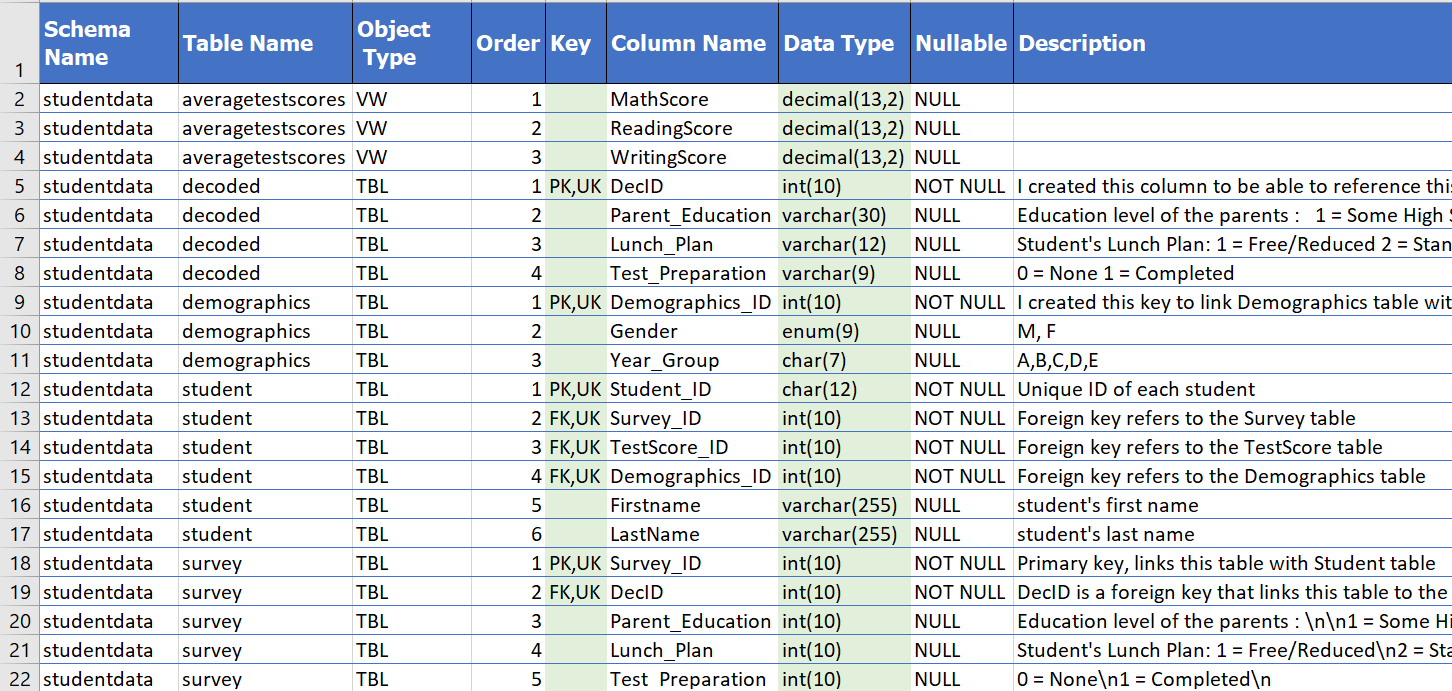
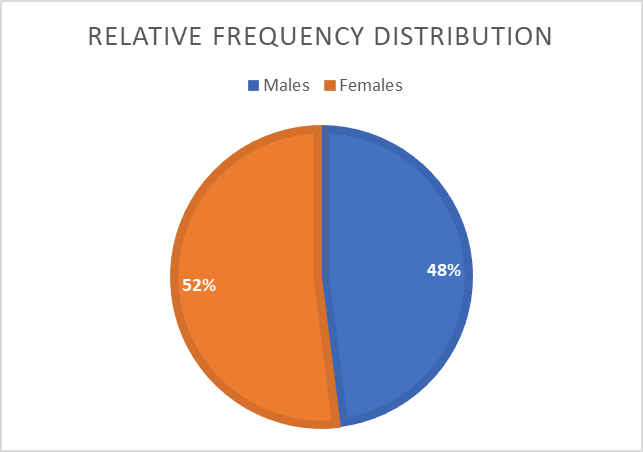
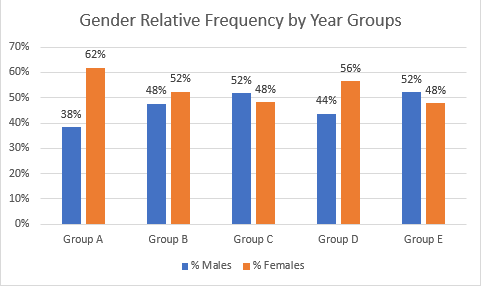
1. **Data Dictionary**

****

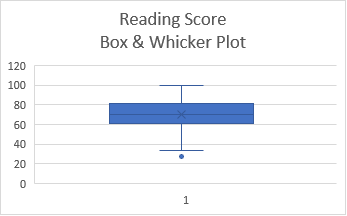
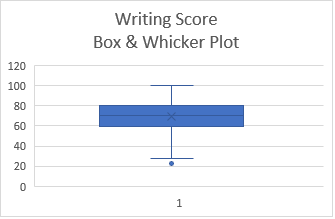
1. **Data Cleaning and Preparation**
2. In order to design a database that would fallow the Star Schema format, I had to modify the columns in the given dataset. As such, columns TestScore\_ID, Demographics\_ID, Survey\_ID, and later, Decoded\_ID were added and the tables were renamed. These columns serve as primary keys in the corresponding tables and as foreign keys in the Fact Table or, in the case with Decoded\_ID, in the Survey table.
3. I created a new table called Student. This table serves as a Fact Table in StarSchema; therefore, it has all the foreign keys mentioned in the paragraph above. Through this table, I intend to query the rest of our database.
   1. Columns Firstname and LastName were auto filled and contain an Easter egg.
   2. I sorted the tables based on the studentID and then added foreign keys from each dimension table into Student table. Then I removed StudentIDs from each of the dimension tables to eliminate redundancy.
      1. I made sure to preserve original data integrity and referential integrity before removing the StudentID column from the Dimension tables.
   3. I wanted to reference Dimension tables from the Student table (fact table) for that reason the StudentID column was redundant in there.
4. Checked for duplicates or null values in the raw data by using Conditional Formatting and Sorting. I did not find any duplicates, nor null values.
5. I will keep to file formats csv for MYSQL and BIGQUERY imports and xls for presentation purposes.
   1. The formatting of csv files changes.

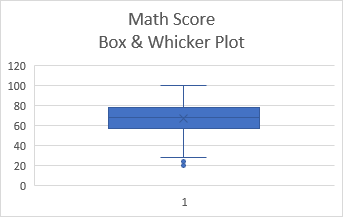
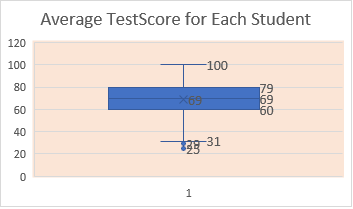
**FACED A BLOCKER:** In the raw data tables were referenced by the studentID. This reversed the relationship that was needed for StarSchema. In the Star Schema the Fact table must reference the Dimension tables, not the other way around. The fact table is the starting point through which the rest of the database is queried and it contains the foreign keys of other tables. Therefore, I had to modify the database by creating new tables, adding new columns, and deleting redundant StudentID column from the tables.

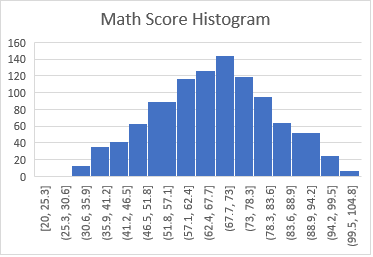
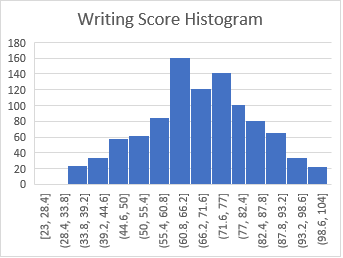
1. **Exploratory Data Analysis**
2. perform exploratory data analysis with plots of all variables, note any potential outliers or influential points, include scatter plots or histograms or pie charts etc, whichever appropriate, any also perform simple two-way comparisons with variables
   1. check for trends etc.
   2. perform comparisons with variables
   3. do appropriate charts based on data
   4. show a lot of visuals of the data
3. Exploration of raw data:
   1. Genders (charts)

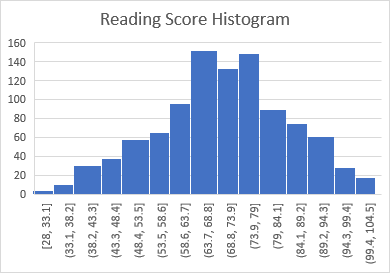
 

* 1. Parent education (charts)
  2. How many did parent lunch vs how many did not

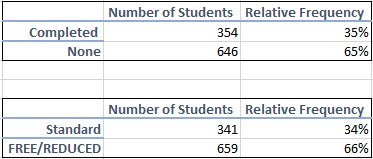
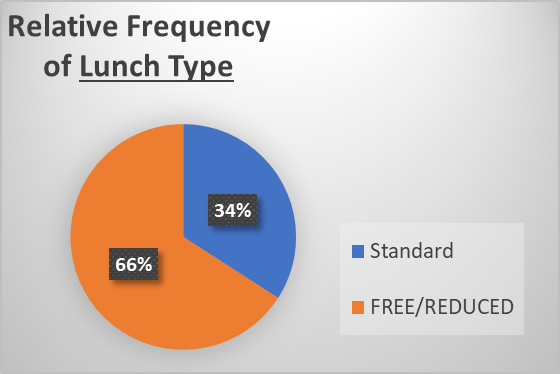
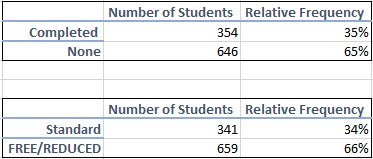
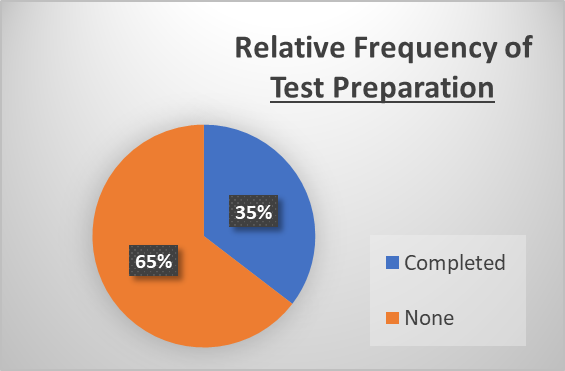
1. Identifying Outliers:
   1. I used Box & Whicker plots to identify outliers in the test scores grouped by subject.

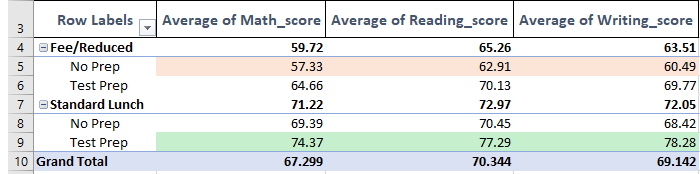
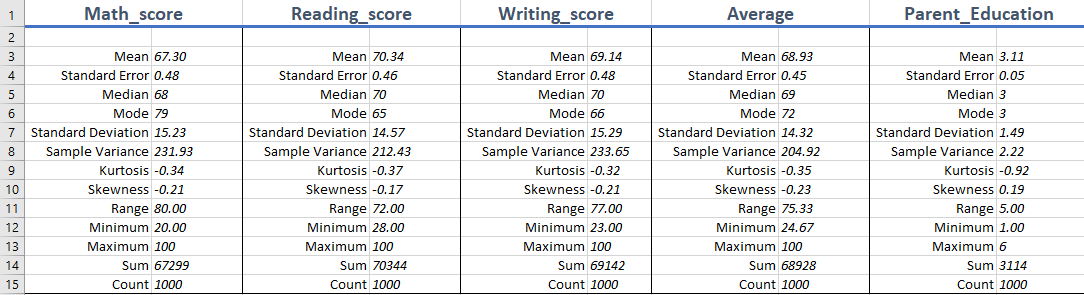
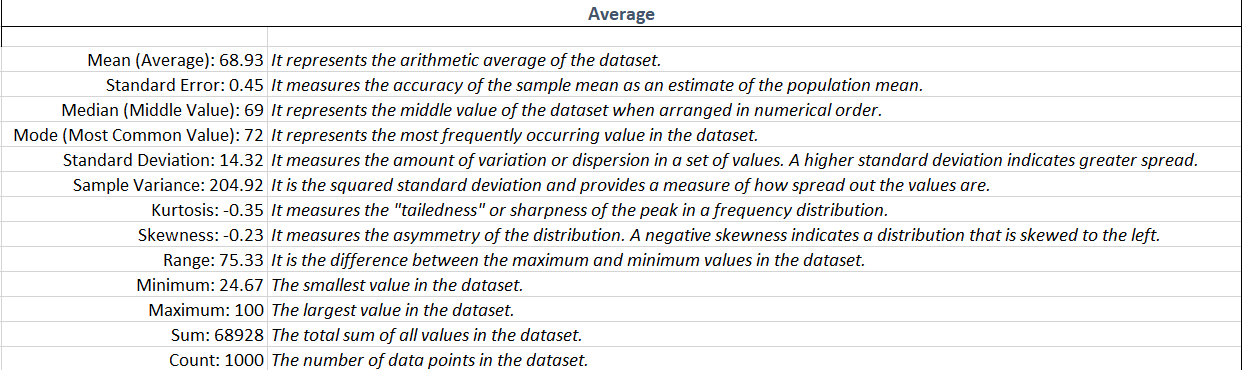


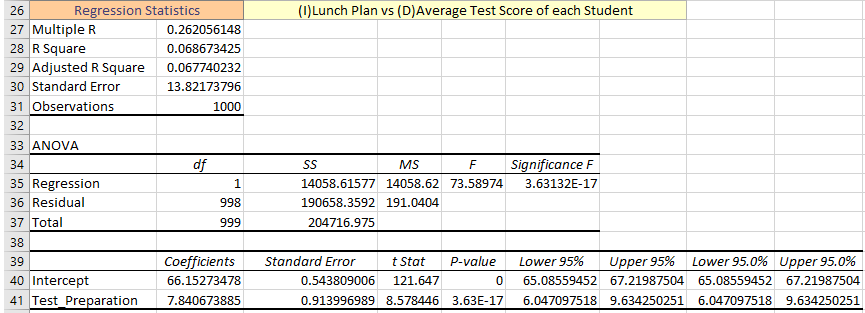
1. Checking distribution of the test scores



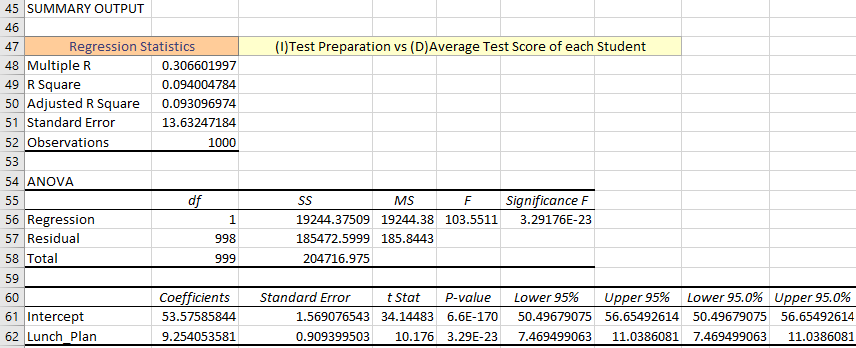
1. Comparing relative frequency of students who had lunch with those who did not. Also, comparing relative frequency of students completing the test preparation against those who did not complete it.



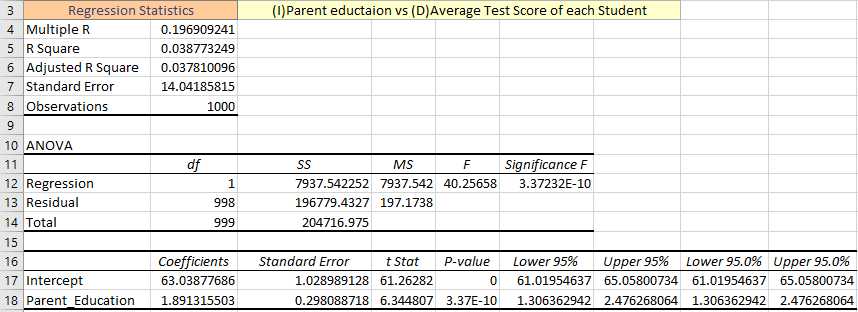
1. Cross tabulations for Average test scores based on the type of lunch and the test preparation.
2. I used Descriptive Statistics function in excel to calculate the various metrics in the Student Data. The table below highlights the results of the relevant columns.
   1. I calculated the combined average of the Test Scores (Average).
   2. We can see the mean, standard deviation, standard error, range, and count of each column.
   3. ****Interpretation of the Average column:
3. Regression Analysis:
   1. Lunch Plan vs Average Test Score



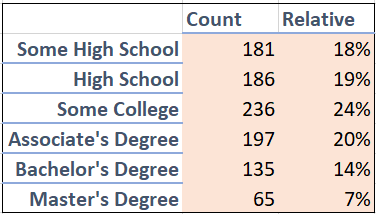
* 1. Test Preparation vs Average Test Scores

****

* 1. Parent Education vs Average Test Scores



1. Distribution of Education Levels

****

**MYSQL SCHEMA:**

CREATE DATABASE StudentData; -- creating database

USE StudentData; -- making StudentData an active database

DROP DATABASE StudentData;

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Creating Tables\*\*\*\*\*\*\*\*\*\*

Drop table Student;

CREATE TABLE Student (

Student\_ID char(12) PRIMARY KEY CHECK (char\_length(Student\_ID) = 12), -- each consists of 12 characters

Survey\_ID INT CHECK (Survey\_ID >= 30000 AND Survey\_ID <=39999),

TestScore\_ID INT CHECK (TestScore\_ID >= 20000 AND TestScore\_ID <=29999),

Demographics\_ID INT CHECK (Demographics\_ID >= 10000 AND Demographics\_ID <=19999),

Firstname VARCHAR(255) DEFAULT 'First Name is Missing!',

LastName VARCHAR(255)DEFAULT 'Last Name is Missing!',

FOREIGN KEY (Survey\_ID) REFERENCES Survey(Survey\_ID) ON DELETE CASCADE,

FOREIGN KEY (TestScore\_ID) REFERENCES TestScore(TestScore\_ID) ON DELETE CASCADE,

FOREIGN KEY (Demographics\_ID) REFERENCES Demographics(Demographics\_ID) ON DELETE CASCADE)

;

Drop table Survey;

CREATE TABLE Survey (

Survey\_ID INT PRIMARY KEY CHECK (Survey\_ID >= 30000 AND Survey\_ID <=39999),

DecID INT CHECK (DecID >= 40000 AND DecID <=49999),

Parent\_Education int DEFAULT 0 CHECK (Parent\_Education >= 1 AND Parent\_Education <= 6),

Lunch\_Plan int DEFAULT 0 CHECK (Lunch\_Plan >= 1 AND Lunch\_Plan <= 6),

Test\_Preparation int DEFAULT 0 CHECK (Test\_Preparation >= 0 AND Test\_Preparation <= 100),

FOREIGN KEY (DecID) REFERENCES Decoded(DecID) ON DELETE CASCADE

);

Drop table TestScore;

CREATE TABLE TestScore (

TestScore\_ID INT PRIMARY KEY CHECK (TestScore\_ID >= 20000 AND TestScore\_ID <=29999),

Math\_Score INT CHECK (Math\_Score >= 0 AND Math\_Score <= 100),

Reading\_Score INT CHECK (Reading\_Score >= 0 AND Reading\_Score <= 100),

Writing\_Score INT CHECK (Writing\_Score >= 0 AND Writing\_Score <= 100)

);

Drop table Demographics;

CREATE TABLE Demographics (

Demographics\_ID INT PRIMARY KEY CHECK (Demographics\_ID >= 10000 AND Demographics\_ID <=19999),

Gender ENUM('M', 'F', 'Undecided') DEFAULT 'Undecided', -- enum data types list a list of possible values

Year\_Group CHAR(7) DEFAULT 'Retain' -- each group has 7 characters; default has to be retained for another year

);

Drop table Decoded;

CREATE TABLE IF NOT EXISTS Decoded (

DecID INT PRIMARY KEY CHECK (DecID >= 40000 AND DecID <=49999),

Parent\_Education VARCHAR(30) DEFAULT 'Aj Aj jaj!',

Lunch\_Plan VARCHAR(12) DEFAULT 'Olegs work',

Test\_Preparation VARCHAR(9) DEFAULT 'Donde?'

);

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

IDdata Table:

IDdata\_ID is the primary key of the IDdata table, ensuring that each record in this table is uniquely identified by its IDdata\_ID.

Student\_ID, TestScore\_ID, and Demographics\_ID are marked as UNIQUE, indicating that each of these IDs must be unique across all records in the IDdata table. This implies that each Student\_ID, TestScore\_ID, and Demographics\_ID in the IDdata table is associated with at most one record in the corresponding tables (Student, TestScore, Demographics).

**Student Table:**

Student\_ID is the primary key of the Student table, ensuring that each student record is uniquely identified by its Student\_ID.

IDdata\_ID is marked as UNIQUE, implying that each Student record is associated with at most one record in the IDdata table.

FOREIGN KEY (IDdata\_ID) REFERENCES IDdata(IDdata\_ID) establishes a foreign key relationship, indicating that the IDdata\_ID in the Student table references the IDdata\_ID in the IDdata table. This implies a one-to-zero-or-one relationship between Student and IDdata.

**TestScore Table:**

TestScore\_ID is the primary key of the TestScore table, ensuring that each test score record is uniquely identified by its TestScore\_ID.

IDdata\_ID is marked as UNIQUE, implying that each TestScore record is associated with at most one record in the IDdata table.

FOREIGN KEY (IDdata\_ID) REFERENCES IDdata(IDdata\_ID) establishes a foreign key relationship, indicating that the IDdata\_ID in the TestScore table references the IDdata\_ID in the IDdata table. This implies a one-to-zero-or-one relationship between TestScore and IDdata.

**Demographics Table:**

Demographics\_ID is the primary key of the Demographics table, ensuring that each demographics record is uniquely identified by its Demographics\_ID.

IDdata\_ID is marked as UNIQUE, implying that each Demographics record is associated with at most one record in the IDdata table.

FOREIGN KEY (IDdata\_ID) REFERENCES IDdata(IDdata\_ID) establishes a foreign key relationship, indicating that the IDdata\_ID in the Demographics table references the IDdata\_ID in the IDdata table. This implies a one-to-zero-or-one relationship between Demographics and IDdata.

Note: **TO DO POSSIBLE**

* Add constraints what happens to these columns when the student is removed. On DELETE xxx
* Look for relationships and correlations, for example students who at xx generally got results that were xx above the mean(standard deviation)

Work:

**Excel**:

1. Created fact table Student
   1. Demographics table could be combined with Student (fact) table without any consequence; however, I will make it into a separate table just to have more tables here for practice. Alternatively, I might create my own, made up, additional columns just to add data in this database for practice and complexity.
2. Created additional columns for each dimension table
3. Removed unnecessary StudentId from dimension tables

**Exploratory analysis in excel:**

1. Count all necessary fields
2. Get % rates relative to one another
3. Count number of students in each:
   1. year group
   2. females and males in each grade
4. % of students in each grade group that got lunch
   1. % of students in each grade group based on gender
   2. % that got lunch based on gender
5. % of students grouped by parent education
6. Created box plots based on test results. This way it is easy to identify outliers.
   1. We do not have context about outliers – are they naturally occurring?
   2. I will keep them for now; however, if removed, there might be a significant effect on the queried data.
7. Created Histograms on Test Score data
   1. Normal distribution for all three tests
8. . Calculate basic summary statistics such as mean, median, mode, standard deviation, and range for each score (Reading Score, Math Score, etc.).

**MySQl:**

1. Established relationships between tables
2. Created database in mysql and created tables with appropriate column names
   1. Added constrains such as on delete
   2. Added defaults
   3. On Delete Cascade – setting constraint on foreign keys in tables that are referencing parent table (student) so they’ll be deleted if Student\_id delted in parent table.
   4. Set up check constrains to primary keys to limit the ids to preset standards; but studentID definitely needs a check constraint since studentIDs are 12 characters long.
3. Generated erd star schema and made editions
4. Combined tables into one document in excel, multiple worksheets
5. Inserted appropriate columns into each table based on star schema
   1. Foreign keys and autogenerated random with RANDBETWEEN (the paste special to paste them as simple values so they won’t change with each worksheet update)
      1. TestScore\_ID: 20000-29999
      2. Demographics\_ID: =RANDBETWEEN(10000,19999)
      3. Survey\_ID: =RANDBETWEEN(30000,39999)
   2. Added student names (why no names in originals?)
      1. Easter egg: Oleg Zasukha 60D2154DAB86
      2. Separated by Data>TextToColumns>Deliminated>space>Text/Text
      3. \*\***maybe make Oleg an outlier – add data**
6. Saved workbooks as comma delaminated to be able to import into mysql

**Back to mysql:**

1. Importing into mysql via table data wizard:
   1. Changing type of StudentID fields in mysql from int to char(12) – all the given studentIDs are 12 characters long
      1. Was going to use modify, but it’s a primary key in Student, so its much easier here to start over rather than drop foreign keys on referencing tables and modify and then add constraint again.
      2. Successfully imported Student table data – had to import it first since other tables reference it.
      3. ERROR when trying to import Demographics table:
         1. Row import failed with error: ("Duplicate entry '17668' for key 'demographics.PRIMARY'", 1062)
         2. I guess there should be one to many relationship; demographics
            1. NOOO I probably generated RANDBETWEEN with duplicates
            2. -------back to excel-----
            3. Used conditional formatting to create new rule to highlight duplicates. Then I sorted ids based on color.
            4. Used =$A$2+RANDBETWEEN(10,1000) to generate numbers for duplicates – worked like a charm!
         3. Will do the same for other tables

------------------------------\*\*\*RESTART\*\*\*-----------------------------------

\***blocker:** the state of the tables: each table had student id column which I had to remove because of the referential relationship which I wanted to implement between the dimension tables and user-created Fact table.

1. In the original raw data, each of the three dimension tables contains studentID. However, I created a separate student table. This student table would be the mina table that we will query (fact table). Therefore, it must reference the dimension tables, not the other way around. This means that it must contain foreign keys to dimension tables instead of having Student\_ID in each table as a foreign key. 2/3/24 looked it today and realized that it should be re-done.
2. Plan:
   1. Move all the data in excel as needed
      1. I can sort by student\_id and then delete it while moving all the primary keys into the Student table as foreign keys.
      2. In mysql do everything by practicing commands – don’t just drop!
      3. Then import the data back into mysql.
3. I sorted every column in each table in excel based on student id and then copied each table’s primary key into student table and removed studentid columns. This way I preserved the original data integrity respective to each student.
   1. Csv files will change formatting of the tables in comparison to excel
      1. To import into databases: save as csv and remove ‘table’ feature
      2. For presentation: format xls files nicely and add screenshots from them.
4. Make appropriate editions to database schema. Later I could alter the columns etc. At the moment, a lot of things need to be moved around and all the data needs to be truncated to preserve the integrity of the original data.
   1. First create Survey, Demographics, and TestScore tables because they will be referenced by Student. Then create Student. Alternative, would be creating Student and then establishing referential relationships.
   2. Now I will import all the data into workbench incase ill need it there later.
      1. Everything went smoothly except for TestScore table import. It reported correctly, but while importing the database would show error message, crush, and close. Yet, despite that, all the data was imported correctly.
   3. I run some queries to check data integrity. The values in my tables much the raw data!
5. Done with data formatting and mySQl, moving on to GCP and BIGQUERY.

**GCP BIG QUERY:**

1. Imported everything – works like a charm!!
2. First Join:

select st.Student\_ID, st.FirstName, st.LastName, avg((ts.Math\_score + ts.Reading\_score + ts.Writing\_score)/3) as CombinedAverage

from `studentdata-project0.StudentData.Student` as st

join `studentdata-project0.StudentData.TestScore` ts

on st.TestScore\_ID = ts.TestScore\_ID

Group by st.Student\_ID, firstname, lastname -- in select using aggregate functions in conjunction with non-aggregate, so group by is necessary to

-- group the results for non-aggregated columns;

order by CombinedAverage desc, st.LastName desc

limit 10;

1. Realized that I need to make another table with decoded values – no bueno
   1. “You may need to create your own data to include the decoded values for student\_exam\_survey.csv.”

**Back to Excel:**

1. Creating new csv file ‘Decoded’
   1. I will copy/paste Survey\_ID
   2. I will generate DecID (easy way this time!) range 40000 – 49999
      1. Used formula =B2+10, took 15 seconds to do all, should have done so on all columns that I’ve defined; I will learn.
      2. Copy/special paste AS values (to transform datatype into simple values)
   3. I will use IF/Then/Else conditions to decode the data
      1. I will copy/paste al the data from Survey table into Survey worksheet on my Decoded table workbook.
         1. ParentEduction:
            1. =IF(Survey!B2=1, "Some High School", IF(Survey!B2=2, "High School", IF(Survey!B2=3, "Some College", IF(Survey!B2=4, "Associates Degree", IF(Survey!B2=4, "Bachelors Degree", “Masters Degree”))))
         2. Lunch Plan
            1. =IF(Survey!C2=1, "Free/Reduced", "Standard")
         3. Test Preparation
            1. =IF(Survey!D2=0, "None", "Completed")
         4. Copy/Paste as values to remove the formulas since they won’t be saved in .csv
   4. Copy/Paste DecID from Decoded.csv to our Survey.csv
      1. Deleted SurveyID from Decoded.csv – we want this table to be referenced by Survey.csv table, so DecID should be in the Survey table.
      2. Data integrity preserved!

**WORKBENCH:**

1. Write new SQL statements to create new table and make additions to Survey table.

CREATE TABLE IF NOT EXISTS Decoded (

DecID INT PRIMARY KEY CHECK (DecID >= 40000 AND DecID <=49999),

Parent\_Education VARCHAR(18) DEFAULT 'Aj Aj jaj!',

Lunch\_Plan VARCHAR(12) DEFAULT 'Donde esta mi gato!',

Test\_Preparation VARCHAR(9) DEFAULT 'Yo no tengo un gato!'

);

* 1. When importing data, only 603 records were imported out of 1000 because “Parent\_Education was too long”
     1. Modifying column:
        1. Alter table Decoded

MODIFY COLUMN Parent\_Education VARCHAR(30) DEFAULT 'AJ AJ AJ';

* + - 1. Truncate table Decoded;
    1. Using ‘wizard import data’ to import data again.
    2. Success!

1. Write mySQL commands to alter Survey table data:

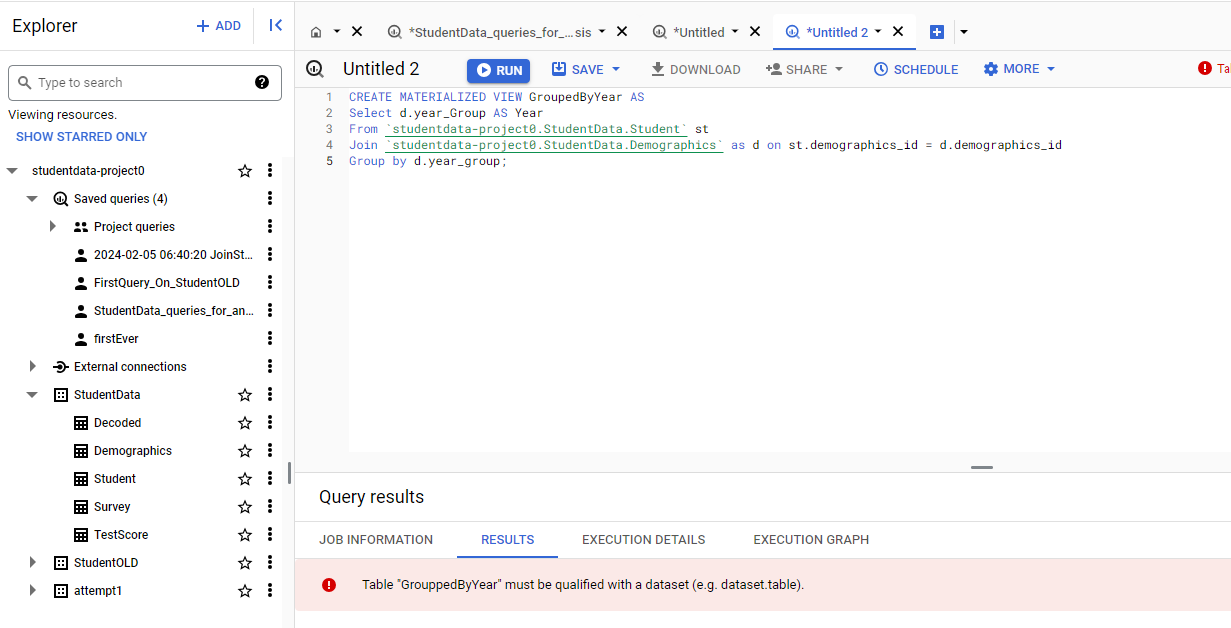
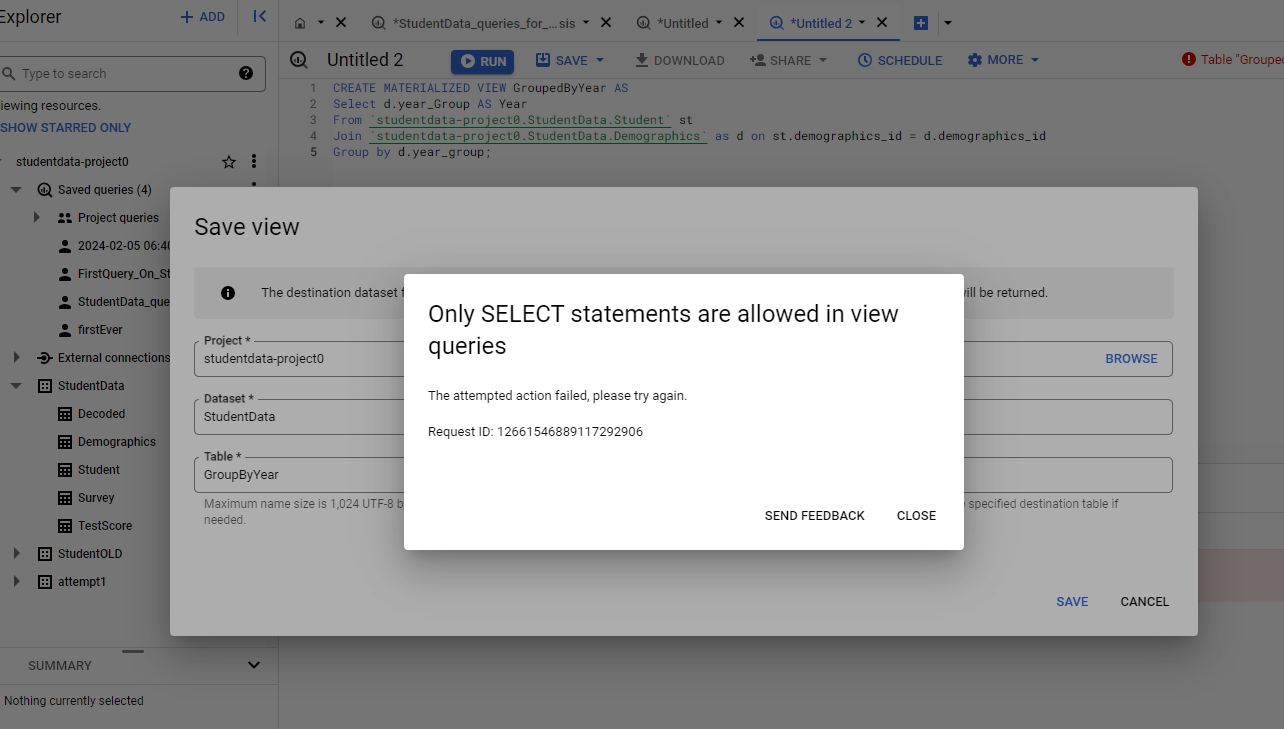
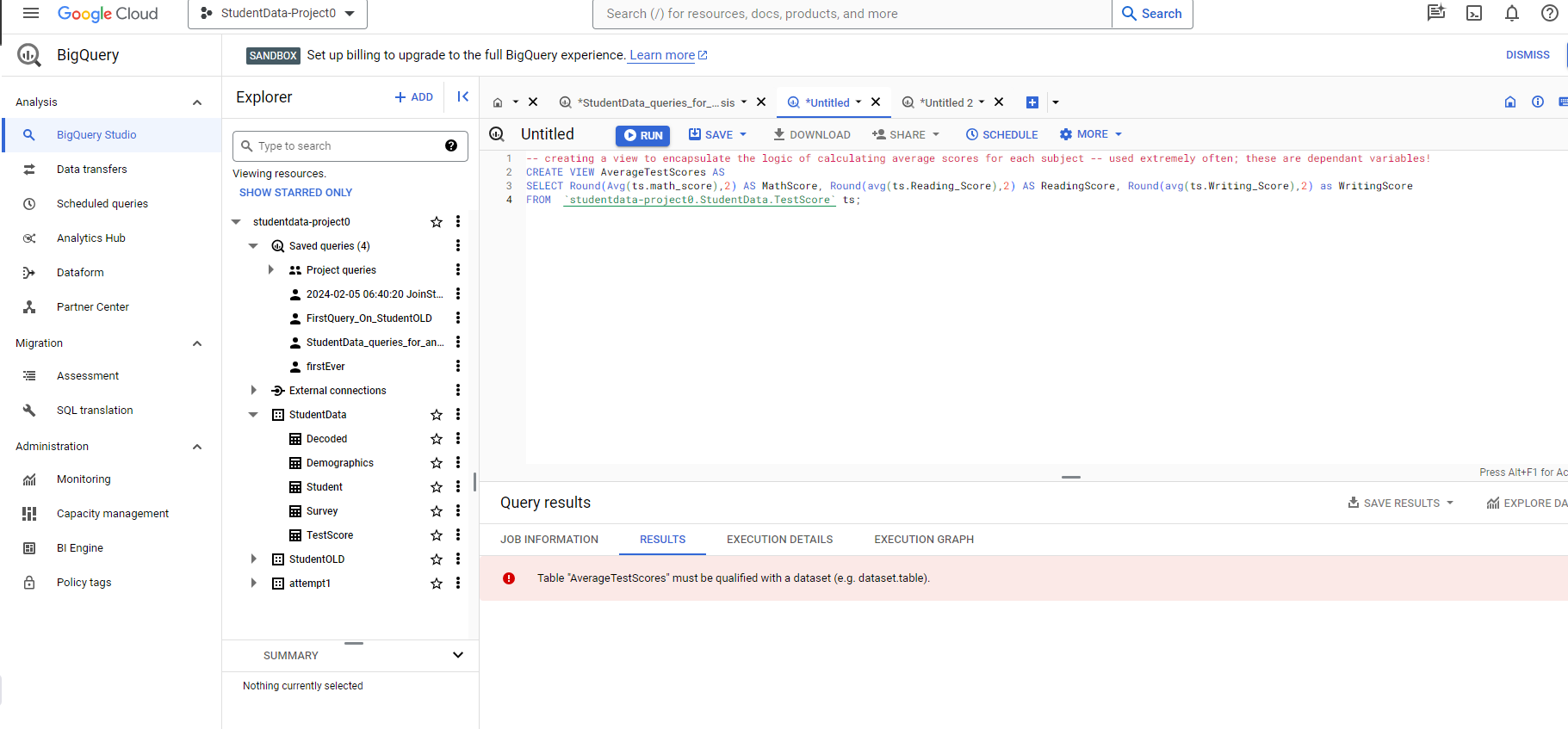
**GCP BigQuery:**

1. Created a new dataset called StudentOld and moved all the data from first attempt into it.

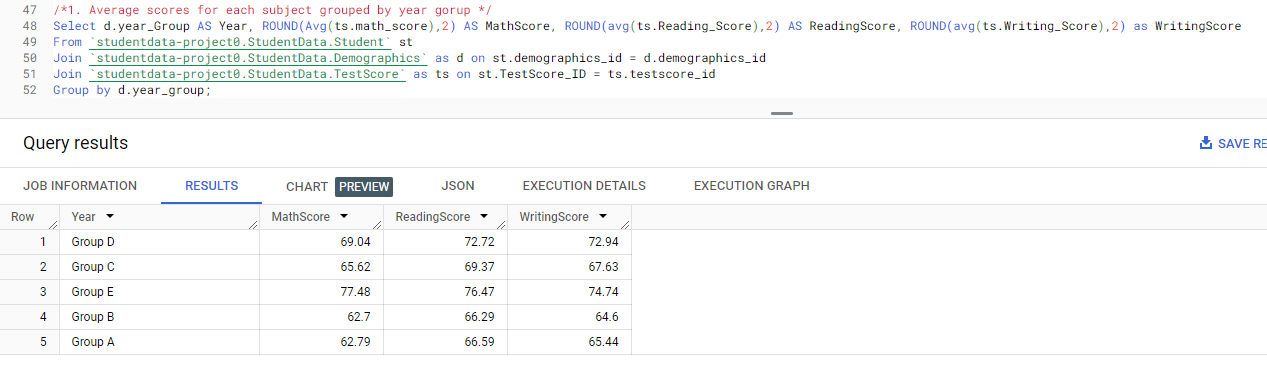
2. imported all the latest data into StudentData.

3. Time to write queries and analyze the data!

4. I tried to create views in BigQuery, but I kept getting an error message:



**My Queries:**

1. ****/\*1. Average scores for each subject grouped by year gorup \*/

Group E has the highest average scores:

* Group E has the highest average scores in all three subjects (Math, Reading, and Writing), indicating strong academic performance.
* They scored highest in every single category

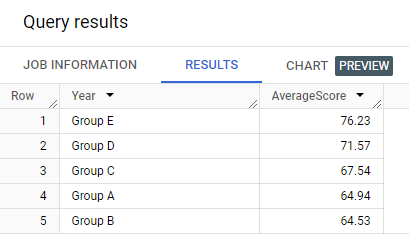
Group D and Group C show consistent scores:

* Both Group D and Group C exhibit consistent scores across the three subjects, with Group D having slightly higher scores.

Group B has lower scores compared to other groups:

* Group B has relatively lower average scores in Math, Reading, and Writing compared to the other groups.
* Scored lowest in every single category

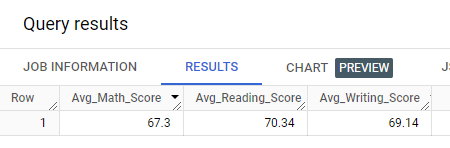
1. /\*2. Combined average scores for each year group \*/



Group E – highest combined average score

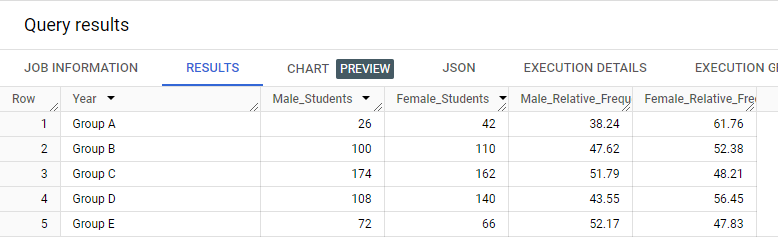
Group B – lowest combined average score

1. 3 Simply Average scores for each subject



On average, students got the highest scores on reading tests and the lowest on math exams.

1. --4. Number of males and females in each year group and relative to group's total number of students



Group C has the largest number of students:

* Group C has the highest number of students (174), indicating the largest population among the five groups.
* Group C has the highest male percentage (51.79%),

Group A has the smallest number of students:

* Group A has the smallest number of students (26), indicating the smallest population among the five groups.
* Group A has the highest female percentage (61.76%).

Overall, the gender distribution varies across groups, and this information could be valuable for further analyses related to academic performance or other factors that may be influenced by gender.

**Queries BigQuery**

SELECT

  s.Student\_ID,

  s.Firstname,

  s.LastName,

  s.Survey\_ID,

  s.TestScore\_ID,

  s.Demographics\_ID,

  d.Gender,

  d.Year\_Group,

  t.Math\_Score,

  t.Reading\_Score,

  t.Writing\_Score,

  dec.Lunch\_Plan,

  dec.Parent\_Education,

  dec.Test\_Preparation

FROM

  `studentdata-project0.StudentData.Student` s

JOIN

  `studentdata-project0.StudentData.Survey` sur ON s.Survey\_ID = sur.Survey\_ID

JOIN

  `studentdata-project0.StudentData.Demographics` d ON s.Demographics\_ID = d.Demographics\_ID

JOIN

  `studentdata-project0.StudentData.TestScore` t ON s.TestScore\_ID = t.TestScore\_ID

JOIN

  `studentdata-project0.StudentData.Decoded` dec ON sur.DecID = dec.DecID;

-- Adding Key constrains to enforce rules on relationships betweeen table; however, in BigQuery they are not inforced.

/\*

ALTER TABLE `studentdata-project0.StudentData.Decoded`

ADD PRIMARY KEY (DecID) NOT ENFORCED;

ALTER TABLE `studentdata-project0.StudentData.Survey`

ADD PRIMARY KEY (Survey\_ID) NOT ENFORCED;

ALTER TABLE `studentdata-project0.StudentData.TestScore`

ADD PRIMARY KEY (TestScore\_ID) NOT ENFORCED;

ALTER TABLE `studentdata-project0.StudentData.Demographics`

ADD PRIMARY KEY (Demographics\_ID) NOT ENFORCED;

ALTER TABLE `studentdata-project0.StudentData.Survey`

ADD FOREIGN KEY (DecID) REFERENCES `studentdata-project0.StudentData.Decoded`(DecID) NOT ENFORCED;

---------------Student Table

ALTER TABLE `studentdata-project0.StudentData.Student`

ADD PRIMARY KEY (Student\_ID) NOT ENFORCED,

ADD FOREIGN KEY (Survey\_ID) REFERENCES `studentdata-project0.StudentData.Survey`(Survey\_ID) NOT ENFORCED,

ADD FOREIGN KEY (TestScore\_ID) REFERENCES `studentdata-project0.StudentData.TestScore`(TestScore\_ID) NOT ENFORCED,

ADD FOREIGN KEY (Demographics\_ID) REFERENCES `studentdata-project0.StudentData.Demographics`(Demographics\_ID) NOT ENFORCED;\*/

/\*1. Jennifer: Average scores for each subject grouped by year gorup \*/

Select d.year\_Group AS Year, ROUND(Avg(ts.math\_score),2) AS MathScore, ROUND(avg(ts.Reading\_Score),2) AS ReadingScore, ROUND(avg(ts.Writing\_Score),2) as WritingScore

From `studentdata-project0.StudentData.Student` st

Join `studentdata-project0.StudentData.Demographics` as d on st.demographics\_id = d.demographics\_id

Join `studentdata-project0.StudentData.TestScore` as ts on st.TestScore\_ID = ts.testscore\_id

Group by d.year\_group;

/\*2.Natalie: Combined average scores for each year group \*/

Select d.year\_Group AS Year, Round(Avg((ts.math\_score +ts.Reading\_Score+ts.Writing\_Score)/3),2) AS AverageScore

From `studentdata-project0.StudentData.Student` st

Join `studentdata-project0.StudentData.Demographics` as d on st.demographics\_id = d.demographics\_id

Join `studentdata-project0.StudentData.TestScore` as ts on st.TestScore\_ID = ts.testscore\_id

Group by d.year\_group

Order by AverageScore desc;

-- 3. Nora: Simply Average scores for each subject

SELECT

  ROUND(Avg(ts.Math\_Score),2) AS Avg\_Math\_Score,

  ROUND(Avg(ts.Reading\_Score),2) AS Avg\_Reading\_Score,

  ROUND(Avg(ts.Writing\_Score),2) AS Avg\_Writing\_Score

FROM

  `studentdata-project0.StudentData.Student` st

JOIN

  `studentdata-project0.StudentData.TestScore` ts ON st.TestScore\_ID = ts.testscore\_id;

--3.Louisa: Average scores across subjects grouped by gender

Select d.Gender AS Gender, Round(Avg(ts.math\_score),2) AS MathScore, Round(avg(ts.Reading\_Score),2) AS ReadingScore, Round(avg(ts.Writing\_Score),2) as WritingScore, Round(Avg(ts.math\_score+ts.Reading\_Score + ts.Writing\_score)/3,2) AS CombinedAverageScore

From `studentdata-project0.StudentData.Student` st

Join `studentdata-project0.StudentData.Demographics` as d on st.demographics\_id = d.demographics\_id

Join `studentdata-project0.StudentData.TestScore` as ts on st.TestScore\_ID = ts.testscore\_id

Group by d.Gender;

--4.Aurora:  Number of males and females in each year group and relative to group's total number of students

SELECT

  d.year\_Group AS Year,

  COUNTIF(d.Gender = 'M') AS Male\_Students,

  COUNTIF(d.Gender = 'F') AS Female\_Students,

  ROUND(COUNTIF(d.Gender = 'M') / COUNT(st.Student\_ID) \* 100, 2) AS Male\_Relative\_Frequency,

  ROUND(COUNTIF(d.Gender = 'F') / COUNT(st.Student\_ID) \* 100, 2) AS Female\_Relative\_Frequency

FROM

  `studentdata-project0.StudentData.Student` st

JOIN

  `studentdata-project0.StudentData.Demographics` as d ON st.demographics\_id = d.demographics\_id

GROUP BY

  Year

ORDER BY

  Year;

--4. Average test scores grouped by parental level

SELECT

  d.Parent\_Education AS ParentDegree, s.Parent\_Education AS ParentEducationLevel,

  Round(AVG(ts.Math\_Score),2) AS AvgMathScore,

  Round(AVG(ts.Reading\_Score),2) AS AvgReadingScore,

  Round(AVG(ts.Writing\_Score), 2) AS AvgWritingScore,

  Round(AVG(ts.Writing\_Score+ts.Math\_score + ts.Reading\_score)/3, 2) AS CombinedAverageScore,

FROM

  `studentdata-project0.StudentData.Student` st

JOIN

  `studentdata-project0.StudentData.Survey`  s ON st.Survey\_ID = s.Survey\_ID

JOIN

  `studentdata-project0.StudentData.Decoded` d ON s.DecID = d.DecID

JOIN

  `studentdata-project0.StudentData.TestScore` ts ON st.TestScore\_ID = ts.TestScore\_ID

GROUP BY

  d.Parent\_Education, s.Parent\_Education

ORDER BY

  s.Parent\_Education desc;

  --\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

  --\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- 4.5 Standard Deviations

SELECT

  Round(STDDEV(ts.reading\_score),2) AS ReadingScoreStdDev,

  Round(STDDEV(ts.writing\_score),2) AS WritingScoreStdDev,

  Round(STDDEV(ts.math\_score),2) AS MathScoreStdDev

FROM

  `studentdata-project0.StudentData.Student` st

JOIN

  `studentdata-project0.StudentData.Demographics` AS d ON st.demographics\_id = d.demographics\_id

JOIN

  `studentdata-project0.StudentData.TestScore` AS ts ON st.TestScore\_ID = ts.testscore\_id;

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

--5.  Did students who took the test prep course scored higher? Does test preparation affect the scores? How different it is for each of the categories?  -

Select d.Test\_Preparation, Round(Avg(ts.math\_score),2) AS MathScore, Round(avg(ts.Reading\_Score),2) AS ReadingScore, Round(avg(ts.Writing\_Score),2) as WritingScore, Round(Avg(ts.math\_score+ ts.Reading\_score+ ts.Writing\_score) / 3,2) AS Combined\_Average\_Score

FROM `studentdata-project0.StudentData.Student` st

Join `studentdata-project0.StudentData.TestScore` as ts on st.TestScore\_ID = ts.testscore\_id

JOIN

  `studentdata-project0.StudentData.Survey`  s ON st.Survey\_ID = s.Survey\_ID

JOIN

  `studentdata-project0.StudentData.Decoded` d ON s.DecID = d.DecID

GROUP BY d.Test\_Preparation;

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- 5.1. Did the meal plan affect test scores?

Select d.Lunch\_Plan, Round(Avg(ts.math\_score),2) AS MathScore, Round(avg(ts.Reading\_Score),2) AS ReadingScore, Round(avg(ts.Writing\_Score),2) as WritingScore, Round(Avg(ts.math\_score+ts.Reading\_score+ts.Writing\_score)/3,2) AS CombinedAverageScore

FROM `studentdata-project0.StudentData.Student` st

Join `studentdata-project0.StudentData.TestScore` as ts on st.TestScore\_ID = ts.testscore\_id

JOIN

  `studentdata-project0.StudentData.Survey`  s ON st.Survey\_ID = s.Survey\_ID

JOIN

  `studentdata-project0.StudentData.Decoded` d ON s.DecID = d.DecID

GROUP BY d.Lunch\_Plan;

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*IMPORTANT\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- 5.5 Test Scores grouped by both Preparaion and Lunch

SELECT

  d.Test\_Preparation,

  d.Lunch\_Plan,

  ROUND(AVG(ts.Math\_Score), 2) AS Avg\_Math\_Score,

  ROUND(AVG(ts.Reading\_Score), 2) AS Avg\_Reading\_Score,

  ROUND(AVG(ts.Writing\_Score), 2) AS Avg\_Writing\_Score,

  Round(Avg(ts.math\_score+ ts.Reading\_score+ ts.Writing\_score) / 3,2) AS Combined\_Average\_Score

FROM

  `studentdata-project0.StudentData.Student` st

JOIN

  `studentdata-project0.StudentData.Survey` s ON st.Survey\_ID = s.Survey\_ID

JOIN

  `studentdata-project0.StudentData.Decoded` d ON s.DecID = d.DecID

JOIN

  `studentdata-project0.StudentData.TestScore` ts ON st.TestScore\_ID = ts.testscore\_id

GROUP BY

  d.Test\_Preparation,

  d.Lunch\_Plan

ORDER BY

  d.Test\_Preparation,

  d.Lunch\_Plan;

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- 5.65.Victoria : Obsolete - MOVE ON TO 5.7: correlation coefficients of student test results based on students who had BOTH standard meal plan AND have completed TEST PREP

-- And it is followed by: FREE meal plan and None Test\_preparation

-- Find correlation coefficients between test scores and multiple independent variables from the Survey table

SELECT

  CORR(ts.Math\_Score, IF(d.Lunch\_Plan = 'Standard' AND d.Test\_Preparation = 'Completed', 1, 0)) AS Correlation\_Math\_Lunch\_TestPrep,

  CORR(ts.Reading\_Score, IF(d.Lunch\_Plan = 'Standard' AND d.Test\_Preparation = 'Completed', 1, 0)) AS Correlation\_Reading\_Lunch\_TestPrep,

  CORR(ts.Writing\_Score, IF(d.Lunch\_Plan = 'Standard' AND d.Test\_Preparation = 'Completed', 1, 0)) AS Correlation\_Writing\_Lunch\_TestPrep

FROM

  `studentdata-project0.StudentData.Student` st

JOIN

  `studentdata-project0.StudentData.Survey` s ON st.Survey\_ID = s.Survey\_ID

JOIN

  `studentdata-project0.StudentData.Decoded` d ON s.DecID = d.DecID

JOIN

  `studentdata-project0.StudentData.TestScore` ts ON st.TestScore\_ID = ts.testscore\_id;

SELECT

  CORR(ts.Math\_Score, IF(d.Lunch\_Plan = 'Free/Reduced' AND d.Test\_Preparation = 'None', 1, 0)) AS Correlation\_Math\_Lunch\_TestPrep,

  CORR(ts.Reading\_Score, IF(d.Lunch\_Plan = 'Free/Reduced' AND d.Test\_Preparation = 'None', 1, 0)) AS Correlation\_Reading\_Lunch\_TestPrep,

  CORR(ts.Writing\_Score, IF(d.Lunch\_Plan = 'Free/Reduced' AND d.Test\_Preparation = 'None', 1, 0)) AS Correlation\_Writing\_Lunch\_TestPrep

FROM

  `studentdata-project0.StudentData.Student` st

JOIN

  `studentdata-project0.StudentData.Survey` s ON st.Survey\_ID = s.Survey\_ID

JOIN

  `studentdata-project0.StudentData.Decoded` d ON s.DecID = d.DecID

JOIN

  `studentdata-project0.StudentData.TestScore` ts ON st.TestScore\_ID = ts.testscore\_id;

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*IMPORTANT\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

  -- 5.7 Susana:  Correlation of combined averages of test scores with BOTH completion of TEST PREP AND STANDARD MEAL

  -- UNION with Correlation of combined averages of test scores with BOTH 'Free/Reduced' and 'no prep.

WITH CorrelationResults AS (

  SELECT

    'Standard\_and\_Completed' AS Lunch\_and\_Preparation,

    CORR(ts.Math\_Score, IF(d.Lunch\_Plan = 'Standard' AND d.Test\_Preparation = 'Completed', 1, 0)) AS Correlation\_Math,

    CORR(ts.Reading\_Score, IF(d.Lunch\_Plan = 'Standard' AND d.Test\_Preparation = 'Completed', 1, 0)) AS Correlation\_Reading,

    CORR(ts.Writing\_Score, IF(d.Lunch\_Plan = 'Standard' AND d.Test\_Preparation = 'Completed', 1, 0)) AS Correlation\_Writing

  FROM

    `studentdata-project0.StudentData.Student` st

  JOIN

    `studentdata-project0.StudentData.Survey` s ON st.Survey\_ID = s.Survey\_ID

  JOIN

    `studentdata-project0.StudentData.Decoded` d ON s.DecID = d.DecID

  JOIN

    `studentdata-project0.StudentData.TestScore` ts ON st.TestScore\_ID = ts.testscore\_id

  UNION ALL

  SELECT

    'Free/Reduced\_And\_None' AS Lunch\_and\_Preparation,

    CORR(ts.Math\_Score, IF(d.Lunch\_Plan = 'Free/Reduced' AND d.Test\_Preparation = 'None', 1, 0)) AS Correlation\_Math,

    CORR(ts.Reading\_Score, IF(d.Lunch\_Plan = 'Free/Reduced' AND d.Test\_Preparation = 'None', 1, 0)) AS Correlation\_Reading,

    CORR(ts.Writing\_Score, IF(d.Lunch\_Plan = 'Free/Reduced' AND d.Test\_Preparation = 'None', 1, 0)) AS Correlation\_Writing

  FROM

    `studentdata-project0.StudentData.Student` st

  JOIN

    `studentdata-project0.StudentData.Survey` s ON st.Survey\_ID = s.Survey\_ID

  JOIN

    `studentdata-project0.StudentData.Decoded` d ON s.DecID = d.DecID

  JOIN

    `studentdata-project0.StudentData.TestScore` ts ON st.TestScore\_ID = ts.testscore\_id

)

SELECT

  Lunch\_and\_Preparation,

  Round(Avg(Correlation\_Math + Correlation\_Reading + Correlation\_Writing) / 3, 2) AS Correlation\_CombAvgSCORE

FROM CorrelationResults

GROUP BY Lunch\_and\_Preparation

ORDER BY Lunch\_and\_Preparation desc;

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*IMPORTANT\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

--6. DO 6.1 INSTEAD!!! ... Test scores grouped by parent level of education and both YES prep and YES lunch

  SELECT d.Parent\_Education,

      Round(AVG(CASE WHEN d.Lunch\_Plan = 'Standard' AND d.Test\_Preparation = 'Completed' THEN ts.Math\_Score END), 2) AS Avg\_Math\_Standard\_Completed,

      Round(AVG(CASE WHEN d.Lunch\_Plan = 'Standard' AND d.Test\_Preparation = 'Completed' THEN ts.Reading\_Score END),2) AS Avg\_Reading\_Standard\_Completed,

      Round(AVG(CASE WHEN d.Lunch\_Plan = 'Standard' AND d.Test\_Preparation = 'Completed' THEN ts.Writing\_Score END),2) AS Avg\_Writing\_Standard\_Completed,

      Round(AVG(CASE WHEN d.Lunch\_Plan = 'Free/Reduced' AND d.Test\_Preparation = 'None' THEN ts.Math\_Score END),2) AS Avg\_Math\_\_Free\_None,

      Round(AVG(CASE WHEN d.Lunch\_Plan = 'Free/Reduced' AND d.Test\_Preparation = 'None' THEN ts.Reading\_Score END),2) AS Avg\_Reading\_Free\_None,

      Round(AVG(CASE WHEN d.Lunch\_Plan = 'Free/Reduced' AND d.Test\_Preparation = 'None' THEN ts.Writing\_Score END),2) AS Avg\_Writing\_Free\_None

FROM `studentdata-project0.StudentData.Student` st

JOIN

  `studentdata-project0.StudentData.Survey`  s ON st.Survey\_ID = s.Survey\_ID

JOIN

  `studentdata-project0.StudentData.Decoded` d ON s.DecID = d.DecID

Join `studentdata-project0.StudentData.TestScore` as ts on st.TestScore\_ID = ts.testscore\_id

GROUP BY d.Parent\_Education, s.Parent\_Education

ORDER BY s.Parent\_Education desc;

--6.1. Maya:  Combined Average Test scores grouped by parent level of education and conditions where they had Standard Lunch and Completed Preparation vs NONE and FREE Lunch

  SELECT d.Parent\_Education,

      Round(

        AVG(

          CASE WHEN d.Lunch\_Plan = 'Standard' AND d.Test\_Preparation = 'Completed' THEN ts.Math\_Score END + CASE WHEN d.Lunch\_Plan = 'Standard' AND d.Test\_Preparation = 'Completed' THEN ts.Reading\_Score END + CASE WHEN d.Lunch\_Plan = 'Standard' AND d.Test\_Preparation = 'Completed' THEN ts.Writing\_Score END)/3, 2) AS Avg\_Score\_Standard\_PrepCompleted,

      Round(

        AVG(

          CASE WHEN d.Lunch\_Plan = 'Free/Reduced' AND d.Test\_Preparation = 'None' THEN ts.Math\_Score END + CASE WHEN d.Lunch\_Plan = 'Free/Reduced' AND d.Test\_Preparation = 'None' THEN ts.Reading\_Score END + CASE WHEN d.Lunch\_Plan = 'Free/Reduced' AND d.Test\_Preparation = 'None' THEN ts.Writing\_Score END )/ 3 ,2) AS Avg\_\_Score\_\_MealFree\_PrepNone,

FROM `studentdata-project0.StudentData.Student` st

JOIN

  `studentdata-project0.StudentData.Survey`  s ON st.Survey\_ID = s.Survey\_ID

JOIN

  `studentdata-project0.StudentData.Decoded` d ON s.DecID = d.DecID

Join `studentdata-project0.StudentData.TestScore` as ts on st.TestScore\_ID = ts.testscore\_id

GROUP BY d.Parent\_Education, s.Parent\_Education

ORDER BY s.Parent\_Education desc;

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- 8.Ivy:  Is there a correlation between parent educaton and meal plan?

SELECT

  CORR(s.Parent\_Education, s.Lunch\_Plan) AS Correlation

FROM

  `studentdata-project0.StudentData.Student` st

JOIN

  `studentdata-project0.StudentData.Survey` s ON st.Survey\_ID = s.Survey\_ID

JOIN

  `studentdata-project0.StudentData.Decoded` d ON s.DecID = d.DecID;

  --8.1.Hazel: Correlation between testPreparation and parent education

SELECT

  CORR(s.Parent\_Education, s.Test\_Preparation) AS Correlation\_TestPrep\_ParentEducation

FROM

  `studentdata-project0.StudentData.Student` st

JOIN

  `studentdata-project0.StudentData.Survey` s ON st.Survey\_ID = s.Survey\_ID;

  -- \*\*\*\*\*\*\*\*

  -- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*IMPORTANT\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- 8.5 Elena: What is the rate of meal purchase for parents of each education level AND joined with rate of completing the test preparation

SELECT

  d.Parent\_Education,

  -- COUNTIF(s.Lunch\_Plan = 1) AS Meal\_Standard,

  -- COUNTIF(s.Lunch\_Plan = 2) AS Meal\_None,

  -- COUNT(st.Student\_ID) AS Total\_Students,

  Round(AVG(ts.Writing\_Score+ts.Math\_score + ts.Reading\_score)/3, 2) AS CombinedAverageScore,

    COUNT(st.Student\_ID) AS StudentsInEachStrata,

  ROUND(COUNTIF(s.Test\_Preparation = 1) / COUNT(st.Student\_ID),3) AS Rate\_Prep\_Completed,

  ROUND(COUNTIF(s.Test\_Preparation = 0) / COUNT(st.Student\_ID),2) AS Rate\_Prep\_None,

  ROUND(COUNTIF(s.Lunch\_Plan = 1) / COUNT(st.Student\_ID),2) AS Rate\_Meal\_Standard,

  ROUND(COUNTIF(s.Lunch\_Plan = 2) / COUNT(st.Student\_ID),2) AS Rate\_Meal\_None

FROM

  `studentdata-project0.StudentData.Student` st

JOIN

  `studentdata-project0.StudentData.Survey` s ON st.Survey\_ID = s.Survey\_ID

JOIN

  `studentdata-project0.StudentData.Decoded` d ON s.DecID = d.DecID

  JOIN

  `studentdata-project0.StudentData.TestScore` ts ON st.TestScore\_ID = ts.TestScore\_ID

GROUP BY

  d.Parent\_Education, s.Parent\_Education

Order by

  s.Parent\_Education desc;

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

--11. Correlation coefficient for ParentEducation and average math scores

SELECT

  CORR(s.Parent\_Education, (ts.Math\_Score + ts.Reading\_Score + ts.Writing\_Score) / 3) AS Correlation\_Combined\_Average

FROM

  `studentdata-project0.StudentData.Student` st

JOIN

  `studentdata-project0.StudentData.Survey` s ON st.Survey\_ID = s.Survey\_ID

JOIN

  `studentdata-project0.StudentData.TestScore` ts ON st.TestScore\_ID = ts.TestScore\_ID;

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- 9.is there a correlation between a test prep (yes) and meal plan (standard)? - tp find out: out of those who test preped rate of standard vs rate of reduced.

SELECT

  CORR(s.Parent\_Education, s.Lunch\_Plan) AS Correlation\_TestPrep\_MealPlan

FROM

  `studentdata-project0.StudentData.Student` st

JOIN

  `studentdata-project0.StudentData.Survey` s ON st.Survey\_ID = s.Survey\_ID;

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

  /\* discarder because indices are not supported and views only allow select clauses

  -- will create some views and indices

  CREATE INDEX IDtestScore -- not supported in BigQuery

  ON `studentdata-project0.StudentData.TestScore` (TestScore\_ID); -- index created for practice; this column is used very often in join clauses not just select;

-- creating a view to encapsulate the logic of calculating average scores for each subject -- used extremely often; these are dependant variables!

CREATE VIEW AverageTestScores AS

SELECT Round(Avg(ts.math\_score),2) AS MathScore, Round(avg(ts.Reading\_Score),2) AS ReadingScore, Round(avg(ts.Writing\_Score),2) as WritingScore

FROM  `studentdata-project0.StudentData.TestScore` ts;

-- AFTER CREATING THE VIEW:

-- Quesiton 1: Rewritten :

/\*1. Average scores for each subject grouped by year gorup \*/

/\*Select d.year\_Group AS Year, ats.math\_score, ats.Reading\_Score, ats.Writing\_Score

From `studentdata-project0.StudentData.Student` st

Join `studentdata-project0.StudentData.Demographics` as d on st.demographics\_id = d.demographics\_id

Join `studentdata-project0.StudentData.TestScore.AverageTestScores` ats ON st.TestScore\_ID = ats.TestScore\_ID

Group by d.year\_group;\*/

**mysql statements to create a schema**

CREATE DATABASE StudentData; -- creating database

USE StudentData; -- making StudentData an active database

DROP DATABASE StudentData;

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Creating Tables\*\*\*\*\*\*\*\*\*\*

Drop table Student;

CREATE TABLE Student (

Student\_ID char(12) PRIMARY KEY CHECK (char\_length(Student\_ID) = 12), -- each consists of 12 characters

Survey\_ID INT CHECK (Survey\_ID >= 30000 AND Survey\_ID <=39999),

TestScore\_ID INT CHECK (TestScore\_ID >= 20000 AND TestScore\_ID <=29999),

Demographics\_ID INT CHECK (Demographics\_ID >= 10000 AND Demographics\_ID <=19999),

Firstname VARCHAR(255) DEFAULT 'First Name is Missing!',

LastName VARCHAR(255)DEFAULT 'Last Name is Missing!',

FOREIGN KEY (Survey\_ID) REFERENCES Survey(Survey\_ID) ON DELETE CASCADE,

FOREIGN KEY (TestScore\_ID) REFERENCES TestScore(TestScore\_ID) ON DELETE CASCADE,

FOREIGN KEY (Demographics\_ID) REFERENCES Demographics(Demographics\_ID) ON DELETE CASCADE)

;

Drop table Survey;

CREATE TABLE Survey (

Survey\_ID INT PRIMARY KEY CHECK (Survey\_ID >= 30000 AND Survey\_ID <=39999),

DecID INT CHECK (DecID >= 40000 AND DecID <=49999),

Parent\_Education int DEFAULT 0 CHECK (Parent\_Education >= 1 AND Parent\_Education <= 6),

Lunch\_Plan int DEFAULT 0 CHECK (Lunch\_Plan >= 1 AND Lunch\_Plan <= 6),

Test\_Preparation int DEFAULT 0 CHECK (Test\_Preparation >= 0 AND Test\_Preparation <= 100),

FOREIGN KEY (DecID) REFERENCES Decoded(DecID) ON DELETE CASCADE

);

Drop table TestScore;

CREATE TABLE TestScore (

TestScore\_ID INT PRIMARY KEY CHECK (TestScore\_ID >= 20000 AND TestScore\_ID <=29999),

Math\_Score INT CHECK (Math\_Score >= 0 AND Math\_Score <= 100),

Reading\_Score INT CHECK (Reading\_Score >= 0 AND Reading\_Score <= 100),

Writing\_Score INT CHECK (Writing\_Score >= 0 AND Writing\_Score <= 100)

);

Drop table Demographics;

CREATE TABLE Demographics (

Demographics\_ID INT PRIMARY KEY CHECK (Demographics\_ID >= 10000 AND Demographics\_ID <=19999),

Gender ENUM('M', 'F', 'Undecided') DEFAULT 'Undecided', -- enum data types list a list of possible values

Year\_Group CHAR(7) DEFAULT 'Retain' -- each group has 7 characters; default has to be retained for another year

);

Drop table Decoded;

CREATE TABLE IF NOT EXISTS Decoded (

DecID INT PRIMARY KEY CHECK (DecID >= 40000 AND DecID <=49999),

Parent\_Education VARCHAR(30) DEFAULT 'Aj Aj jaj!',

Lunch\_Plan VARCHAR(12) DEFAULT 'Olegs work',

Test\_Preparation VARCHAR(9) DEFAULT 'Donde?'

);

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*MY Queries\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- Average scores for each subject grouped by year group

Select d.year\_Group AS Year, Avg(ts.math\_score) AS MathScore, avg(ts.Reading\_Score) AS ReadingScore, avg(ts.Writing\_Score) as WritingScore

From Student st

Join demographics as d on st.demographics\_id = d.demographics\_id

Join testscore as ts on st.TestScore\_ID = ts.testscore\_id

Group by d.year\_group;

-- Combined average scores for each year group

Select d.year\_Group AS Year, Format(Avg((ts.math\_score +ts.Reading\_Score+ts.Writing\_Score)/3), 'xx') AS AverageScore

From Student st

Join demographics as d on st.demographics\_id = d.demographics\_id

Join testscore as ts on st.TestScore\_ID = ts.testscore\_id

Group by d.year\_group

Order by AverageScore desc;

-- average test scores goruped by parent educaiton level

SELECT

d.Parent\_Education,

AVG(ts.Math\_Score) AS AvgMathScore,

AVG(ts.Reading\_Score) AS AvgReadingScore,

AVG(ts.Writing\_Score) AS AvgWritingScore

FROM

Decoded d

JOIN

Survey s ON d.DecID = s.DecID

JOIN

Student st ON s.Survey\_ID = st.Survey\_ID

JOIN

TestScore ts ON st.TestScore\_ID = ts.TestScore\_ID

GROUP BY

d.Parent\_Education

ORDER BY

d.Parent\_Education;

--

-- will create some views and indices

CREATE INDEX IDtestScore -- not supported in BigQuery

ON TestScore (TestScore\_ID); -- index created for practice; this column is used very often in join clauses not just select;

-- View

CREATE VIEW AverageTestScores AS

SELECT Round(Avg(ts.math\_score),2) AS MathScore, Round(avg(ts.Reading\_Score),2) AS ReadingScore, Round(avg(ts.Writing\_Score),2) as WritingScore

FROM TestScore ts;

---

SELECT

d.Parent\_Education, s.Parent\_Education,

AVG(ts.Math\_Score) AS AvgMathScore,

AVG(ts.Reading\_Score) AS AvgReadingScore,

AVG(ts.Writing\_Score) AS AvgWritingScore

FROM

Student st

JOIN

Survey s ON st.Survey\_ID = s.Survey\_ID

JOIN

decoded d ON s.DecID = d.DecID

JOIN

TestScore ts ON st.TestScore\_ID = ts.TestScore\_ID

GROUP BY

d.Parent\_Education, s.Parent\_Education

ORDER BY

s.Parent\_Education desc;

--

-- 10. Test scores grouped by parent level of education and both YES prep and YES lunch

SELECT d.Parent\_Education,

AVG(CASE WHEN d.Lunch\_Plan = 'Standard' AND d.Test\_Preparation = 'Completed' THEN ts.Math\_Score END) AS Avg\_Math\_Standard\_Completed,

AVG(CASE WHEN d.Lunch\_Plan = 'Standard' AND d.Test\_Preparation = 'Completed' THEN ts.Reading\_Score END) AS Avg\_Reading\_Standard\_Completed,

AVG(CASE WHEN d.Lunch\_Plan = 'Standard' AND d.Test\_Preparation = 'Completed' THEN ts.Writing\_Score END) AS Avg\_Writing\_Standard\_Completed,

AVG(CASE WHEN d.Lunch\_Plan = 'Free/Reduced' AND d.Test\_Preparation = 'none' THEN ts.Math\_Score END) AS Avg\_Math\_\_Free\_None,

AVG(CASE WHEN d.Lunch\_Plan = 'Free/Reduced' AND d.Test\_Preparation = 'none' THEN ts.Reading\_Score END) AS Avg\_Reading\_Free\_None,

AVG(CASE WHEN d.Lunch\_Plan = 'Free/Reduced' AND d.Test\_Preparation = 'none' THEN ts.Writing\_Score END) AS Avg\_Writing\_Free\_None

FROM Student st

JOIN Survey s ON st.Survey\_ID = s.Survey\_ID

JOIN TestScore ts ON st.TestScore\_ID = ts.TestScore\_ID

JOIN Decoded d ON s.DecID = d.DecID

GROUP BY d.Parent\_Education;

-- Integrity check

Select \*

from student

join survey on student.survey\_ID = survey.survey\_ID

join testscore on student.testscore\_id = testscore.testscore\_id

join decoded on survey.decid = decoded.decid

order by testscore.math\_score desc;

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- not everything was imported, stated data too long in Parent\_Education, so i will alter that column here:

Alter table decoded

drop column Parent\_Education;

Alter table decoded

add column Parent\_Education VARCHAR(30) DEFAULT 'Aj Aj jaj!';

Alter table Decoded

MODIFY COLUMN Parent\_Education VARCHAR(30) DEFAULT 'AJ AJ AJ';

Truncate table Decoded;

-- Altering Survey table: 1. create a new foreign key

ALTER TABLE Survey

DROP COLUMN DecID;

Alter table Survey

Add column DecID INT DEFAULT 0 CHECK (DecID >= 40000 AND DecID <=49999);

ALTER TABLE Survey

ADD FOREIGN KEY (DecID) REFERENCES Decoded(DecID);

ALTER TABLE Survey

MODIFY COLUMN DECID INT DEFAULT NULL CHECK (DecID >= 40000 AND DecID <=49999);

SHOW CREATE TABLE Survey;

ALTER TABLE Survey

DROP FOREIGN KEY DecID;

Select count(Parent\_Education)from Decoded;

select \* from Survey;

Select \* from Demographics;

Select \* from TestScore;

Select \* from Student;

select count(TestScore\_ID)from student;

--

SELECT \*

FROM Student

JOIN Demographics ON Student.Student\_ID = Demographics.Student\_ID

JOIN TestScore ON Student.Student\_ID = TestScore.Student\_ID

JOIN Survey ON Student.Student\_ID = Survey.Student\_ID;

--

select count(student\_id)

from Student;

/\*

alter table student

drop column iddata\_id; \*/

-- changing datatype of student\_id from int to char(12)

-- ALTER TABLE Student

-- MODIFY COLUMN Student\_ID char(12);

select st.Student\_ID, st.FirstName, st.LastName,ts.Math\_score, ts.Reading\_score, ts.Writing\_score, avg((ts.Math\_score + ts.Reading\_score + ts.Writing\_score)/3) as CombinedAverage

from student as st

join testscore as ts

on st.TestScore\_ID = ts.TestScore\_ID

Group by Student\_ID -- in select using aggregate function in conjunction with non-aggregate, so group by is necessary to

-- group the results for non-aggregated columns;

order by CombinedAverage desc

limit 10;

/\*

-- modifying survey table

ALTER TABLE `studentdata`.`survey`

DROP FOREIGN KEY `survey\_ibfk\_1`;

ALTER TABLE `studentdata`.`survey`

CHANGE COLUMN `DecID` `DecID` INT NOT NULL COMMENT 'DecID is a foreign key that links this table to the Decoded table ' ,

ADD UNIQUE INDEX `Survey\_ID\_UNIQUE` (`Survey\_ID` ASC) VISIBLE,

ADD UNIQUE INDEX `DecID\_UNIQUE` (`DecID` ASC) VISIBLE;

;

-- surveyID

ALTER TABLE `studentdata`.`survey`

ADD CONSTRAINT `survey\_ibfk\_1`

FOREIGN KEY (`DecID`)

REFERENCES `studentdata`.`decoded` (`DecID`)

ON DELETE CASCADE;

ALTER TABLE `studentdata`.`survey`

CHANGE COLUMN `Survey\_ID` `Survey\_ID` INT NOT NULL COMMENT 'Primary key, links this table with Student table ' ;\*/

/\* Modifying testScore table

ALTER TABLE `studentdata`.`testscore`

ADD UNIQUE INDEX `TestScore\_ID\_UNIQUE` (`TestScore\_ID` ASC) VISIBLE;

;

ALTER TABLE `studentdata`.`testscore`

CHANGE COLUMN `Math\_Score` `Math\_Score` INT NULL DEFAULT NULL COMMENT 'Student\'s score on Math test 0-100%' ;

/\*

/\*

-- Modifying Student table; editing key constraints; adding discriptions to all columns

ALTER TABLE `studentdata`.`student`

DROP FOREIGN KEY `student\_ibfk\_1`,

DROP FOREIGN KEY `student\_ibfk\_2`,

DROP FOREIGN KEY `student\_ibfk\_3`;

ALTER TABLE `studentdata`.`student`

CHANGE COLUMN `Student\_ID` `Student\_ID` CHAR(12) NOT NULL COMMENT 'Unique ID of each student' ,

CHANGE COLUMN `Survey\_ID` `Survey\_ID` INT NOT NULL COMMENT 'Foreign key refers to the Survey table' ,

CHANGE COLUMN `TestScore\_ID` `TestScore\_ID` INT NOT NULL COMMENT 'Foreign key refers to the TestScore table' ,

CHANGE COLUMN `Demographics\_ID` `Demographics\_ID` INT NOT NULL COMMENT 'Foreign key refers to the Demographics table' ,

CHANGE COLUMN `Firstname` `Firstname` VARCHAR(255) NULL DEFAULT 'First Name is Missing!' COMMENT 'student\'s first name ' ,

CHANGE COLUMN `LastName` `LastName` VARCHAR(255) NULL DEFAULT 'Last Name is Missing!' COMMENT 'student\'s last name ' ,

ADD UNIQUE INDEX `Student\_ID\_UNIQUE` (`Student\_ID` ASC) VISIBLE,

ADD UNIQUE INDEX `Survey\_ID\_UNIQUE` (`Survey\_ID` ASC) VISIBLE,

ADD UNIQUE INDEX `Demographics\_ID\_UNIQUE` (`Demographics\_ID` ASC) VISIBLE,

ADD UNIQUE INDEX `TestScore\_ID\_UNIQUE` (`TestScore\_ID` ASC) VISIBLE;

;

ALTER TABLE `studentdata`.`student`

ADD CONSTRAINT `student\_ibfk\_1`

FOREIGN KEY (`Survey\_ID`)

REFERENCES `studentdata`.`survey` (`Survey\_ID`)

ON DELETE CASCADE,

ADD CONSTRAINT `student\_ibfk\_2`

FOREIGN KEY (`TestScore\_ID`)

REFERENCES `studentdata`.`testscore` (`TestScore\_ID`)

ON DELETE CASCADE,

ADD CONSTRAINT `student\_ibfk\_3`

FOREIGN KEY (`Demographics\_ID`)

REFERENCES `studentdata`.`demographics` (`Demographics\_ID`)

ON DELETE CASCADE;

\*/