

# Project Name: MIS Admissions Fall 2018

Version 1.0

**Business Requirements Document (BRD)**

**Version and Approvals**

|  |  |
| --- | --- |
| **Version History** | |
| **Version #** | **Date** | | **Revised By** | **Reason for change** |
| **1.0** | **12/23/17** | |  |  |
|  |  | |  |  |
|  |  | |  |  |
|  |  | |  |  |

|  |  |
| --- | --- |
| **Document Approvals** | |
| **Approver Name** | **Project Role** | | **Signature/Electronic Approval** | **Date** |
| **Pavankumar Mulgund** |  | |  |  |
|  |  | |  |  |
|  |  | |  |  |
|  |  | |  |  |

**Table of Contents**

[Project Name: MIS Admissions Fall 2018](#_Toc501880113)

Business Requirement document [Version and Approvals](#_Toc501880114)

[1. Project Details](#_Toc501880115)

[2. Document Resources](#_Toc501880116)

[3. Glossary of Terms](#_Toc501880117)

[4. Project Overview](#_Toc501880118)

[4.1 Project Overview and Background](#_Toc501880119)

[4.2 Project Dependencies](#_Toc501880120)

[4.3 Stakeholders](#_Toc501880121)

[7. ENTITY RELATIONSHIP DIAGRAM](#_Toc501880124)

[8. ENTITY RELATIONSHIP DIAGRAM EXPLAINATION](#_Toc501880124)

[9. LOGICAL MODEL](#_Toc501880124)

[10. DATABASE STRUCTURES EXPLAINATION](#_Toc501880124)

[11. QUERIES](#_Toc501880124)

Project Details

|  |  |
| --- | --- |
| **Project Name** | MIS Admissions Fall 2018 |
| **Project Type** | New Initiative |
| **Project Start Date** | November |
| **Project End Date** |  |
| **Project Sponsor** |  |
| **Primary Driver** | DBMS related project |
| **Secondary Driver** |  |
| **Division** |  |

1. Document Resources

| **Name** | **Business Unit** | **Role** |
| --- | --- | --- |
| Rushabh Mehtalia |  |  |
| Doyita Mitra |  |  |
| Garima Mathur |  |  |

1. Glossary of Terms

| **Term/Acronym** | **Definition** |
| --- | --- |
| UGPA | The score in terms of grade point average scored by the student in his/her undergraduate course. |

1. Project Overview

## Project Overview and Background

Database management system for MIS Admissions Fall 2018 stores the information about all the students who have applied to the MS-MIS program of UB. With the help of this information, UB assigns points to each student based upon the student’s GRE or GMAT score, TOEFL or IELTS score, UGPA and work experience. The decision to whether admit a student into the MIS program or not to admit the student depends on whether the point based score exceeds a certain threshold or not. Apart from that, the application will also store background and personal data of the student like First Name, Last Name, Name of the UG institution, the type of UG major pursued by the student, Name of the previous employer if any and such.

The application system will load the information from various tables that have been created in third normal form into the SQL for querying purposes. Querying will be done for purposes of analysis like for example finding the average verbal score or quant score of the entire applicant pool or just for a selected pool of applicants. The select admissions committee will also be able to classify students and admit them based on not just academic achievements but also work experience. (Note: Ideally the admissions committee will look at the overall profile of the student but the scope of this project limits the student profiling to only academic and work related achievements. Also the scope of the project limits the personal information of the student to only certain fields and not all that are typically asked. The terms applicant and student are used interchangeably.)

## Project Dependencies

N/A

## 4.3 Stakeholders

The following comprises the internal and external stakeholders whose requirements are represented by this document:

| **Num** | **Stakeholders** |
| --- | --- |
| 1. |  |
| 2. |  |
| 3. |  |

1. ENITITY RELATIONSHIP DIAGRAM

**GMAT IELTS Test Details**

Student Id(FK)

Exam Id (FK)

Verbal Raw

Verbal %

Quant Raw

Quant%

AWA Raw

AWA%

IR Raw

IR %

Listening

Speaking

IELTS\_Total

**GRE IELETS Test Details**

Student Id(FK)

Exam Id (FK)

Verbal Raw

Verbal %

Quant Raw

Quant%

AWA Raw

AWA%

Listening

Speaking

IELTS\_Total

**Undergraduate Majors**

Student Id(FK)

UG Major(PK)

**Student**

Student Id (PK)

Last Name

First Name

UG Institution(FK)

UG Major(FK)

UGPA

Current Country(FK)

Company(FK)

Work Experience (Months)

**Test Details**

Student Id (FK)

Exam Type (ID) (PK)

Exam Description

**GRE TOEFL Test Exam Details**

Student Id(FK)

Exam Id (PK)

Verbal Raw

Verbal %

Quant Raw

Quant%

AWA Raw

AWA%

Listening

Speaking

TOEFL\_Total

**GMAT TOEFL Test Exam Details**

Student Id(FK)

Exam Id (FK)

Verbal Raw

Verbal %

Quant Raw

Quant%

AWA Raw

AWA%

IR Raw

IR %

Listening

Speaking

TOEFL\_Total

**Institution**

Student Id(FK)

UG Institution(PK)

**Company**

Student Id(FK)

Company(PK)

**Country**

Student Id(PK)

Current Country(PK)

**Decision**

Student Id (FK)

Decision

1. Enity Relationship Diagram EXPLAINATION

**Student Admission:**

**Entities:**

**Student –** will contain details of the students applied for the course

**Institution –** will contain details of the institution in which students studied

**Undergraduate Majors -** will contain the list of Majors specialization in their under graduation

**Company –** will contain the details of students’ company in which they have worked before joining the college

**Country –** will contain the students’ country

**Decision –** will contain the decision whether the student got admitted in the college for that particular course

**Test Details –** will contain the details about the exam type and its description.

**GRE TOEFL Test Details–** will contain the details about the TOEFL and GREExam like Quant raw etc.

**GMAT TOEFL Test Details–** will contain the details about the GMAT Exam like Quant raw and TOEFL .

**GRE IELTS Test Details–** will contain the details about the GRE and IELTS Exam like Listening etc.

**GMAT IELTS Test Details–** will contain the details about the GMAT and IELTS Exam like Reading etc.

**Description:**

A **Student** must belong to One **Institution** only but each **Institution** can belong to one or many **students**

A **Student** must belong to One **Undergraduate Major** only but each **Undergraduate Major** can belong to one or many **students**

A **Student** must belong to One **Country** only but each **Country** can belong to one or many **students**

A **Student** maybelong to One or many **Companies** only and each **Company** can belong to one or many **students**

A **Student** must have One **Test Details** only but each **Test Details** can belong to one or many **students**

A **Student** may belong to One **GRE Test Details** only but each **GRE Test Details** can belong to one or many **students**

A **Student** may belong to One **GMAT TOEFL Test Details** only but each **GMAT TOEFL Test Details** can belong to one or many **students**

A **Student** may belong to One **GRE TOEFL Test Details** only but each **GRE TOEFL Test Details** can belong to one or many **students**

A **Student** may belong to One **GMAT** **IELTS Test Details** only but each **GMAT** **IELTS Test Details** can belong to one or many **student**

1. LOGICAL DIAGRAM

Student\_DB

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Student Id(PK) | Last Name | First Name | UG Institution(FK) | UG Major(FK) | UGPA | Current Country(FK) | Company(FK) | Work Experience (Months) |
|  |  |  |  |  |  |  |  |  |

Test Details

|  |  |  |
| --- | --- | --- |
| Student Id(FK) | Exam ID(PK) | Exam Description |
|  |  |  |

Primary Key:Student\_Id,Exam\_Id

GRE TOEFL Exam Details

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Student Id(FK) | Exam ID(FK) | Verbal Raw | Verbal % | Quant Raw | Quant % | AWA | AWA % | Listening | Speaking | TOEFL\_Total |
|  |  |  |  |  |  |  |  |  |  |  |

Primary Key:Student\_Id,Exam\_Id

GRE IELTS Exam Details

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Student Id(FK) | Exam ID(FK) | Verbal Raw | Verbal % | Quant Raw | Quant % | AWA | AWA % | Listening | Speaking | IELTS\_Total |
|  |  |  |  |  |  |  |  |  |  |  |

Primary Key:Student\_Id,Exam\_Id

GMAT IELTS Exam Details

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Student Id(FK) | Exam ID(FK) | Verbal Raw | Verbal % | Quant Raw | Quant % | AWA | AWA % | IR Raw | IR % | Listening | Speaking | IELTS\_Total |
|  |  |  |  |  |  |  |  |  |  |  |  |  |

Primary Key:Student\_Id,Exam\_Id

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Student Id(FK) | Exam ID(FK) | Verbal Raw | Verbal % | Quant Raw | Quant % | AWA | AWA % | IR Raw | IR % | Listening | Speaking | TOEFL\_Total |
|  |  |  |  |  |  |  |  |  |  |  |  |  |

GMAT TOEFL Exam Details

Primary Key:Student\_Id,Exam\_Id

Decision Details

|  |  |
| --- | --- |
| Student ID | Decision |
|  |  |

Primary Key:Student\_Id,Decision

Company Details

|  |  |
| --- | --- |
| Student Id(FK) | Company(PK) |
|  |  |

Primary Key:Student\_Id,Company

Institution Details

|  |  |
| --- | --- |
| Student Id(FK) | UG Institution(PK) |
|  |  |

Primary Key:Student\_Id,Institution

Undergraduate Majors

|  |  |
| --- | --- |
| Student Id(FK) | UG Major(PK) |
|  |  |

Primary Key:Student\_Id,UG\_Major

Country Details

|  |  |
| --- | --- |
| Student Id(FK) | Current Country(PK) |
|  |  |

Primary Key:Student\_Id,Country

1. Database Structures

**PFB the .sql file for the below queries**

****

**PFB the Excel file for all the data which was used for importing the tables in Oracle**

****

**Database Schema for Student Admission DB**

--Tables in this DB are Student,Company,Institution,UnderGraduate Major,Country,Decision,Test Details,GRE TOEFL Test Details,GMAT TOEFL Test Details,GRE IELETS Test Details,GMAT IELTS Test Details

--**DDL statement for Student table**

CREATE TABLE Student\_DB

( Student\_Id NUMBER(8) NOT NULL,

First\_Name VARCHAR2(100) ,

Last\_Name VARCHAR2(100) ,

UG\_Institution VARCHAR2(200),

UG\_Major VARCHAR2(200),

UGPA NUMBER(8,2) ,

Current\_Country VARCHAR(20),

Company VARCHAR(200),

Work\_experience NUMBER(8,2),

CONSTRAINT studentdb\_pk PRIMARY KEY (Student\_Id));

--**DDL statement for Company table**

CREATE TABLE Company

( Student\_Id NUMBER(8) NOT NULL,

Company VARCHAR(200),

CONSTRAINT Company\_pk PRIMARY KEY (Student\_Id,Company),

CONSTRAINT fk\_Company FOREIGN KEY (Student\_Id) REFERENCES Student\_DB(Student\_Id));

--**DDL statement for UG Details table**

CREATE TABLE UG\_MajorName

( Student\_Id NUMBER(8) NOT NULL,

UG\_Major VARCHAR2(200),

CONSTRAINT UGMajor\_pk PRIMARY KEY (Student\_Id,UG\_Major),

CONSTRAINT UGMajor\_FK FOREIGN KEY (Student\_Id) REFERENCES Student\_DB(Student\_Id));

--**DDL statement for Country table**

CREATE TABLE Country

( Student\_Id NUMBER(8) NOT NULL,

Current\_Country VARCHAR2(200),

CONSTRAINT Country\_pk PRIMARY KEY (Student\_Id,Current\_Country),

CONSTRAINT Country\_FK FOREIGN KEY (Student\_Id) REFERENCES Student\_DB(Student\_Id));

--**DDL statement for Institution table**

CREATE TABLE Institution

( Student\_Id NUMBER(8) NOT NULL,

UG\_Institution VARCHAR2(200),

CONSTRAINT Institution\_pk PRIMARY KEY (Student\_Id,UG\_Institution),

CONSTRAINT Institution\_FK FOREIGN KEY (Student\_Id) REFERENCES Student\_DB(Student\_Id));

--**DDL statement for Decisison table**

CREATE TABLE Decision

( Student\_Id NUMBER(8) NOT NULL,

Decision VARCHAR(200),

CONSTRAINT Decison\_pk PRIMARY KEY (Student\_Id,Decision),

CONSTRAINT Decision\_FK FOREIGN KEY (Student\_Id) REFERENCES Student\_DB(Student\_Id));

--**DDL statement for Test\_Details table**

CREATE TABLE Test\_Details

( Student\_Id NUMBER(8) NOT NULL,

Exam\_Id NUMBER(20),

Exam\_Description VARCHAR(200),

CONSTRAINT TestDetails\_pk PRIMARY KEY (Student\_Id,Exam\_Id),

CONSTRAINT Testdetails3\_FK FOREIGN KEY (Student\_Id) REFERENCES Student\_DB(Student\_Id));

--**DDL statement for GRE\_TOEFL\_Test\_Details table**

CREATE TABLE GRE\_TOEFL\_Test\_Details

( Student\_Id NUMBER(8) NOT NULL ,

Exam\_Id NUMBER(20) ,

Verbal\_Raw NUMBER(20),

VerbalPercentage NUMBER(6,2),

Quant\_Raw NUMBER(20),

QuantPercentage NUMBER(6,2),

AWA\_Raw NUMBER(20,2),

AWAPercentage NUMBER(6,2),

Listening NUMBER(20),

Speaking NUMBER(20),

TOEFL\_Total NUMBER(20),

CONSTRAINT GRETOTestDetails\_pk PRIMARY KEY (Student\_Id,Exam\_Id),

CONSTRAINT GRETOTestdetails\_FK FOREIGN KEY (Student\_Id,Exam\_Id) REFERENCES Test\_Details(Student\_Id,Exam\_Id));

--**DDL statement for GRE\_IELTS\_Test\_Details table**

CREATE TABLE GRE\_IELTS\_Test\_Details

( Student\_Id NUMBER(8) NOT NULL ,

Exam\_Id NUMBER(20) ,

Verbal\_Raw NUMBER(20),

VerbalPercentage NUMBER(6,2),

Quant\_Raw NUMBER(20),

QuantPercentage NUMBER(6,2),

AWA\_Raw NUMBER(20,2),

AWAPercentage NUMBER(6,2),

Listening NUMBER(20),

Speaking NUMBER(20),

IELTS\_Total NUMBER(20),

CONSTRAINT GREIETestdetails\_FK FOREIGN KEY (Student\_Id,Exam\_Id) REFERENCES Test\_Details(Student\_Id,Exam\_Id),

CONSTRAINT GREIETestDetails\_pk PRIMARY KEY (Student\_Id,Exam\_Id));

--**DDL statement for GMAT\_TOEFL\_Test\_Details table**

CREATE TABLE GMAT\_TOEFL\_Test\_Details

( Student\_Id NUMBER(8) NOT NULL,

Exam\_Id NUMBER(20),

Verbal\_Raw NUMBER(20),

VerbalPercentage NUMBER(6,2),

Quant\_Raw NUMBER(20),

QuantPercentage NUMBER(6,2),

AWA\_Raw NUMBER(20,2),

AWAPercentage NUMBER(6,2),

IR\_Raw NUMBER(20),

IRPercentage NUMBER(6,2),

Listening NUMBER(20),

Speaking NUMBER(20),

TOEFL\_Total NUMBER(20),

CONSTRAINT GMATTOTestDetails\_pk PRIMARY KEY (Student\_Id,Exam\_Id),

CONSTRAINT GMATTOTestdetails\_FK FOREIGN KEY (Student\_Id,Exam\_Id) REFERENCES Test\_Details(Student\_Id,Exam\_Id));

--**DDL statement for GMAT\_IELTS\_Test\_Details table**

CREATE TABLE GMAT\_IELTS\_Test\_Details

( Student\_Id NUMBER(8) NOT NULL,

Exam\_Id NUMBER(20),

Verbal\_Raw NUMBER(20),

VerbalPercentage NUMBER(6,2),

Quant\_Raw NUMBER(20),

QuantPercentage NUMBER(6,2),

AWA\_Raw NUMBER(20,2),

AWAPercentage NUMBER(6,2),

IR\_Raw NUMBER(20),

IRPercentage NUMBER(6,2),

Listening NUMBER(20),

Speaking NUMBER(20),

IELTS\_Total NUMBER(20),

CONSTRAINT GMATIETestDetails\_pk PRIMARY KEY (Student\_Id,Exam\_Id),

CONSTRAINT GMATIETestdetails\_FK FOREIGN KEY (Student\_Id,Exam\_Id) REFERENCES Test\_Details(Student\_Id,Exam\_Id));

1. QUERIES

**1)Is your design in 3NF,Justify**

**Sol:**

Yes,our design is in 3NF because of the following features removed making it to 3NF :

1) There are no table with multivalued attributes .If there are any attributes like that for example Company then it is resolved by making Company as a different table

2) There are no partial dependencies present.

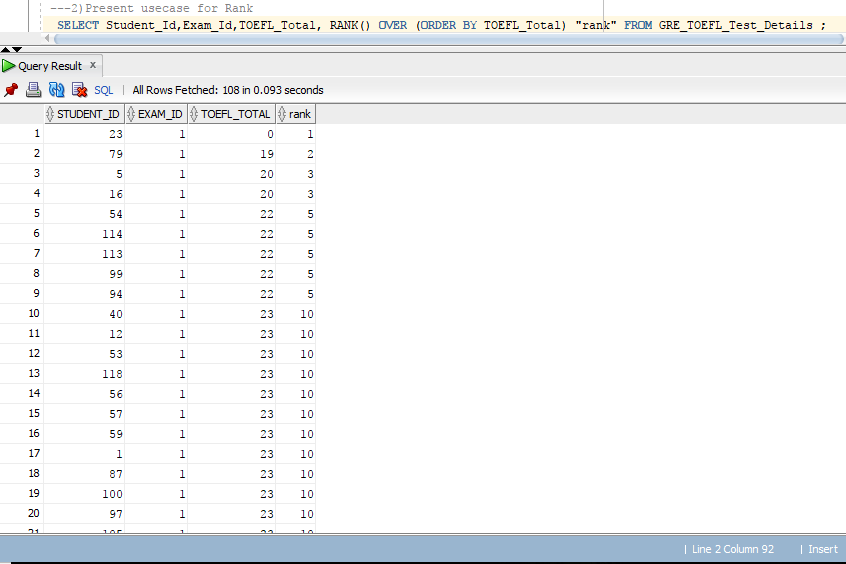
3) If there are any attributes like that for example undergraduation Major is transitively dependent on Institution which is dependent on Student hence different tables are created each for Insitution, UnderGraduate Major and Student

**2)Present usecase for Rank**

|  |  |
| --- | --- |
| ***Use Case #*** |  |
| Use Case Description | Students’s Rank by TOEFL Total who have written GRE also |
| Detailed Description | This will contain a report of all the students ranks who have written GRE and is based on TOEFL Total score. |

**Query in Oracle:**

SELECT Student\_Id,Exam\_Id,TOEFL\_Total, RANK() OVER (ORDER BY TOEFL\_Total) "rank" FROM GRE\_TOEFL\_Test\_Details ;

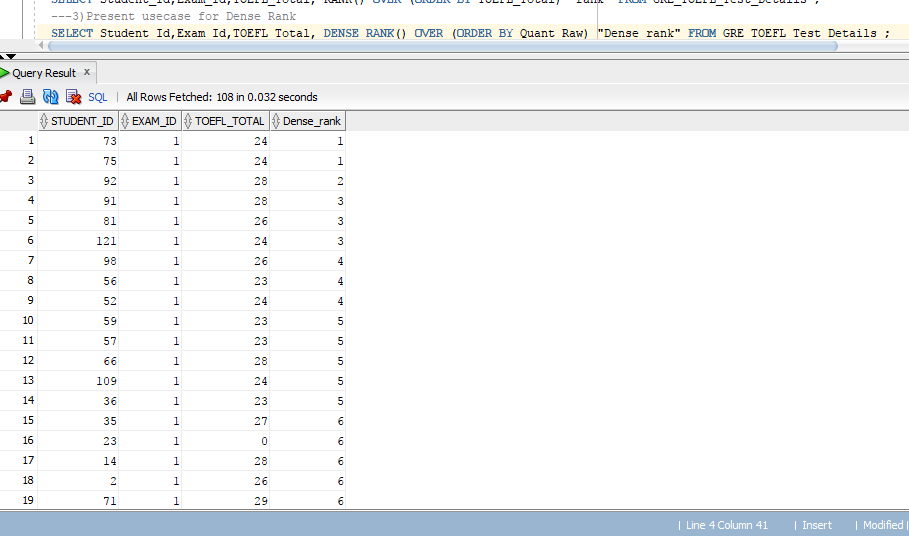


**3)Present usecase for Dense Rank**

|  |  |
| --- | --- |
| ***Use Case #*** |  |
| Use Case Description | Students’s Dense Rank by TOEFL Total who have written GRE also |
| Detailed Description | This will contain a report of all the students Dense ranks who have written GRE and is based on TOEFL Total score. |

**Query in Oracle:**

SELECT Student\_Id,Exam\_Id,TOEFL\_Total, DENSE\_RANK() OVER (ORDER BY Quant\_Raw) "Dense\_rank" FROM GRE\_TOEFL\_Test\_Details ;



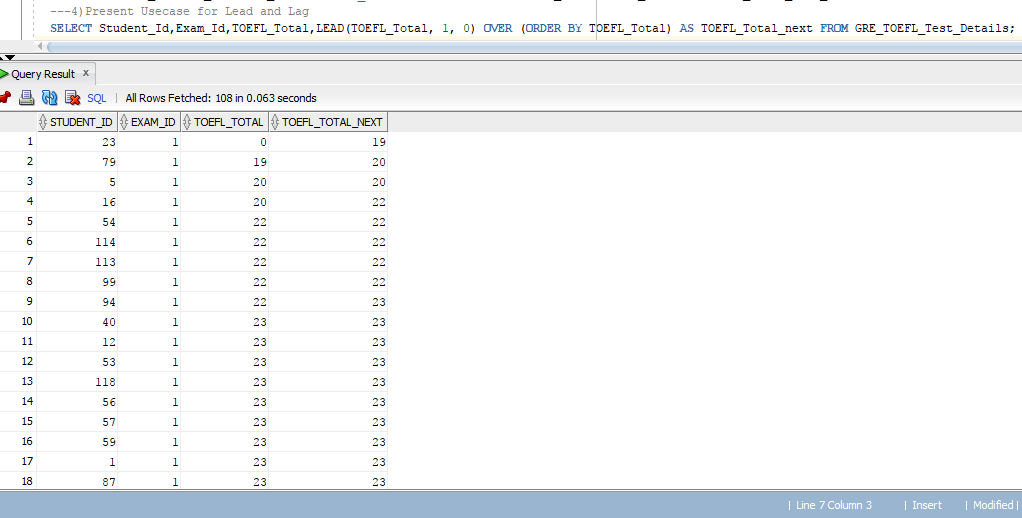
**4)Present Usecase for Lead**

**Lead Usecase**

|  |  |
| --- | --- |
| ***Use Case #*** | ***UC.Lead*** |
| Use Case Description | Students’s who have written GRE and TOEFL Total score and other Test details from the next 1 rows based on TOEFL Total Score |
| Detailed Description | This will contain a report of all the students who have written GRE and their TOEFL Total score Details from the next 1 row based on the TOEFL Total Score secured by each student. |

**Query in Oracle:**

SELECT Student\_Id,Exam\_Id,TOEFL\_Total,LEAD(TOEFL\_Total, 1, 0) OVER (ORDER BY TOEFL\_Total) AS TOEFL\_Total\_next FROM GRE\_TOEFL\_Test\_Details;



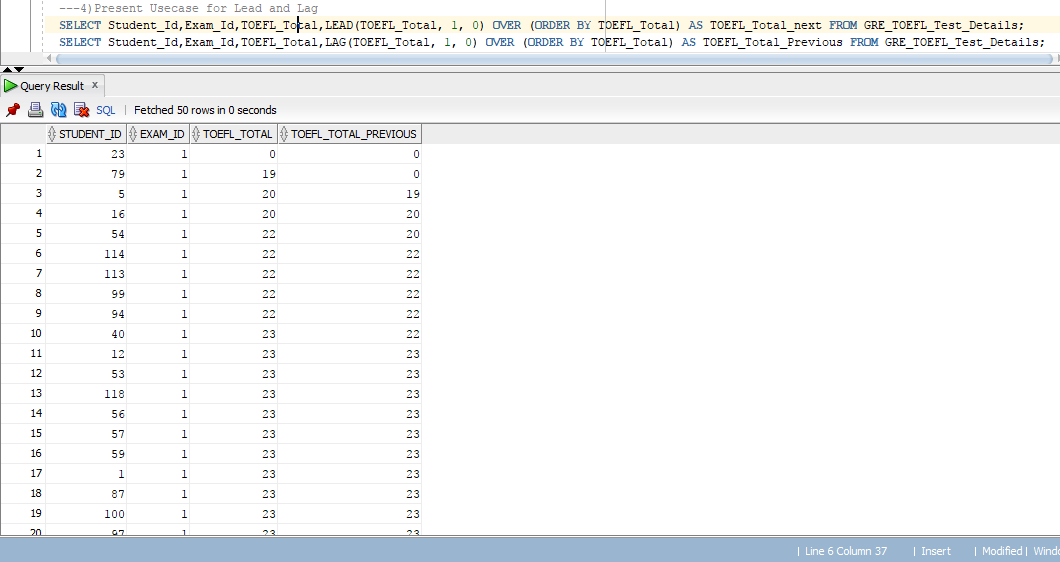
**5)Present Usecase for Lag**

**Lag Usecase**

|  |  |
| --- | --- |
| ***Use Case #*** | ***UC.Lag*** |
| Use Case Description | Students’s who have written GRE and TOEFL Total score and other Test details from the next 1 rows based on TOEFL Total Score |
| Detailed Description | This will contain a report of all the students who have written GRE and their TOEFL Total score Details from the pervious 1 row based on the TOEFL Total Score secured by each student. |

**Query in Oracle:**

SELECT Student\_Id,Exam\_Id,TOEFL\_Total,LAG(TOEFL\_Total, 1, 0) OVER (ORDER BY TOEFL\_Total) AS TOEFL\_Total\_Previous FROM GRE\_TOEFL\_Test\_Details;



**6)Bring Individual student level information with Test Result and Verbal Avg. Score and Avg Quant Raw Score using**

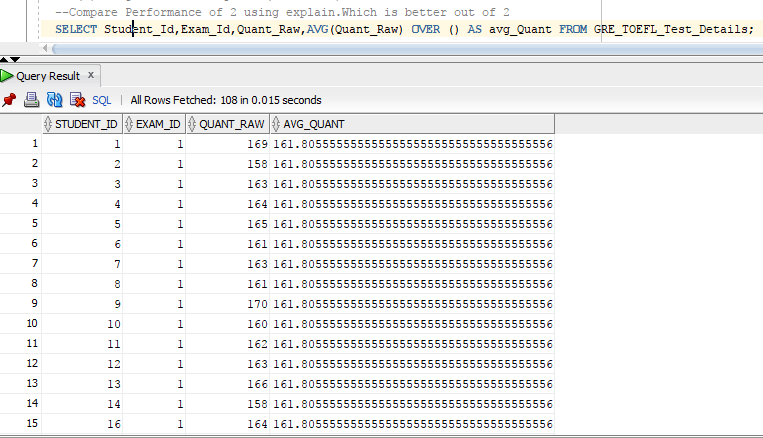
**(a)Using Over()(Partition)**

**(b)Using Without Using Over(Partition)**

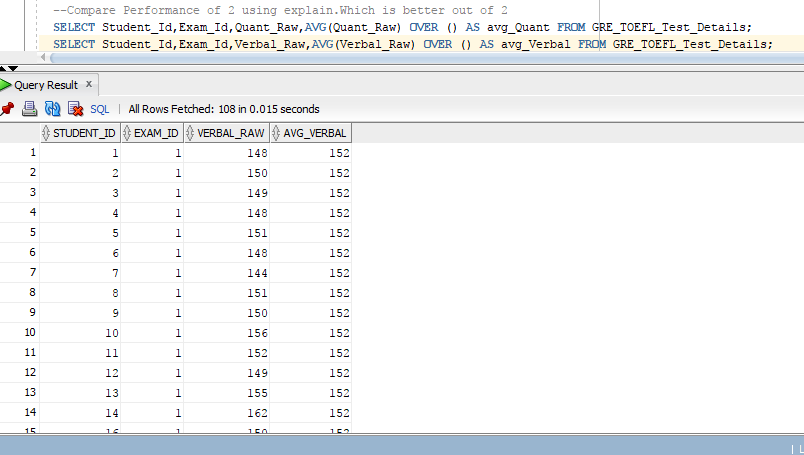
**Compare Performance of 2 using explain.Which is better out of 2**

**Sol**Using Over() analytical functionQueries in SQL

SELECT Student\_Id,Exam\_Id,Quant\_Raw,AVG(Quant\_Raw) OVER () AS avg\_Quant FROM GRE\_TOEFL\_Test\_Details;

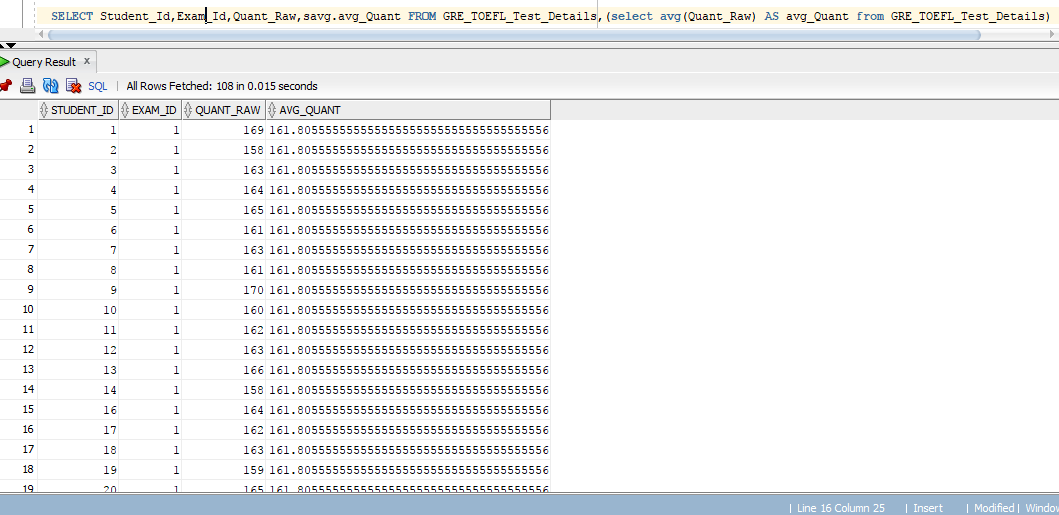


SELECT Student\_Id,Exam\_Id,Verbal\_Raw,AVG(Verbal\_Raw) OVER () AS avg\_Verbal FROM GRE\_TOEFL\_Test\_Details;

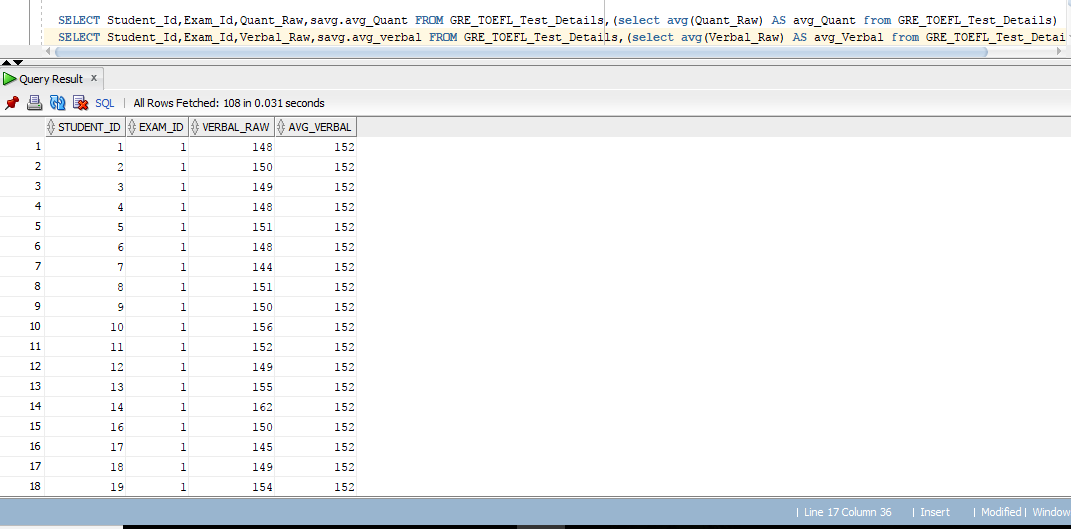


(b)Using Without Using Over(Partition)

SELECT Student\_Id,Exam\_Id,Quant\_Raw,savg.avg\_Quant FROM GRE\_TOEFL\_Test\_Details,(select avg(Quant\_Raw) AS avg\_Quant from GRE\_TOEFL\_Test\_Details) savg ;



SELECT Student\_Id,Exam\_Id,Verbal\_Raw,savg.avg\_verbal FROM GRE\_TOEFL\_Test\_Details,(select avg(Verbal\_Raw) AS avg\_Verbal from GRE\_TOEFL\_Test\_Details) savg ;



**--Comparision between 2 methods**

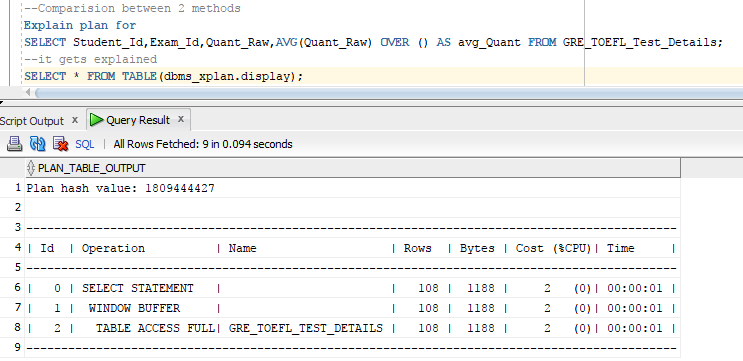
**Explain Plan for OVER() Analytical function**

Explain plan for

SELECT Student\_Id,Exam\_Id,Quant\_Raw,AVG(Quant\_Raw) OVER () AS avg\_Quant FROM GRE\_TOEFL\_Test\_Details;

SELECT \* FROM TABLE(dbms\_xplan.display);

Results:



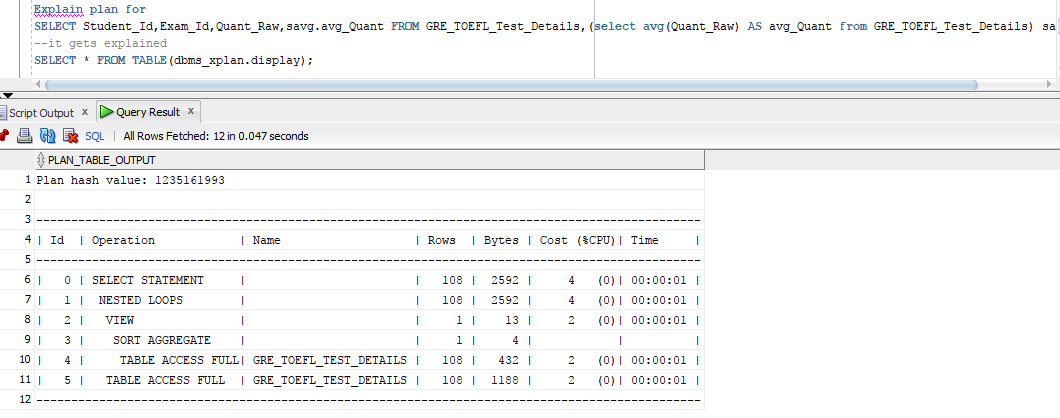
**Explain Plan without OVER() Analytical function**

Explain plan for

SELECT Student\_Id,Exam\_Id,Quant\_Raw,savg.avg\_Quant FROM GRE\_TOEFL\_Test\_Details,(select avg(Quant\_Raw) AS avg\_Quant from GRE\_TOEFL\_Test\_Details) savg ;

SELECT \* FROM TABLE(dbms\_xplan.display);

Results:



**Comparison between two methods:**

As we can see more Cost units are being utilized in the case without Using OVER() Analytical function as compared to the one with using OVER() Function.

Hence Method using OVER() analytical function should be a better choice out of 2

Q1: Is our design in third normal form? Justify:

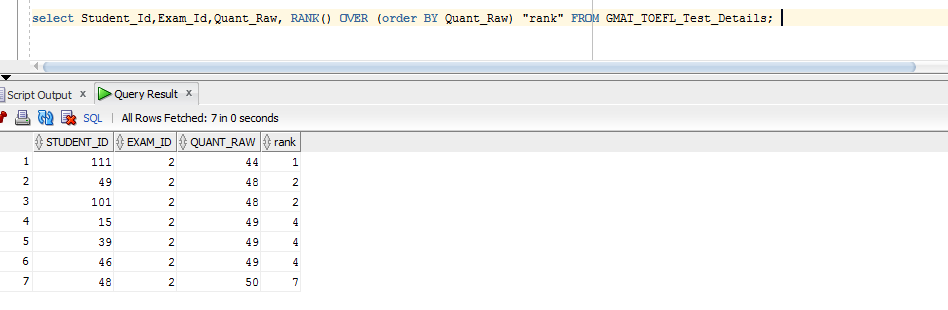
* Yes, our design is in the normal form.

Justification: We reduced the insert, delete and update anomalies by breaking down the table into smaller chunks, which removed the functional dependencies. This made the tables more informative and helped increase the performance for transaction processing.

Ex: The company, country and college attributes were dependent on the Student database and any changes made to them would have removed them from the main table. The partial dependencies were removed and the table were constructed in such a manner that the columns are dependent on the primary key.

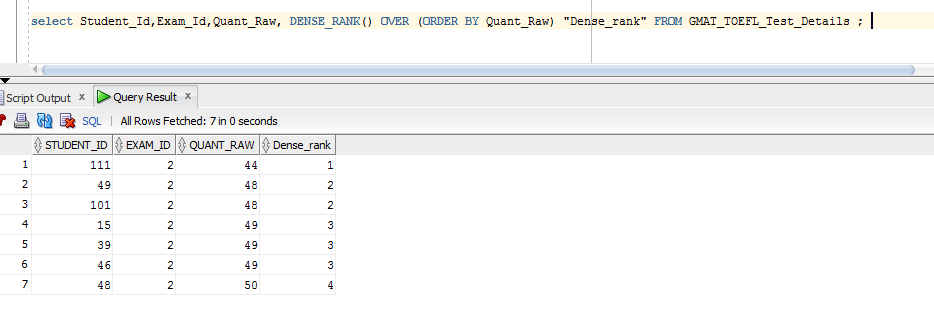
Q2. Present the use case, query and result for rank function.

- Select Student\_Id,Exam\_Id,Quant\_Raw, RANK() OVER (order BY Quant\_Raw) "rank" FROM GMAT\_TOEFL\_Test\_Details;



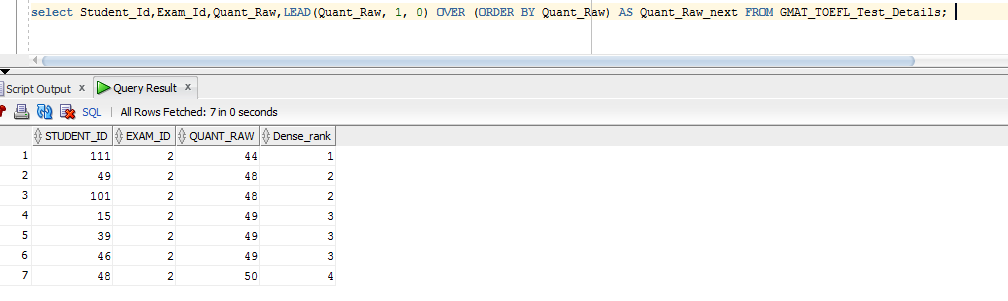
Q3. Present the use case, query and result for dense\_rank function.

-Select Student\_Id,Exam\_Id,Quant\_Raw, DENSE\_RANK() OVER (ORDER BY Quant\_Raw) "Dense\_rank" FROM GMAT\_TOEFL\_Test\_Details ;



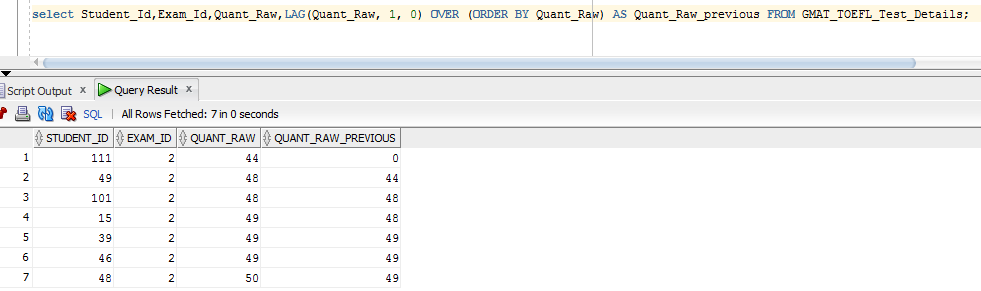
Q4: Present the use case, query and result for lead function.

- Select Student\_Id,Exam\_Id,Quant\_Raw,LEAD(Quant\_Raw, 1, 0) OVER (ORDER BY Quant\_Raw) AS Quant\_Raw\_next FROM GMAT\_TOEFL\_Test\_Details;



Q5. Present the use case, query and result for lag function

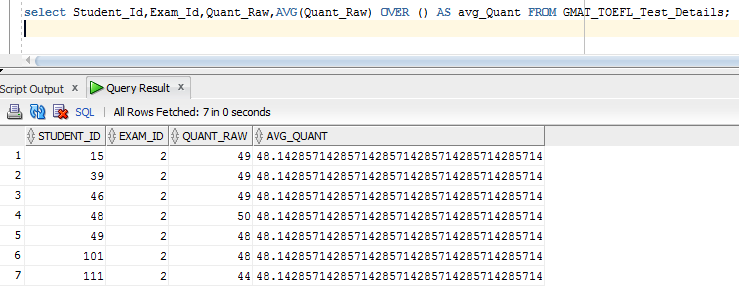
- Select Student\_Id,Exam\_Id,Quant\_Raw,LAG(Quant\_Raw, 1, 0) OVER (ORDER BY Quant\_Raw) AS Quant\_Raw\_previous FROM GMAT\_TOEFL\_Test\_Details;



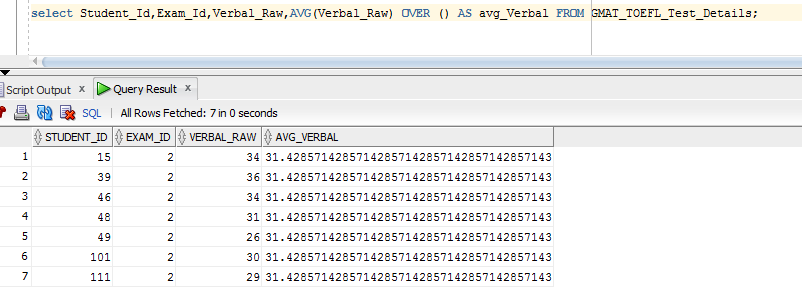
Q6: Bring all the individual student level information along with test information containing quant avg score and verbal avg score

-- Using over()

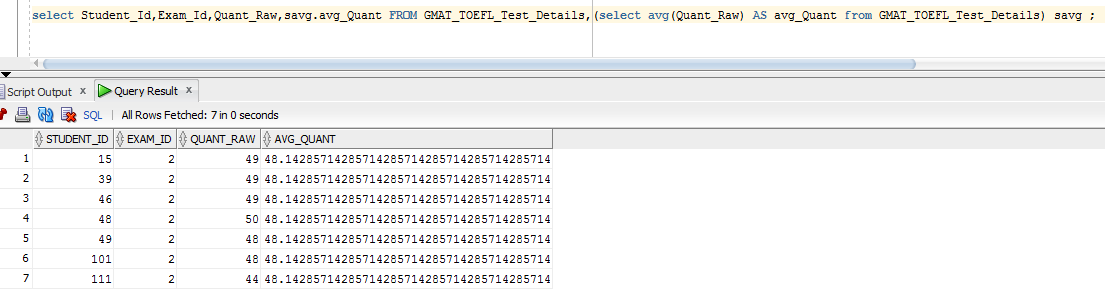
- Select Student\_Id,Exam\_Id,Quant\_Raw,AVG(Quant\_Raw) OVER () AS avg\_Quant FROM GMAT\_TOEFL\_Test\_Details;



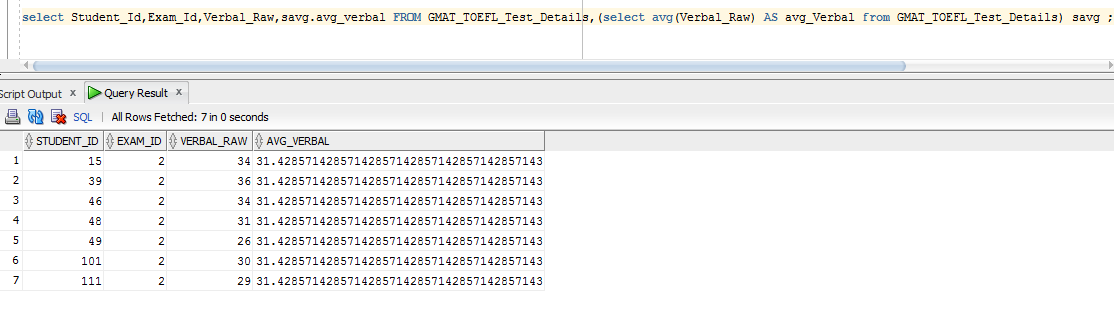
- select Student\_Id,Exam\_Id,Verbal\_Raw,AVG(Verbal\_Raw) OVER () AS avg\_Verbal FROM GMAT\_TOEFL\_Test\_Details;



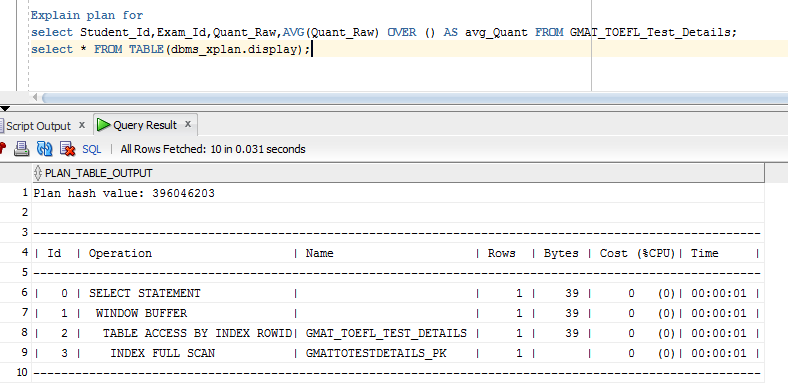
--Without Over()

-Select Student\_Id,Exam\_Id,Quant\_Raw,savg.avg\_Quant FROM GMAT\_TOEFL\_Test\_Details,(select avg(Quant\_Raw) AS avg\_Quant from GMAT\_TOEFL\_Test\_Details) savg ; 

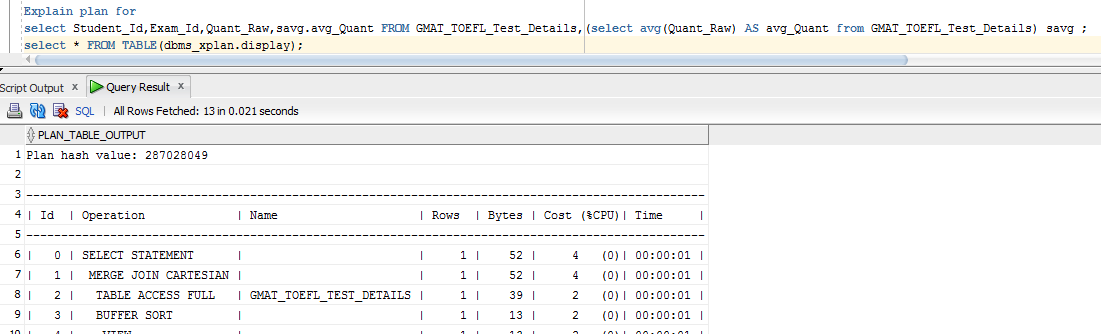
-Select Student\_Id,Exam\_Id,Verbal\_Raw,savg.avg\_verbal FROM GMAT\_TOEFL\_Test\_Details,(select avg(Verbal\_Raw) AS avg\_Verbal from GMAT\_TOEFL\_Test\_Details) savg ;



-Explain plan for  
select Student\_Id,Exam\_Id,Quant\_Raw,AVG(Quant\_Raw) OVER () AS avg\_Quant FROM GMAT\_TOEFL\_Test\_Details;  
select \* FROM TABLE(dbms\_xplan.display);



-Explain plan for  
select Student\_Id,Exam\_Id,Quant\_Raw,savg.avg\_Quant FROM GMAT\_TOEFL\_Test\_Details,(select avg(Quant\_Raw) AS avg\_Quant from GMAT\_TOEFL\_Test\_Details) savg ;  
select \* FROM TABLE(dbms\_xplan.display);



DBMS Project Queries

Q1: Is our design in third normal form? Justify:

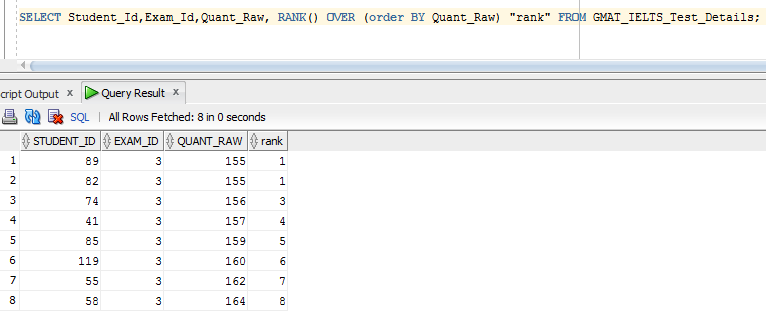
Sol1:

* Our earlier design was not in the third normal form.
* However, we brought it into third normal form which we are submitting herewith.
* There were update, delete and insert anomalies in the earlier design which we removed in this one.
* There were functional dependencies on the non-key fields in the earlier design which we eliminated.
* And all the attributes were not totally dependent on the primary key in the earlier case which we ensured otherwise in this one.

Q2: Present the use case, query and result for rank function:

Sol2:

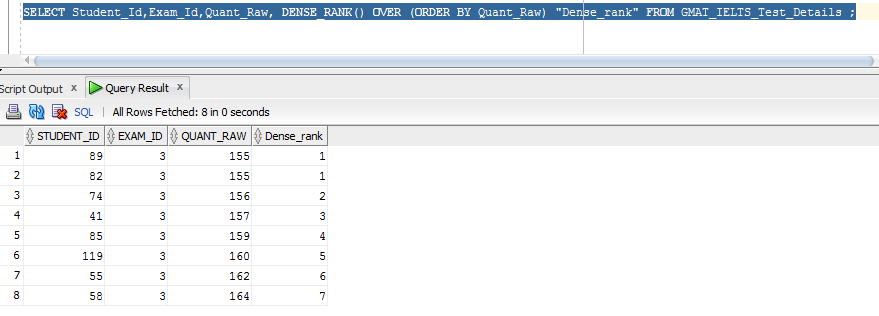
SELECT Student\_Id,Exam\_Id,Quant\_Raw, RANK() OVER (order BY Quant\_Raw) "rank" FROM GMAT\_IELTS\_Test\_Details;



Q3: Present the use case, query and result for dense\_rank function:

Sol3:

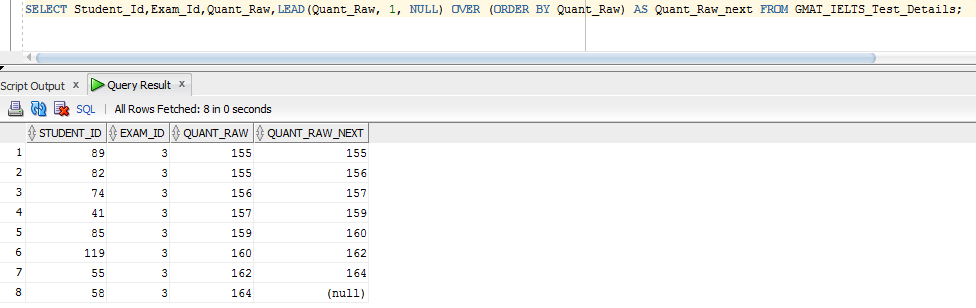
SELECT Student\_Id,Exam\_Id,Quant\_Raw, DENSE\_RANK() OVER (ORDER BY Quant\_Raw) "Dense\_rank" FROM GMAT\_IELTS\_Test\_Details ;



Q4: Present the use case, query and result for lead function:

Sol4:

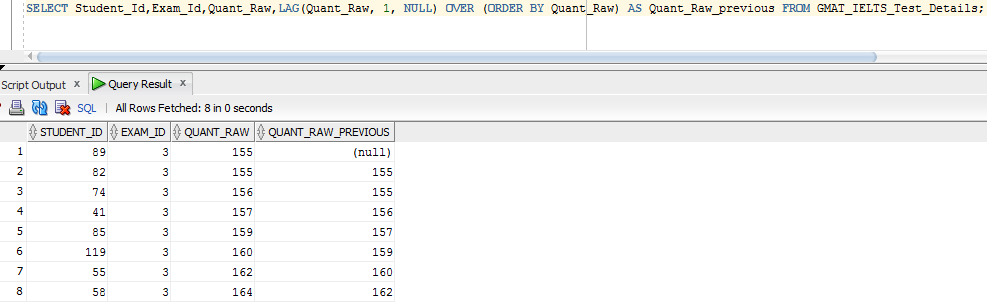
SELECT Student\_Id,Exam\_Id,Quant\_Raw,LEAD(Quant\_Raw, 1, NULL) OVER (ORDER BY Quant\_Raw) AS Quant\_Raw\_next FROM GMAT\_IELTS\_Test\_Details;



Q5: Present the use case, query and result for lag function:

Sol5:

SELECT Student\_Id,Exam\_Id,Quant\_Raw,LAG(Quant\_Raw, 1, NULL) OVER (ORDER BY Quant\_Raw) AS Quant\_Raw\_previous FROM GMAT\_IELTS\_Test\_Details;

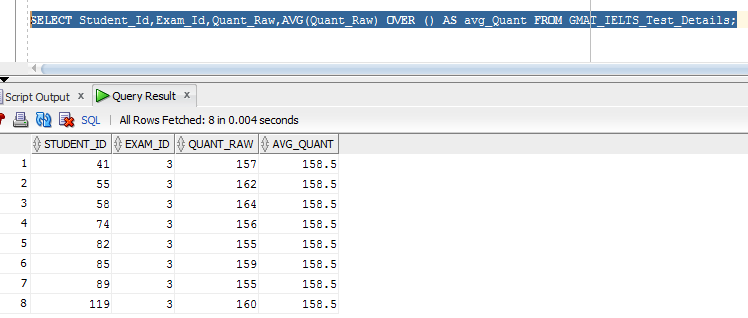


Q6: Bring all the individual student level information along with test information containing quant avg score and verbal avg score:

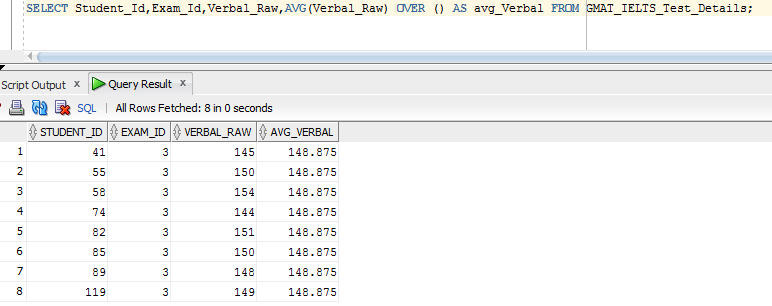
1. Using over() and partition by:

Sol 6a:

SELECT Student\_Id,Exam\_Id,Quant\_Raw,AVG(Quant\_Raw) OVER () AS avg\_Quant FROM GMAT\_IELTS\_Test\_Details;



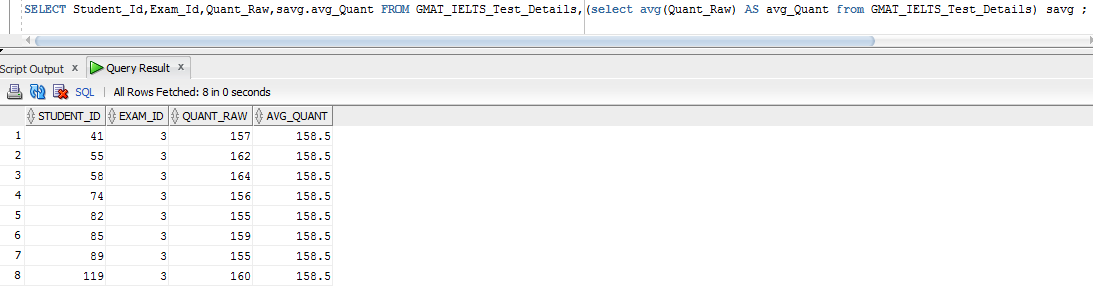
SELECT Student\_Id,Exam\_Id,Verbal\_Raw,AVG(Verbal\_Raw) OVER () AS avg\_Verbal FROM GMAT\_IELTS\_Test\_Details;



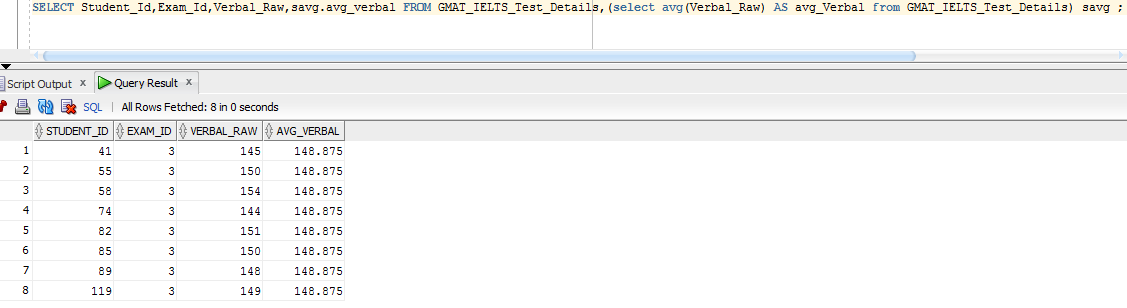
1. Without using over() and partition by:

Sol 6b:

SELECT Student\_Id,Exam\_Id,Quant\_Raw,savg.avg\_Quant FROM GMAT\_IELTS\_Test\_Details,(select avg(Quant\_Raw) AS avg\_Quant from GMAT\_IELTS\_Test\_Details) savg ;



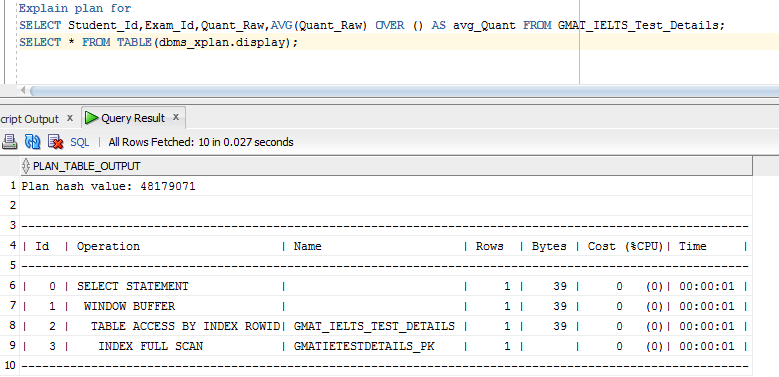
SELECT Student\_Id,Exam\_Id,Verbal\_Raw,savg.avg\_verbal FROM GMAT\_IELTS\_Test\_Details,(select avg(Verbal\_Raw) AS avg\_Verbal from GMAT\_IELTS\_Test\_Details) savg ;



1. Explain plan for both, compare and mention which one is better:

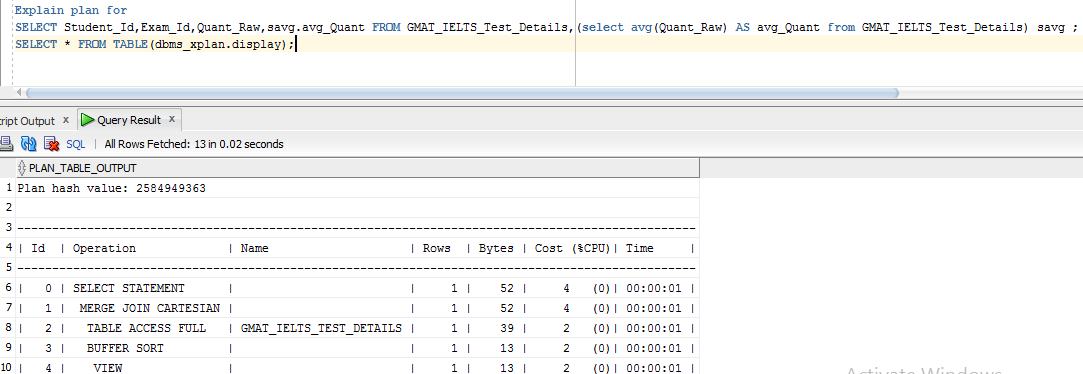
Sol 6c:

Below is the explain plan for the query with over () and partition by-



In this case, the cost is zero and the no of bytes used is 39

Now, below is the explain plan for the query without using over() and partition by-



In this case, we can see that the cost is substantially increased as also the number of bytes used are higher.

Therefore, considering the cost and the number of bytes used, it is better to query using over() and partition by.