TUS THE TECHNOLOGICAL UNIVERSITY OF THE SHANNON: MIDLANDS MIDWEST

Student Name:	**Nur Alya Maisarah Abd Karim**
Student Id Number:	**A00268252**
Course:	**MSc in Data Analytics**
Title of Assignment:	**Relational Database Project**
Date:	**10th December 2024**

Table of Contents

Original Project Outline

Project Title:

Database Design for Capturing Research Project and Scientist Information

Objective:

To design and implement a relational database that captures essential details about scientists and their research projects. This database will ensure comprehensive tracking of projects, funding agencies, and related scientist information, facilitating efficient querying and reporting.

Scope:

- Capture scientist demographics, qualifications, and career highlights
- Record research project details, including focus, funding, and progress
- Use data from universities in Ireland, Scotland, Wales, and England
- Enforce a one-to-many relationship: a scientist can have multiple projects, but each project is linked to one scientist

Proposed Tables:

1. **Scientist Table**:

    - **Variables:** *Scientist_Id, ForeName, Surname, Town, County, Nationality, Date_Born, University, Degree, Highest_Qual, Yrs_Exp, Num_Publications, Largest_Grant, Income*
    - **Purpose:** To store information about each scientist, their educational background, and professional achievements

2. **Research_Project Table**:

    - **Variables:** *Project_Id, Description, Start_Year, End_Year, Discipline, Subject_Area, Funding, Project_Value, Progress_Report*
    - **Purpose:** To track the details of each research project, including its focus, funding source, and value

Relationships:

- **One-to-Many**: A scientist can undertake many research projects, but each project is supervised by one and only one scientist

Expected Outcomes:

- A fully functional database schema that enforces relationships and allows data entry
- Capability to generate reports, such as total project funding by a specific scientist or trends in funding by discipline

Revisions/Fixes to the Project Outline

Changes Made:

1. **Towns Added**: Expanded the list of valid towns to include Dublin, Cork, Edinburgh, Glasgow, Cardiff, Swansea, Oxford, and Athlone to provide a more diverse geographic representation.
2. **Universities Added**: Included Trinity College Dublin, University College Cork, University of Edinburgh, University of Glasgow, Cardiff University, Swansea University, University of Oxford, University of Cambridge, and TUS: Midlands Midwest to ensure realistic educational data.
3. **Degree Options**: Defined the degree options as Physics, Biology, Microbiology, Chemistry, and Computer Science to align with common scientific disciplines.
4. **Highest Qualification Options**: Limited qualifications to PhD, Masters, and Degree to standardize data entry and maintain consistency.
5. **Subject Area Expanded**: Added Polymer/Nanomaterial, Medicine, Renewable Energy, Space Science, Climate Change, and Artificial Intelligence as options to reflect current research trends and priorities.
6. **Discipline Matching Degree**: Ensured alignment between Discipline and Degree values to maintain data integrity and logical coherence.
7. **Funding Sources Defined**: Standardized funding sources with options including EU, IRC, Private Donation, Department of Education, WHO, and Government of Ireland for clearer categorization and reporting.

Data Validation Rules: Added constraints for numeric fields like Yrs_Exp, Num_Publications, Largest_Grant, and Project_Value to ensure values are non-negative and within reasonable ranges.

Default Values: Introduced default values for Progress_Report (e.g., Quarterly) to simplify data entry and standardize reporting.

Date Constraints: Enforced that End_Year cannot be earlier than Start_Year and restricted Date_Born to past dates only.

Unique Identifiers: Ensured all primary keys, such as Scientist_Id and Project_Id, are unique and auto-incrementing to avoid duplication and streamline record creation.

Currency Format: Standardized monetary fields like Largest_Grant and Project_Value to use Euros (€) with appropriate formatting, ensuring consistency in financial reporting.

Additional Relationships for Future Scalability: Considered introducing a table for Funding_Agencies to capture more detailed information about funding sources, allowing for more granular reporting and potential extensions of the database.

ERD Diagram

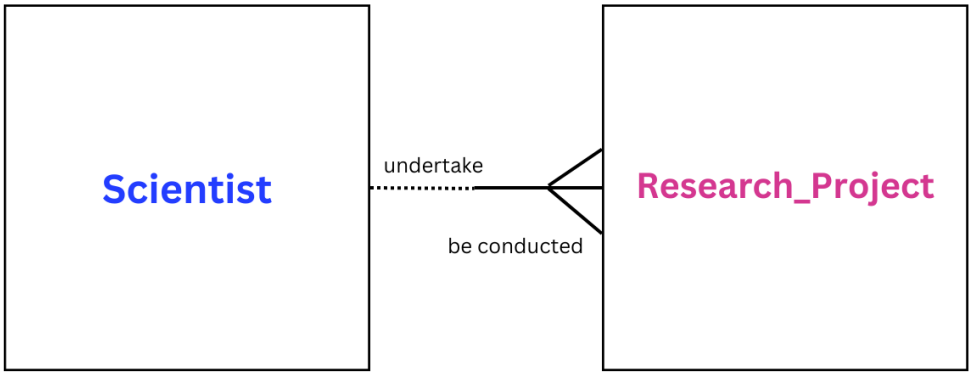


Figure 1 ERD Diagram showing the entities, relationships (cardinality/optionality)

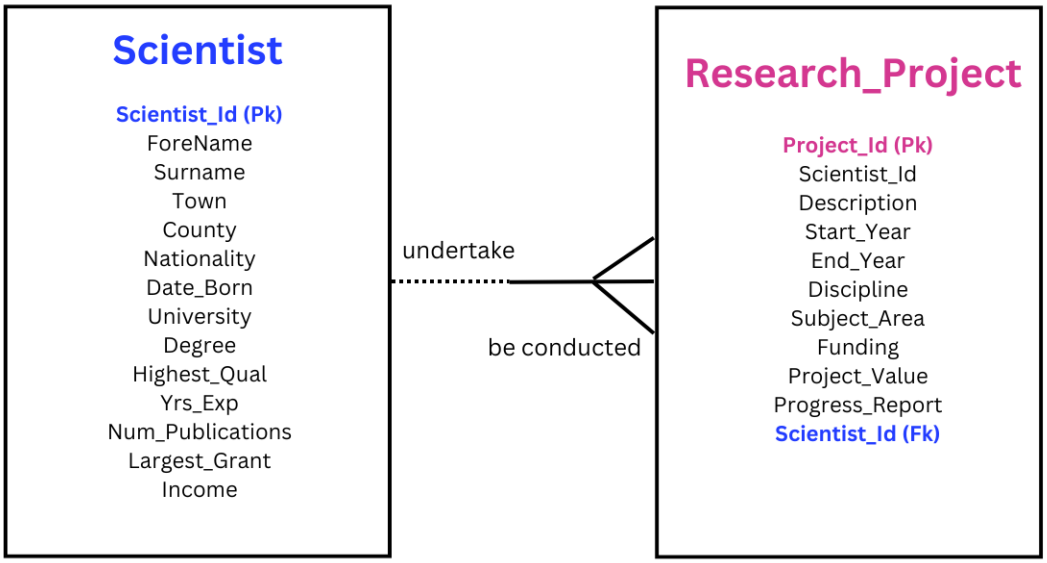


Figure 2 Fully Attributed ERD Diagram that shows all the attributes in each table including colour-coded Primary Keys and Matching Foreign Keys

Create Tables and Inserts

The provided SQL statements define the structure for two tables: **Scientist** and **Research_Project**. The Drop Table commands are used to delete these tables if they already exist. The Create Table for **Scientist** establishes columns to store personal, academic, and professional details of scientists, including constraints such as a primary key on Scientist_Id. The create Table for **Research_Project** defines columns for project-specific data, including a foreign key constraint linking Scientist_Id to the **Scientist** table, ensuring that each research project is associated with a valid scientist. The tables together track scientists, their projects, and related details like funding and progress reporting.

```
Drop Table  Research_Project;
Drop Table  Scientist;

Create Table        Scientist(
Scientist_Id        Number(4),
ForeName        Varchar2(20),
Surname         Varchar2(10),
Town            Varchar2(10),
County          Varchar2(15),
Nationality         Varchar2(15),
Date_Born       Date,
University      Varchar2(25),
Degree          Varchar2(20),
Highest_Qual        Varchar2(10),
Yrs_Exp             Number(2),
Num_Publications    Number(2),
Largest_Grant       Number(7),
Income          Number(20),
 Constraint Scientist_Scientist_Id_Pk Primary Key (Scientist_Id));

Create Table        Research_Project(
Project_Id      Number(5),
Scientist_Id        Number(4),
Description     Varchar2(105),
Start_Year      Number(4),
End_Year        Number(4),
Discipline      Varchar2(20),
Subject_Area        Varchar2(25),
Funding         Varchar2(25),
Project_Value       Number(7),
Progress_Report     Varchar2(10),
 Constraint Research_Project_Scientist_Id   Foreign Key (Scientist_Id) References Scientist (Scientist_Id),
 Constraint Research_Project_Project_Id_Pk  Primary Key (Project_Id));
```

The provided Insert Into statements populate the **Scientist** and **Research_Project** tables with sample data. For the **Scientist** table, entries capture detailed personal and professional information, such as João Silva from Brazil, who studied Physics at Trinity College Dublin and holds a PhD. The data includes a range of scientists from various nationalities, disciplines, and universities, illustrating diversity in academic and research backgrounds. Similarly, the **Research_Project** table records projects with descriptions like "Innovative Catalysts for Green Applications," linking each project to a scientist via Scientist_Id. These projects span disciplines such as Chemistry, Physics, and Biology, with varied funding sources like the EU and Government of Ireland, and include project-specific details like start and end years, subject areas, and progress reporting schedules.

```
--Scientist*/
--Scientist_Id, ForeName, Surname, Town, County, Nationality, Date_Born, University, Degree, Highest_Qual, Yrs_Exp, Num_Publications, Largest_Grant, Income*/
INSERT INTO Scientist VALUES(1001,'João','Silva','Dublin','Dublin','Brazil','29-Mar-92','Trinity College Dublin','Physics','PhD',1,5,450000,9375);
INSERT INTO Scientist VALUES(1002,'Ahmad','Ali','Cork','Cork','Malaysia','15-Jul-92','University College Cork','Biology','PhD',3,6,370000,7708);
INSERT INTO Scientist VALUES(1003,'James','Wong','Edinburgh','Edinburgh','United Kingdom','08-Jan-95','University of Edinburgh','Microbiology','Degree',3,1,Null,Null);
INSERT INTO Scientist VALUES(1004,'Emma','Johnson','Glasgow','Glasgow','United Kingdom','10-Oct-96','University of Glasgow','Microbiology','PhD',4,6,450000,9375);
INSERT INTO Scientist VALUES(1005,'Li','Wei','Cardiff','Cardiff','China','14-Dec-97','Cardiff University','Chemistry','PhD',3,6,280000,5833);
INSERT INTO Scientist VALUES(1006,'Ana','Costa','Swansea','Swansea','Brazil','13-Jan-92','Swansea University','Chemistry','PhD',3,4,350000,7292);
INSERT INTO Scientist VALUES(1007,'Siti Nurhaliza','Ahmad','Oxford','Oxfordshire','Malaysia','22-May-99','University of Oxford','Physics','Masters',3,3,90000,2500);
INSERT INTO Scientist VALUES(1008,'Muhammad Haziq','Syafiq','Cambridge','Cambridgeshire','Malaysia','11-Jan-93','University of Cambridge','Computer Science','PhD',2,4,445000,9271);
INSERT INTO Scientist VALUES(1009,'Nurul','Aisyah','Edinburgh','Edinburgh','Malaysia','08-Oct-97','University of Edinburgh','Chemistry','Masters',2,2,70000,1944);
INSERT INTO Scientist VALUES(1010,'Conor','O Neill','Cardiff','Cardiff','Ireland','13-Nov-93','Cardiff University','Computer Science','PhD',4,4,320000,6667);
INSERT INTO Scientist VALUES(1011,'Carlos','Santos','Cork','Cork','Brazil','17-Jun-92','University College Cork','Microbiology','Masters',4,2,70000,1944);
INSERT INTO Scientist VALUES(1012,'Lee','Wei','Athlone','Westmeath','Malaysia','30-Apr-92','TUS: Midlands Midwest','Physics','PhD',1,4,200000,4167);
```

Figure 3 Screenshot of Partial Insert Into Scientist Table

Figure 4 Figure 3 Screenshot of Partial Insert Into Research_Project Table

```
--Research_Project*/
--Project_Id, Scientist_Id, Descriptiom, Start_Year, End_Year, Discipline, Subject_Area, Funding, Project_Value, Progress_Report*/
INSERT INTO Research_Project VALUES(1,1015,'Innovative Catalysts for Green Applications',2020,2024,'Chemistry','Polymer/Nanomaterial','Government of Ireland',1000000,'Quarterly');
INSERT INTO Research_Project VALUES(2,1045,'The Role of Microbiomes in Ecosystem Health',2018,2021,'Physics','Medicine','Government of Ireland',500000,'Annually');
INSERT INTO Research_Project VALUES(3,1038,'Advances in Organic Photovoltaics for Sustainable Energy',2022,2026,'Biology','Renewable Energy','EU',3000000,'Annually');
INSERT INTO Research_Project VALUES(4,1055,'Modeling Gravitational Lensing in Astrophysics',2018,2021,'Chemistry','Space Science','IRC',1000000,'Monthly');
INSERT INTO Research_Project VALUES(5,1003,'Microbial Resistance Mechanisms in Pathogenic Bacteria',2022,2025,'Microbiology','Medicine','EU',500000,'Monthly');
INSERT INTO Research_Project VALUES(6,1010,'AI-Driven Simulations for Drug Discovery in Chemistry',2019,2023,'Microbiology','Artificial Intelligence','IRC',1000000,'Annually');
INSERT INTO Research_Project VALUES(7,1006,'Chemical Strategies for CO2 Capture and Utilization',2018,2022,'Physics','Climate Change','Private Donation',2000000,'Annually');
INSERT INTO Research_Project VALUES(8,1029,'Genetic Adaptations in Plants Under Climate Stress',2018,2022,'Physics','Climate Change','Department of Education',500000,'Quarterly');
INSERT INTO Research_Project VALUES(9,1057,'Quantum Mechanics in Material Science Innovations',2021,2025,'Biology','Polymer/Nanomaterial','Private Donation',1000000,'Annually');
INSERT INTO Research_Project VALUES(10,1027,'Impact of Urbanization on Wildlife Population Dynamics',2018,2021,'Microbiology','Climate Change','WHO',500000,'Monthly');
INSERT INTO Research_Project VALUES(11,1013,'Exploring the Effects of Quantum Entanglement in Computing',2020,2024,'Microbiology','Artificial Intelligence','EU',2000000,'Quarterly');
INSERT INTO Research_Project VALUES(12,1017,'Blockchain Technology for Securing Biomedical Data',2019,2023,'Chemistry','Artificial Intelligence','WHO',500000,'Annually');
INSERT INTO Research_Project VALUES(13,1032,'Epigenetics and the Regulation of Gene Expression',2020,2023,'Microbiology','Medicine','WHO',1000000,'Annually');
```

Query Questions And SQL Syntax And Snips

Basic Aggregation and Counting (Beginner)

1. Calculate Total Project Value by Scientist

Select

Scientist_Id,

Sum(Project_Value) Over (Partition By Scientist_Id) As Total_Project_Value,

Avg(Project_Value) Over (Partition By Scientist_Id) As Avg_Project_Value

From Research_Project;

```
A00268252_SQL>Select
  2      Scientist_Id,
  3      Sum(Project_Value) Over (Partition By Scientist_Id) As Total_Project_Value,
  4      Avg(Project_Value) Over (Partition By Scientist_Id) As Avg_Project_Value
  5  From Research_Project;

SCIENTIST_ID TOTAL_PROJECT_VALUE AVG_PROJECT_VALUE
------------ ------------------- -----------------
        1001              500000            500000
        1002             2500000           1250000
        1002             2500000           1250000
        1003              500000            500000
        1004              500000            500000
        1005             3000000           1500000
        1005             3000000           1500000
        1006             2000000           2000000
        1007             2500000           1250000
        1007             2500000           1250000
        1008             1000000            500000
        1052             2500000           1250000
        1052             2500000           1250000
        1053             1500000            750000
        1053             1500000            750000
        1054              500000            500000
        1055             3000000           1500000
        1055             3000000           1500000
        1056             1000000           1000000
        1057             1000000           1000000
        1059             1000000            500000
        1059             1000000            500000

92 rows selected.

A00268252_SQL>
```

This query calculates the total and average project value for each scientist using window functions.

- **Sum()** and **Avg()** with **Over (Partition By Scientist_Id**) aggregate project values for each scientist across all their projects
- The window functions retain individual project records, providing detailed insights into each scientist's project funding

2. **Find the percentage of each project’s value within a scientist's total**

Cl scr
Select Scientist_Id, Project_Id, Project_Value,
Round(Project_Value * 100 / Sum(Project_Value) Over (Partition By Scientist_Id), 2) As Percent_Contribution,
Rank() Over (Partition By Scientist_Id Order By Project_Value Desc) As Rank From Research_Project
Where Project_Value > 0;

```
A00268252_SQL>Select Scientist_Id, Project_Id, Project_Value,
  2  Round(Project_Value * 100 / Sum(Project_Value) Over (Partition By Scientist_Id), 2) As Percent_Contribution,
  3  Rank() Over (Partition By Scientist_Id Order By Project_Value Desc) As Rank From Research_Project
  4  Where Project_Value > 0;

SCIENTIST_ID PROJECT_ID PROJECT_VALUE PERCENT_CONTRIBUTION       RANK
------------ ---------- ------------- -------------------- ----------
        1001         14        500000                  100          1
        1002         81       2000000                   80          1
        1002         41        500000                   20          2
        1003          5        500000                  100          1
        1004         54        500000                  100          1
        1005         19       2000000                66.67          1
        1005         74       1000000                33.33          2
        1006          7       2000000                  100          1
        1007         42       2000000                   80          1
        1007         80        500000                   20          2
        1052         86        500000                   20          2
        1053         22       1000000                66.67          1
        1053         94        500000                33.33          2
        1054         23        500000                  100          1
        1055         99       2000000                66.67          1
        1055          4       1000000                33.33          2
        1056         52       1000000                  100          1
        1057          9       1000000                  100          1
        1059         98        500000                   50          1
        1059         46        500000                   50          1

92 rows selected.

A00268252_SQL>
```

This query calculates each project's percentage contribution to the total project value for its scientist and ranks projects based on their value.

- The formula `**Project_Value * 100 / Sum(Project_Value) Over (Partition By Scientist_Id)**` calculates the percentage contribution
- `**Rank()**` assigns a rank to each project, ordered by Project_Value in descending order
- The `**Where Project_Value > 0**` clause filters out projects with non-positive values

3. Find the maximum and minimum project value for each scientist

```
Cl scr
Select
   Scientist_Id, Project_Id, Project_Value,
   Max(Project_Value) Over (Partition By Scientist_Id) As Max_Value,
   Min(Project_Value) Over (Partition By Scientist_Id) As Min_Value,
   Dense_Rank() Over (Partition By Scientist_Id Order By Project_Value Desc) As Dense_Rank
From Research_Project
Where Project_Value > 0;
```

```
A00268252_SQL>Select
  2      Scientist_Id, Project_Id, Project_Value,
  3      Max(Project_Value) Over (Partition By Scientist_Id) As Max_Value,
  4      Min(Project_Value) Over (Partition By Scientist_Id) As Min_Value,
  5      Dense_Rank() Over (Partition By Scientist_Id Order By Project_Value Desc) As Dense_Rank
  6  From Research_Project
  7  Where Project_Value > 0;

SCIENTIST_ID PROJECT_ID PROJECT_VALUE  MAX_VALUE  MIN_VALUE DENSE_RANK
------------ ---------- ------------- ---------- ---------- ----------
        1001         14        500000     500000     500000          1
        1002         81       2000000    2000000     500000          1
        1002         41        500000    2000000     500000          2
        1003          5        500000     500000     500000          1
        1004         54        500000     500000     500000          1
        1005         19       2000000    2000000    1000000          1
        1005         74       1000000    2000000    1000000          2
        1006          7       2000000    2000000    2000000          1
        ----         --       -------    -------     ------          -
        1052         86        500000    2000000     500000          2
        1053         22       1000000    1000000     500000          1
        1053         94        500000    1000000     500000          2
        1054         23        500000     500000     500000          1
        1055         99       2000000    2000000    1000000          1
        1055          4       1000000    2000000    1000000          2
        1056         52       1000000    1000000    1000000          1
        1057          9       1000000    1000000    1000000          1
        1059         98        500000     500000     500000          1
        1059         46        500000     500000     500000          1

92 rows selected.

A00268252_SQL>
```

This query identifies the maximum and minimum project values for each scientist and ranks projects based on their value.

- **Max()** and **Min()** with **Over (Partition By Scientist_Id)** identify the maximum and minimum project values for each scientist
- **Dense_Rank()** assigns a dense rank to each project, ordered by Project_Value in descending order
- The **Where Project_Value > 0** condition ensures only projects with positive values are considered

4. **Rank projects by value within each scientist**

Cl scr
Select
Scientist_Id, Project_Id, Project_Value,
Rank() Over (Partition By Scientist_Id Order By Project_Value Desc) As Rank, Dense_Rank()
Over (Partition By Scientist_Id Order By Project_Value Desc) As Dense_Rank
From Research_Project Where Project_Value > 0;

```
A00268252_SQL>Select
  2     Scientist_Id, Project_Id, Project_Value,
  3     Rank() Over (Partition By Scientist_Id Order By Project_Value Desc) As Rank, Dense_Rank()
  4     Over (Partition By Scientist_Id Order By Project_Value Desc) As Dense_Rank
  5  From Research_Project Where Project_Value > 0;

SCIENTIST_ID PROJECT_ID PROJECT_VALUE       RANK DENSE_RANK
------------ ---------- ------------- ---------- ----------
        1001         14        500000          1          1
        1002         81       2000000          1          1
        1002         41        500000          2          2
        1003          5        500000          1          1
        1004         54        500000          1          1
        1005         19       2000000          1          1
        1005         74       1000000          2          2
        1006          7       2000000          1          1



        1053         22       1000000          1          1
        1053         94        500000          2          2
        1054         23        500000          1          1
        1055         99       2000000          1          1
        1055          4       1000000          2          2
        1056         52       1000000          1          1
        1057          9       1000000          1          1
        1059         98        500000          1          1
        1059         46        500000          1          1

92 rows selected.

A00268252 SQL>
```

This query ranks projects within each scientist's portfolio based on value.

- **Rank()** assigns a rank to each project based on Project_Value in descending order, with ties receiving different ranks
- **Dense_Rank()** assigns a dense rank, where tied projects get the same rank, and the next rank is consecutive
- Both functions use **Over (Partition By Scientist_Id)** to group the projects by scientist
- The **Where Project_Value > 0** condition ensures only projects with positive values are included

5. **Compute the running average project value by scientist**

Cl scr
Select
Scientist_Id, Project_Id, Project_Value,
Avg(Project_Value) Over (Partition By Scientist_Id Order By Project_Id Rows Between 2 Preceding And Current Row) As Moving_Avg,
Stddev_Samp(Project_Value) Over (Partition By Scientist_Id) As Sample_StdDev
From Research_Project Where Project_Value > 0;

```
A00268252_SQL>Select
  2     Scientist_Id, Project_Id, Project_Value,
  3     Avg(Project_Value) Over (Partition By Scientist_Id Order By Project_Id Rows Between 2 Preceding And Current Row) As Moving_Avg,
  4     Stddev_Samp(Project_Value) Over (Partition By Scientist_Id) As Sample_StdDev
  5  From Research_Project Where Project_Value > 0;

SCIENTIST_ID PROJECT_ID PROJECT_VALUE MOVING_AVG SAMPLE_STDDEV
------------ ---------- ------------- ---------- -------------
        1001         14        500000     500000
        1002         41        500000     500000    1060660.17
        1002         81       2000000    1250000    1060660.17
        1003          5        500000     500000
        1004         54        500000     500000
        1005         19       2000000    2000000    707106.781
        1005         74       1000000    1500000    707106.781
        1006          7       2000000    2000000
        1007         42       2000000    2000000    1060660.17
        1007         80        500000    1250000    1060660.17
        1008         26        500000     500000             0
        1008         88        500000     500000             0
        1009         35       1000000    1000000    707106.781
        1009        100       2000000    1500000    707106.781
        1011         17       1000000    1000000    353553.391
        1011         71        500000     750000    353553.391
        ----          --       -------    -------               -
        1052         53       2000000    2000000    1060660.17
        1052         86        500000    1250000    1060660.17
        1053         22       1000000    1000000    353553.391
        1053         94        500000     750000    353553.391
        1054         23        500000     500000
        1055          4       1000000    1000000    707106.781
        1055         99       2000000    1500000    707106.781
        1056         52       1000000    1000000
        1057          9       1000000    1000000
        1059         46        500000     500000             0
        1059         98        500000     500000             0

92 rows selected.

A00268252_SQL>
```

This query computes the running average and standard deviation of project values for each scientist.

- • **Avg()** calculates the moving average of project values, considering the current and two preceding projects using **Over (Partition By Scientist_Id Order By Project_Id Rows Between 2 Preceding And Current Row).** This means for each project, the average is calculated based on the project value of the current project and the two previous projects, creating a rolling 3-project average for each scientist.
- **Stddev_Samp()** calculates the sample standard deviation of project values for each scientist
- The **Where Project_Value > 0** condition ensures only projects with positive values are included

Windowing/Partition Functions

1. Find the top 3 projects by value for each scientist

```
Cl scr
Select *
From (
   Select
      Scientist_Id, Project_Id, Project_Value,
      Rank() Over (Partition By Scientist_Id Order By Project_Value Desc) As Rank
   From Research_Project
)
Where Rank <= 3;
```

```
A00268252_SQL>Select *
  2  From (
  3      Select
  4          Scientist_Id, Project_Id, Project_Value,
  5          Rank() Over (Partition By Scientist_Id Order By Project_Value Desc) As Rank
  6      From Research_Project
  7  )
  8  Where Rank <= 3;

SCIENTIST_ID PROJECT_ID PROJECT_VALUE       RANK
------------ ---------- -------------- ----------
        1001         14         500000          1
        1002         81        2000000          1
        1002         41         500000          2
        1003          5         500000          1
        1004         54         500000          1
        1005         19        2000000          1
        1005         74        1000000          2
        1006          7        2000000          1
        1007         42        2000000          1
        1007         80         500000          2
        1008         26         500000          1
        1008         88         500000          1
        1009        100        2000000          1
        1009         35        1000000          2
        1011         17        1000000          1
        1050         47         500000          1
        1050         61         500000          1
        1051         60        1000000          1
        1051         84        1000000          1
        1052         53        2000000          1
        1052         86         500000          2
        1053         22        1000000          1
        1053         94         500000          2
        1054         23         500000          1
        1055         99        2000000          1
        1055          4        1000000          2
        1056         52        1000000          1
        1057          9        1000000          1
        1059         46         500000          1
        1059         98         500000          1

92 rows selected.

A00268252_SQL>
```

This query retrieves the top 3 projects by value for each scientist.

- **Rank()** orders projects by Project_Value in descending order, assigning ranks
- Only the top 3 projects (Rank <= 3) are included in the result
- **Over (Partition By Scientist_Id Order By Project_Value Desc)** ranks projects within each scientist's portfolio
- The **Where** condition ensures only projects with positive values are included

2. Calculate cumulative distribution of project values for each scientist

```
Cl scr
Select
    Scientist_Id, Project_Id, Project_Value,
    Cume_Dist() Over (Partition By Scientist_Id Order By Project_Value) As Cumulative_Dist,
    Rank() Over (Partition By Scientist_Id Order By Project_Value Desc) As Rank
From Research_Project Where Project_Value > 0;
```

```
A00268252_SQL>Select
  2     Scientist_Id, Project_Id, Project_Value,
  3     Cume_Dist() Over (Partition By Scientist_Id Order By Project_Value) As Cumulative_Dist,
  4     Rank() Over (Partition By Scientist_Id Order By Project_Value Desc) As Rank
  5  From Research_Project Where Project_Value > 0;

SCIENTIST_ID PROJECT_ID PROJECT_VALUE CUMULATIVE_DIST       RANK
------------ ---------- ------------- --------------- ----------
        1001         14        500000               1          1
        1002         41        500000              .5          2
        1002         81       2000000               1          1
        1003          5        500000               1          1
        1004         54        500000               1          1
        1005         74       1000000              .5          2
        1005         19       2000000               1          1
        1006          7       2000000               1          1
        1053         94        500000              .5          2
        1053         22       1000000               1          1
        1054         23        500000               1          1
        1055          4       1000000              .5          2
        1055         99       2000000               1          1
        1056         52       1000000               1          1
        1057          9       1000000               1          1
        1059         98        500000               1          1
        1059         46        500000               1          1

92 rows selected.

A00268252_SQL>
```

This query calculates the cumulative distribution and rank of project values for each scientist, offering insights into the relative position of each project within their portfolio.

- **Cume_Dist()** computes the cumulative distribution of project values, representing the percentage of projects with a value less than or equal to the current project, ordered by Project_Value within each scientist's portfolio
- **Rank()** assigns a rank to each project based on Project_Value in descending order, ensuring higher-value projects receive a higher rank, with ties receiving different ranks
- **Over (Partition By Scientist_Id Order By Project_Value)** ensures that both the cumulative distribution and rank are computed within each scientist's project set, allowing comparisons across a scientist's projects
- The **Where** condition filters out projects with non-positive values, ensuring that only valid projects are considered in the analysis

3. Compute the percentage rank of each project value for a scientist

Cl scr
Select
Scientist_Id, Project_Id, Project_Value,
Percent_Rank() Over (Partition By Scientist_Id Order By Project_Value) As Percent_Rank,
Corr(Project_Value, Project_Value) Over (Partition By Scientist_Id) As Project_Correlation
From Research_Project Where Project_Value > 0;

```
A00268252_SQL>      Select
  2     Scientist_Id, Project_Id, Project_Value,
  3     Percent_Rank() Over (Partition By Scientist_Id Order By Project_Value) As Percent_Rank,
  4     Corr(Project_Value, Project_Value) Over (Partition By Scientist_Id) As Project_Correlation
  5        From Research_Project Where Project_Value > 0;

SCIENTIST_ID PROJECT_ID PROJECT_VALUE PERCENT_RANK PROJECT_CORRELATION
------------ ---------- ------------- ------------ --------------------
        1001         14        500000            0
        1002         41        500000            0                    1
        1002         81       2000000            1                    1
        1003          5        500000            0
        1004         54        500000            0
        1005         74       1000000            0                    1
        1005         19       2000000            1                    1
        1006          7       2000000            0
        1007         80        500000            0                    1
        1007         42       2000000            1                    1
        1008         26        500000            0
        1008         88        500000            0
        1051         60       1000000            0
        1052         86        500000            0                    1
        1052         53       2000000            1                    1
        1053         94        500000            0                    1
        1053         22       1000000            1                    1
        1054         23        500000            0
        1055          4       1000000            0                    1
        1055         99       2000000            1                    1
        1056         52       1000000            0
        1057          9       1000000            0
        1059         98        500000            0
        1059         46        500000            0

92 rows selected.

A00268252_SQL>
```

This query calculates the percentage rank and correlation of project values for each scientist.

- **Percent_Rank()** computes the percentage rank of each project's value, representing its relative position within a scientist's portfolio, ordered by Project_Value
- **Over (Partition By Scientist_Id Order By Project_Value)** groups the projects by scientist and orders them by project value, enabling the percentage rank calculation
- **Corr(Project_Value, Project_Value)** calculates the correlation of each project's value with itself, which always returns 1, indicating perfect self-correlation
- The **Where** condition ensures that only projects with positive values are included in the analysis

4. Find the scientist with the highest total project value

```
Cl scr
Select
   Scientist_Id,
   Sum(Project_Value) As Total_Value,
   Rank() Over (Order By Sum(Project_Value) Desc) As Overall_Rank
From Research_Project
Group By Scientist_Id
Having Sum(Project_Value) > 0;
```

```
A00268252_SQL>Select
  2      Scientist_Id,
  3      Sum(Project_Value) As Total_Value,
  4      Rank() Over (Order By Sum(Project_Value) Desc) As Overall_Rank
  5  From Research_Project
  6  Group By Scientist_Id
  7  Having Sum(Project_Value) > 0;

SCIENTIST_ID TOTAL_VALUE OVERALL_RANK
------------ ----------- ------------
        1048     4000000            1
        1020     3000000            2
        1038     3000000            2
        1013     3000000            2
        1009     3000000            2
        1015     3000000            2
        1021     1000000           29
        1025     1000000           29
        1045     1000000           29
        1014     1000000           29
        1008     1000000           29
        1041     1000000           29
        1049     1000000           29
        1044     1000000           29
        1057     1000000           29
        1056     1000000           29
        1001      500000           46
        1047      500000           46
        1004      500000           46
        1040      500000           46
        1017      500000           46
        1039      500000           46
        1003      500000           46
        1054      500000           46
        1043      500000           46
        1035      500000           46
        1033      500000           46

56 rows selected.

A00268252_SQL>
```

This query identifies the scientist with the highest total project value.

- **Sum(Project_Value)** calculates the total project value for each scientist, grouped by Scientist_Id
- **Rank()** assigns a rank based on the total project value, with the highest total receiving rank 1, ordered by **Sum(Project_Value) Desc**
- **Over (Order By Sum(Project_Value) Desc)** applies the ranking function in descending order of project value
- The **Having** condition ensures only scientists with a positive total project value are considered

5. Identify the moving average of project values for each scientist

```
Cl scr
Select
    Scientist_Id, Project_Id, Project_Value,
    Avg(Project_Value) Over (Partition By Scientist_Id Order By Project_Id Rows Between 3 Preceding And Current Row) As Moving_Avg,
    Cume_Dist() Over (Partition By Scientist_Id Order By Project_Value) As Cumulative_Dist
From Research_Project
Where Project_Value > 0;
```

```
A00268252_SQL>Select
  2     Scientist_Id, Project_Id, Project_Value,
  3     Avg(Project_Value) Over (Partition By Scientist_Id Order By Project_Id Rows Between 3 Preceding And Current Row) As Moving_Avg,
  4     Cume_Dist() Over (Partition By Scientist_Id Order By Project_Value) As Cumulative_Dist
  5  From Research_Project
  6  Where Project_Value > 0;

SCIENTIST_ID PROJECT_ID PROJECT_VALUE MOVING_AVG CUMULATIVE_DIST
------------ ---------- ------------- ---------- ---------------
        1001         14        500000     500000               1
        1002         41        500000     500000              .5
        1002         81       2000000    1250000               1
        1003          5        500000     500000               1
        1004         54        500000     500000               1
        1005         19       2000000    2000000               1
        1005         74       1000000    1500000              .5
        1006          7       2000000    2000000               1
        1007         42       2000000    2000000               1
          1052          86         500000    1250000                .5
          1053          22        1000000    1000000                 1
          1053          94         500000     750000                .5
          1054          23         500000     500000                 1
          1055           4        1000000    1000000                .5
          1055          99        2000000    1500000                 1
          1056          52        1000000    1000000                 1
          1057           9        1000000    1000000                 1
          1059          46         500000     500000                 1
          1059          98         500000     500000                 1

92 rows selected.

A00268252_SQL>
```

This query calculates the moving average and cumulative distribution of project values for each scientist.

- **Avg()** computes the moving average of project values, considering the current project and the three preceding ones using **Over (Partition By Scientist_Id Order By Project_Id Rows Between 3 Preceding And Current Row)**
- **Cume_Dist()** calculates the cumulative distribution, representing the proportion of projects with a value less than or equal to the current project
- The **Where Project_Value** > 0 condition ensures that only projects with positive values are included

Statistical Functions

1. Find the scientist's rank based on the sum of project values

```
Cl scr
Select
    Scientist.Scientist_Id,
    (Select Sum(Project_Value) From Research_Project Where Scientist_Id = Scientist.Scientist_Id) As Total_Value,
    Rank() Over (Order By (Select Sum(Project_Value) From Research_Project Where Scientist_Id = Scientist.Scientist_Id) Desc) As Rank,
    Dense_Rank() Over (Order By (Select Sum(Project_Value) From Research_Project Where Scientist_Id = Scientist.Scientist_Id) Desc) As Dense_Rank
From Scientist;
```

```
A00268252_SQL>Select
  2     Scientist.Scientist_Id,
  3     (Select Sum(Project_Value) From Research_Project Where Scientist
_Id = Scientist.Scientist_Id) As Total_Value,
  4     Rank() Over (Order By (Select Sum(Project_Value) From Research_P
roject Where Scientist_Id = Scientist.Scientist_Id) Desc) As Rank,
  5     Dense_Rank() Over (Order By (Select Sum(Project_Value) From Rese
arch_Project Where Scientist_Id = Scientist.Scientist_Id) Desc) As Dense
_Rank
  6  From Scientist;

SCIENTIST_ID TOTAL_VALUE       RANK DENSE_RANK
------------ ----------- ---------- ----------
        1048     4000000          1          1
        1005     3000000          2          2
        1023     3000000          2          2
        1013     3000000          2          2
        1018     3000000          2          2
        1055     3000000          2          2
        1015     3000000          2          2
        1009     3000000          2          2
        1038     3000000          2          2



             1050     1000000         29          6
             1024     1000000         29          6
             1027     1000000         29          6
             1028     1000000         29          6
             1059     1000000         29          6
             1057     1000000         29          6
             1047      500000         46          7
             1039      500000         46          7
             1040      500000         46          7
             1003      500000         46          7
             1004      500000         46          7
             1017      500000         46          7
             1033      500000         46          7
             1043      500000         46          7
             1035      500000         46          7
             1054      500000         46          7
             1001      500000         46          7

     56 rows selected.

     A00268252_SQL>
```

This query ranks scientists based on the sum of their project values.

- A subquery calculates the total project value for each scientist using **Sum(Project_Value)**
- **Rank()** assigns a rank based on total project value in descending order
- **Dense_Rank()** assigns a rank without gaps for ties
- Both rankings are applied using **Order By** to order scientists by total project value

2. Find the scientist with the highest value project within each discipline

```
Cl Scr
Select
        Discipline, Scientist_Id, Project_Id, Project_Value,
        First_Value(Project_Value) Over (Partition By Discipline Order By Project_Value Desc)
As First_Project_Value From Research_Project
Where Project_Value > 0;
```

```
A00268252_SQL>Select
  2     Discipline, Scientist_Id, Project_Id, Project_Value,
  3     First_Value(Project_Value) Over (Partition By Discipline Order By Project_Value Desc) As First_Project_Value From Research_Project
  4  Where Project_Value > 0;

DISCIPLINE                                         SCIENTIST_ID PROJECT_ID PROJECT_VALUE FIRST_PROJECT_VALUE
-------------------------------------------------- ------------ ---------- ------------- -------------------
Biology                                                    1038          3       3000000             3000000
Biology                                                    1026         40       2000000             3000000
Biology                                                    1036         38       1000000             3000000
Biology                                                    1036         68       1000000             3000000
Biology                                                    1031         15       1000000             3000000
Biology                                                    1057          9       1000000             3000000
Biology                                                    1028         62        500000             3000000
Biology                                                    1044         79        500000             3000000
Biology                                                    1026         76        500000             3000000
Biology                                                    1031         85        500000             3000000
Biology                                                    1044         20        500000             3000000
Biology                                                    1028         24        500000             3000000
Biology                                                    1001         14        500000             3000000
Biology                                                    1039         34        500000             3000000
Chemistry                                                  1005         19       2000000             2000000
Chemistry                                                  1055         99       2000000             2000000

Physics                                                    1011         17       1000000             2000000
Physics                                                    1029         82       1000000             2000000
Physics                                                    1019         93       1000000             2000000
Physics                                                    1054         23        500000             2000000
Physics                                                    1034         39        500000             2000000
Physics                                                    1021         55        500000             2000000
Physics                                                    1045          2        500000             2000000
Physics                                                    1052         86        500000             2000000
Physics                                                    1021         70        500000             2000000
Physics                                                    1035         33        500000             2000000
Physics                                                    1034         75        500000             2000000
Physics                                                    1045         96        500000             2000000
Physics                                                    1029          8        500000             2000000
Physics                                                    1011         71        500000             2000000
Physics                                                    1047         25        500000             2000000

92 rows selected.

A00268252_SQL>
```

This query retrieves the first project value within each discipline based on the highest project value.

- **First_Value()** selects the first project value with the highest amount for each discipline using Over (Partition By Discipline Order By Project_Value Desc)
- **Partition By Discipline** ensures the calculation is performed separately for each discipline
- **Order By Project_Value Desc** sorts projects in descending order, so the project with the highest value is selected first
- **Where Project_Value > 0** filters out projects with non-positive values

The query returns the discipline, scientist ID, project ID, project value, and the highest project value within each discipline, offering insights into the most significant project in each field.

3. Identify the moving average of project values for each scientist over a 5-project window

```
Cl Scr
Select
    Scientist_Id, Project_Id, Project_Value,
    Avg(Project_Value) Over (Partition By Scientist_Id Order By Project_Id Rows Between 4
Preceding And Current Row) As Moving_Avg,
    Lead(Project_Value, 1, 0) Over (Partition By Scientist_Id Order By Project_Id) As
Next_Project_Value
From Research_Project
Where Project_Value > 0;
```

```
A00268252_SQL>Select
  2     Scientist_Id, Project_Id, Project_Value,
  3     Avg(Project_Value) Over (Partition By Scientist_Id Order By Project_Id Rows Between 4 Preceding And Current Row) As Moving_Avg,
  4     Lead(Project_Value, 1, 0) Over (Partition By Scientist_Id Order By Project_Id) As Next_Project_Value
  5  From Research_Project
  6  Where Project_Value > 0;

SCIENTIST_ID PROJECT_ID PROJECT_VALUE MOVING_AVG NEXT_PROJECT_VALUE
------------ ---------- ------------- ---------- ------------------
        1001         14        500000     500000                  0
        1002         41        500000     500000            2000000
        1002         81       2000000    1250000                  0
        1003          5        500000     500000                  0
        1004         54        500000     500000                  0
        1005         19       2000000    2000000            1000000
        1005         74       1000000    1500000                  0
        1006          7       2000000    2000000                  0
        1007         42       2000000    2000000             500000
        1007         80        500000    1250000                  0
        1008         26        500000     500000             500000

         1050          61        500000     500000                  0
         1051          60       1000000    1000000            1000000
         1051          84       1000000    1000000                  0
         1052          53       2000000    2000000             500000
         1052          86        500000    1250000                  0
         1053          22       1000000    1000000             500000
         1053          94        500000     750000                  0
         1054          23        500000     500000                  0
         1055           4       1000000    1000000            2000000
         1055          99       2000000    1500000                  0
         1056          52       1000000    1000000                  0
         1057           9       1000000    1000000                  0
         1059          46        500000     500000             500000
         1059          98        500000     500000                  0

92 rows selected.

A00268252_SQL>
```

This query calculates the moving average of project values for each scientist over a 5-project window.

- **Avg()** computes the moving average for each project, considering the current and the four preceding projects using Over (Partition By Scientist_Id Order By Project_Id Rows Between 4 Preceding And Current Row)
- **Lead()** retrieves the next project value within the same scientist's projects, defaulting to 0 if no subsequent project exists
- **Where Project_Value > 0** filters out non-positive project values

4. Rank projects within each discipline and find the cumulative distribution

```
Cl Scr
Select
    Discipline, Project_Id, Project_Value,
    Rank() Over (Partition By Discipline Order By Project_Value Desc) As Rank,
    Cume_Dist() Over (Partition By Discipline Order By Project_Value) As Cumulative_Dist,
    Count(*) Over (Partition By Discipline) As Project_Count
From Research_Project;
```

```
A00268252_SQL>Select
  2     Discipline, Project_Id, Project_Value,
  3     Rank() Over (Partition By Discipline Order By Project_Value Desc) As Rank,
  4     Cume_Dist() Over (Partition By Discipline Order By Project_Value) As Cumulative_Dist,
  5     Count(*) Over (Partition By Discipline) As Project_Count
  6  From Research_Project;

DISCIPLINE                                         PROJECT_ID PROJECT_VALUE       RANK CUMULATIVE_DIST PROJECT_COUNT
-------------------------------------------------- ---------- ------------- ---------- --------------- -------------
Biology                                                     3       3000000          1               1            14
Biology                                                    40       2000000          2      .928571429            14
Biology                                                    15       1000000          3      .857142857            14
Biology                                                     9       1000000          3      .857142857            14
Biology                                                    68       1000000          3      .857142857            14
Biology                                                    38       1000000          3      .857142857            14
Biology                                                    76        500000          7      .571428571            14
Biology                                                    14        500000          7      .571428571            14
Biology                                                    79        500000          7      .571428571            14
Biology                                                    24        500000          7      .571428571            14
Biology                                                    34        500000          7      .571428571            14



Physics                                                    71        500000         13              .5            24
Physics                                                    70        500000         13              .5            24
Physics                                                     8        500000         13              .5            24
Physics                                                    55        500000         13              .5            24
Physics                                                    39        500000         13              .5            24
Physics                                                     2        500000         13              .5            24
Physics                                                    23        500000         13              .5            24
Physics                                                    96        500000         13              .5            24
Physics                                                    86        500000         13              .5            24
Physics                                                    75        500000         13              .5            24
Physics                                                    33        500000         13              .5            24

92 rows selected.

A00268252_SQL>
```

This query ranks projects within each discipline and computes their cumulative distribution.

- **Rank()** assigns a rank to each project within its discipline, ordered by project value in descending order using Over (Partition By Discipline Order By Project_Value Desc)
- **Cume_Dist()** calculates the cumulative distribution, representing the proportion of projects with a value less than or equal to the current project within each discipline
- **Count()** counts the total number of projects within each discipline using Over (Partition By Discipline)

Additional Windowing/Partitioning Functions

1. Calculate the population covariance between two variables

```
Cl Scr
Select
   Research_Project.Discipline,
   Covar_Pop(Scientist.Num_Publications, Sum(Research_Project.Project_Value))
      Over (Partition By Research_Project.Discipline) As Population_Covariance
From
   Scientist, Research_Project
Where
   Scientist.Scientist_Id = Research_Project.Scientist_Id
Group By
   Research_Project.Discipline, Scientist.Num_Publications;
```

```
A00268252_SQL>SELECT
  2      Research_Project.Discipline,
  3      COVAR_POP(Scientist.Num_Publications, SUM(Research_Project.Project_Value))
  4          OVER (PARTITION BY Research_Project.Discipline) AS Population_Covariance
  5  FROM
  6      Scientist, Research_Project
  7  WHERE
  8      Scientist.Scientist_Id = Research_Project.Scientist_Id
  9  GROUP BY
 10      Research_Project.Discipline, Scientist.Num_Publications;

DISCIPLINE           POPULATION_COVARIANCE
-------------------- ---------------------
Biology                              920000
Biology                              920000
Biology                              920000
Biology                              920000
Biology                              920000
Chemistry                           1820000
Chemistry                           1820000
Chemistry                           1820000
Chemistry                           1820000
Chemistry                           1820000
Computer Science                     750000
Computer Science                     750000
Computer Science                     750000
Computer Science                     750000
Computer Science                     750000
Computer Science                     750000
Microbiology                        3960000
Microbiology                        3960000
Microbiology                        3960000
Microbiology                        3960000
Microbiology                        3960000
Physics                          1166666.67
Physics                          1166666.67
Physics                          1166666.67
Physics                          1166666.67
Physics                          1166666.67
Physics                          1166666.67

27 rows selected.

A00268252_SQL>
```

This query ranks projects within each discipline and computes their cumulative distribution.

- **Rank()** assigns a rank to each project within its discipline, ordered by project value in descending order using Over (Partition By Discipline Order By Project_Value Desc)
- **Cume_Dist()** calculates the cumulative distribution, representing the proportion of projects with a value less than or equal to the current project within each discipline
- **Count()** counts the total number of projects within each discipline using Over (Partition By Discipline)

2. Compute the ratio to report for each project

This query calculates the ratio of each project value relative to the total for each scientist.

```
Cl scr
Select
   Scientist_Id, Project_Id, Project_Value,
   Ratio_To_Report(Project_Value) Over (Partition By Scientist_Id) As Value_Percentage
From Research_Project;
```

```
A00268252_SQL>Select
  2      Scientist_Id, Project_Id, Project_Value,
  3      Ratio_To_Report(Project_Value) Over (Partition By Scientist_Id) As Value_Percentage
  4  From Research_Project;

SCIENTIST_ID PROJECT_ID PROJECT_VALUE VALUE_PERCENTAGE
------------ ---------- ------------- ----------------
        1001         14        500000                1
        1002         81       2000000               .8
        1002         41        500000               .2
        1003          5        500000                1
        1004         54        500000                1
        1005         74       1000000        .333333333
        1005         19       2000000        .666666667
        1006          7       2000000                1
        1007         42       2000000               .8
        1007         80        500000               .2
        1008         26        500000               .5
        1053         22       1000000        .666666667
        1053         94        500000        .333333333
        1054         23        500000                1
        1055          4       1000000        .333333333
        1055         99       2000000        .666666667
        1056         52       1000000                1
        1057          9       1000000                1
        1059         46        500000               .5
        1059         98        500000               .5

92 rows selected.

A00268252_SQL>
```

This query calculates the percentage of each project's value relative to the total project value for each scientist.

- **Ratio_To_Report()** computes the ratio of each project's value to the total value of all projects for the given scientist
- The calculation is partitioned by Scientist_Id to ensure it's done for each scientist individually
- The query returns Scientist_Id, Project_Id, Project_Value, and the calculated Value_Percentage

3. **Calculate Moving Average Income for 30 Days Before and After Birthdate**

This query is designed to calculate the moving average income over a 30-day window before and after each scientist's birthdate. Specifically, it calculates the average income for scientists within a 30-day range before and after each individual's birthdate, providing insights into trends or shifts in income relative to a scientist's age or other time-based factors.

```
Cl scr
Select Surname, Date_Born, Date_Born - 30 “30DyEar”, Date_Born + 30 “30DyLat”, Income,
        Avg(Income) Over (Order By Date_Born Asc Range 30 Preceding)
Av_Income_30Dy_Before,
        Avg(Income) Over (Order By Date_Born Desc Range 30 Preceding)
Av_Income_30Dy_After
From Scientist
Order By Date_Born;
```

```
A00268252_SQL>Select Surname, Date_Born, Date_Born - 30 “30DyEar”, Date_Born + 30 “30DyLat”, Income,
  2  Avg(Income) Over (Order By Date_Born Asc Range 30 Preceding) Av_Income_30Dy_Before,
  3  Avg(Income) Over (Order By Date_Born Desc Range 30 Preceding) Av_Income_30Dy_After
  4  From Scientist
  5  Order By Date_Born;

SURNAME                                            DATE_BORN 30DyEar   30DyLat      INCOME AV_INCOME_30DY_BEFORE AV_INCOME_30DY_AFTER
-------------------------------------------------- --------- --------- --------- ---------- --------------------- --------------------
Li                                                 18-JAN-91 19-DEC-90 17-FEB-91       7917                  7917               7604.5
Oliveira                                           14-FEB-91 15-JAN-91 16-MAR-91       7292                7604.5                 7292
Gallagher                                          01-MAY-91 01-APR-91 31-MAY-91       7292                  7292               4812.5
Murphy                                             13-MAY-91 13-APR-91 12-JUN-91       2333                4812.5                 2333
Cooper                                             11-DEC-91 11-NOV-91 10-JAN-92       8125                  8125                 8125
Costa                                              13-JAN-92 14-DEC-91 12-FEB-92       7292                  7292                 7292
Walker                                             25-JAN-92 26-DEC-91 24-FEB-92       7292                  7292                 7292
Silva                                              29-MAR-92 28-FEB-92 28-APR-92       9375                  9375                 9375
Wei                                                30-APR-92 31-MAR-92 30-MAY-92       4167                  4167                 4167
Santos                                             17-JUN-92 18-MAY-92 17-JUL-92       1944                  1944                 4826
Ali                                                15-JUL-92 15-JUN-92 14-AUG-92       7708                  4826                 7708
Martins                                            23-JUL-92 23-JUN-92 22-AUG-92       7708                  7708                 7708
Fitzgerald                                         16-OCT-92 16-SEP-92 15-NOV-92       7500                  7500                 7500
Brown                                              28-NOV-92 29-OCT-92 28-DEC-92       6667                  6667                 6667
Syafiq                                             11-JAN-93 12-DEC-92 10-FEB-93       9271                  9271                 9271
Clark                                              26-JUL-93 26-JUN-93 25-AUG-93       2500                  2500               2333.5
Souza                                              28-JUL-93 28-JUN-93 27-AUG-93                             2500                 2167
Silva                                              15-AUG-93 16-JUL-93 14-SEP-93       2167                2333.5                 2167


 Wei                                               14-DEC-97 14-NOV-97 13-JAN-98       5833                  5833                 5833
 Li Jia                                            25-JAN-98 26-DEC-97 24-FEB-98       1667                  1667                 1667
 Silva                                             05-JUN-98 06-MAY-98 05-JUL-98       9861                  9861          9259.33333
 Almeida                                           23-JUN-98 24-MAY-98 23-JUL-98       9375                  9618                 8958.5
 Fang                                              26-JUN-98 27-MAY-98 26-JUL-98       8542          9259.33333                 8542
 Lima                                              09-OCT-98 09-SEP-98 08-NOV-98       2222                  2222               4496.5
 Xiao                                              27-OCT-98 27-SEP-98 26-NOV-98       6771                4496.5               4635.5
 Wei                                               22-NOV-98 23-OCT-98 22-DEC-98       2500                4635.5                 2500
 Ibrahim                                           04-JAN-99 05-DEC-98 03-FEB-99       9375                  9375                 9375
 Doyle                                             21-APR-99 22-MAR-99 21-MAY-99       2361                  2361                 5868
 Pereira                                           10-MAY-99 10-APR-99 09-JUN-99       9375                  5868               5937.5
 Ahmad                                             22-MAY-99 22-APR-99 21-JUN-99       2500                5937.5                 2500

 56 rows selected.

 A00268252_SQL>
```

This query calculates the average income of scientists over a 30-day window before and after each scientist's birthdate, along with the surname, birthdate, and income.

- The **Avg(Income)** function is used with a windowing clause: Range 30 Preceding for calculating the moving average of income for the 30 days preceding the birthdate, and **Range 30 Preceding (in reverse order)** for the 30 days following the birthdate.
- The **Date_Born - 30 and Date_Born + 30** create a range of 30 days before and after the birthdate.
- The **Order By Date_Born** ensures the results are sorted chronologically by birthdate, providing a temporal view of the scientists' income trends.

4. Calculate Cumulative Income and Sequence for Each Scientist Within Their University

This query calculates cumulative income totals and assigns a row sequence number to each scientist within their university. It also computes the total income per university and individual cumulative income. The results are grouped by university and ordered by surname for clarity.

```
Cl scr
Break On University Skip 1
Select Surname, University, Income,
        Sum(Income) Over
                (Order By University, Surname) Income_Total,
        Sum(Income) Over
                (Partition By University
                Order By Surname) Scientist_Total,
        Row_Number() Over
                (Order By Surname) Seq
From Scientist
Order By University, Surname;
```

```
A0068252_SQL>Break On University Skip 1
A0068252_SQL>Select Surname, University, Income,
  2     Sum(Income) Over
  3             (Order By University, Surname) Income_Total,
  4     Sum(Income) Over
  5             (Partition By University
  6             Order By Surname) Scientist_Total,
  7  Row_Number() Over
  8  (Order By Surname) Seq
  9  From Scientist
 10  Order By University, Surname;

SURNAME          UNIVERSITY              INCOME INCOME_TOTAL SCIENTIST_TOTAL        SEQ
---------------- ----------------------- ------ ------------ --------------- ----------
Noor             Cardiff University        9375         9375            9375         33
Parker                                      556         9931            9931         36
Walker                                     7292        17223           17223         49
Wei                                        5833        23056           23056         51

Cooper           Swansea University        8125        31181            8125          9
Costa                                       556        39029           15973         12
Costa                                      7292        39029           15973         10
Murphy                                      833        39862           16806         32
```

```
Xiao                                       6771       277189           62327         55

Ahmad            University of Oxford      2500       279689            2500          1
Harris                                     2500       282189            5000         19
Oliveira                                   9375       298856           21667         35
Oliveira                                   7292       298856           21667         34
Wei                                        2500       301356           24167         52

56 rows selected.

A0068252_SQL>
```

This query calculates the cumulative income totals for scientists grouped by university and provides a row number for each scientist.

- **Break On University Skip 1**: This clause groups the results by University and skips the first row after each university grouping to create a break in the output.
- **Sum(Income) Over (Order By University, Surname):** This computes the cumulative total income for each scientist, ordered by University and Surname.
- **Sum(Income) Over (Partition By University Order By Surname):** This calculates the total income for all scientists within the same university, ordered by Surname.
- **Row_Number() Over (Order By Surname**): This assigns a sequential row number to each scientist, ordered by Surname, for unique identification.

The query returns the Surname, University, Income, and the cumulative income calculations, sorted by University and Surname.

5. Calculate Cumulative Income and Row Sequence for Each Scientist

This query calculates the cumulative income totals and assigns a row sequence number to each scientist, based on their scientist ID and surname. It also computes the total income per scientist and the cumulative income across all scientists.

```
Cl scr
Set Autotrace Traceonly
Select Surname, Scientist_ID, Income,
        Sum(Income) Over
                (Order By Scientist_ID, Surname) Income_Total,
        Sum(Income) Over
                (Partition By Scientist_ID
                Order By Surname) Scientist_Total,
        Row_Number() Over
                (Partition By Scientist_ID
                Order By Surname) Seq
From Scientist
Order By University, Surname;
```

```
SQL> Set Autotrace Traceonly
SQL> Select Surname, Scientist_ID, Income,
  2     Sum(Income) Over
  3             (Order By Scientist_ID, Surname) Income_Total,
  4     Sum(Income) Over
  5             (Partition By Scientist_ID
  6             Order By Surname) Scientist_Total,
  7  Row_Number() Over
  8  (Partition By Scientist_ID
  9  Order By Surname) Seq
 10  From Scientist
 11  Order By University, Surname;

56 rows selected.

Elapsed: 00:00:00.00

Execution Plan
----------------------------------------------------------
Plan hash value: 853376216

--------------------------------------------------------------------------------
| Id  | Operation           | Name      | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT    |           |    56 |  2128 |     4  (50)| 00:00:01 |
|   1 |  SORT ORDER BY      |           |    56 |  2128 |     4  (50)| 00:00:01 |
|   2 |   WINDOW SORT       |           |    56 |  2128 |     4  (50)| 00:00:01 |
|   3 |    TABLE ACCESS FULL| SCIENTIST |    56 |  2128 |     2   (0)| 00:00:01 |
--------------------------------------------------------------------------------


Statistics
----------------------------------------------------------
          0  recursive calls
          0  db block gets
          2  consistent gets
          0  physical reads
          0  redo size
       3004  bytes sent via SQL*Net to client
        557  bytes received via SQL*Net from client
          5  SQL*Net roundtrips to/from client
          2  sorts (memory)
          0  sorts (disk)
         56  rows processed

SQL>
```

This query is designed to retrieve the Surname, Scientist_ID, Income, and additional calculated columns, using various window functions.

1. **Set Autotrace Traceonly**: Enables execution plan tracing without returning query results, allowing for performance analysis
2. **Cumulative Income Total (Income_Total)**: Computes the running total of income for each scientist, ordered by scientist ID and surname.
3. **Total Income by Scientist (Scientist_Total)**: Calculates the total income for each scientist, partitioned by scientist ID and ordered by surname.
4. **Row Sequence (Seq)**: Assigns a unique sequential number to each scientist within their scientist ID group, ordered by surname.

Advanced SQL Queries with Windowing, Partitioning, and Statistical Functions

1. Calculate Year-over-Year Change in Project Value per Scientist

This query calculates the year-over-year change in project values, including moving averages, rank, and cumulative distribution.

```
Cl scr
Select
    Scientist_Id, Project_Id, Project_Value,
    Start_Year As Year, Lag(Project_Value) Over (Partition By Scientist_Id Order By Start_Year) As Prev_Year_Value,
    Project_Value - Lag(Project_Value) Over (Partition By Scientist_Id Order By Start_Year) As Yoy_Change,
    Sum(Project_Value) Over (Partition By Scientist_Id Order By Start_Year Rows Between 1 Preceding And Current Row) As Rolling_Sum,
    Rank() Over (Partition By Scientist_Id Order By Start_Year Desc, Project_Id) As Year_Rank,
    Cume_Dist() Over (Partition By Scientist_Id Order By Project_Value Desc) As Cumulative_Distribution
From Research_Project Where Project_Value > 0
Order By Scientist_Id, Start_Year;
```

```
A00268252_SQL>Select
  2       Scientist_Id, Project_Id, Project_Value,
  3       Start_Year As Year, Lag(Project_Value) Over (Partition By Scientist_Id Order By Start_Year) As Prev_Year_Value,
  4       Project_Value - Lag(Project_Value) Over (Partition By Scientist_Id Order By Start_Year) As Yoy_Change,
  5       Sum(Project_Value) Over (Partition By Scientist_Id Order By Start_Year Rows Between 1 Preceding And Current Row) As Rolling_Sum,
  6       Rank() Over (Partition By Scientist_Id Order By Start_Year Desc, Project_Id) As Year_Rank,
  7       Cume_Dist() Over (Partition By Scientist_Id Order By Project_Value Desc) As Cumulative_Distribution
  8  From Research_Project Where Project_Value > 0
  9  Order By Scientist_Id, Start_Year;

SCIENTIST_ID PROJECT_ID PROJECT_VALUE       YEAR PREV_YEAR_VALUE YOY_CHANGE ROLLING_SUM  YEAR_RANK CUMULATIVE_DISTRIBUTION
------------ ---------- ------------- ---------- --------------- ---------- ----------- ---------- ------------------------
        1001         14        500000       2022                               500000          1                        1
        1002         81       2000000       2018                              2000000          2                       .5
        1002         41        500000       2018         2000000   -1500000     2500000          1                        1
        1003          5        500000       2022                               500000          1                        1
        1004         54        500000       2018                               500000          1                        1
        1005         19       2000000       2020                              2000000          1                       .5
        1005         74       1000000       2020         2000000   -1000000     3000000          2                        1
        1006          7       2000000       2018                              2000000          1                        1
        1007         42       2000000       2021                              2000000          1                       .5
        1007         80        500000       2021         2000000   -1500000     2500000          2                        1
        1008         26        500000       2021                               500000          1                        1
        1008         88        500000       2021          500000          0     1000000          2                        1
        1009        100       2000000       2021                              2000000          2                       .5


        1052         53       2000000       2020                              2000000          1                       .5
        1052         86        500000       2020         2000000   -1500000     2500000          2                        1
        1053         22       1000000       2021                              1000000          1                       .5
        1053         94        500000       2021         1000000    -500000     1500000          2                        1
        1054         23        500000       2022                               500000          1                        1
        1055         99       2000000       2018                              2000000          2                       .5
        1055          4       1000000       2018         2000000   -1000000     3000000          1                        1
        1056         52       1000000       2019                              1000000          1                        1
        1057          9       1000000       2021                              1000000          1                        1
        1059         46        500000       2018                               500000          1                        1
        1059         98        500000       2018          500000          0     1000000          2                        1

92 rows selected.

A00268252_SQL>
```

This query analyzes project values for each scientist, including year-on-year (YoY) changes, cumulative distribution, and rolling sums.

Query Breakdown:

- **Autotrace Traceonly:** Enables execution plan tracing for performance analysis.
- **Prev Year Value:**Uses Lag() to retrieve the previous year's project value
- **Year-over-Year Change (YoY Change):** Calculates the difference between current and previous year's project values
- **Rolling Sum:** Computes the cumulative sum of project values, considering the previous row
- **Year Rank:** Ranks projects by year with tie-breaking on project ID
- **Cumulative Distribution:** Calculates the cumulative distribution of project values by scientist ID

Explanation:

- **Blank Prev_Year_Value and YoY_Change:** Lag() returns Null for the first project, causing blanks
- **Year Rank Calculation:** Rank() ties on year and project ID, causing potential ranking issues
- **Cumulative Distribution:** Cume_Dist() shows constant values if project values are similar

Key Issues:

- **Null Values:** Expected for first projects
- **Year Rank:** Tie-breaking may cause ranking issues
- **Cumulative Distribution:** Minimal variation for similar project values

2. Find the Moving Average and Variability of Project Values per Scientist

This query calculates the moving average, standard deviation, and variability of project values per scientist.

```
Cl scr
Select
   Scientist_Id, Project_Id, Project_Value,
   Start_Year As Year,
   Avg(Project_Value) Over (Partition By Scientist_Id Order By Start_Year Rows Between 2
Preceding And Current Row) As Moving_Avg_3_Years,
   Stddev_Samp(Project_Value) Over (Partition By Scientist_Id Order By Start_Year Rows
Between 1 Preceding And Current Row) As Stddev_Moving,
   Lag(Project_Value, 1, 0) Over (Partition By Scientist_Id Order By Start_Year) As
Prev_Year_Value,
   Row_Number() Over (Partition By Scientist_Id Order By Start_Year Desc) As Row_Num,
   Var_Pop(Project_Value) Over (Partition By Scientist_Id) As Project_Variance
From Research_Project
Where Project_Value > 0;
```

```
A00268252_SQL>Select
  2      Scientist_Id, Project_Id, Project_Value,
  3      Start_Year As Year,
  4      Avg(Project_Value) Over (Partition By Scientist_Id Order By Start_Year Rows Between 2 Preceding And Current Row) As Moving_Avg_3_Years,
  5      Stddev_Samp(Project_Value) Over (Partition By Scientist_Id Order By Start_Year Rows Between 1 Preceding And Current Row) As Stddev_Moving,
  6      Lag(Project_Value, 1, 0) Over (Partition By Scientist_Id Order By Start_Year) As Prev_Year_Value,
  7      Row_Number() Over (Partition By Scientist_Id Order By Start_Year Desc) As Row_Num,
  8      Var_Pop(Project_Value) Over (Partition By Scientist_Id) As Project_Variance
  9  From Research_Project
 10  Where Project_Value > 0;

SCIENTIST_ID PROJECT_ID PROJECT_VALUE       YEAR MOVING_AVG_3_YEARS STDDEV_MOVING PREV_YEAR_VALUE    ROW_NUM PROJECT_VARIANCE
------------ ---------- ------------- ---------- ----------------- ------------- ---------------- ---------- ----------------
        1001         14        500000       2022            500000                             0          1                0
        1002         81       2000000       2018           2000000                             0          1       5.6250E+11
        1002         41        500000       2018           1250000    1060660.17         2000000          2       5.6250E+11
        1003          5        500000       2022            500000                             0          1                0
        1004         54        500000       2018            500000                             0          1                0
        1005         74       1000000       2020           1000000                             0          1       2.5000E+11
        1005         19       2000000       2020           1500000    707106.781         1000000          2       2.5000E+11
        1006          7       2000000       2018           2000000                             0          1                0
        1007         42       2000000       2021           2000000                             0          1       5.6250E+11


        1052         53       2000000       2020           1250000    1060660.17          500000          2       5.6250E+11
        1053         94        500000       2021            500000                             0          1       6.2500E+10
        1053         22       1000000       2021            750000    353553.391          500000          2       6.2500E+10
        1054         23        500000       2022            500000                             0          1                0
        1055         99       2000000       2018           2000000                             0          1       2.5000E+11
        1055          4       1000000       2018           1500000    707106.781         2000000          2       2.5000E+11
        1056         52       1000000       2019           1000000                             0          1                0
        1057          9       1000000       2021           1000000                             0          1                0
        1059         98        500000       2018            500000                             0          1                0
        1059         46        500000       2018            500000             0          500000          2                0

92 rows selected.

A00268252_SQL>
```

This query provides a detailed analysis of project values by calculating various statistical measures for each scientist.

Query Breakdown:

- **Autotrace Traceonly**: Enables execution plan tracing for performance analysis
- **Moving Average (Moving_Avg_3_Years)**: Uses Avg() to compute the average of the current and the previous two years' project values
- **Standard Deviation (StdDev_Moving)**: Uses Stddev_Samp() to calculate the standard deviation of project values for the current and previous year
- **Previous Year Value (Prev_Year_Value)**: Uses Lag() to retrieve the previous year's project value, defaulting to 0 if not available
- **Row Number (Row_Num)**: Assigns a unique sequential number to each project per scientist, ordered by year (descending)
- **Project Variance (Project_Variance)**: Uses Var_Pop() to calculate the population variance of project values for each scientist

Explanation:

- **Blank Prev_Year_Value**: The Lag() function returns NULL for the first year of data, leading to blank values for Prev_Year_Value.
- **Moving Average and StdDev**: Both Avg() and Stddev_Samp() calculate rolling metrics, using data from the current and previous years.
- **Year Rank**: The Row_Number() function ranks projects for each scientist, ordered by descending year. It will always give unique ranks.
- **Variance**: The Var_Pop() function calculates the population variance for the project values partitioned by scientist.

Key Issues:

- **Null Values**: Expected for the first project of each scientist (no previous year data).
- **Rolling Calculations**: Functions like Avg() and Stddev_Samp() depend on the window frame, which might not be populated for the first rows.
- **Year Rank**: No issues with ranking since Row_Number() generates unique ranks for each project.

3. **Calculate Yearly Ranking of Scientists by Total Project Value with Previous Year's Rank Comparison**

This query computes the total project value per year and compares rankings between years.

```
Cl scr
Select
        Scientist_Id, Total_Project_Value, Year_Rank,
        Nvl(Lag(Year_Rank, 1) Over (Order By Total_Project_Value Desc), 0) As
Prev_Year_Rank,
        Nvl(Lead(Year_Rank, 1) Over (Order By Total_Project_Value Desc), 0) As
Next_Year_Rank,
        Cume_Dist() Over (Order By Total_Project_Value Desc) As Cumulative_Distribution
From (
        Select
                Scientist_Id,
                Sum(Project_Value) As Total_Project_Value,
                Rank() Over (Order By Sum(Project_Value) Desc) As Year_Rank
        From Research_Project
        Group By Scientist_Id
)
Order By
Total_Project_Value Desc;
```

```
A00268252_SQL>Select
  2  Scientist_Id, Total_Project_Value, Year_Rank,
  3  Nvl(Lag(Year_Rank, 1) Over (Order By Total_Project_Value Desc), 0) As Prev_Year_Rank,
  4  Nvl(Lead(Year_Rank, 1) Over (Order By Total_Project_Value Desc), 0) As Next_Year_Rank,
  5  Cume_Dist() Over (Order By Total_Project_Value Desc) As Cumulative_Distribution
  6  From (
  7  Select
  8  Scientist_Id,
  9  Sum(Project_Value) As Total_Project_Value,
 10  Rank() Over (Order By Sum(Project_Value) Desc) As Year_Rank
 11  From Research_Project
 12  Group By Scientist_Id
 13  )
 14  Order By
 15  Total_Project_Value Desc;

SCIENTIST_ID TOTAL_PROJECT_VALUE  YEAR_RANK PREV_YEAR_RANK NEXT_YEAR_RANK CUMULATIVE_DISTRIBUTION
------------ ------------------- ---------- -------------- -------------- ----------------------
        1048             4000000          1              0              2              .017857143
        1013             3000000          2              1              2              .178571429
        1038             3000000          2              2              2              .178571429
        1020             3000000          2              2              2              .178571429
        1005             3000000          2              2              2              .178571429
        1009             3000000          2              2              2              .178571429
        1055             3000000          2              2              2              .178571429
        1023             3000000          2              2              2              .178571429
        1018             3000000          2              2              2              .178571429
        1015             3000000          2              2             11              .178571429
        1052             2500000         11              2             11                     .25
        1017              500000         46             29             46                       1
        1040              500000         46             46             46                       1
        1004              500000         46             46             46                       1
        1047              500000         46             46             46                       1
        1001              500000         46             46             46                       1
        1033              500000         46             46             46                       1
        1043              500000         46             46             46                       1
        1054              500000         46             46             46                       1
        1003              500000         46             46             46                       1
        1039              500000         46             46             46                       1
        1035              500000         46             46              0                       1

56 rows selected.

A00268252_SQL>
```

This query calculates various rankings and distributions based on the total project value for each scientist. It first calculates the total project value for each scientist by summing the Project_Value for all projects linked to each Scientist_Id. A ranking (Year_Rank) is then assigned based on these totals in descending order.

Query Breakdown:

- **Autotrace Traceonly**: Enables execution plan tracing to monitor SQL query execution.
- **Moving Average (Moving_Avg_3_Years)**: Calculates a rolling average of the current and previous two years' project values using Avg().
- **Standard Deviation (StdDev_Moving)**: Computes the sample standard deviation of project values for the current and previous year using Stddev_Samp().
- **Previous Year Value (Prev_Year_Value)**: Retrieves the previous year's project value using Lag(), defaulting to 0 if unavailable.
- **Row Number (Row_Num)**: Assigns a unique rank to each project per scientist based on year using Row_Number().
- **Project Variance (Project_Variance)**: Calculates the population variance of project values for each scientist using Var_Pop().

Explanation:

- **Blank Prev_Year_Value**: The Lag() function returns Null for the first project since there is no previous year to reference, which is replaced by Nvl() with 0.
- **Rolling Calculations**: Avg() and Stddev_Samp() calculate metrics over the current and prior years; incomplete data for the first rows may result in missing values.
- **Year Rank**: Row_Number() ensures unique ranks, guaranteeing no duplicates even with similar project values.
- **Variance**: Var_Pop() measures the spread of project values, helping to identify how much the values vary.

Key Issues:

- **Null Values:** The first project for each scientist will have a Null Prev_Year_Value which is handled using Nvl().
- **Rolling Calculations:** For the first few rows, window functions like Avg() and Stddev_Samp() might not return results due to insufficient data in the window.
- **Year Rank:** Row_Number() generates unique ranks, preventing any rank duplication for the same year or project value.

4. Calculate University-Level Research Project Statistics and Rankings

This query calculates various statistics for each university, including the count of distinct nationalities and scientists, the total project value, and university rankings. It also computes quartile rankings, previous year project values, and the highest qualification of the scientist with the most project value contribution.

```
Cl scr
Select
   University As Uni,
   Count(Distinct Nat) As Nationality_Count,
   Count(S_Id) As Scientist_Count,
   Sum(Tot_Proj_Val) As Tot_Proj_Val,
   Row_Number() Over (Order By Sum(Tot_Proj_Val) Desc) As University_Rank,
   Ntile(4) Over (Order By Sum(Tot_Proj_Val) Desc) As Quart,
   Lag(Sum(Tot_Proj_Val)) Over (Order By Sum(Tot_Proj_Val) Desc) As Prev_Proj_Val,
   Max(Highest_Qual) Keep (Dense_Rank First Order By Tot_Proj_Val Desc) As
Highest_Qual_Contrib,
   Sum(Running_Total_Proj_Val) As Running_Total_Proj_Val
From (
   Select
      S.Nationality As Nat,
      S.University,
      S.Scientist_Id As S_Id,
      Rp.Project_Value As Tot_Proj_Val,
      S.Highest_Qual,
      Sum(Rp.Project_Value) Over (Partition By S.University Order By Rp.Start_Year Rows
Between Unbounded Preceding And Current Row) As Running_Total_Proj_Val
   From Scientist S, Research_Project Rp
   Where Rp.Scientist_Id = S.Scientist_Id
)
Group By University
Order By University;
```

```
A00268252_SQL>Select
  2      University As Uni,
  3      Count(Distinct Nat) As Nationality_Count,
  4      Count(S_Id) As Scientist_Count,
  5      Sum(Tot_Proj_Val) As Tot_Proj_Val,
  6      Row_Number() Over (Order By Sum(Tot_Proj_Val) Desc) As University_Rank,
  7      Ntile(4) Over (Order By Sum(Tot_Proj_Val) Desc) As Quart,
  8      Lag(Sum(Tot_Proj_Val)) Over (Order By Sum(Tot_Proj_Val) Desc) As Prev_Proj_Val,
  9      Max(Highest_Qual) Keep (Dense_Rank First Order By Tot_Proj_Val Desc) As Highest_Qual_Contrib,
 10      Sum(Running_Total_Proj_Val) As Running_Total_Proj_Val
 11  From (
 12     Select
 13        S.Nationality As Nat,
 14        S.University,
 15        S.Scientist_Id As S_Id,
 16        Rp.Project_Value As Tot_Proj_Val,
 17        S.Highest_Qual,
 18        Sum(Rp.Project_Value) Over (Partition By S.University Order By Rp.Start_Year Rows Between Unbounded Preceding And Current Row) As Running_Total_Proj_Val
 19     From Scientist S, Research_Project Rp
 20     Where Rp.Scientist_Id = S.Scientist_Id
 21  )
 22  Group By University
 23  Order By University;

UNI                       NATIONALITY_COUNT SCIENTIST_COUNT TOT_PROJ_VAL UNIVERSITY_RANK      QUART PREV_PROJ_VAL HIGHEST_QU RUNNING_TOTAL_PROJ_VAL
------------------------- ----------------- --------------- ------------ --------------- ---------- ------------ --------- ---------------------
Cardiff University                        4               8      7000000               8          4      8500000 PhD                     29500000
Swansea University                        3               8      8500000               7          3      9000000 PhD                     41500000
TUS: Midlands Midwest                     4              18     12000000               4          2     12500000 PhD                    112500000
Trinity College Dublin                    3               5      5500000               9          4      7000000 PhD                     17500000
University College Cork                   3              14     14000000               2          1     16000000 PhD                     93000000
University of Cambridge                   4              11     12500000               3          1     14000000 Masters                 75500000
University of Edinburgh                   3               8      9500000               5          2     12000000 PhD                     46500000
University of Glasgow                     4              19     16000000               1          1              PhD                    182500000
University of Oxford                      4               9      9000000               6          3      9500000 PhD                     42000000

9 rows selected.

A00268252_SQL>
```

This query provides a comprehensive overview of university-level research projects, ranks them based on total project value, and offers additional insights into the distribution and performance of research projects across different universities.

Query Breakdown:

- **Distinct Nationality Count (Nationality_Count):** This counts the distinct nationalities of scientists per university
- **Scientist Count (Scientist_Count):** This counts the total number of scientists per university
- **Total Project Value (Tot_Proj_Val):** This sums the total project value for each university
- **University Rank (University_Rank):** This ranks universities based on their total project value in descending order using Row_Number()
- **Quartile Ranking (Quart):** Divides universities into four quartiles based on total project value using Ntile(4)
- **Previous Year Project Value (Prev_Proj_Val):** This calculates the previous year's project value for each university using Lag(), which helps in comparing the current year with the prior year's project total
- **Highest Qualification Contribution (Highest_Qual_Contrib):** This identifies the highest qualification associated with the university's highest project value using Max() and Dense_Rank()
- **Running Total of Project Value (Running_Total_Proj_Val):** This computes the cumulative project value for each university using a window function to calculate the sum of project values across all years, starting from the first project

Explanation:

- **Subquery Structure:** The inner query aggregates project values per scientist and computes the running total of project values for each university over time
- **Window Functions:** Window functions such as Lag(), Ntile(), and Row_Number() are used to calculate various statistical measures and rankings. These functions provide insights into the relative standing of each university in comparison to others
- **Grouping by University:** The outer query groups results by university, ensuring that all metrics (e.g., nationalities, total project values, ranks) are calculated at the university level

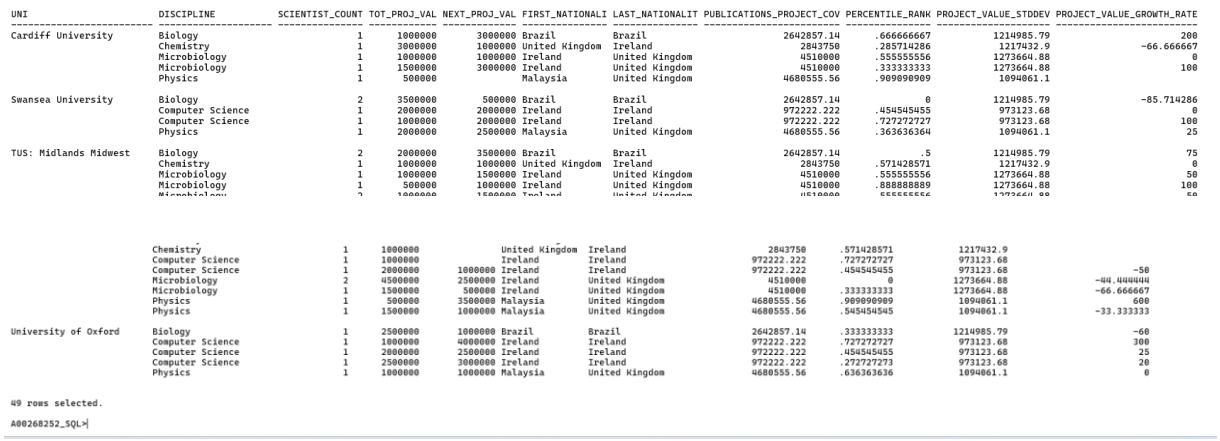
Key Issues:

- **Lag Function for Previous Year:** Lag() calculates the previous year's project value for each university, but this might return NULL for the first entry, which is handled with NVL() in the outer query
- **Handling Nulls:** The NVL() function is used to replace NULL values from the Lag() function with 0, ensuring no gaps in the data
- **Running Total Calculations:** The cumulative sum of project values across all years for each university is calculated using Sum() Over()

5. Calculate Discipline-Level Research Project Statistics and Growth Rates

This query calculates various statistics for each university and discipline, including the number of scientists, total project value, project value growth rates, and the relationship between publications and project value.

```
Cl scr
Break On Uni Skip 1
Select
   Uni, Discipline, Scientist_Count, Tot_Proj_Val,
   Lead(Tot_Proj_Val) Over (Partition By Discipline Order By Start_Year) As Next_Proj_Val,
   First_Value(Nationality) Over (Partition By Discipline Order By Start_Year Rows Between
Unbounded Preceding And Unbounded Following) As First_Nationality,
   Last_Value(Nationality) Over (Partition By Discipline Order By Start_Year Rows Between
Unbounded Preceding And Unbounded Following) As Last_Nationality,
   Covar_Pop(Num_Publications, Tot_Proj_Val) Over (Partition By Discipline) As
Publications_Project_Cov,
   Percent_Rank() Over (Partition By Discipline Order By Tot_Proj_Val Desc) As
Percentile_Rank,
   Stddev(Tot_Proj_Val) Over (Partition By Discipline) As Project_Value_Stddev,
   (Lead(Tot_Proj_Val) Over (Partition By Discipline Order By Start_Year) - Tot_Proj_Val) /
Tot_Proj_Val * 100 As Project_Value_Growth_Rate
From (
   Select
      Rp.Discipline,
      S.University As Uni,
      Count(Distinct S.Scientist_Id) As Scientist_Count,
      Sum(Rp.Project_Value) As Tot_Proj_Val,
      Min(Rp.Start_Year) As Start_Year,
      S.Nationality,
      Sum(S.Num_Publications) As Num_Publications
   From
      Scientist S, Research_Project Rp
   Where
      S.Scientist_Id = Rp.Scientist_Id
   Group By
      Rp.Discipline, S.University, S.Nationality
)
Order By
   Uni, Discipline;
```



UNI	DISCIPLINE	SCIENTIST_COUNT	TOT_PROJ_VAL	NEXT_PROJ_VAL	FIRST_NATIONALI	LAST_NATIONALIT	PUBLICATIONS_PROJECT_COV	PERCENTILE_RANK	PROJECT_VALUE_STDDEV	PROJECT_VALUE_GROWTH_RATE
Cardiff University	Biology	1	1000000	3000000	Brazil	Brazil	2642857.14	.666666667	1214985.79	200
	Chemistry	1	3000000	1000000	United Kingdom	Ireland	2843750	.285714286	1217432.9	-66.666667
	Microbiology	1	1000000	1000000	Ireland	United Kingdom	4510000	.555555556	1273664.88	0
	Microbiology	1	1500000	3000000	Ireland	United Kingdom	4510000	.333333333	1273664.88	100
	Physics	1	500000		Malaysia	United Kingdom	4680555.56	.909090909	1094061.1	
Swansea University	Biology	2	3500000	500000	Brazil	Brazil	2642857.14	0	1214985.79	-85.714286
	Computer Science	1	2000000	2000000	Ireland	Ireland	972222.222	.454545455	973123.68	0
	Computer Science	1	1000000	2000000	Ireland	Ireland	972222.222	.727272727	973123.68	100
	Physics	1	2000000	2500000	Malaysia	United Kingdom	4680555.56	.363636364	1094061.1	25
TUS: Midlands Midwest	Biology	2	2000000	3500000	Brazil	Brazil	2642857.14	.5	1214985.79	75
	Chemistry	1	1000000	1000000	United Kingdom	Ireland	2843750	.571428571	1217432.9	0
	Microbiology	1	1000000	1500000	Ireland	United Kingdom	4510000	.555555556	1273664.88	50
	Microbiology	1	500000	1000000	Ireland	United Kingdom	4510000	.888888889	1273664.88	100
	Microbiology	2	1000000	1500000	Ireland	United Kingdom	4510000	.555555556	1273664.88	50
	Chemistry	1	1000000		United Kingdom	Ireland	2843750	.571428571	1217432.9	
	Computer Science	1	1000000		Ireland	Ireland	972222.222	.727272727	973123.68	
	Computer Science	1	2000000	1000000	Ireland	Ireland	972222.222	.454545455	973123.68	-50
	Microbiology	2	4500000	2500000	Ireland	United Kingdom	4510000	0	1273664.88	-44.444444
	Microbiology	1	1500000	500000	Ireland	United Kingdom	4510000	.333333333	1273664.88	-66.666667
	Physics	1	500000	3500000	Malaysia	United Kingdom	4680555.56	.909090909	1094061.1	600
	Physics	1	1500000	1000000	Malaysia	United Kingdom	4680555.56	.545454545	1094061.1	-33.333333
University of Oxford	Biology	1	2500000	1000000	Brazil	Brazil	2642857.14	.333333333	1214985.79	-60
	Computer Science	1	1000000	4000000	Ireland	Ireland	972222.222	.727272727	973123.68	300
	Computer Science	1	2000000	2500000	Ireland	Ireland	972222.222	.454545455	973123.68	25
	Computer Science	1	2500000	3000000	Ireland	Ireland	972222.222	.272727273	973123.68	20
	Physics	1	1000000	1000000	Malaysia	United Kingdom	4680555.56	.636363636	1094061.1	0

49 rows selected.

A00268252_SQL>

Query Breakdown:

- **Scientist Count (Scientist_Count):** Counts the distinct scientists per university and discipline
- **Total Project Value (Tot_Proj_Val):** Sums the total project value for each university and discipline
- **Next Year's Project Value (Next_Proj_Val):** Uses Lead() to find the next year's project value for each discipline
- **First/Last Nationality (First_Nationality/Last_Nationality):** Retrieves the first and last nationality within each discipline using First_Value() and Last_Value()
- **Publications-Project Value Covariance (Publications_Project_Cov):** Calculates the covariance between the number of publications and project value using Covar_Pop()
- **Percentile Rank (Percentile_Rank):** Computes the percentile rank of total project value within each discipline using Percent_Rank()
- **Standard Deviation (Project_Value_Stddev):** Measures the standard deviation of project values within each discipline using Stddev()
- **Project Value Growth Rate (Project_Value_Growth_Rate):** Calculates the growth rate in project value between consecutive years

Explanation:

- The **inner query** aggregates data by university and discipline, calculating the total project value and number of publications
- **Window functions** such as Lead(), First_Value(), Last_Value(), and Covar_Pop() are used to analyze trends and relationships in the data
- **Grouping** is done by university and discipline to ensure proper aggregation

Key Issues:

- The **Lead()** function may return NULL for the final project, making the growth rate calculation for the last year incomplete
- **Covariance** may be skewed by extreme values
- **Growth rate** might not be meaningful for the first or last years due to missing data

Youtube Links

Video 1: https://youtu.be/-6CFcrz3PiY

Video 2: https://youtu.be/6SM4HiEQFRs