Cross-Tabulation and extraction of Valuable Insights Using SQL

DECEMBER 15

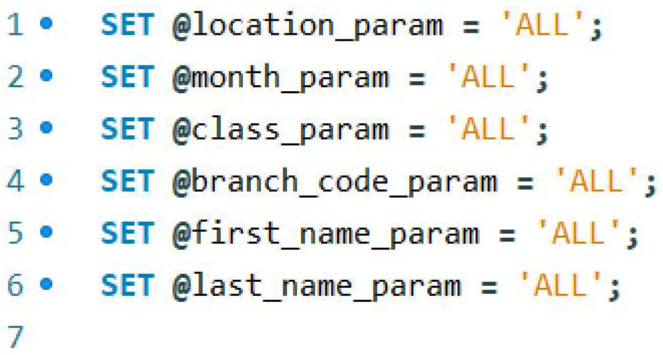
Amol Pagare
Student at Indian Institute of Technology Bombay

SQL Query Solution Explanation

Problem Statement: In an academic year, two vital datasets—demographics and assessments—are integrated for cross-tabulations. The goal is to extract insights on the correlation between learning outcomes and engagement metrics, particularly student attendance and retention, at school and network levels.

All the queries and the Databases are uploaded to GitHub
https://github.com/Amolpagare10/SQL_Database_Integration.git

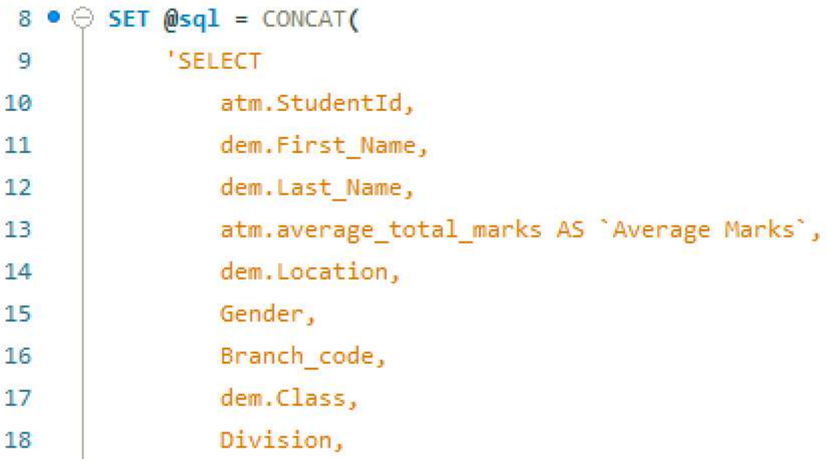
Solution to the above problem statement is a SQL Query which involves integration of databases using the appropriate joins and the use of functions to extract necessary insights.



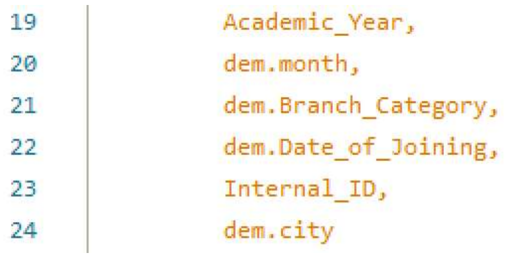
```sql
SET @location_param = 'ALL';
SET @month_param = 'ALL';
SET @class_param = 'ALL';
SET @branch_code_param = 'ALL';
SET @first_name_param = 'ALL';
SET @last_name_param = 'ALL';

```

These lines set up parameters with default values. These parameters will be used to filter the data in the subsequent query.

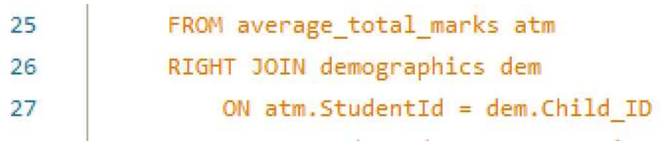


```sql
SET @sql = CONCAT(
    'SELECT
        atm.StudentId,
        dem.First_Name,
        dem.Last_Name,
        atm.average_total_marks AS `Average Marks`,
        dem.Location,
        Gender,
        Branch_code,
        dem.Class,
        Division,
```



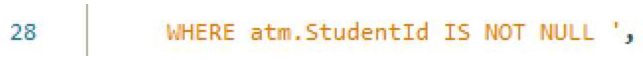
```
        Academic_Year,
        dem.month,
        dem.Branch_Category,
        dem.Date_of_Joining,
        Internal_ID,
        dem.city
```

SELECT Clause: Selects specific columns from the tables `average_total_marks` and `demographics`.



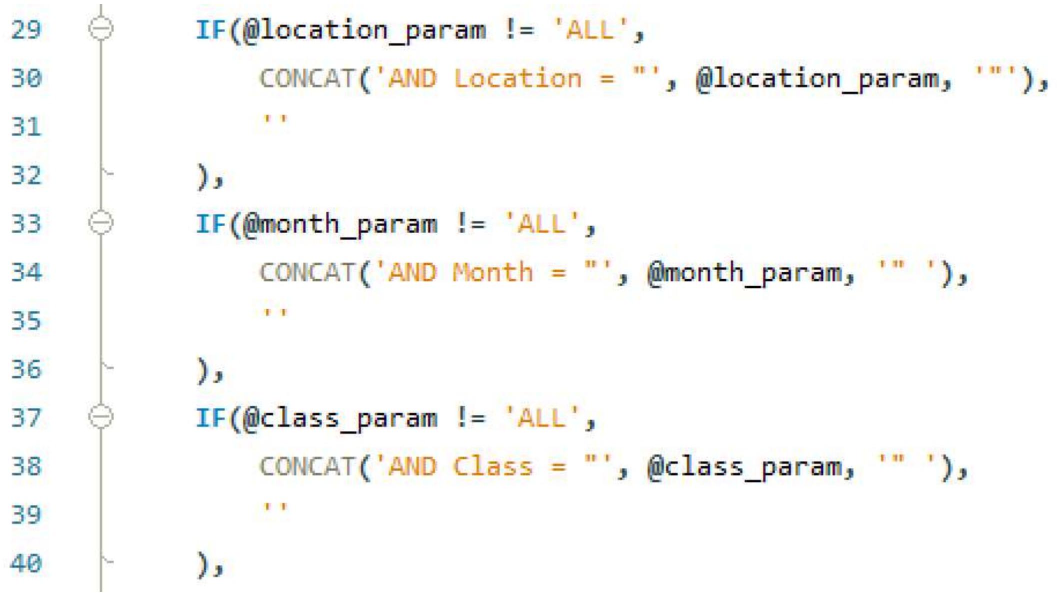
```
    FROM average_total_marks atm
    RIGHT JOIN demographics dem
        ON atm.StudentId = dem.Child_ID
```

FROM Clause: Specifies the tables being queried and establishes a RIGHT JOIN relationship between `average_total_marks` and `demographics` based on the `StudentId` and `Child_ID` columns, respectively.

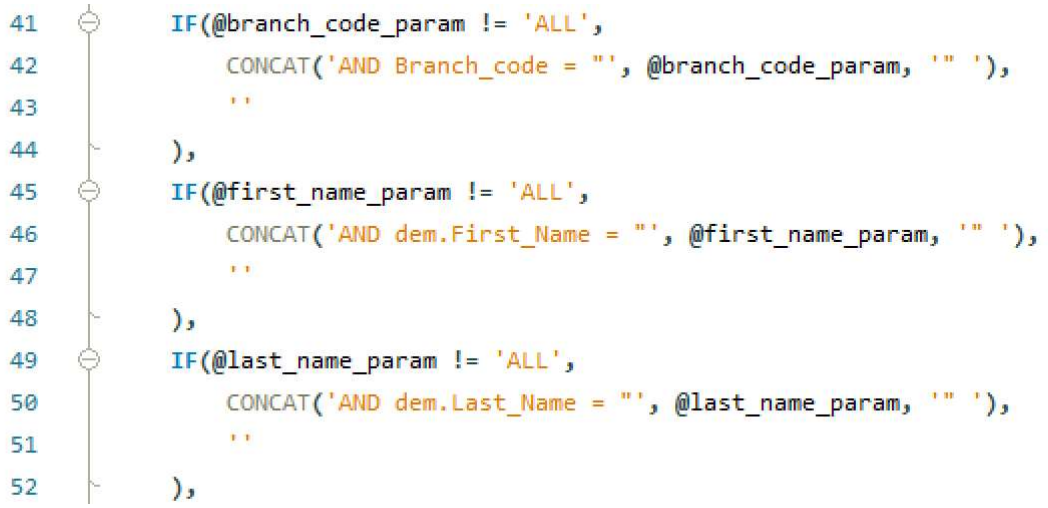


```
    WHERE atm.StudentId IS NOT NULL ',
```

WHERE Clause: Filters the results based on the provided parameters. If a parameter is set to 'ALL', the corresponding condition is omitted.



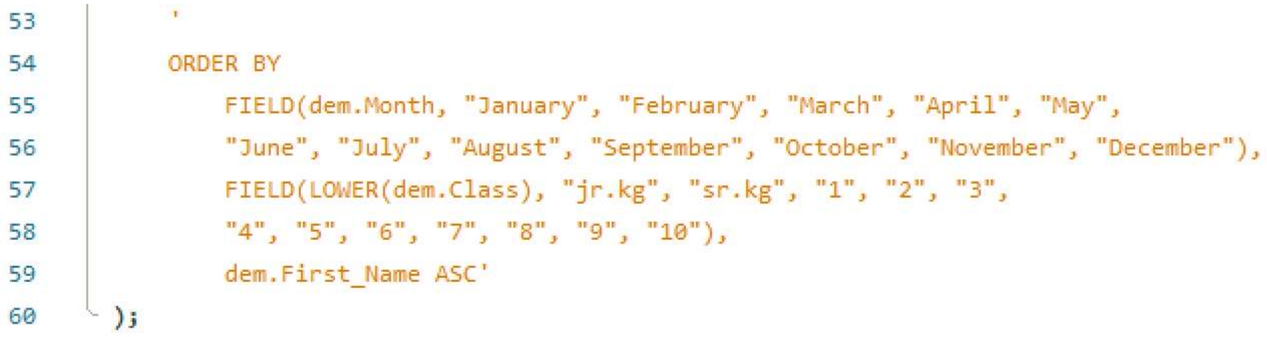
```
    IF(@location_param != 'ALL',
        CONCAT('AND Location = "', @location_param, '"'),
        ''
    ),
    IF(@month_param != 'ALL',
        CONCAT('AND Month = "', @month_param, '" '),
        ''
    ),
    IF(@class_param != 'ALL',
        CONCAT('AND Class = "', @class_param, '" '),
        ''
    ),
```



```
    IF(@branch_code_param != 'ALL',
        CONCAT('AND Branch_code = "', @branch_code_param, '" '),
        ''
    ),
    IF(@first_name_param != 'ALL',
        CONCAT('AND dem.First_Name = "', @first_name_param, '" '),
        ''
    ),
    IF(@last_name_param != 'ALL',
        CONCAT('AND dem.Last_Name = "', @last_name_param, '" '),
        ''
    ),
```

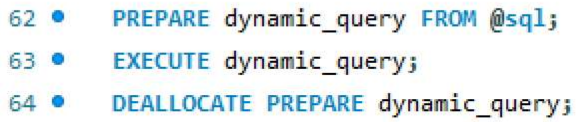
This line is using the IF statement to dynamically construct a condition for the Month parameter. The logic is as follows:

If @month_param is not equal to 'ALL', it concatenates the condition 'AND Month = "<value>" ' to the SQL query.
If ‘@month_param’ is 'ALL', it appends an empty string ' '.
This pattern is similarly followed by the location, class, first_name, last_name and Branch_code filters.



```
        '
        ORDER BY
            FIELD(dem.Month, "January", "February", "March", "April", "May",
            "June", "July", "August", "September", "October", "November", "December"),
            FIELD(LOWER(dem.Class), "jr.kg", "sr.kg", "1", "2", "3",
            "4", "5", "6", "7", "8", "9", "10"),
            dem.First_Name ASC'
);
```

ORDER BY Clause: Orders the result set based on the specified criteria. The `FIELD` function is used to create a custom order for months and classes. Results are ordered first by month, then by class in a case-insensitive manner, and finally by the first name in ascending order.

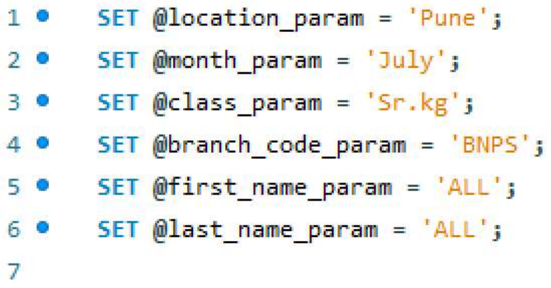


```
PREPARE dynamic_query FROM @sql;
EXECUTE dynamic_query;
DEALLOCATE PREPARE dynamic_query;
```

These lines prepare, execute, and deallocate the dynamically constructed query using the parameters set earlier. The `PREPARE` statement creates a prepared statement from the dynamic SQL string, the `EXECUTE` statement runs the prepared statement, and `DEALLOCATE` frees up the resources associated with the prepared statement.

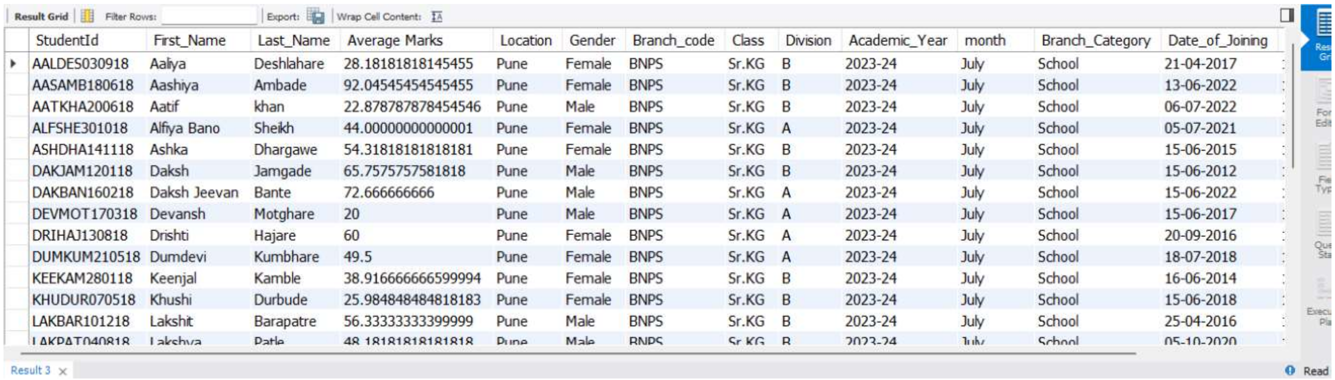
Demonstrating the use of Query for effective retrieval of data for any set of filters.

For example, if we were to get the detail for the corresponding entities, we fill up the parameters with the required values.

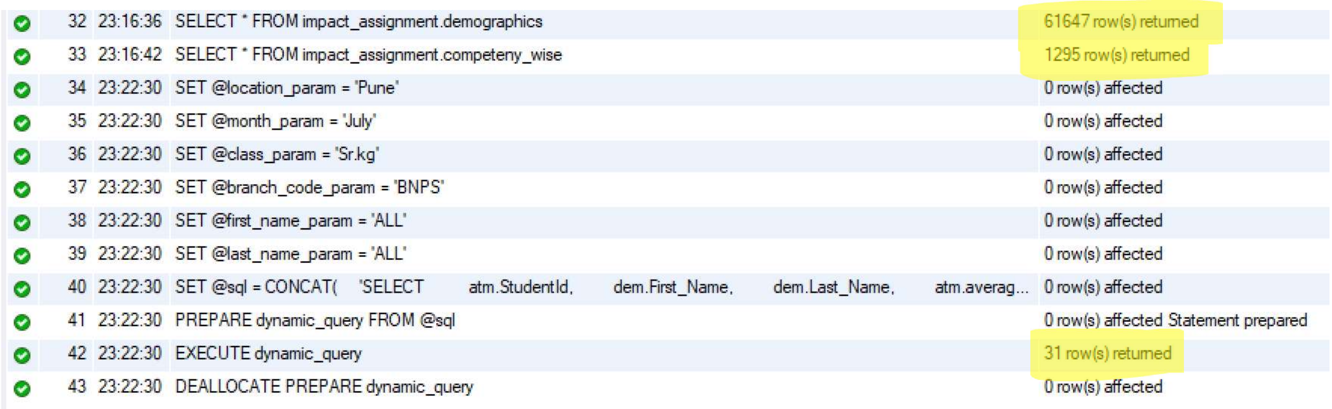


```
SET @location_param = 'Pune';
SET @month_param = 'July';
SET @class_param = 'Sr.kg';
SET @branch_code_param = 'BNPS';
SET @first_name_param = 'ALL';
SET @last_name_param = 'ALL';
```

We get the required details of the students with their average marks and other necessary details also them being arranged according to the class hierarchy and the sequence of months in a year.



StudentId	First_Name	Last_Name	Average Marks	Location	Gender	Branch_code	Class	Division	Academic_Year	month	Branch_Category	Date_of_Joining
AALDES030918	Aaliya	Deshlahare	28.18181818145455	Pune	Female	BNPS	Sr.KG	B	2023-24	July	School	21-04-2017
AASAMB180618	Aashiya	Ambade	92.04545454545455	Pune	Female	BNPS	Sr.KG	B	2023-24	July	School	13-06-2022
AATKHA200618	Aatif	khan	22.878787878454546	Pune	Male	BNPS	Sr.KG	B	2023-24	July	School	06-07-2022
ALFSHE301018	Alfiya Bano	Sheikh	44.00000000000001	Pune	Female	BNPS	Sr.KG	A	2023-24	July	School	05-07-2021
ASHDHA141118	Ashka	Dhargawe	54.31818181818181	Pune	Female	BNPS	Sr.KG	B	2023-24	July	School	15-06-2015
DAKJAM120118	Daksh	Jamgade	65.7575757581818	Pune	Male	BNPS	Sr.KG	B	2023-24	July	School	15-06-2012
DAKBAN160218	Daksh Jeevan	Bante	72.666666666	Pune	Male	BNPS	Sr.KG	A	2023-24	July	School	15-06-2022
DEVMOT170318	Devansh	Motghare	20	Pune	Male	BNPS	Sr.KG	A	2023-24	July	School	15-06-2017
DRIHAJ130818	Drishti	Hajare	60	Pune	Female	BNPS	Sr.KG	A	2023-24	July	School	20-09-2016
DUMKUM210518	Dumdevi	Kumbhare	49.5	Pune	Female	BNPS	Sr.KG	A	2023-24	July	School	18-07-2018
KEEKAM280118	Keenjal	Kamble	38.916666666599994	Pune	Female	BNPS	Sr.KG	B	2023-24	July	School	16-06-2014
KHUDUR070518	Khushi	Durbude	25.984848484818183	Pune	Female	BNPS	Sr.KG	B	2023-24	July	School	15-06-2018
LAKBAR101218	Lakshit	Barapatre	56.33333333399999	Pune	Male	BNPS	Sr.KG	B	2023-24	July	School	25-04-2016
LAKPAT040818	Lakshya	Patle	48.18181818181818	Pune	Male	BNPS	Sr.KG	B	2023-24	July	School	05-10-2020

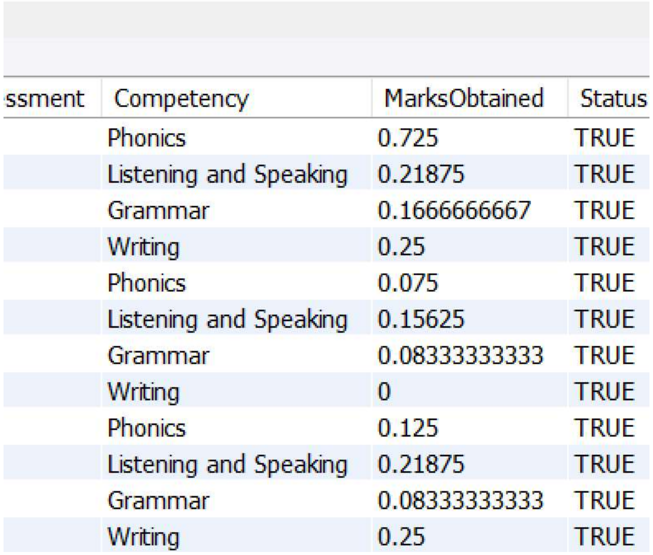


```
32  23:16:36  SELECT * FROM impact_assignment.demographics                                          61647 row(s) returned
33  23:16:42  SELECT * FROM impact_assignment.competeny_wise                                        1295 row(s) returned
34  23:22:30  SET @location_param = 'Pune'                                                          0 row(s) affected
35  23:22:30  SET @month_param = 'July'                                                             0 row(s) affected
36  23:22:30  SET @class_param = 'Sr.kg'                                                            0 row(s) affected
37  23:22:30  SET @branch_code_param = 'BNPS'                                                       0 row(s) affected
38  23:22:30  SET @first_name_param = 'ALL'                                                         0 row(s) affected
39  23:22:30  SET @last_name_param = 'ALL'                                                          0 row(s) affected
40  23:22:30  SET @sql = CONCAT(   'SELECT     atm.StudentId,     dem.First_Name,     dem.Last_Name,     atm.averag...   0 row(s) affected
41  23:22:30  PREPARE dynamic_query FROM @sql                                                       0 row(s) affected Statement prepared
42  23:22:30  EXECUTE dynamic_query                                                                 31 row(s) returned
43  23:22:30  DEALLOCATE PREPARE dynamic_query                                                      0 row(s) affected
```

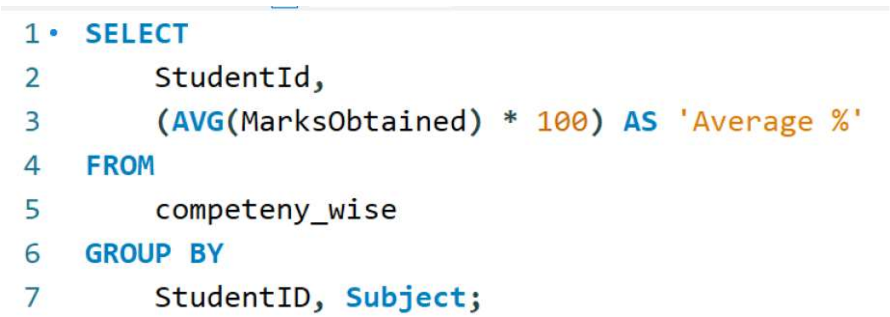
The output pane displays the exact number of data entries which satisfy the filters out of the total 61647 Unique entries.
In this example 31 entries satisfy the given specifications.

I have also created a separate query to calculate the per-cent average total marks of each student for all the subject English, Marathi, Math and Hindi.

Note: Assuming the marks given were scaled down to 1 for the total marks.

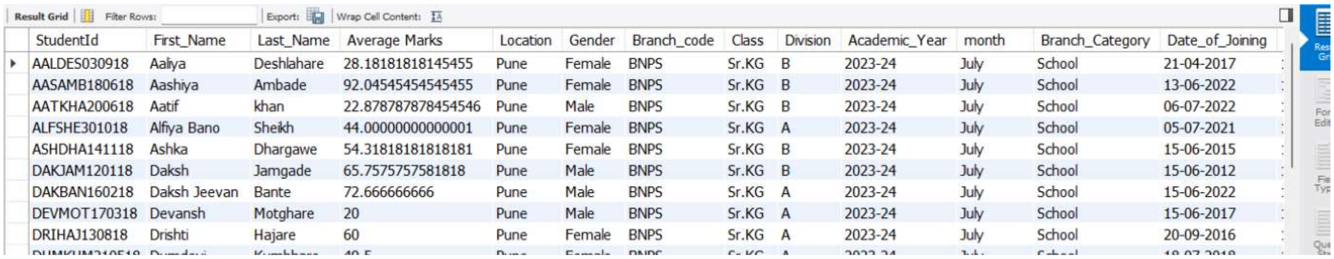


ssment	Competency	MarksObtained	Status
	Phonics	0.725	TRUE
	Listening and Speaking	0.21875	TRUE
	Grammar	0.1666666667	TRUE
	Writing	0.25	TRUE
	Phonics	0.075	TRUE
	Listening and Speaking	0.15625	TRUE
	Grammar	0.08333333333	TRUE
	Writing	0	TRUE
	Phonics	0.125	TRUE
	Listening and Speaking	0.21875	TRUE
	Grammar	0.08333333333	TRUE
	Writing	0.25	TRUE



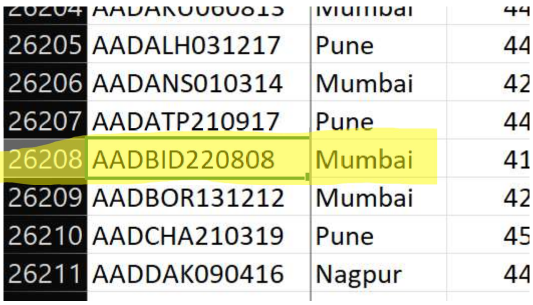
```
SELECT
      StudentId,
      (AVG(MarksObtained) * 100) AS 'Average %'
FROM
      competeny_wise
GROUP BY
      StudentID, Subject;
```

Joined this result with the “Demographics” database to get a holistic view of the student’s performance relative to the demographic perspective.

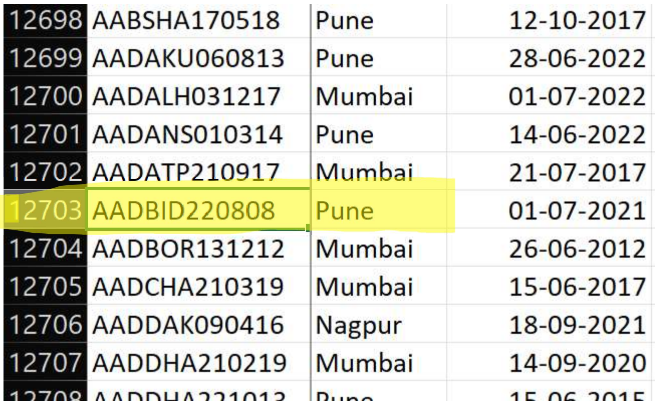


StudentId	First_Name	Last_Name	Average Marks	Location	Gender	Branch_code	Class	Division	Academic_Year	month	Branch_Category	Date_of_Joining
AALDES030918	Aaliya	Deshlahare	28.18181818145455	Pune	Female	BNPS	Sr.KG	B	2023-24	July	School	21-04-2017
AASAMB180618	Aashiya	Ambade	92.04545454545455	Pune	Female	BNPS	Sr.KG	B	2023-24	July	School	13-06-2022
AATKHA200618	Aatif	khan	22.878787878454546	Pune	Male	BNPS	Sr.KG	B	2023-24	July	School	06-07-2022
ALFSHE301018	Alfiya Bano	Sheikh	44.00000000000001	Pune	Female	BNPS	Sr.KG	A	2023-24	July	School	05-07-2021
ASHDHA141118	Ashka	Dhargawe	54.31818181818181	Pune	Female	BNPS	Sr.KG	B	2023-24	July	School	15-06-2015
DAKJAM120118	Daksh	Jamgade	65.7575757581818	Pune	Male	BNPS	Sr.KG	B	2023-24	July	School	15-06-2012
DAKBAN160218	Daksh Jeevan	Bante	72.666666666	Pune	Male	BNPS	Sr.KG	A	2023-24	July	School	15-06-2022
DEVMOT170318	Devansh	Motghare	20	Pune	Male	BNPS	Sr.KG	A	2023-24	July	School	15-06-2017
DRIHAJ130818	Drishti	Hajare	60	Pune	Female	BNPS	Sr.KG	A	2023-24	July	School	20-09-2016

Note: Not all entries of students in the Demographic database were listed in the assessment databases. Also some students had their locations changed in different months, in the Demographics database.



26204	AADAKU060813	Mumbai	44
26205	AADALH031217	Pune	44
26206	AADANS010314	Mumbai	42
26207	AADATP210917	Pune	44
26208	AADBID220808	Mumbai	41
26209	AADBOR131212	Mumbai	42
26210	AADCHA210319	Pune	45
26211	AADDAK090416	Nagpur	44



12698	AABSHA170518	Pune	12-10-2017
12699	AADAKU060813	Pune	28-06-2022
12700	AADALH031217	Mumbai	01-07-2022
12701	AADANS010314	Pune	14-06-2022
12702	AADATP210917	Mumbai	21-07-2017
12703	AADBID220808	Pune	01-07-2021
12704	AADBOR131212	Mumbai	26-06-2012
12705	AADCHA210319	Mumbai	15-06-2017
12706	AADDAK090416	Nagpur	18-09-2021
12707	AADDHA210219	Mumbai	14-09-2020
12708	AADDHA221013	Pune	15-06-2015