Manufacturer:           Microsoft Corporation
OS Configuration:          Standalone Workstation
OS Build Type:             Multiprocessor Free
Registered Owner:          [redacted]@gmail.com
Registered Organization:
Product ID:                [redacted]
Original Install Date:     6/29/2020, 5:12:15 PM
System Boot Time:          12/7/2020, 1:37:54 PM
System Manufacturer:       [redacted]
System Model:              [redacted]
System Type:               x64-based PC
Processor(s):              1 Processor(s) Installed.
                           [01] [redacted]
BIOS Version:              American Megatrends Inc. F11, 12/6/2019
Windows Directory:         C:\Windows
System Directory:          C:\Windows\system32
Boot Device:               \Device\HarddiskVolume1
System Locale:             en-us;English (United States)
Input Locale:              en-us;English (United States)
Time Zone:                 (UTC-06:00) Central Time (US & Canada)
Total Physical Memory:     16,331 MB
Available Physical Memory: 8,694 MB
Virtual Memory: Max Size:  20,683 MB
```

Using the "OS Name:" and "System Type:" information you should be able to determine what platform installed.

Step 2. Connecting to the Database

Conveniently Oracle has developed its software to allow for easy communication between databases and their SQL software. First, open and run the SQL Developer application, you should see a window that looks like this. Select the "+" option and enter the credentials on the pop up.



Name	oralab
Database Type	Oracle
Authentication Type	Default
Username	S111220
Role	default
Password	••••••
Save Password	☐
Connection Type	Basic
Hostname	lhoracle.serv14.eiu.edu
Port	1521
SID ●	orcl
Service name ○	

- ❖ Name: oralab (this option can be named anything you like)
- ❖ Username: S111220
- ❖ Password: p11121
- ❖ Hostname: lhoracle.serv14.eiu.edu
- ❖ Port: 1521
- ❖ SID: orcl

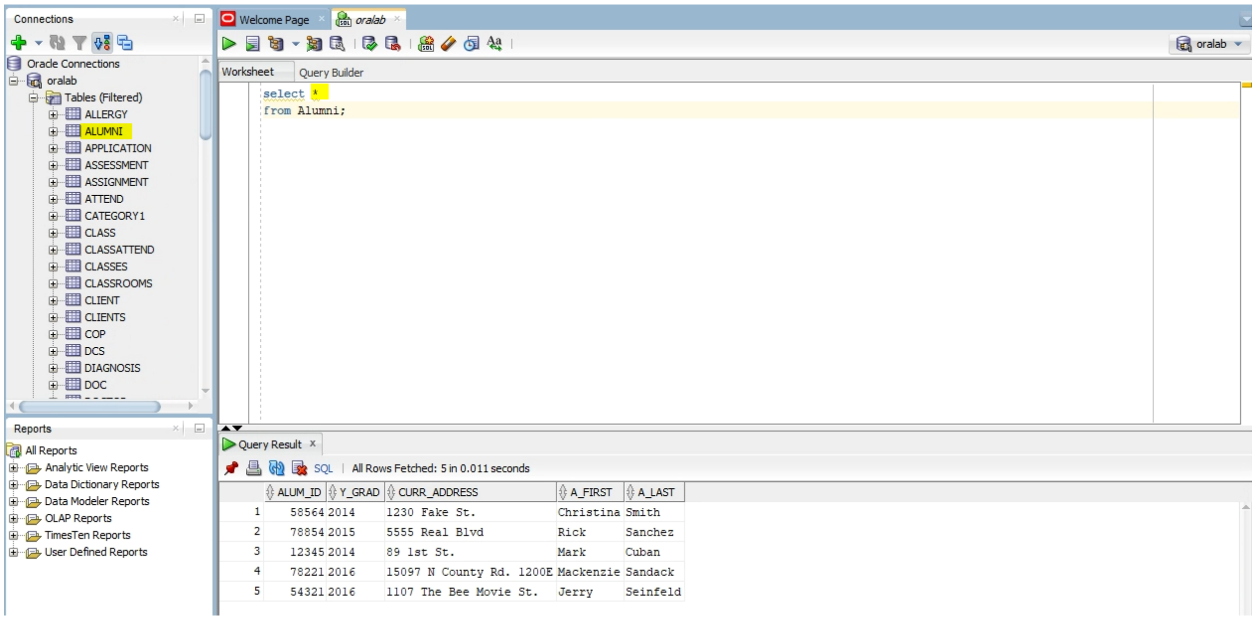
Press connect and you will be prompted to enter your password again. After this you'll be connected to the database and can begin exploring the environment.

Your password is case sensitive.

Step 3. Finding Basic Information

APPENDIX A — DATABASE USER MANUAL

Each table on the Logical model in this packet has its own table that stores information for reference. The information on the table will require you to enter some basic SQL code...



```
select *
from Alumni;
```

All Rows Fetched: 5 in 0.011 seconds

	ALUM_ID	Y_GRAD	CURR_ADDRESS	A_FIRST	A_LAST
1	58564	2014	1230 Fake St.	Christina	Smith
2	78854	2015	5555 Real Blvd	Rick	Sanchez
3	12345	2014	89 1st St.	Mark	Cuban
4	78221	2016	15097 N County Rd. 1200E	Mackenzie	Sandack
5	54321	2016	1107 The Bee Movie St.	Jerry	Seinfeld

```
SELECT *
FROM Alumni;
```

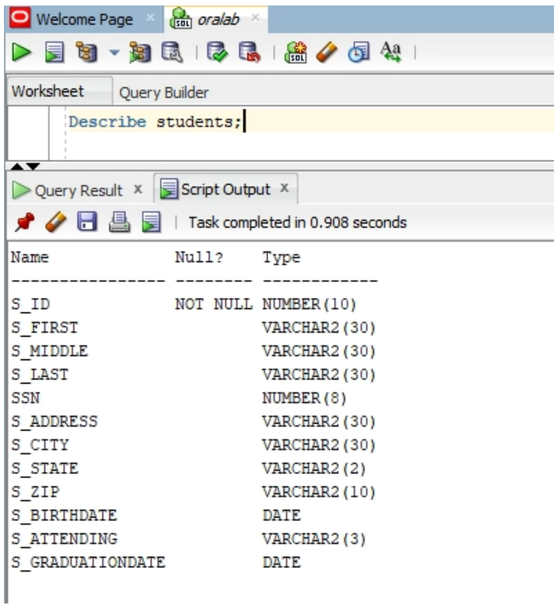
The code above will display your queried results at the bottom of the dialogue box based on the SQL you designed. There are a few rules associated with writing SQL that has to abide by some basic syntax. For example, each piece of SQL should end in a ";" that denotes the end of your query. Luckily the queries LCCB designed, found in the appendix, address the syntax necessary to retrieve the information requested, but some basic code that can help is the the above code.

These are the basic commands for a query, the order will not change much and depends on need.

```
Select * = Distinct Columns
From * = Tables
```

```
Where * = Conditional
Group By = Group by Column
Having * = Conditional (for group by)
Order By * = Order for Select
```

Using an asterisk represents a wildcard and will return all results matching the "From" and "Where" statement of the query. To find what kind of information a table has you can also use a suite of specialized information that is relevant to the table. Though the options are vast, a basic one to know is "describe" which will return the table's formatting from when it was created.



```
Describe students;
```

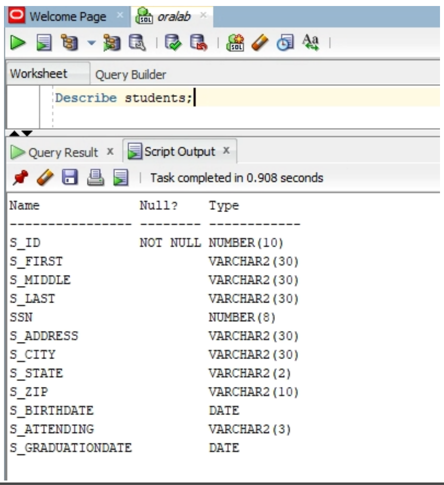
Task completed in 0.908 seconds

```
Name             Null?    Type
---------------- -------- ------------
S_ID             NOT NULL NUMBER(10)
S_FIRST                   VARCHAR2(30)
S_MIDDLE                  VARCHAR2(30)
S_LAST                    VARCHAR2(30)
SSN                       NUMBER(8)
S_ADDRESS                 VARCHAR2(30)
S_CITY                    VARCHAR2(30)
S_STATE                   VARCHAR2(2)
S_ZIP                     VARCHAR2(10)
S_BIRTHDATE               DATE
S_ATTENDING               VARCHAR2(3)
S_GRADUATIONDATE          DATE
```

Using this information, we can determine what can be done with the data to display our ideal query. Very rarely will you have to write your own query when dealing with the database, but it is helpful to have an idea of some of the basics of what is involved.

Step 4. How to Enter Data

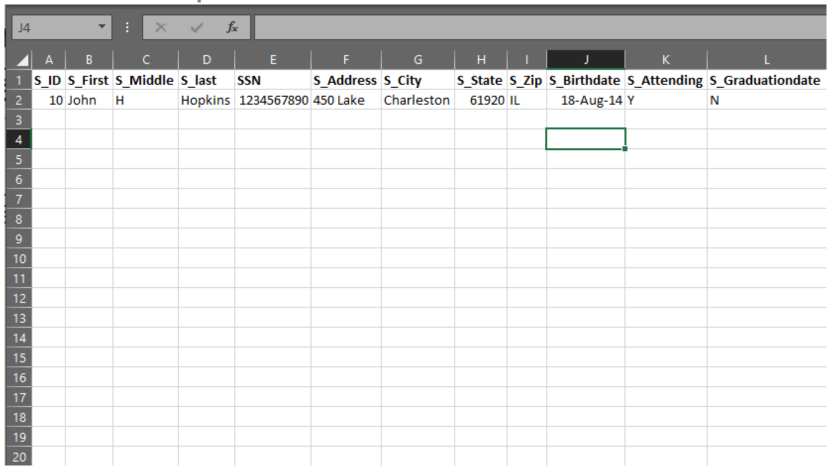
Data entry is a monotonous task for a database but can be simplified quite easily if you use the right tools. First open an Excel document, or your preferred spreadsheet application, and create a table based off the description of the table on Oracle.



```
Describe students;
```

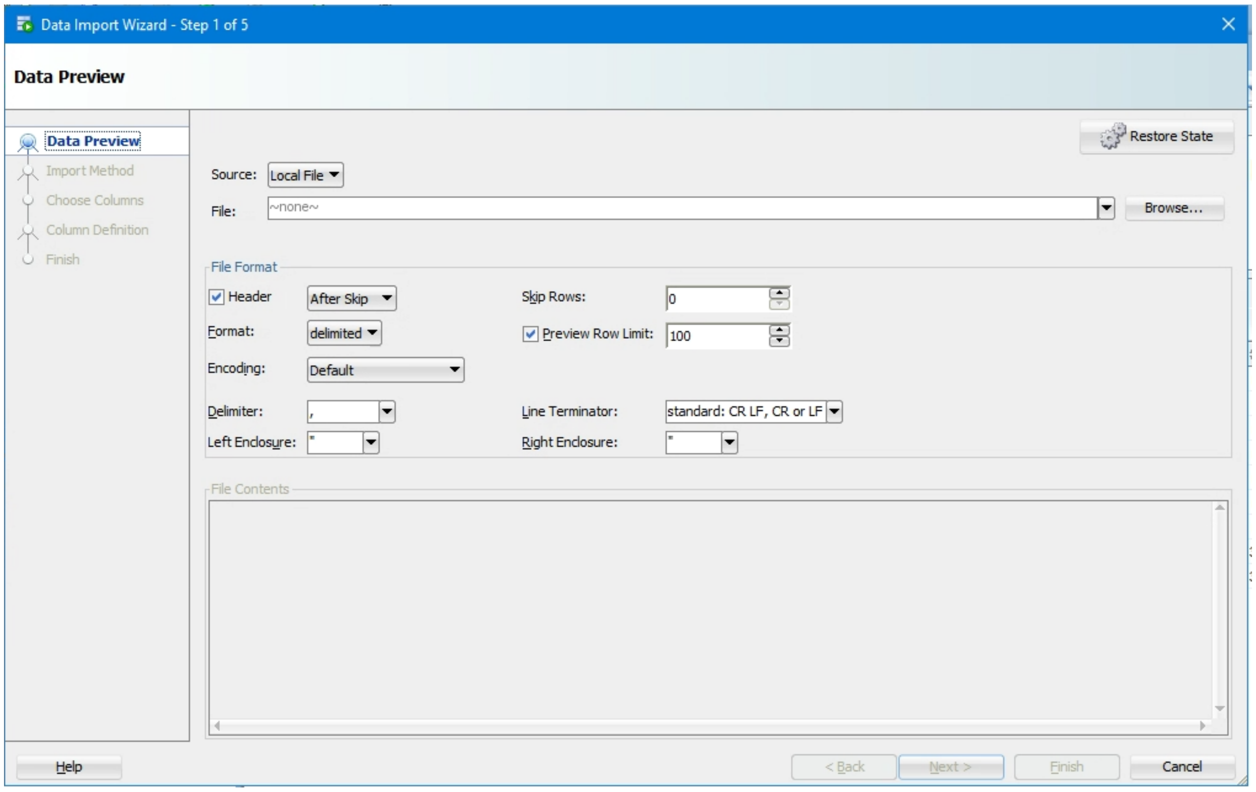
Task completed in 0.908 seconds

```
Name             Null?    Type
---------------- -------- ------------
S_ID             NOT NULL NUMBER(10)
S_FIRST                   VARCHAR2(30)
S_MIDDLE                  VARCHAR2(30)
S_LAST                    VARCHAR2(30)
SSN                       NUMBER(8)
S_ADDRESS                 VARCHAR2(30)
S_CITY                    VARCHAR2(30)
S_STATE                   VARCHAR2(2)
S_ZIP                     VARCHAR2(10)
S_BIRTHDATE               DATE
S_ATTENDING               VARCHAR2(3)
S_GRADUATIONDATE          DATE
```



S_ID	S_First	S_Middle	S_last	SSN	S_Address	S_City	S_State	S_Zip	S_Birthdate	S_Attending	S_Graduationdate
10	John	H	Hopkins	1234567890	450 Lake	Charleston	61920	IL	18-Aug-14	Y	N

Likely you will already have a sheet prepared for you depending on who handles data entry. But the idea is to match the number of columns and their headers (the headers do not need to match if the number of columns matches the oracle table). Right click on the table you want to update and select import data; the following window will pop up.


Data Import Wizard - Step 1 of 5

Data Preview

Source: Local File

File: ~none~

File Format

☑ Header After Skip

Skip Rows: 0

Format: delimited

☑ Preview Row Limit: 100

Encoding: Default

Delimiter: ,

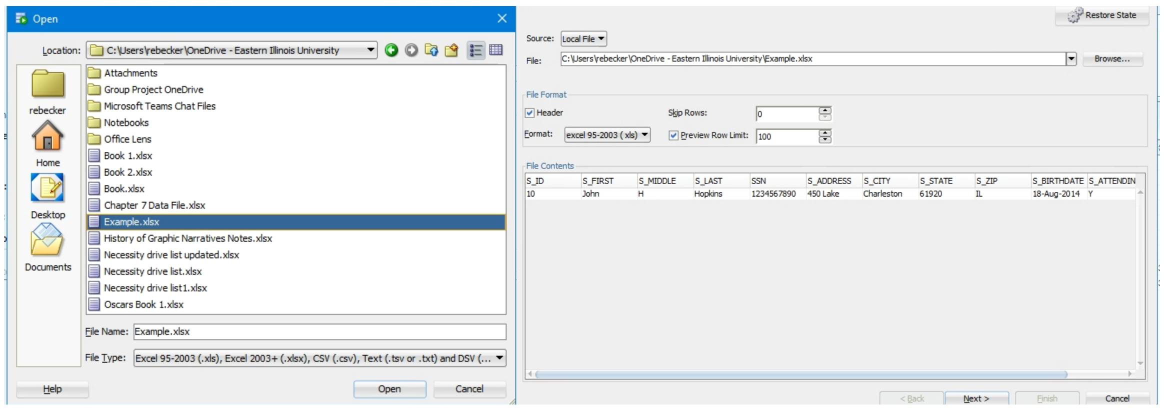
Line Terminator: standard: CR LF, CR or LF

Left Enclosure: "

Right Enclosure: "

File Contents

Select browse and navigate file explorer to where you saved the spreadsheet, highlight it and select “Open.”


Open

Location: C:\Users\rebecker\OneDrive - Eastern Illinois University

Attachments
Group Project OneDrive
Microsoft Teams Chat Files
Notebooks
Office Lens
Book 1.xlsx
Book 2.xlsx
Book.xlsx
Chapter 7 Data File.xlsx
Example.xlsx
History of Graphic Narratives Notes.xlsx
Necessity drive list updated.xlsx
Necessity drive list.xlsx
Necessity drive list1.xlsx
Oscars Book 1.xlsx

File Name: Example.xlsx

File Type: Excel 95-2003 (.xls), Excel 2003+ (.xlsx), CSV (.csv), Text (.tsv or .txt) and DSV (...

Source: Local File

File: C:\Users\rebecker\OneDrive - Eastern Illinois University\Example.xlsx

File Format

☑ Header

Skip Rows: 0

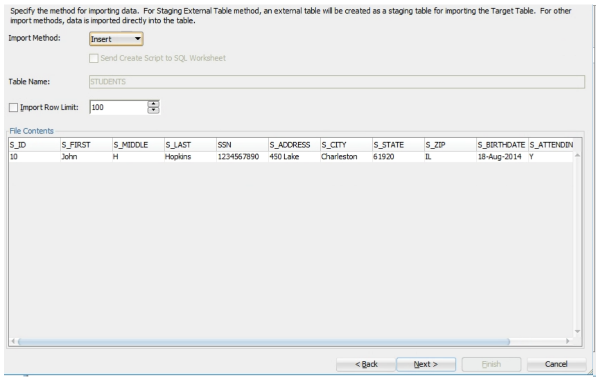
Format: excel 95-2003 (xls)

☑ Preview Row Limit: 100

File Contents

S_ID	S_FIRST	S_MIDDLE	S_LAST	SSN	S_ADDRESS	S_CITY	S_STATE	S_ZIP	S_BIRTHDATE	S_ATTENDIN
10	John	H	Hopkins	1234567890	450 Lake	Charleston	61920	IL	18-Aug-2014	Y

It will read the file and attempt to match the table you have selected to import the data into. Select “Next” and make sure the import method is set to insert. Also, make sure the “Selected Columns” all match the headers of the chart, otherwise it will create a new column on the table and will be more complex to fix later.


Specify the method for importing data. For Staging External Table method, an external table will be created as a staging table for importing the Target Table. For other import methods, data is imported directly into the table.

Import Method: Insert

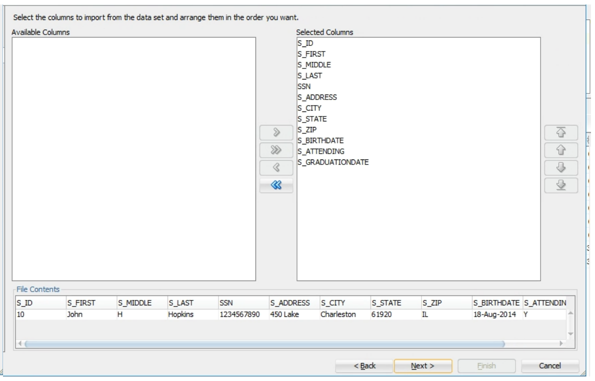
Send Create Script to SQL Worksheet

Table Name: STUDENTS

Import Row Limit: 100

File Contents

S_ID	S_FIRST	S_MIDDLE	S_LAST	SSN	S_ADDRESS	S_CITY	S_STATE	S_ZIP	S_BIRTHDATE	S_ATTENDIN
10	John	H	Hopkins	1234567890	450 Lake	Charleston	61920	IL	18-Aug-2014	Y


Select the columns to import from the data set and arrange them in the order you want.

Available Columns

Selected Columns

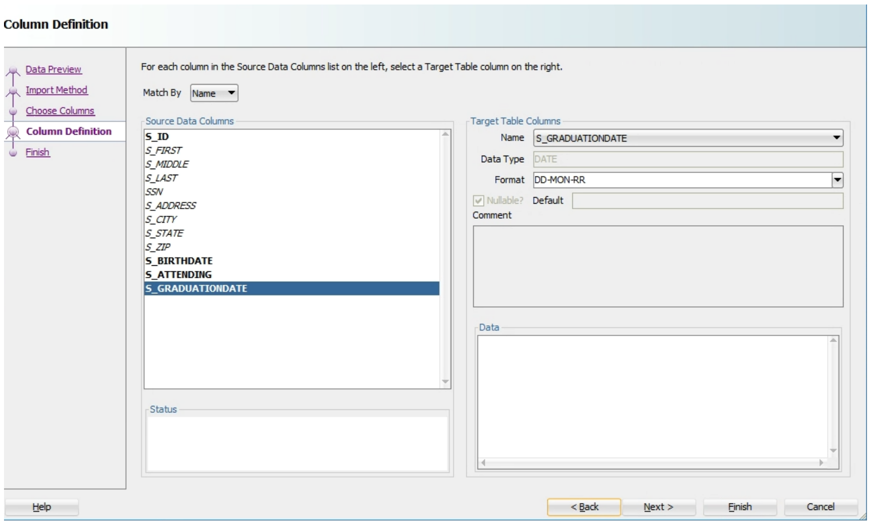
- S_ID
- S_FIRST
- S_MIDDLE
- S_LAST
- SSN
- S_ADDRESS
- S_CITY
- S_STATE
- S_ZIP
- S_BIRTHDATE
- S_ATTENDING
- S_GRADUATIONDATE

File Contents

S_ID	S_FIRST	S_MIDDLE	S_LAST	SSN	S_ADDRESS	S_CITY	S_STATE	S_ZIP	S_BIRTHDATE	S_ATTENDIN
10	John	H	Hopkins	1234567890	450 Lake	Charleston	61920	IL	18-Aug-2014	Y

After the information selected looks correct hit the “Next” button.

Ensure there are not errors on the next page, you can now select finish if all issues are addressed.


Column Definition

For each column in the Source Data Columns list on the left, select a Target Table column on the right.

Match By: Name

Source Data Columns

- S_ID
- S_FIRST
- S_MIDDLE
- S_LAST
- SSN
- S_ADDRESS
- S_CITY
- S_STATE
- S_ZIP
- S_BIRTHDATE
- S_ATTENDING
- S_GRADUATIONDATE

Status

Target Table Columns

Name: S_GRADUATIONDATE

Data Type: DATE

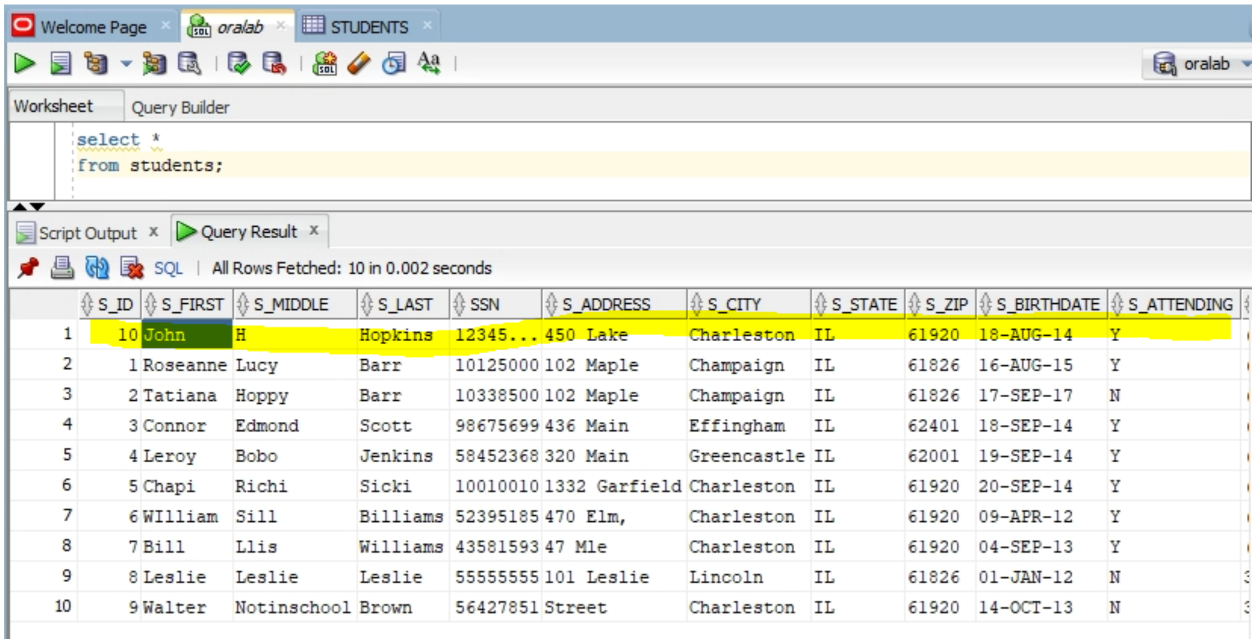
Format: DD-MON-RR

Nullable? ✓ Default

Comment

Data

Double check your information has coverted properly by checking the data of your tables, and it should now be ready for SQL querying.



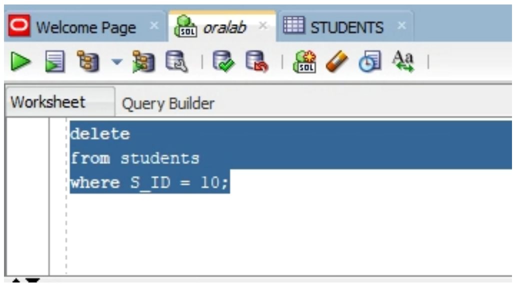
```
select *
from students;
```

All Rows Fetched: 10 in 0.002 seconds

	S_ID	S_FIRST	S_MIDDLE	S_LAST	SSN	S_ADDRESS	S_CITY	S_STATE	S_ZIP	S_BIRTHDATE	S_ATTENDING
1	10	John	H	Hopkins	12345...	450 Lake	Charleston	IL	61920	18-AUG-14	Y
2	1	Roseanne	Lucy	Barr	10125000	102 Maple	Champaign	IL	61826	16-AUG-15	Y
3	2	Tatiana	Hoppy	Barr	10338500	102 Maple	Champaign	IL	61826	17-SEP-17	N
4	3	Connor	Edmond	Scott	98675699	436 Main	Effingham	IL	62401	18-SEP-14	Y
5	4	Leroy	Bobo	Jenkins	58452368	320 Main	Greencastle	IL	62001	19-SEP-14	Y
6	5	Chapi	Richi	Sicki	10010010	1332 Garfield	Charleston	IL	61920	20-SEP-14	Y
7	6	WIlliam	Sill	Billiams	52395185	470 Elm,	Charleston	IL	61920	09-APR-12	Y
8	7	Bill	Llis	Williams	43581593	47 Mle	Charleston	IL	61920	04-SEP-13	Y
9	8	Leslie	Leslie	Leslie	55555555	101 Leslie	Lincoln	IL	61826	01-JAN-12	N
10	9	Walter	Notinschool	Brown	56427851	Street	Charleston	IL	61920	14-OCT-13	N

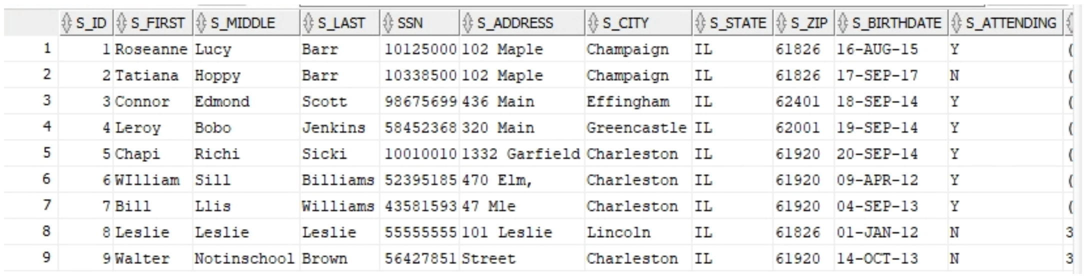
Step 4a. How to Remove Data

Though you will not often want to delete the data from a table there me an exception where an entry mistake has been made. To achieve this, you will simply have to use the “delete” argument to clear your rows, using the primary ID of the entry to delete.



```
delete
from students
where S_ID = 10;
```

Using this syntax, we will ask oracle to delete out our entry from the specified tables where the condition is true. This is just one of the methods we can use. The safest way to avoid deleting data you wish to maintain is to only create conditionals relevant to the primary keys or unique values. If we go to our table now, John Hopkins has been deleted.

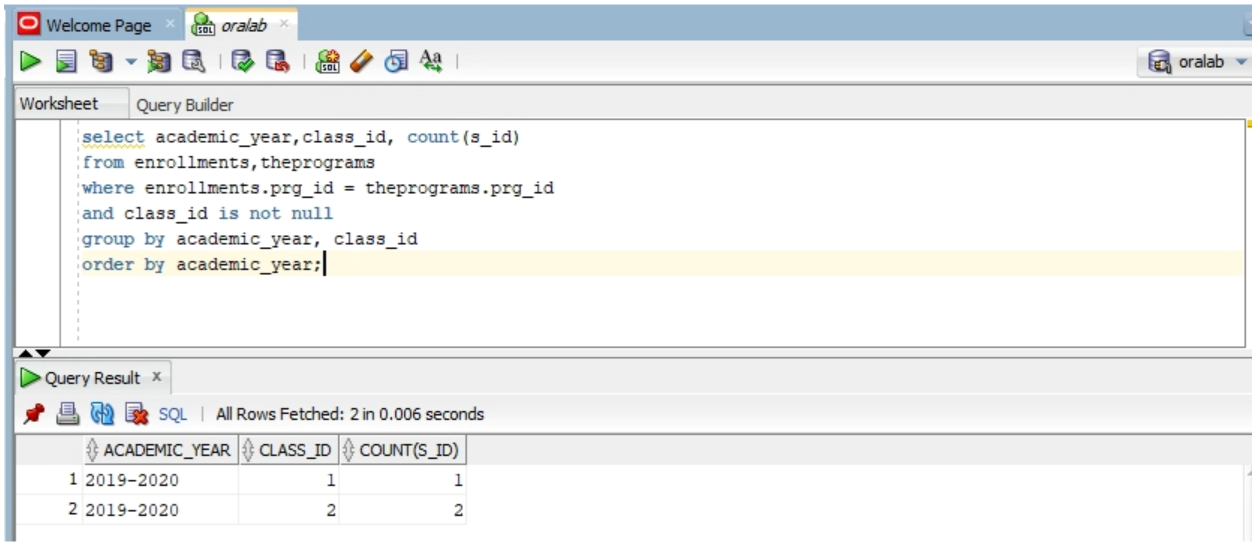


	S_ID	S_FIRST	S_MIDDLE	S_LAST	SSN	S_ADDRESS	S_CITY	S_STATE	S_ZIP	S_BIRTHDATE	S_ATTENDING
1	1	Roseanne	Lucy	Barr	10125000	102 Maple	Champaign	IL	61826	16-AUG-15	Y
2	2	Tatiana	Hoppy	Barr	10338500	102 Maple	Champaign	IL	61826	17-SEP-17	N
3	3	Connor	Edmond	Scott	98675699	436 Main	Effingham	IL	62401	18-SEP-14	Y
4	4	Leroy	Bobo	Jenkins	58452368	320 Main	Greencastle	IL	62001	19-SEP-14	Y
5	5	Chapi	Richi	Sicki	10010010	1332 Garfield	Charleston	IL	61920	20-SEP-14	Y
6	6	WIlliam	Sill	Billiams	52395185	470 Elm,	Charleston	IL	61920	09-APR-12	Y
7	7	Bill	Llis	Williams	43581593	47 Mle	Charleston	IL	61920	04-SEP-13	Y
8	8	Leslie	Leslie	Leslie	55555555	101 Leslie	Lincoln	IL	61826	01-JAN-12	N
9	9	Walter	Notinschool	Brown	56427851	Street	Charleston	IL	61920	14-OCT-13	N

Step 5. How to Return a Query

Open your oracle database query developer. Copy and paste the query you would like to use (refer to the queries in section 8).

Run the query and enjoy.



```
select academic_year,class_id, count(s_id)
from enrollments,theprograms
where enrollments.prg_id = theprograms.prg_id
and class_id is not null
group by academic_year, class_id
order by academic_year;
```

All Rows Fetched: 2 in 0.006 seconds

```
  ACADEMIC_YEAR  CLASS_ID  COUNT(S_ID)
1 2019-2020             1            1
2 2019-2020             2            2
```

APPENDIX B — FRESH-INSTALL DATABASE CREATION SYNTAX

```
CREATE TABLE students
(s_id NUMBER(10),
s_first VARCHAR2(30),
s_middle VARCHAR2(30),
s_last VARCHAR2(30),
ssn NUMERIC(8),
s_address VARCHAR2(30),
s_city VARCHAR2(30),
s_state VARCHAR2(2),
s_zip VARCHAR2(10),
s_birthdate DATE,
s_attending VARCHAR2(3) NULL,
s_graduationDate DATE NULL,
CONSTRAINT students_s_id_pk PRIMARY KEY(s_id));

CREATE TABLE sibling_relation
(sr_id NUMBER(10),
st1_id NUMBER(10),
st2_id NUMBER(10),
CONSTRAINT sibling_relation_sr_id_pk PRIMARY KEY (sr_id),
CONSTRAINT sibling_relation_c_id_fk FOREIGN KEY (st1_id) REFERENCES students(s_id),
CONSTRAINT sibling_relation_os_id_fk FOREIGN KEY (st2_id) REFERENCES
students(s_id));

CREATE TABLE allergy
(a_id NUMBER(10),
a_name VARCHAR2(30),
CONSTRAINT allergy_a_id_pk PRIMARY KEY (a_id));

CREATE TABLE student_allergy
(sa_id NUMBER(10),
a_id NUMBER(10),
s_id NUMBER(10),
CONSTRAINT student_allergy_sa_id_pk PRIMARY KEY (sa_id),
CONSTRAINT student_allergy_a_id_fk FOREIGN KEY (a_id) REFERENCES allergy(a_id),
CONSTRAINT student_allergy_s_id_fk FOREIGN KEY (s_id) REFERENCES students(s_id));

CREATE TABLE doctors
(d_id NUMBER(10),
d_name VARCHAR2(30),
d_phone VARCHAR2(14),
CONSTRAINT doctors_d_id_pk PRIMARY KEY (d_id));

CREATE TABLE student_doctor
(sd_id NUMBER(10),
d_id NUMBER(10),
s_id NUMBER(10),
CONSTRAINT student_doctor_sd_id_pk PRIMARY KEY (sd_id),
CONSTRAINT student_doctor_a_id_fk FOREIGN KEY (d_id) REFERENCES doctors(d_id),
CONSTRAINT student_doctor_s_id_fk FOREIGN KEY (s_id) REFERENCES students(s_id));

CREATE TABLE theparents
(parent_id NUMBER(10),
p_first VARCHAR2(30),
p_middle VARCHAR2(30),
p_last VARCHAR2(30),
p_birthdate DATE,
p_address VARCHAR2(30),
p_city VARCHAR2(30),
p_state VARCHAR2(2),
p_zip VARCHAR2(10),
CONSTRAINT theparents_parent_id_pk PRIMARY KEY (parent_id));

CREATE TABLE parents_phone
(phone_id NUMBER(14),
parent_id NUMBER(14),
phone VARCHAR2(14),
CONSTRAINT parents_phone_phone_id_pk PRIMARY KEY (phone_id),
CONSTRAINT parents_phone_parent_id_fk FOREIGN KEY (parent_id) REFERENCES
theparents(parent_id));

CREATE TABLE student_parents
(sp_id NUMBER(10),
s_id NUMBER(10),
parent_id NUMBER(10),
can_pick_up VARCHAR2(3),
gets_bill VARCHAR2(3),
contact_type VARCHAR2(30),
CONSTRAINT student_parents_sp_id_pk PRIMARY KEY (sp_id),
CONSTRAINT student_parents_s_id_fk FOREIGN KEY (s_id) REFERENCES students(s_id),
CONSTRAINT student_parents_a_id_fk FOREIGN KEY (parent_id) REFERENCES
theparents(parent_id));

CREATE TABLE theprograms
(prg_id NUMBER(10),
academic_year VARCHAR2(20),
prg_name VARCHAR(30),
prg_capacity NUMBER(3),
prg_type VARCHAR(30),
reg_fee NUMBER(10,2),
deposit NUMBER(10,2),
pay_amt_10 NUMBER(10,2),
pay_amt_2 NUMBER(10,2),
pay_amt_1 NUMBER(10,2),
CONSTRAINT theprograms_prg_id_pk PRIMARY KEY (prg_id));

CREATE TABLE theclassrooms
(class_id NUMBER(10),
location VARCHAR2(30),
CONSTRAINT theclassrooms_class_id_pk PRIMARY KEY (class_id));

CREATE TABLE enrollments
(enroll_id NUMBER(10),
s_id NUMBER(10),
prg_id NUMBER(10),
class_id NUMBER(10),
dateOfEntry DATE,
pay_frequency_choice NUMBER(2),
eval1_pls VARCHAR(30) NULL,
eval1_sa VARCHAR(30) NULL,
eval1_ld VARCHAR(30) NULL,
eval1_math VARCHAR(30) NULL,
eval1_culture VARCHAR(30) NULL,
eval2_pls VARCHAR(30) NULL,
eval2_sa VARCHAR(30) NULL,
eval2_ld VARCHAR(30) NULL,
eval2_math VARCHAR(30) NULL,
eval2_culture VARCHAR(30) NULL,
status VARCHAR2(10),
CONSTRAINT enrollments_enroll_id_pk PRIMARY KEY (enroll_id),
CONSTRAINT enrollments_s_id_fk FOREIGN KEY (s_id) REFERENCES students(s_id),
CONSTRAINT enrollments_prg_id_fk FOREIGN KEY (prg_id) REFERENCES
theprograms(prg_id),
CONSTRAINT enrollments_class_id_fk FOREIGN KEY (class_id) REFERENCES
theclassrooms(class_id));

CREATE TABLE tuition_pmts
(pay_id NUMBER(10),
sp_id NUMBER(10),
enroll_id NUMBER(10),
day_of_payment DATE,
tp_methpmt VARCHAR2(10),
pay_amt NUMBER(10,2),
completed VARCHAR2(3),
CONSTRAINT tuition_pmts_pay_id_pk PRIMARY KEY (pay_id),
CONSTRAINT tuition_pmts_sp_id_fk FOREIGN KEY (sp_id) REFERENCES
student_parents(sp_id),
CONSTRAINT tuition_pmts_enroll_id_fk FOREIGN KEY (enroll_id) REFERENCES
```

```
enrollments(enroll_id));

CREATE TABLE theteachers
(teacher_id NUMBER(10),
t_first VARCHAR2(30),
t_middle VARCHAR2(30),
t_last VARCHAR2(30),
cert_status VARCHAR(30),
CONSTRAINT theteachers_teacher_id_pk PRIMARY KEY (teacher_id));

CREATE TABLE teach_class
(tc_id NUMBER(10),
teacher_id NUMBER(10),
class_id NUMBER(10),
position VARCHAR2(30),
CONSTRAINT teach_class_tc_id_pk PRIMARY KEY (tc_id),
CONSTRAINT teach_class_teacher_id_fk FOREIGN KEY (teacher_id) REFERENCES
theteachers(teacher_id),
CONSTRAINT teach_class_class_id_fk FOREIGN KEY (class_id) REFERENCES
theclassrooms(class_id));

CREATE TABLE classattend
(attend_id NUMBER(10),
enroll_id NUMBER(10),
teacher_id NUMBER(10),
attend_date DATE,
present VARCHAR(3),
CONSTRAINT classattend_attend_id_pk PRIMARY KEY (attend_id),
CONSTRAINT classattend_enroll_id_fk FOREIGN KEY (enroll_id) REFERENCES
enrollments(enroll_id),
CONSTRAINT classattend_teacher_id_fk FOREIGN KEY (teacher_id) REFERENCES
theteachers(teacher_id));

CREATE TABLE teach_eval
(te_id NUMBER(10),
teacher_id NUMBER(10),
question VARCHAR2(50),
CONSTRAINT teach_eval_te_id_pk PRIMARY KEY (te_id),
CONSTRAINT teach_eval_teacher_id_fk FOREIGN KEY (teacher_id) REFERENCES
theteachers(teacher_id));

COMMIT;
```

APPENDIX C — SYNTAX TO POPULATE DATABASE WITH TEST DATA

```
--- inserting records into students
INSERT INTO students VALUES
(1, 'Roseanne', 'Lucy', 'Barr', 10125000, '102
Maple','Champaign','IL',61826,to_date('08/16/2015', 'mm/dd/yyyy'),'Y',NULL);

INSERT INTO students VALUES
(2, 'Tatiana', 'Hoppy', 'Barr',10338500,'102
Maple','Champaign','IL',61826,to_date('09/17/2017', 'mm/dd/yyyy'),'N',NULL);

INSERT INTO students VALUES
(3, 'Connor', 'Edmond', 'Scott',98675699,'436
Main','Effingham','IL',62401,to_date('09/18/2014', 'mm/dd/yyyy'),'Y',NULL);

INSERT INTO students VALUES
(4, 'Leroy', 'Bobo', 'Jenkins',58452368,'320
Main','Greencastle','IL',62001,to_date('09/19/2014', 'mm/dd/yyyy'),'Y',NULL);

INSERT INTO students VALUES
(5, 'Chapi', 'Richi', 'Sicki',10010010,'1332
Garfield','Charleston','IL',61920,to_date('09/20/2014', 'mm/dd/yyyy'),'Y',NULL);

INSERT INTO students VALUES
(6, 'WIlliam', 'Sill', 'Billiams',52395185,'470
Elm,','Charleston','IL',61920,to_date('04/09/2012', 'mm/dd/yyyy'),'Y',NULL);

INSERT INTO students VALUES
(7, 'Bill', 'Llis', 'Williams',43581593,'47
Mle','Charleston','IL',61920,to_date('09/04/2013', 'mm/dd/yyyy'),'Y',NULL);

INSERT INTO students VALUES
(8, 'Leslie', 'Leslie', 'Leslie',55555555,'101
Leslie','Lincoln','IL',61826,to_date('01/01/2012',
'mm/dd/yyyy'),'N',to_date('05/30/2020', 'mm/dd/yyyy'));

INSERT INTO students VALUES
(9, 'Walter', 'Notinschool',
'Brown',56427851,'Street','Charleston','IL',61920,to_date('10/14/2013',
'mm/dd/yyyy'),'N','30-MAY-88');

INSERT INTO students VALUES
(10,'John','H','Hopkins',1234567890,'450
Lake','Charleston','IL',61920,to_date('12/18/2014', 'mm/dd/yyyy'),'Y',null);

--- inserting values into sibling_relation
INSERT INTO sibling_relation VALUES (1,1,2);
INSERT INTO sibling_relation VALUES (2,2,1);
```

APPENDIX C — SYNTAX TO POPULATE DATABASE WITH TEST DATA

```
--- inserting values into doctor
INSERT INTO doctors VALUES
(1, 'Dr. Leslie', '(217)-304-9875');
INSERT INTO doctors VALUES
(2, 'Dr. Oz', '(217)-509-5555');
INSERT INTO doctors VALUES
(3, 'Dr. Jekyll', '(217)-555-5555');
INSERT INTO doctors VALUES
(4, 'Dr. Stewart', '(217)-555-5555');
INSERT INTO doctors VALUES
(5, 'Dr. Waldo', '(217)-555-5555');

--- inserting values into student_doctor
INSERT INTO student_doctor VALUES (1,1,1);
INSERT INTO student_doctor VALUES (2,1,2);
INSERT INTO student_doctor VALUES (3,2,3);
INSERT INTO student_doctor VALUES (4,3,4);
INSERT INTO student_doctor VALUES (5,4,5);
INSERT INTO student_doctor VALUES (6,5,6);
INSERT INTO student_doctor VALUES (7,2,7);
INSERT INTO student_doctor VALUES (8,4,8);
INSERT INTO student_doctor VALUES (9,3,9);

-- inserting values into allergy
INSERT INTO allergy VALUES (1, 'Peanuts');
INSERT INTO allergy VALUES (2, 'Tylenol');
INSERT INTO allergy VALUES (3, 'Treenuts');
INSERT INTO allergy VALUES (4, 'Latex');
INSERT INTO allergy VALUES (5, 'Cats');

-- inserting values into student_allergy
INSERT INTO student_allergy VALUES (1,1,1);
INSERT INTO student_allergy VALUES (2,2,1);
INSERT INTO student_allergy VALUES (3,1,2);
INSERT INTO student_allergy VALUES (4,3,3);
INSERT INTO student_allergy VALUES (5,5,4);
INSERT INTO student_allergy VALUES (6,4,6);

-- inserting values into theparents
INSERT INTO theparents VALUES
(1,'Tina','M','Barr','15-JAN-80','102 Maple','Champaign','IL',61826);
INSERT INTO theparents VALUES
(2,'Tim','N','Barr','14-OCT-80','102 Maple','Champaign','IL',61826);
INSERT INTO theparents VALUES
(3,'Linda','L','Peterson','19-MAY-59','900 Street','Charleston','IL',61920);
```

```
INSERT INTO theparents VALUES
(4,'John','Q','Scott','15-AUG-77','436 Main','Effingham','IL',62401);
INSERT INTO theparents VALUES
(5,'Levon','S','Jenkins','15-OCT-82','320 Main','Greencastle','IL',62001);
INSERT INTO theparents VALUES
(6,'Vicki','M','Sicki','13-FEB-79','1332 Garfield','Charleston','IL',61920);
INSERT INTO theparents VALUES
(7,'Bill','B','Billiams','26-DEC-67','470 Elm','Charleston','IL',61920);

-- inserting values into parents_phone
INSERT INTO parents_phone VALUES (1,1,'(217)-555-5555');
INSERT INTO parents_phone VALUES (2,1,'(217)-999-9999');
INSERT INTO parents_phone VALUES (3,2,'(217)-555-5555');
INSERT INTO parents_phone VALUES (4,3,'(217)-555-5555');
INSERT INTO parents_phone VALUES (5,3,'(217)-999-5555');
INSERT INTO parents_phone VALUES (6,4,'(217)-555-5555');
INSERT INTO parents_phone VALUES (7,5,'(217)-555-5555');
INSERT INTO parents_phone VALUES (8,6,'(217)-555-5555');
INSERT INTO parents_phone VALUES (9,7,'(217)-555-5555');

-- inserting values into student_parents
INSERT INTO student_parents VALUES
(1,1,1,1,1,'parent');
INSERT INTO student_parents VALUES
(2,1,2,1,0,'parent');
INSERT INTO student_parents VALUES
(3,1,3,1,0,'emergency');
INSERT INTO student_parents VALUES
(4,2,1,1,1,'parent');
INSERT INTO student_parents VALUES
(5,2,2,1,0,'parent');
INSERT INTO student_parents VALUES
(6,2,3,1,0,'emergency');
INSERT INTO student_parents VALUES
(7,3,4,1,1,'parent');
INSERT INTO student_parents VALUES
(8,4,5,1,1,'parent');
INSERT INTO student_parents VALUES
(9,5,6,1,1,'parent');
INSERT INTO student_parents VALUES
(10,6,7,1,1,'parent');

-- inserting values into theprograms
INSERT INTO theprograms VALUES
(1,'2019-2020','Primary Morning Only',50,'regular',100,475,440,2200,4275);
INSERT INTO theprograms VALUES
```

```
(2,'2019-2020','Primary All-Day',50,'regular',100,525,485,2425,4725);
INSERT INTO theprograms VALUES
(3,'2019-2020','Elementary I',50,'regular',100,575,530,2600,5175);
INSERT INTO theprograms VALUES
(4,'2019-2020','Summer Session (One)',50,'summer',100,0,0,0,475);
INSERT INTO theprograms VALUES
(5,'2019-2020','Summer Session (Two)',50,'summer',100,0,0,525,1050);
INSERT INTO theprograms VALUES
(6,'2019-2020','Afterschool (5 days)',50,'afterschool',0,0,225,1125,2250);
INSERT INTO theprograms VALUES
(7,'2019-2020','Afterschool (4 days)',50,'afterschool',0,0,200,990,1980);
INSERT INTO theprograms VALUES
(8,'2019-2020','Afterschool (3 days)',50,'afterschool',0,0,150,743,1485);
INSERT INTO theprograms VALUES
(9,'2019-2020','Afterschool (2 days)',50,'afterschool',0,0,100,495,990);
INSERT INTO theprograms VALUES
(10,'2019-2020','Afterschool (1 day)',50,'afterschool',0,0,50,248,495);

-- inserting values into theclassrooms
INSERT INTO theclassrooms VALUES (1,'Room One');
INSERT INTO theclassrooms VALUES (2,'Room Two');
INSERT INTO theclassrooms VALUES (3,'Room Three');

-- inserting values into enrollments
INSERT INTO enrollments VALUES
(1,1,2,2,'12-AUG-19',10,'A+','A+','B','A','C',null,null,null,null,null,'enrolled');
INSERT INTO enrollments VALUES
(2,1,8,null,'13-AUG-
19',2,null,null,null,null,null,null,null,null,null,null,'waitlist');
INSERT INTO enrollments VALUES
(3,2,3,null,'13-AUG-
19',10,null,null,null,null,null,null,null,null,null,null,'waitlist');
INSERT INTO enrollments VALUES
(4,2,8,null,'13-AUG-
19',2,null,null,null,null,null,null,null,null,null,null,'waitlist');
INSERT INTO enrollments VALUES
(5,3,1,1,'08-AUG-19',2,'A','A','A','A','B',null,null,null,null,null,'enrolled');
INSERT INTO enrollments VALUES
(6,4,3,null,'07-AUG-
19',10,null,null,null,null,null,null,null,null,null,null,'waitlist');
INSERT INTO enrollments VALUES
(7,5,2,2,'14-AUG-19',1,'A','A','A','A','A',null,null,null,null,null,'enrolled');
INSERT INTO enrollments VALUES
(8,6,3,null,'06-AUG-
19',10,null,null,null,null,null,null,null,null,null,null,'waitlist');
INSERT INTO enrollments VALUES
```

```
(9,7,3,null,'13-AUG-
19',2,null,null,null,null,null,null,null,null,null,null,'waitlist');

-- inserting values into tuition_pmts
INSERT INTO tuition_pmts VALUES
(1,1,1,'12-AUG-19','check',625,'Y');
INSERT INTO tuition_pmts VALUES
(2,1,1,'01-SEP-19','autopay',485,'Y');
INSERT INTO tuition_pmts VALUES
(3,1,1,'01-OCT-19','autopay',485,'Y');
INSERT INTO tuition_pmts VALUES
(4,1,1,'01-NOV-19','autopay',485,'Y');
INSERT INTO tuition_pmts VALUES
(5,1,1,'01-DEC-19','autopay',485,'Y');
INSERT INTO tuition_pmts VALUES
(6,7,5,'08-AUG-19','check',575,'Y');
INSERT INTO tuition_pmts VALUES
(7,7,5,'01-OCT-19','check',2200,'Y');
INSERT INTO tuition_pmts VALUES
(8,9,7,'14-AUG-19','cash',625,'Y');

-- inserting values into theteachers
INSERT INTO theteachers VALUES
(1,'Karen','D','Moore','certified');
INSERT INTO theteachers VALUES
(2,'Margaret','M','Avelar','needs');
INSERT INTO theteachers VALUES
(3,'Deborah','K','Bannister','pending');
INSERT INTO theteachers VALUES
(4,'Leo','W','Crum','certified');
INSERT INTO theteachers VALUES
(5,'Bradley','L','McKean','certified');
INSERT INTO theteachers VALUES
(6,'Patricia','J','Cohen','certified');

-- inserting values into teach_class
INSERT INTO teach_class VALUES
(1,1,1,'lead');
INSERT INTO teach_class VALUES
(2,2,1,'colead');
INSERT INTO teach_class VALUES
(3,3,2,'colead');
INSERT INTO teach_class VALUES
(4,5,2,'lead');
INSERT INTO teach_class VALUES
(5,4,3,'lead');
```

APPENDIX C — SYNTAX TO POPULATE DATABASE WITH TEST DATA

```
INSERT INTO teach_class VALUES
(6,6,3,'colead');

-- inserting values into classattend
INSERT INTO classattend VALUES
(1,1,3,'03-SEP-19','Y');
INSERT INTO classattend VALUES
(2,1,3,'04-SEP-19','N');
INSERT INTO classattend VALUES
(3,1,3,'05-SEP-19','Y');
INSERT INTO classattend VALUES
(4,1,3,'06-SEP-19','Y');
INSERT INTO classattend VALUES
(5,1,3,'07-SEP-19','Y');

-- inserting values into teach_eval
INSERT INTO teach_eval VALUES (1, 1, 'Perfect teacher');
INSERT INTO teach_eval VALUES (2, 2, 'Great teacher');
INSERT INTO teach_eval VALUES (3, 3, 'Room for improvement');
INSERT INTO teach_eval VALUES (4, 4, 'Good');
INSERT INTO teach_eval VALUES (5, 5, 'Yay');

COMMIT;
```

APPENDIX D — TEAM LCCB CONTRACT

Our team is **Team "LCCB" (Lower-Carr-Chen-Becker),** and the motto we plan to follow is "Let's get it done." It is quite simple, yes, but it is this simplicity that makes for a great guiding principle. We aim to complete our assignments/projects in a timely manner, and to perform to the best of our abilities while doing so.

Team LCC is composed of:

Lucas Lower -- lmlower@eiu.edu
Jennifer Carr -- jjcarr3@eiu.edu
ZhiYong Chen -- zchen@eiu.edu
Randall Becker -- rebcker@eiu.edu

Our team has some unique capabilities: Jennifer and Lucas have real-world database experience (with Oracle and MySQL), and Chen is skilled in organized problem solving for programming team competitions (picoCTF). Randall brings business experience per his studies in accounting, which is quite the data-focused field. Jennifer, Lucas, and Chen have shared quite a few courses over the past few years, and have proven able to work with each other multiple times. Randall, while a new addition, brings a fresh set of eyes to problems, and we anticipate working with him to be quite easy and enjoyable. With our combination of existing database and business knowledge, quick learning ability, and problem solving perseverance, we hope to meet or exceed assignment goals in a quick and efficient way.

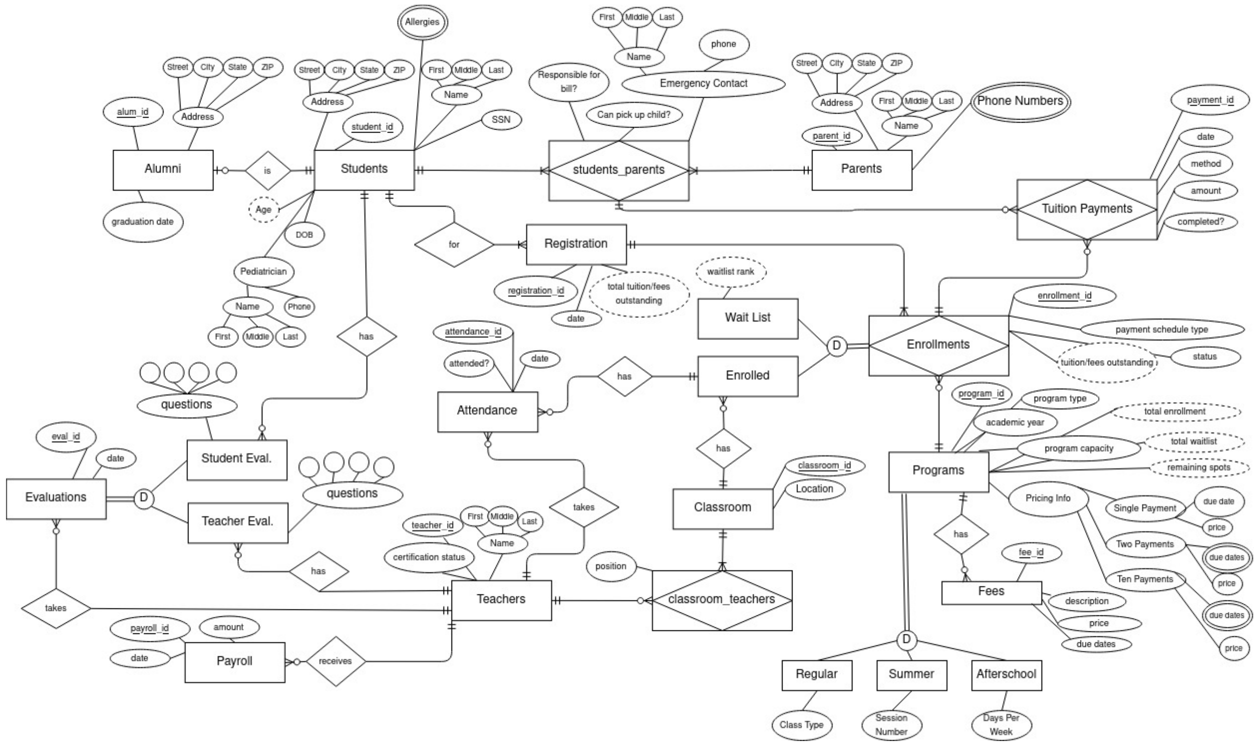
There are some "ground rules" we have agreed to follow:

1. We will use email as the preferred method of communication, and emails should be responded to within 24 hours (or the other team members should be notified of an expected delay in response).
2. We will use Google Calendar to keep track of scheduled meetings, and use a Google Drive shared folder to store collaborative group work.
3. Tasks will be assigned in an organic, volunteer-like manner; i.e. through discussion of the assignment's tasks and matching them to each members' strong suits.
4. Team contributions should be as equal as possible in amount and quality.
5. Team members should attend class as much as possible; when an expected absence can be planned for, other members should be notified (especially on days when group work will be a focus of the class meeting).

6. As with the assignment of tasks, general decisions will be made through group discussion and consensus.
7. Conflicts and disagreements will be avoided through compromise and discussion; if they arise, we ask that you, Dr. Wang, be a mediator to help reach a solution (though we do not expect this to be necessary).
8. Members will be held accountable through the honor system; as with above, non-internally-resolvable issues may require outside mediation, though with our past experiences working together we do not expect this to be necessary.

Our team is excited to take on this class, and the projects that come with it!

APPENDIX E — INITIAL REVISION ERD



Alumni
alum_id
Address
Street
City
State
ZIP
graduation date
is
Students
student_id
Address
Name
First
Middle
Last
SSN
Allergies
Age
DOB
Pediatrician
Phone
students_parents
Responsible for bill?
Can pick up child?
Emergency Contact
phone
Parents
parent_id
Phone Numbers
Tuition Payments
payment_id
date
method
amount
completed?
for
Registration
registration_id
date
total tuition/fees outstanding
Wait List
waitlist rank
D
Enrollments
enrollment_id
payment schedule type
tuition/fees outstanding
status
has
Attendance
attendance_id
attended?
date
Enrolled
Classroom
classroom_id
Location
takes
Programs
program_id
program type
academic year
program capacity
total enrollment
total waitlist
remaining spots
Pricing Info
Single Payment
Two Payments
Ten Payments
due date
due dates
price
Fees
fee_id
description
price
due dates
Regular
Summer
Afterschool
Class Type
Session Number
Days Per Week
questions
Student Eval.
Teacher Eval.
Evaluations
eval_id
date
Teachers
teacher_id
certification status
classroom_teachers
position
Payroll
payroll_id
amount
date
receives