**PART II**

**ALL table that has been used in the Query**

* The Create is use to create the table.
* Create is the sql query used to add the new table in the database.
* It is DDL based query in the oracle.
* SQL Query

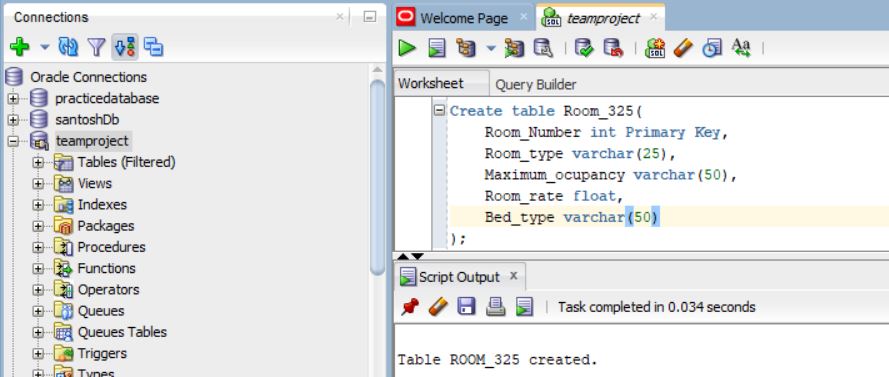
|  |
| --- |
| Create table Room\_325(  Room\_Number int Primary Key,  Room\_type varchar(25),  Maximum\_ocupancy varchar(50),  Room\_rate float,  Bed\_type varchar(50)  ); |

* SQL Query Explanation

Create is used to make the table name Room\_325 with Room\_number as the primary key.

OUTPUT

SQL Query in Oracle

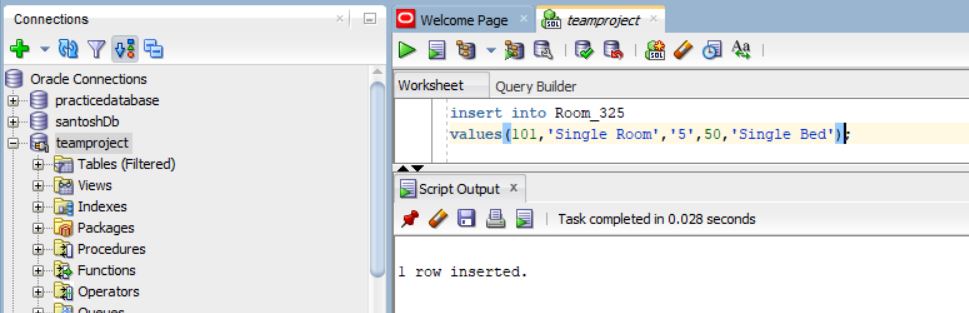


* Inserted into Room
* Insert is used enter the data into the database.
* It is DDL based query
* SQL Query

|  |
| --- |
| insert into Room\_325  values(101,'Single Room','5',50,'Single Bed');  insert into Room\_325  values(102,'Double Room','2',50,'Double Bed'); |

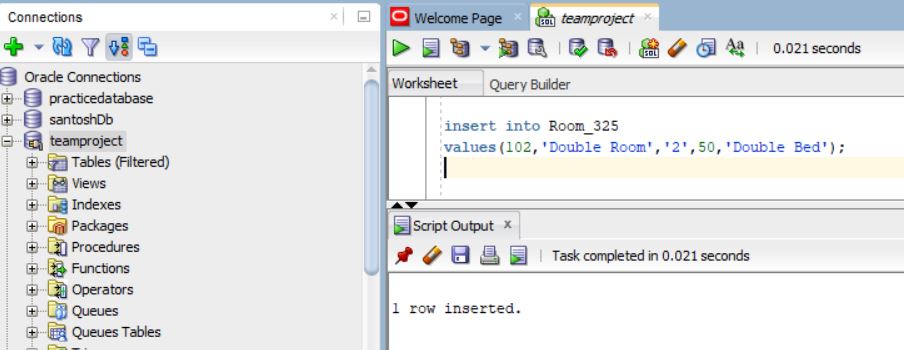
OUTPUT

SQL Query in Oracle



OUTPUT

SQL Query in Oracle



**Customer\_325 table**

* Foreign key is used to get the column that is been assigned to the other table in the same database.
* References is used to refer the table from where the foreign key is selected.
* SQL Query Explanation :

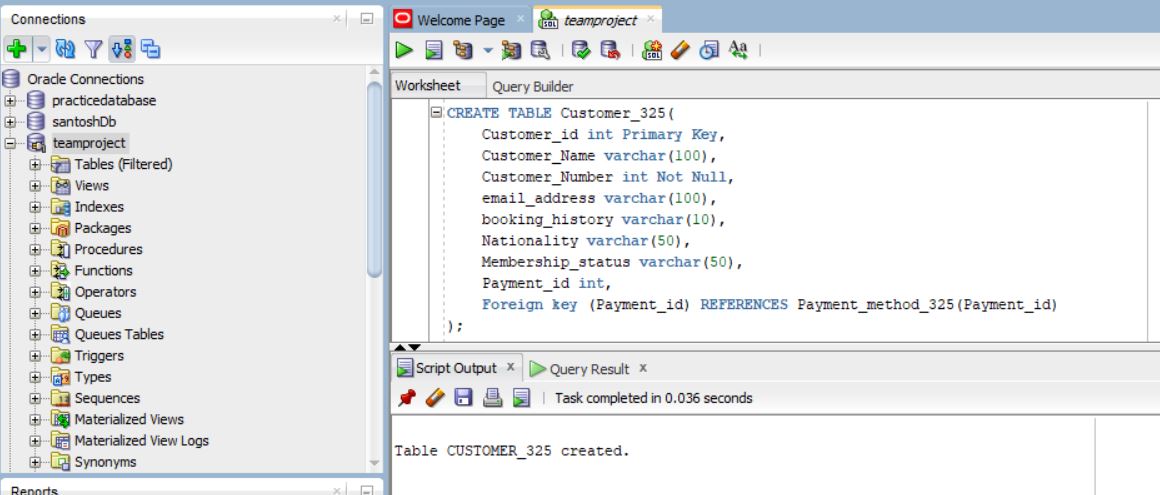
Customer\_325 with the primary key customer\_id , customer\_number is also not null and Payment\_id is the foreign key used from the payment\_method\_325.

* SQL Query

|  |
| --- |
| CREATE TABLE Customer\_325(  Customer\_id int Primary Key,  Customer\_Name varchar(100),  Customer\_Number int Not Null,  email\_address varchar(100),  booking\_history varchar(10),  Nationality varchar(50),  Membership\_status varchar(10),  Payment\_id int,  Foreign key (Payment\_id) REFERENCES Payment\_method\_325(Payment\_id)  ); |

SQL Query in Oracle

OUTPUT



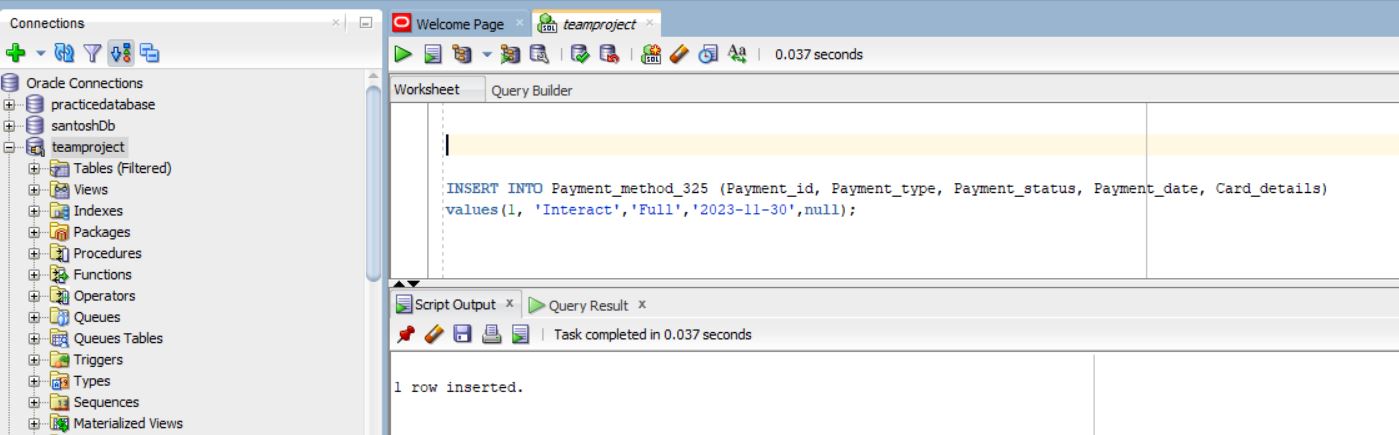
**Inserted into Customer\_325**

* Valued are inserted according to the data types that has been created using the table.
* SQL Query

|  |
| --- |
| insert into Customer\_325  values(201,'Santosh Acharya',4379551559,'acharyaanish16@gmail.com',null,'Nepali','Full Member',1); |

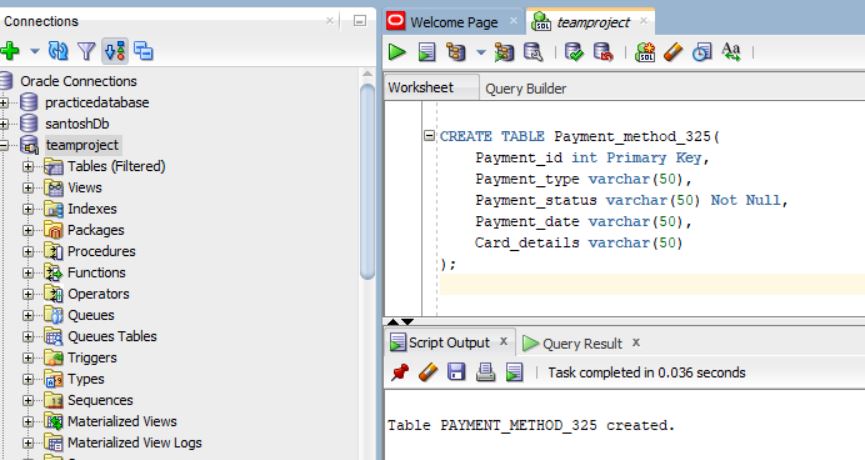
SQL Query in Oracle

OUTPUT



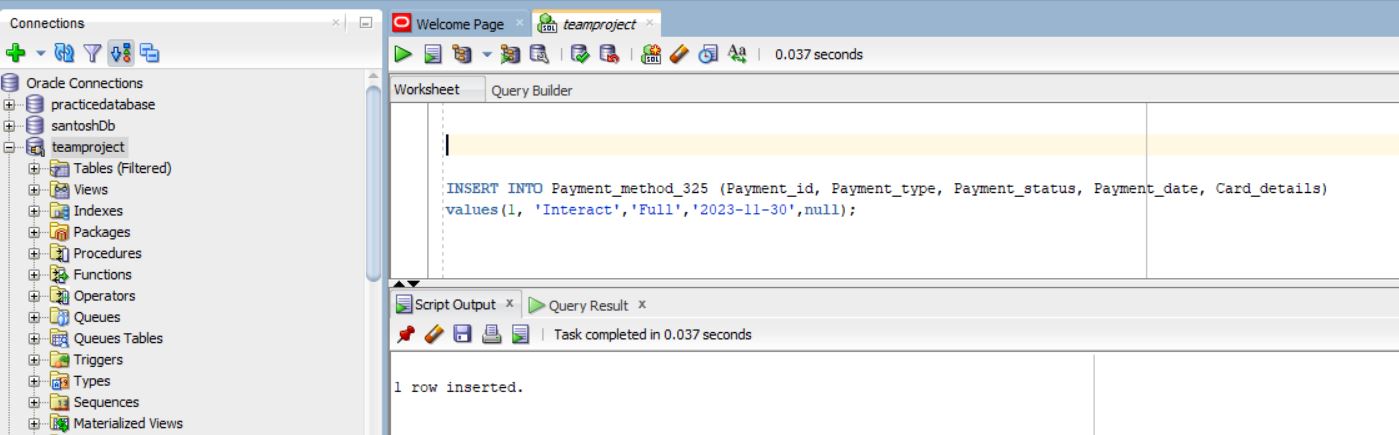
**Payment\_method\_325**

|  |
| --- |
| CREATE TABLE Payment\_method\_325(  Payment\_id int Primary Key,  Payment\_type varchar(50),  Payment\_status varchar(50) Not Null,  Payment\_date date,  Card\_details varchar(50)  ); |

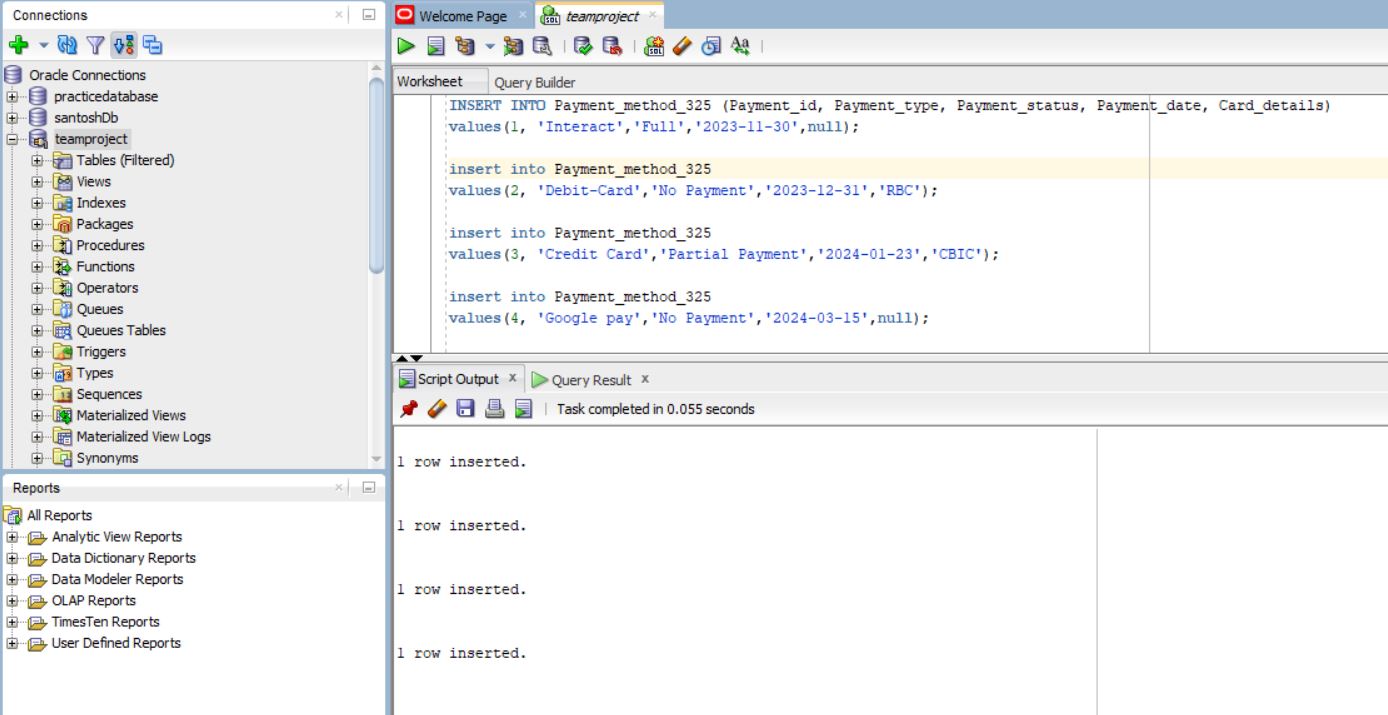


**Inserted into payment\_method\_325**

|  |
| --- |
| insert into Payment\_method\_325  values(1, 'Interact','Full Payment','2023-11-31',null); |



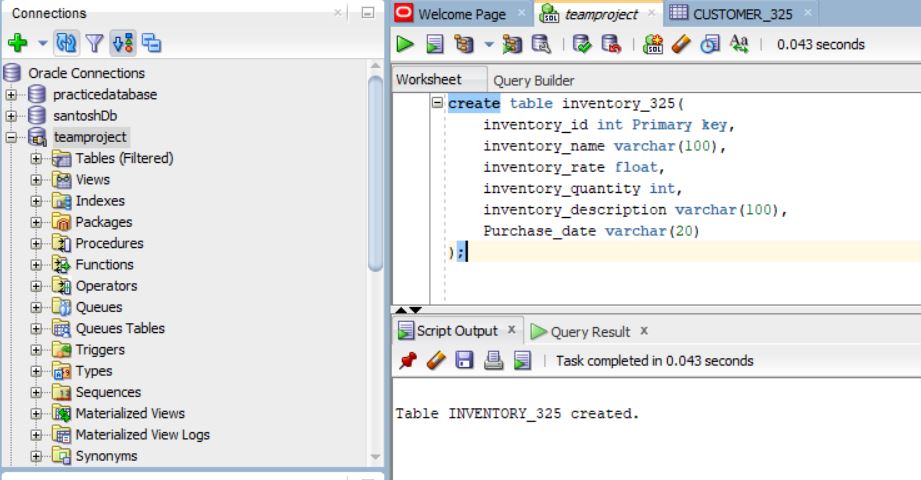
|  |
| --- |
| insert into Payment\_method\_325  values(2, 'Debit Card','No Payment','2023-12-31','RBC');  insert into Payment\_method\_325  values(3, 'Credit Card','Partial Payment','2024-01-23','CBIC');  insert into Payment\_method\_325  values(4, 'Google pay','No Payment','2024-03-15',null);  insert into Payment\_method\_325  values(5, 'Bank Transfer','Partial Payment','2024-01-23','TD'); |



**Inventory\_325**

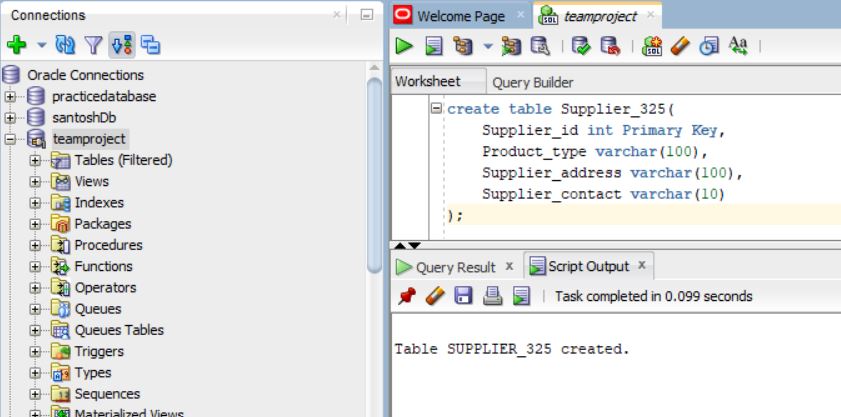
* It is the table used to store the inventory related information in the hotel reservation system.
* SQL Query :

|  |
| --- |
| create table inventory\_325(  inventory\_id int Primary key,  inventory\_name varchar(100),  inventory\_rate float,  inventory\_quantity int,  inventory\_description varchar(100),  Purchase\_date varchar(100)    ); |



**Supplier\_325**

|  |
| --- |
| create table Supplier\_325(  Supplier\_id int Primary Key,  Product\_type varchar(100),  Supplier\_address varchar(100),  Supplier\_contact varchar(10)  ); |



**Category A**

**1**

* Select is the query used to view the data in the table. It is the used to select and view the particular table.
* SQL Query

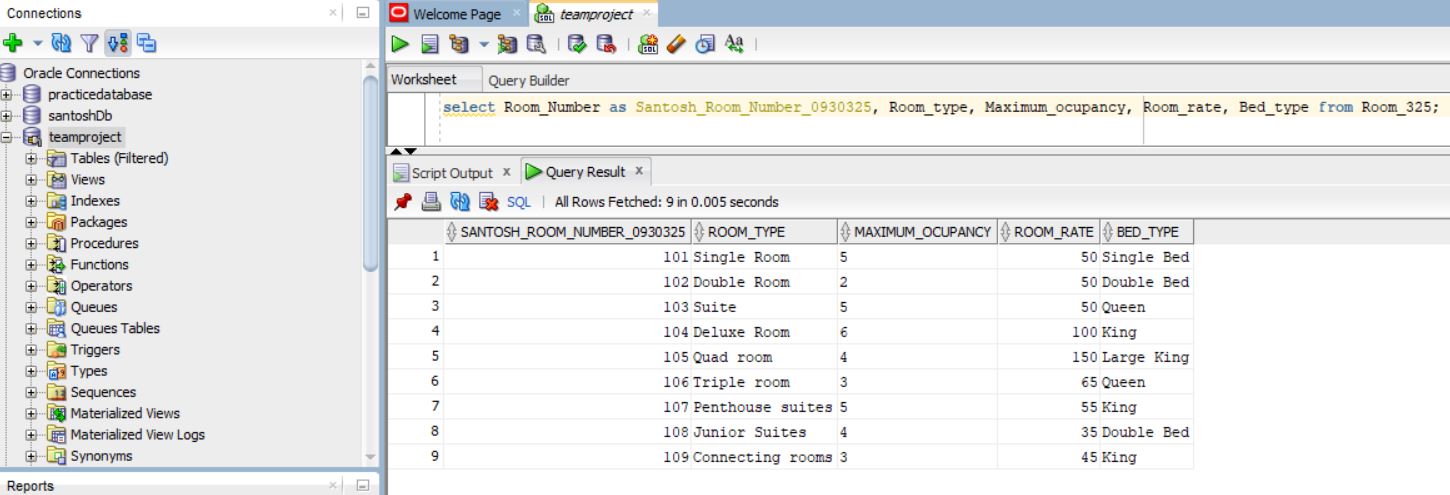
|  |
| --- |
| select Room\_Number as Santosh\_Room\_Number\_0930325, Room\_type, Maximum\_ocupancy, Room\_rate, Bed\_type from Room\_325; |

* SQL Query Explanation

In the sql query room\_number, room type, maximum\_ocupancy, room\_rate, bed\_type is selected from the room\_325 table.

SQL Query in Oracle

OUTPUT



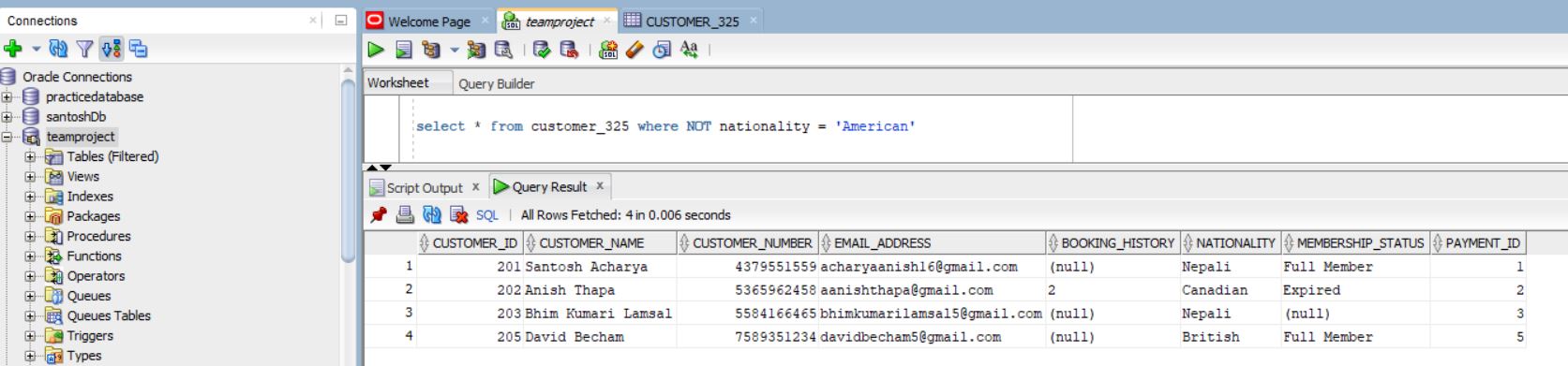
**2**

* NOT is the operator used to denote the select query that will select all other value except the assigned value.
* SQL Query :

|  |
| --- |
| Select \* from Customer\_325 where NOT Nationality = “American” |

* SQL Query Explanation :

In the query the table will show the value except the value that contain Nationality = “American”



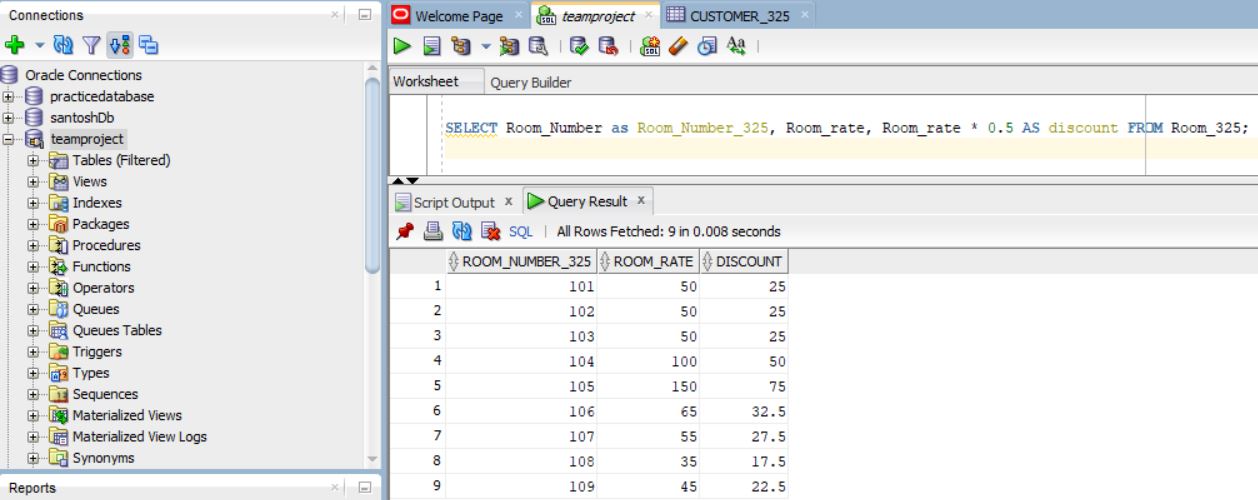
**3**

* Multiplication is a mathematical operation used to multiply numeric values, typically within SELECT statements or as part of UPDATE or INSERT statements.
* SQL Query:

|  |
| --- |
| SELECT Room\_Number as Room\_Number\_325, Room\_rate, Room\_rate \* 0.5 AS discount FROM Room\_325; |

* SQL Query Explanation:

In the query the multiplication is used to get the discount rate of 50% to the room rate from room\_325



**4**

* Concatenation is used to combine the string together. It is useful to join the string
* SQL Query :

|  |
| --- |
| SELECT Room\_Number as Room\_Number\_325, Room\_type || ' - ' || Bed\_type AS Room\_Details\_325 FROM Room\_325; |

* SQL query explanation:

In the query, room type and bed type is connected with each other with the ‘– ‘ sign.



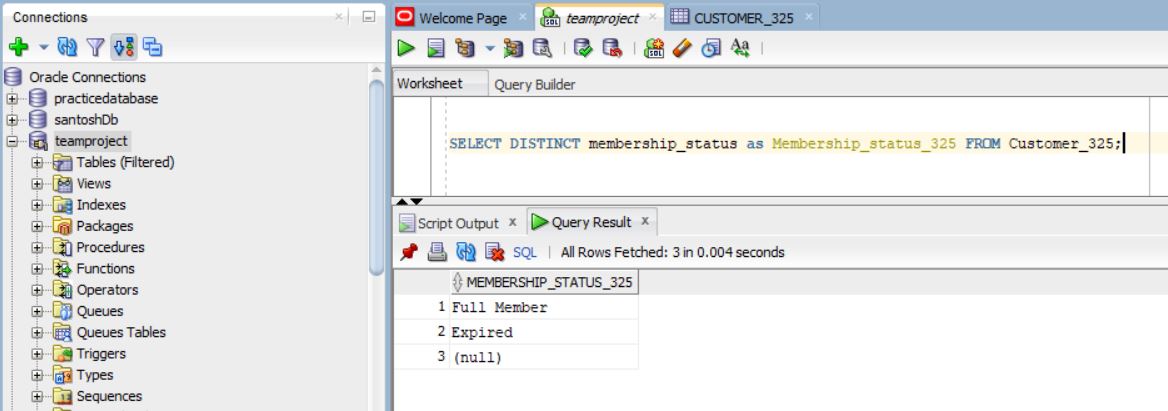
**5**

* DISTINCT is use to return the different value.
* SQL Query

|  |
| --- |
| SELECT DISTINCT membership\_status as Membership\_status\_325 FROM Customer\_325; |

* SQL Explanation:

In the query, the membership status with different value is selected.



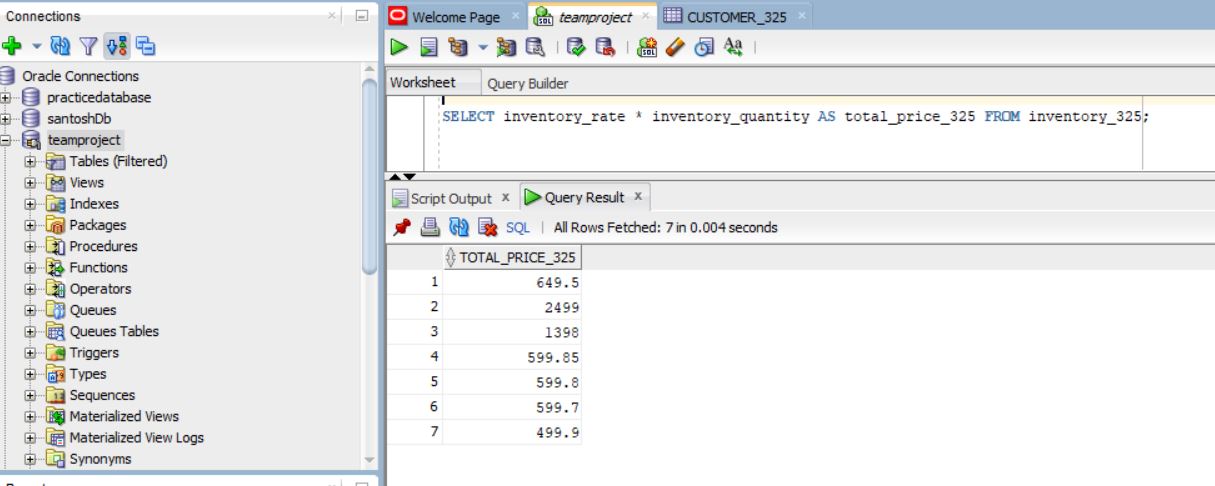
**6**

* Addition, Subtraction , Multiplication is the mathematical operand used with the select in the sql query.
* SQL Query

|  |
| --- |
| Select inventory\_rate \* inventory\_quantity as total\_price\_325 from inventory\_325; |

* SQL Query Explanation:

In the query the inventory rate is multiplied with the inventory quantity to get the total\_price from the inventory\_325 table.



**Category B**

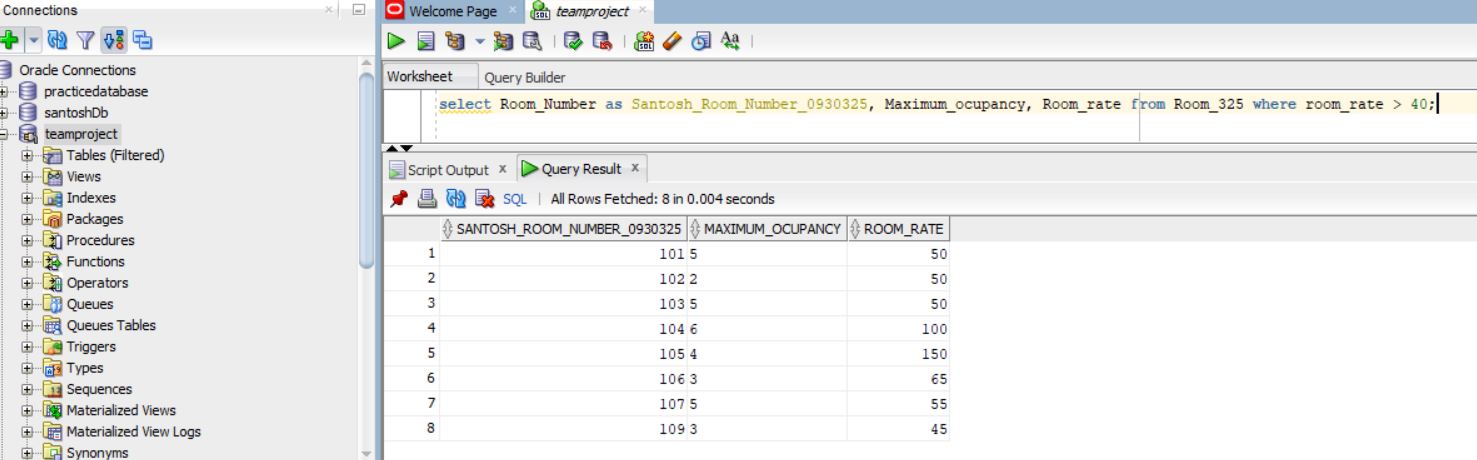
**1**

* Comparison operator helps to know the certain condition in the database and select the data according to the condition.
* Greater than sign will compare the value and determine if the value is greater than the given condition
* SQL query explanation :

In the query the room rate greater than 40 will be selected from the room\_325 table.

* SQL Query

|  |
| --- |
| select Room\_Number as Santosh\_Room\_Number\_0930325, Maximum\_ocupancy, Room\_rate from Room\_325 where room\_rate > 40; |



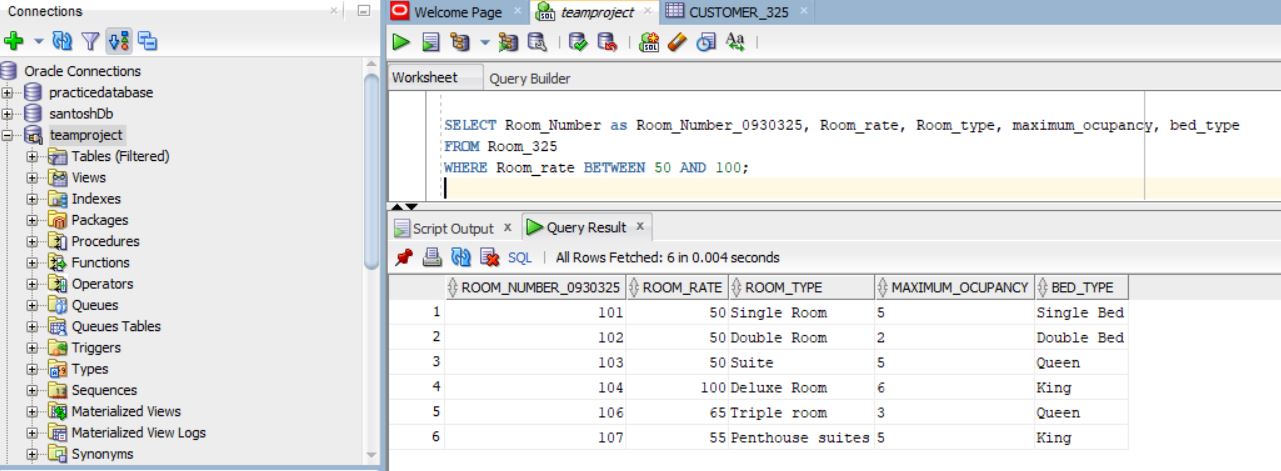
**2**

* Between Operands will help to get the value within a given range.
* SQL Query

|  |
| --- |
| SELECT Room\_Number as Room\_Number\_0930325, Room\_rate, Room\_type, maximum\_ocupancy, bed\_type FROM Room\_325 WHERE Room\_rate BETWEEN 50 AND 100; |

* SQL Query explanation:

In the query the Room Number, Room rate, Room type , maximum occupancy and bed type is selected from the room\_325 table where room rate is between 50 and 100.



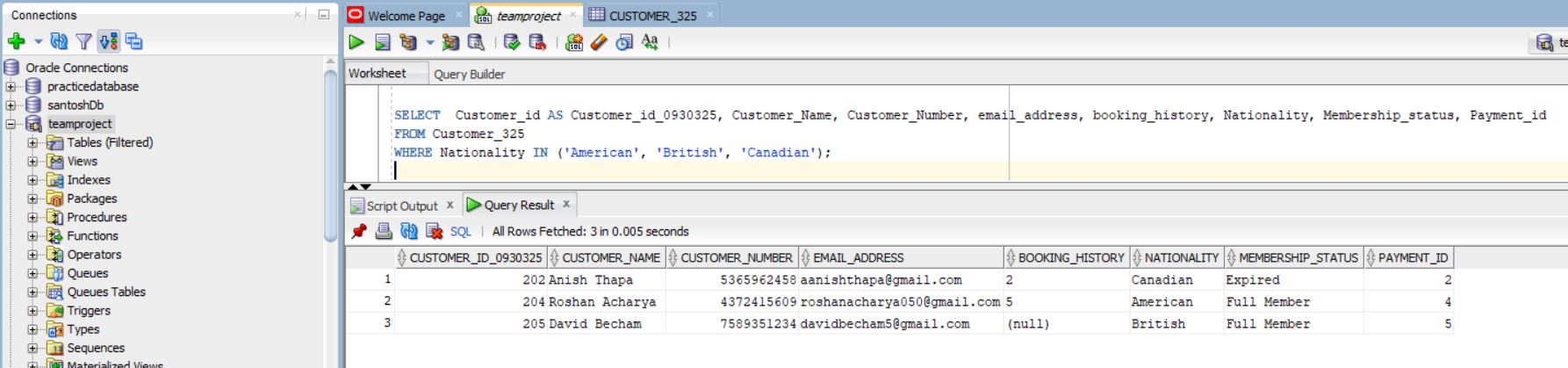
**3**

* IN operands helps to specify multiple values in a Where clause. It is similar of using multiple OR condition
* SQL Query

|  |
| --- |
| SELECT Customer\_id AS Customer\_id\_0930325, Customer\_Name, Customer\_Number, email\_address, booking\_history, Nationality, Membership\_status, Payment\_id  FROM Customer\_325  WHERE Nationality IN ('American', 'British', 'Canadian'); |

* SQL Explanation :

In the query the customer id, customer name, customer number, email address, booking history , membership status and payment id is selected when nationality is either American, british or Canadian.



**4**

* Like operand is used to search the similar pattern in the table. Percentage % and Underscore \_ are often used with like operand.
* SQL Query

|  |
| --- |
| SELECT Customer\_id AS Customer\_id\_0930325, Customer\_Name, Customer\_Number, Nationality, Membership\_status, Payment\_id  FROM Customer\_325  WHERE Customer\_Name LIKE '%Sa%'; |

* SQL Explanation:

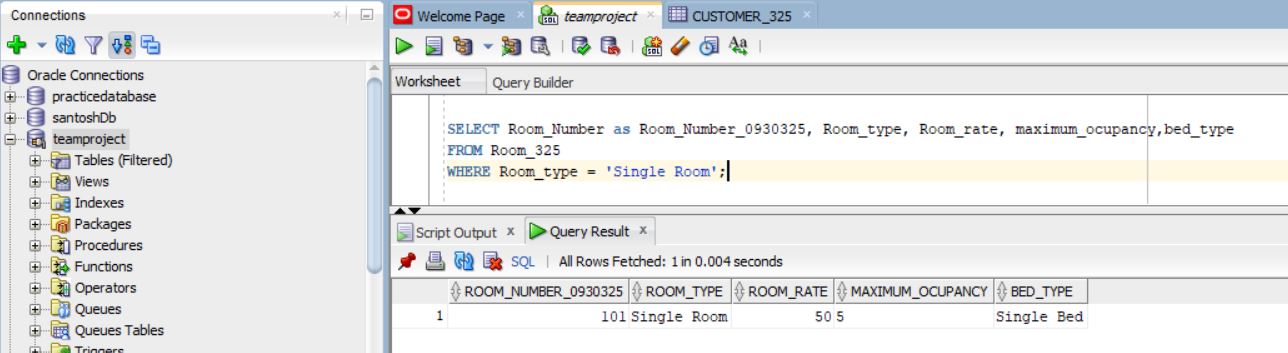
In the query, customer id,customer name, customer number, nationality, membership status and payment id is selected from the customer\_325 where customer name is like ‘%Sa%’.



**5**

* = sign is used to compare the exact value within the table.
* It will provide the value that match the condition.
* SQL Query

|  |
| --- |
| SELECT Room\_Number as Room\_Number\_0930325, Room\_type, Room\_rate, maximum\_ocupancy,bed\_type  FROM Room\_325  WHERE Room\_type = 'Single Room'; |



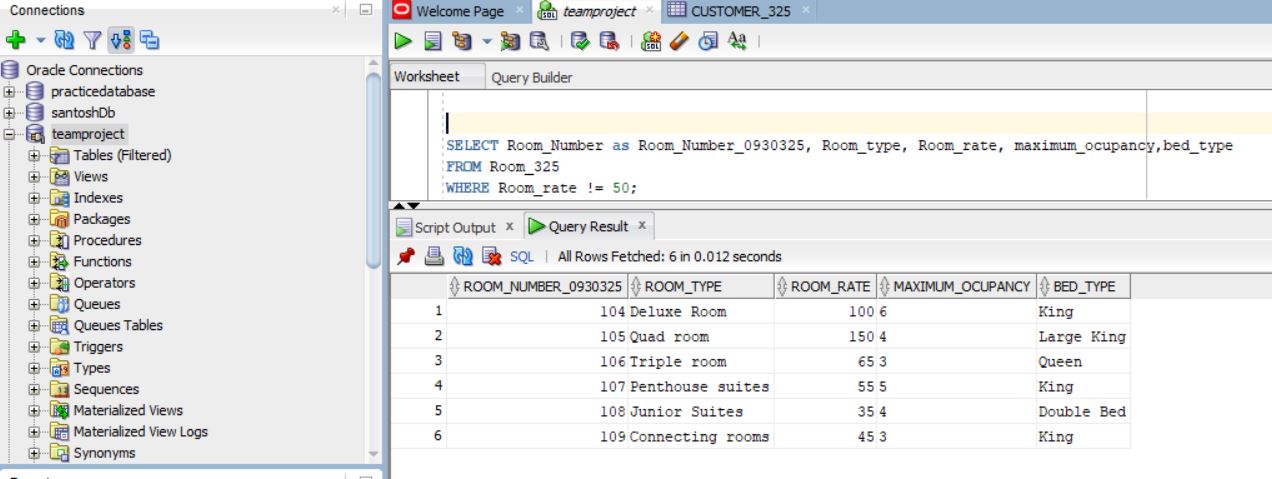
**6**

* != also Know as Not Equal to is used select the value that is not equal to the given value.
* It is the comparison operand used in the sql query.
* SQL Query

|  |
| --- |
| SELECT Room\_Number as Room\_Number\_0930325, Room\_type, Room\_rate, maximum\_ocupancy,bed\_type  FROM Room\_325  WHERE Room\_rate != 50; |

* SQL Query Explanation:

IN the query the given value is used to select the room number, room type, room rate, maximum occupancy and bed type from room\_325 table that is not equal to 50.



**Category C**

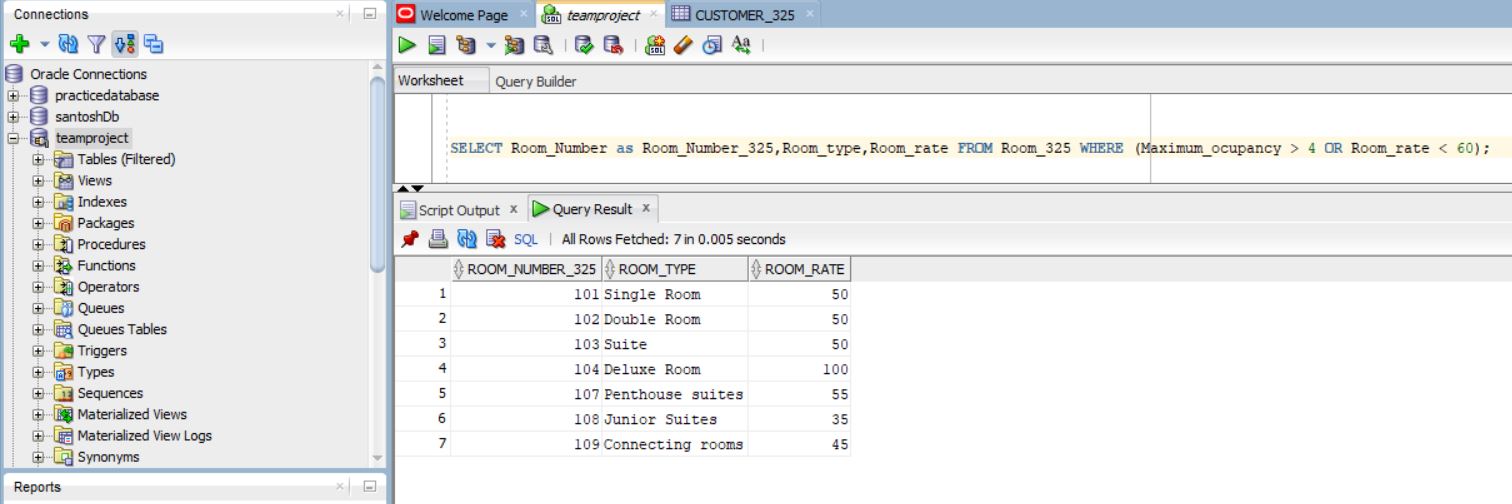
**1**

* Logical Operator are the operator test for the truth of some condition.
* SQL Query

|  |
| --- |
| SELECT Room\_Number as Room\_Number\_325,Room\_type,Room\_rate FROM Room\_325 WHERE (Maximum\_ocupancy > 4 OR Room\_rate < 60); |

* SQL query Explanation:

In the query, the room number, room type, room rate from room\_325 where maximum occupancy is greater than 4 or room rate is less than 60.



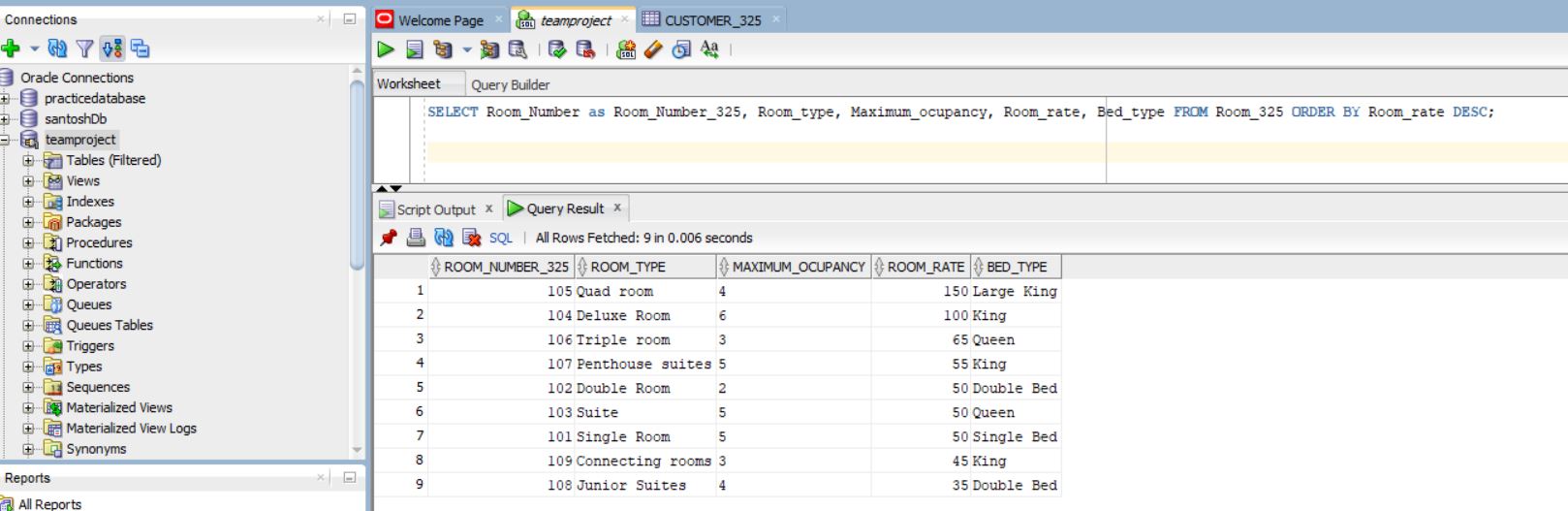
**2**

* Order By is use to sort the result in ascending or descending order.
* It will display the result in the either from low value to high or vice-versa.
* SQL Query

|  |
| --- |
| SELECT Room\_Number as Room\_Number\_325, Room\_type, Maximum\_ocupancy, Room\_rate, Bed\_type FROM Room\_325 ORDER BY Room\_rate DESC; |

* SQL Query Explanation:

In the query, the selected table will be display the value with the descending order by the room rate.



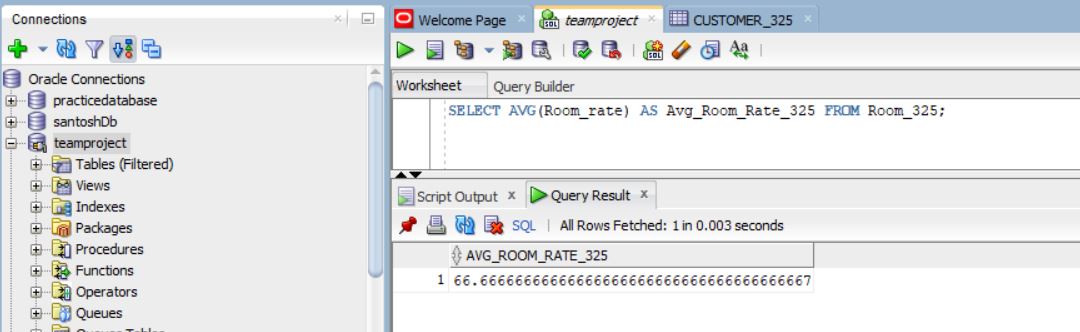
**3**

* Average also known as AVG is function that will return the average value of the column.
* SQL Query

|  |
| --- |
| SELECT AVG(Room\_rate) AS Avg\_Room\_Rate\_325 FROM Room\_325; |

* SQL Query Explanation:

In the query the average of the room rate is displayed from the room\_325



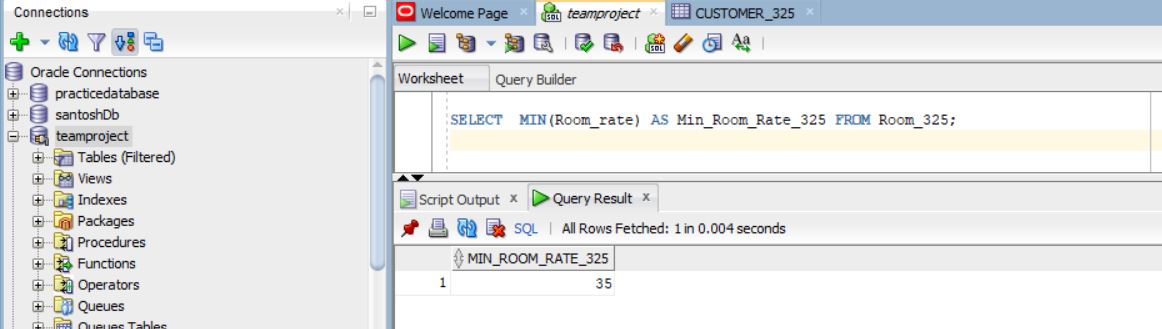
**4**

* Min is the function that will return the minimum value of the column.
* SQL Query

|  |
| --- |
| SELECT MIN(Room\_rate) AS Min\_Room\_Rate\_325 FROM Room\_325; |

* SQL Query Explanation:

In the query the the room rate minimum value is return from the room\_325



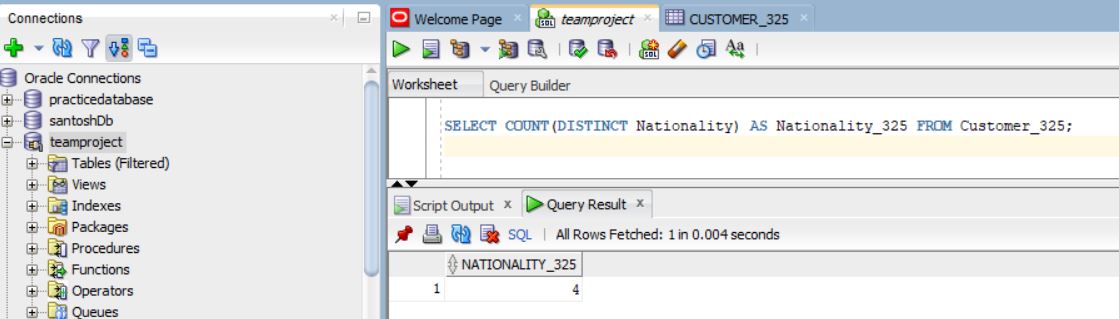
**5**

* Count is the function is the used to return the count of the value of the column.
* SQL Query

|  |
| --- |
| SELECT COUNT(DISTINCT Nationality) AS Nationality\_325 FROM Customer\_325; |

* SQL Query Explanation:

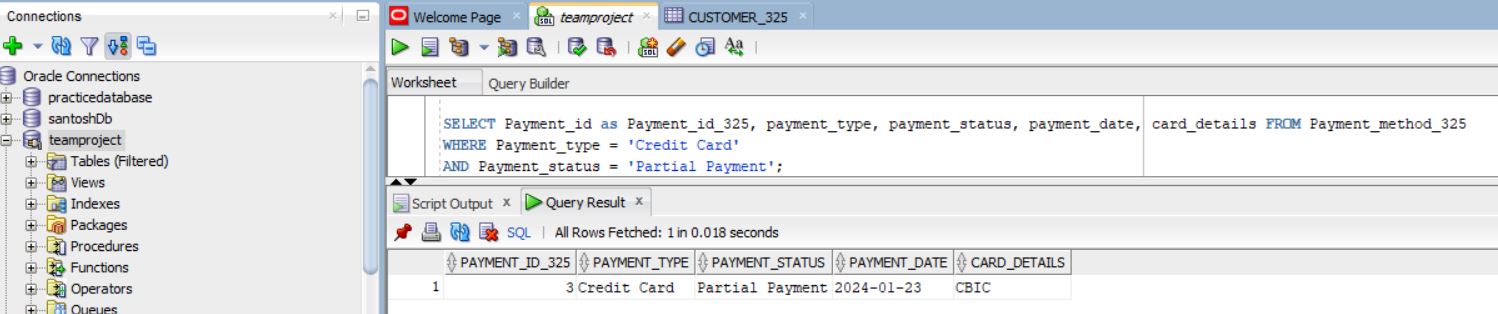
In the query the distinct nationality that count from the customer\_325



**6**

* And operator is used to determine the value that will match all the case before returning the value.
* SQL Query

|  |
| --- |
| SELECT Payment\_id as Payment\_id\_325, payment\_type, payment\_status, payment\_date, card\_details FROM Payment\_method\_325 WHERE Payment\_type = 'Credit Card'  AND Payment\_status = 'Partial Payment'; |

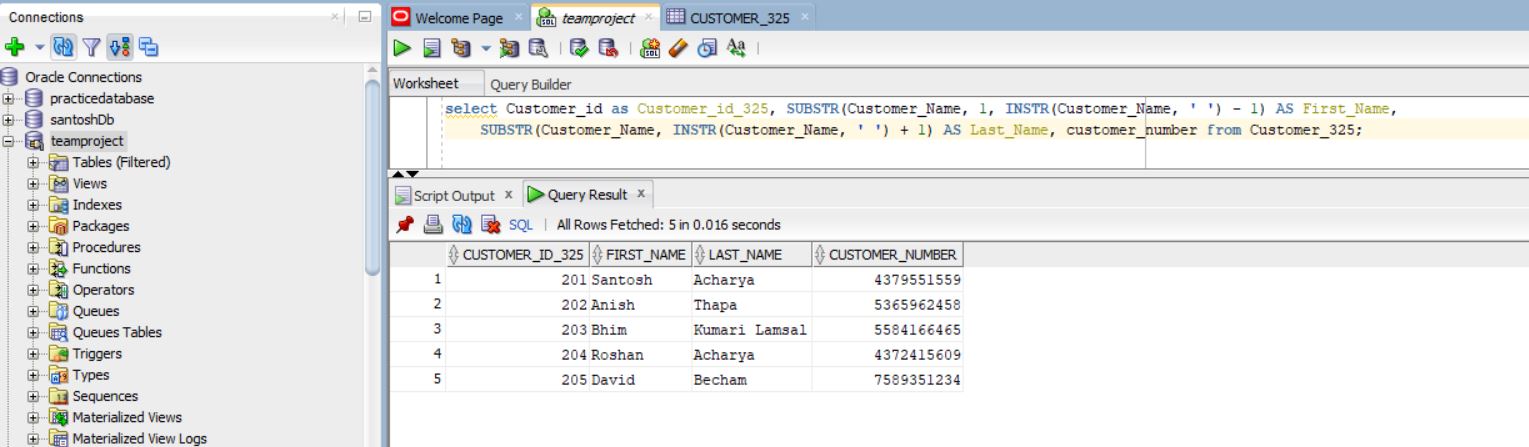


**Category D**

**1**

* INSTR is the function that return the first occurance position of a string in another string.
* SUBSTR is the function that extract substring from a string.
* SQL Query

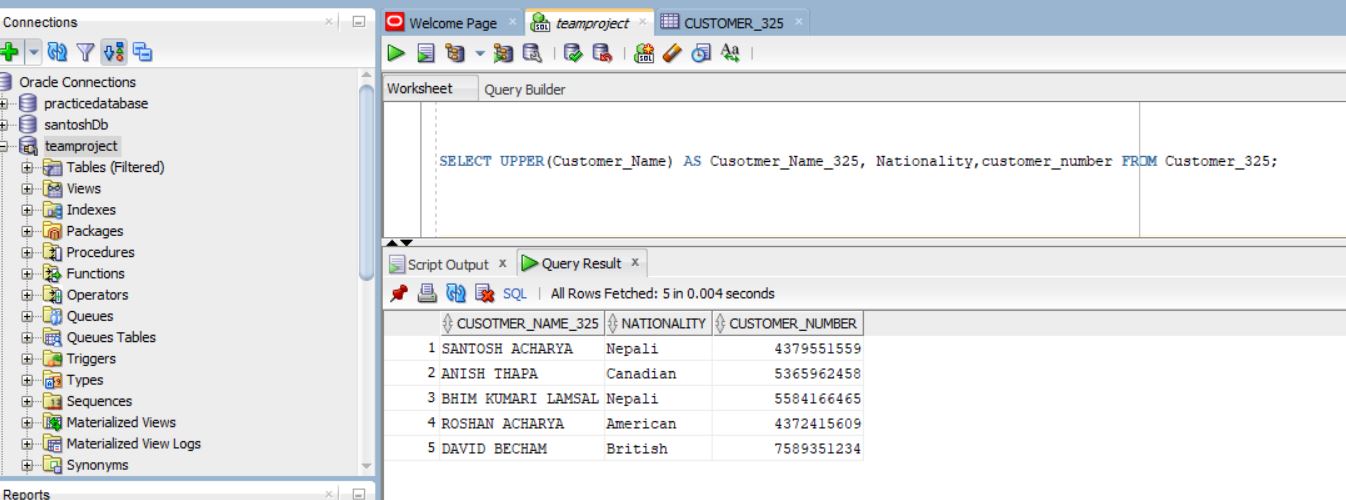
|  |
| --- |
| select Customer\_id as Customer\_id\_325, SUBSTR(Customer\_Name, 1, INSTR(Customer\_Name, ' ') - 1) AS First\_Name, SUBSTR(Customer\_Name, INSTR(Customer\_Name, ' ') + 1) AS Last\_Name, customer\_number from Customer\_325; |



**2**

* Upper is the function that will return the all the string in the column with the upper case.
* SQL Query

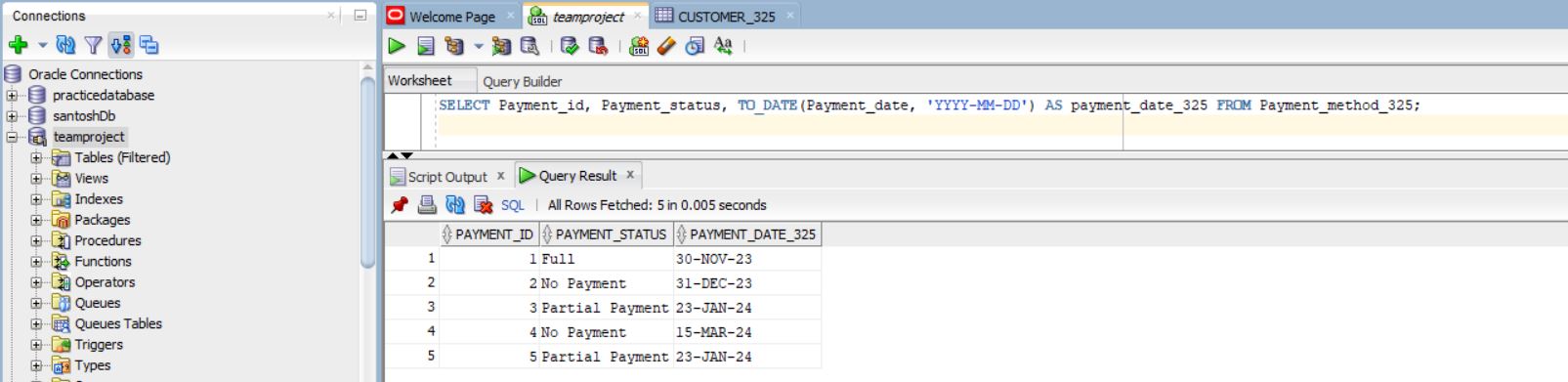
|  |
| --- |
| SELECT UPPER(Customer\_Name) AS Cusotmer\_Name\_325, Nationality,customer\_number FROM Customer\_325; |



**3**

* TO\_DATE is the function that will convert the string to the date format in the SQL.
* SQL Query

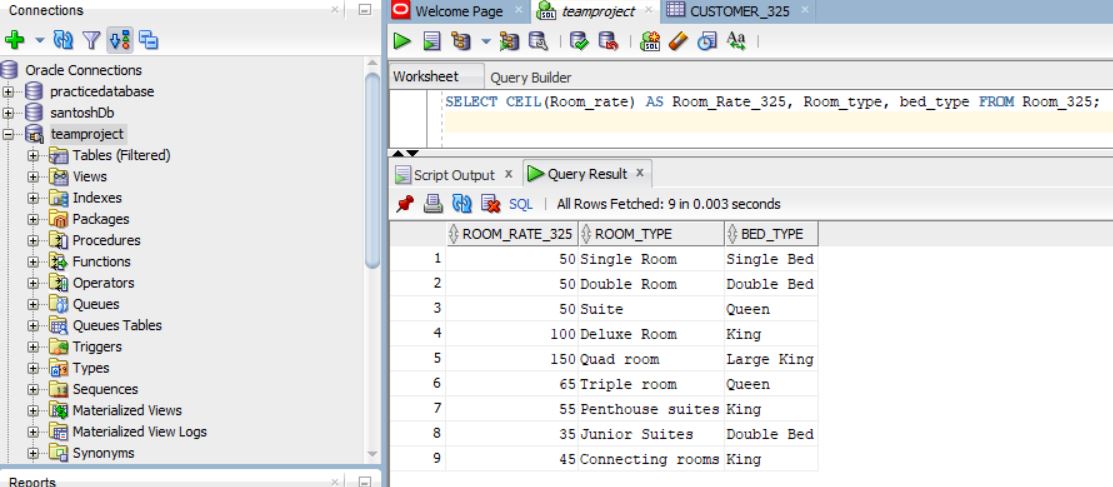
|  |
| --- |
| SELECT Payment\_id, Payment\_status, TO\_DATE(Payment\_date, 'YYYY-MM-DD') AS payment\_date\_325 FROM Payment\_method\_325; |



**4**

* CEIL is the function that will return the nearest small value that is greater than or less than the number.
* SQL Query

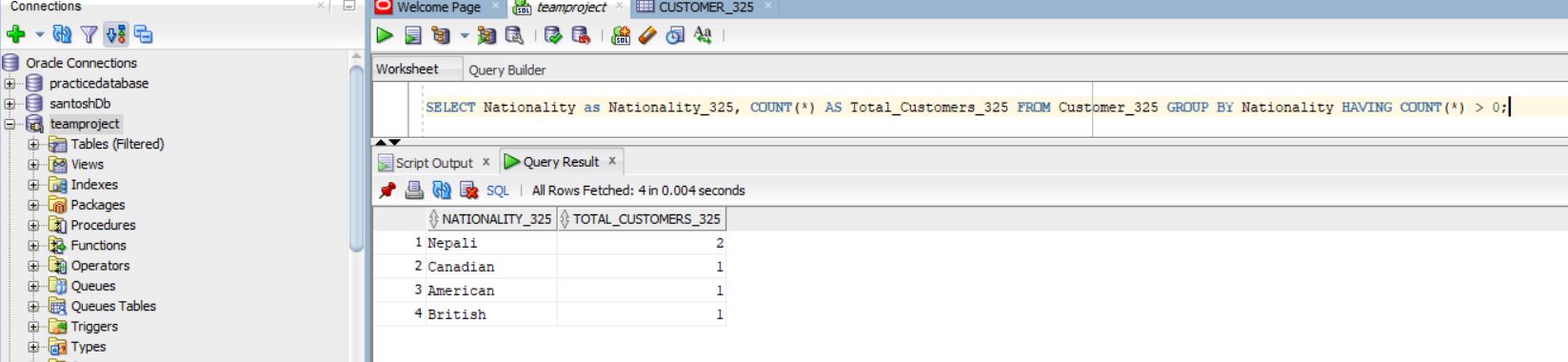
|  |
| --- |
| SELECT CEIL(Room\_rate) AS Room\_Rate\_325, Room\_type, bed\_type FROM Room\_325; |



**5**

* COUNT is thr function that is used to count the value in the table.
* Having is used in the case when the when clause can’t be use in the query.
* SQL Query

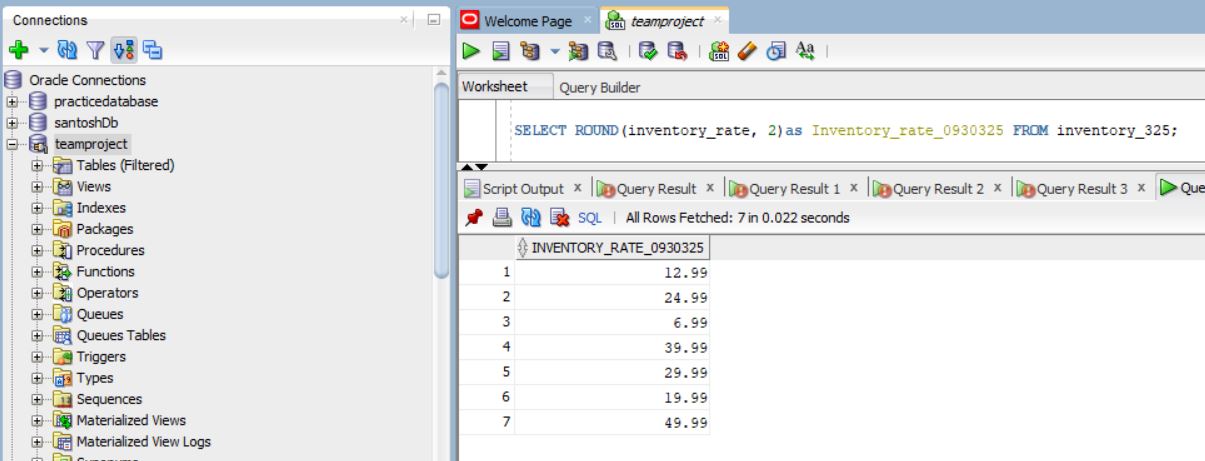
|  |
| --- |
| SELECT Nationality as Nationality\_325, COUNT(\*) AS Total\_Customers\_325 FROM Customer\_325 GROUP BY Nationality HAVING COUNT(\*) > 0; |



**6**

* Round is the function that is used to round of the value to a specified number of decimal places.
* SQL Query

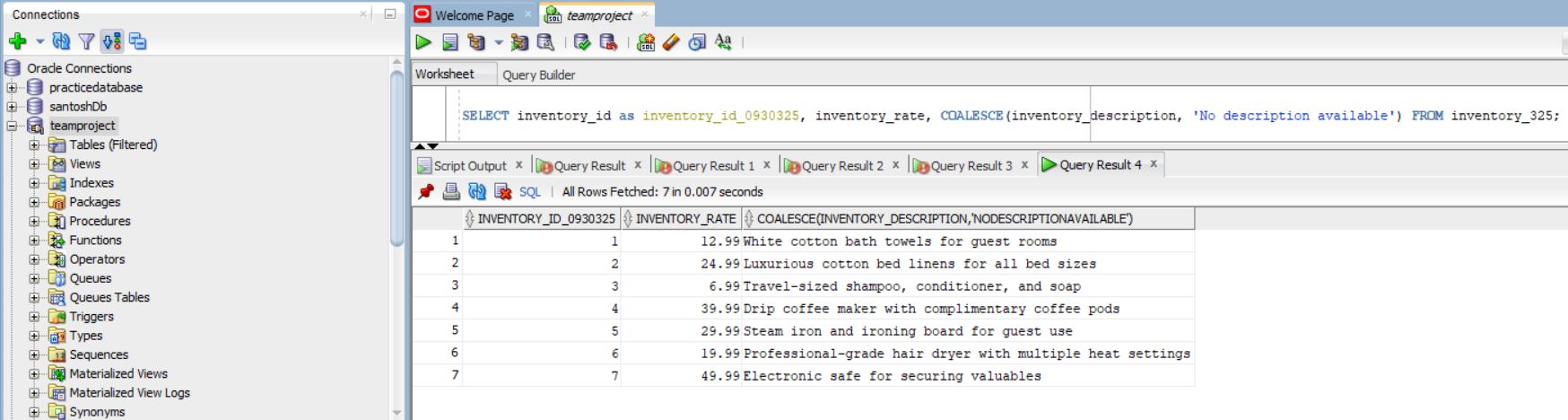
|  |
| --- |
| SELECT ROUND(inventory\_rate, 2)as Inventory\_rate\_0930325 FROM inventory\_325; |



**7**

* COALESCE is used to return the first non-value in the list.
* It used two argument within it.
* SQL Query

|  |
| --- |
| SELECT inventory\_id as inventory\_id\_0930325, inventory\_rate, COALESCE(inventory\_description, 'No description available') FROM inventory\_325; |



**Category E**

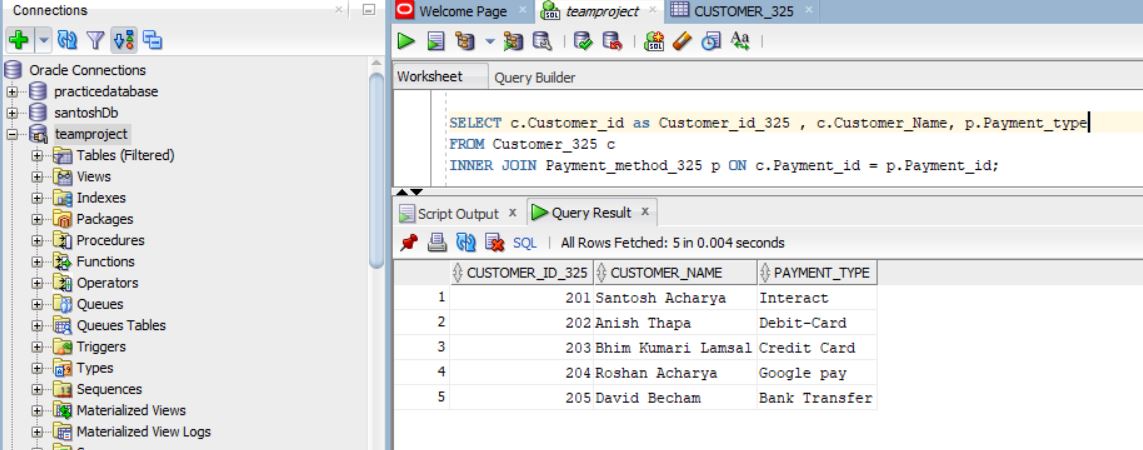
**1**

* Inner join is the function that is used to join the table when both the table has some matching values in a field that is common to both the table.
* SQL Query

|  |
| --- |
| SELECT c.Customer\_id as Customer\_id\_325 , c.Customer\_Name, p.Payment\_type  FROM Customer\_325 c  INNER JOIN Payment\_method\_325 p ON c.Payment\_id = p.Payment\_id; |

* SQL Explanation:

In the query, the Customer\_325 is join with thr Payment\_method\_325 where customer.payemntId is equal to payment.paymentid. c denote as the alias for the table customer and p for the payment.



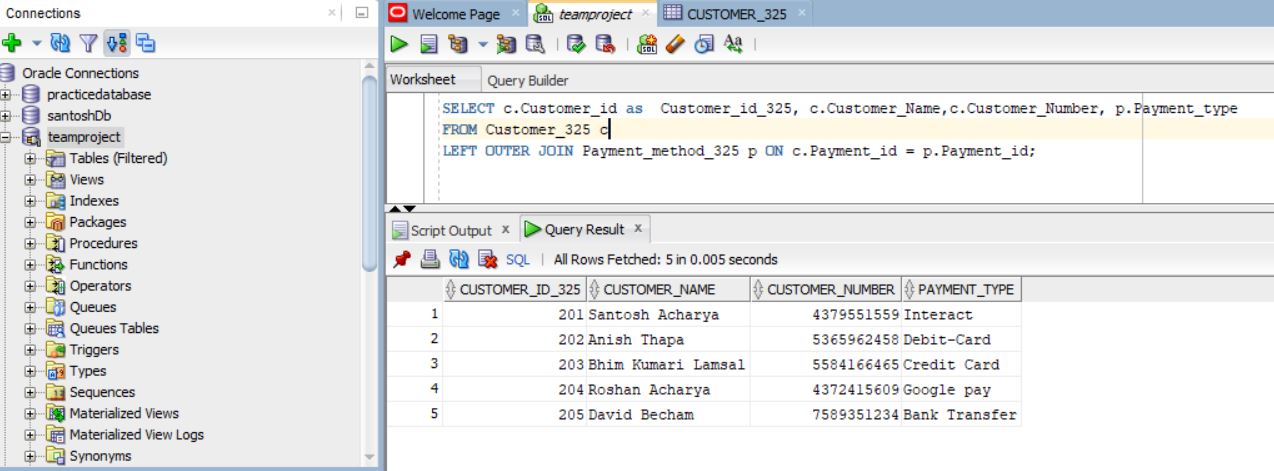
**2**

* Left outer join will return all the record from the left table and the matching record from the right table in the sql.
* SQL Query

|  |
| --- |
| SELECT c.Customer\_id as Customer\_id\_325, c.Customer\_Name,c.Customer\_Number, p.Payment\_type FROM Customer\_325 c  LEFT OUTER JOIN Payment\_method\_325 p ON c.Payment\_id = p.Payment\_id; |

* SQL Query Explanation:

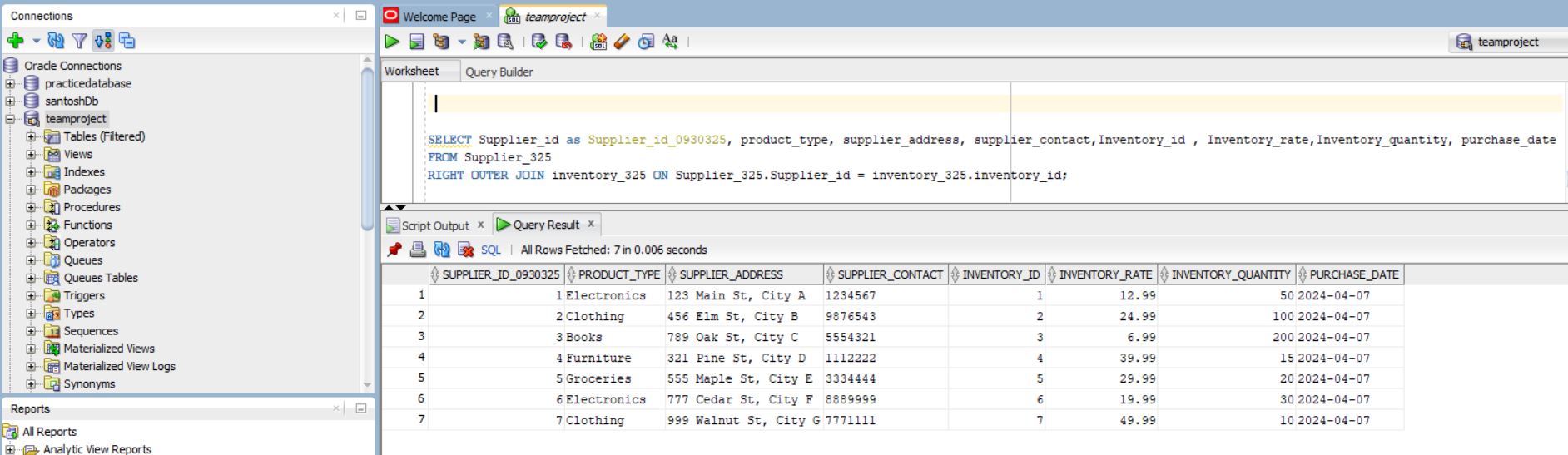
In the query , the table will return all the record from the customer and matching record from the payment method.



**3**

* Right outer join is similar to the left outer join but in Right outer join all the record from the right table is recorded and all the matching record from the left table is extracted.
* SQL Query

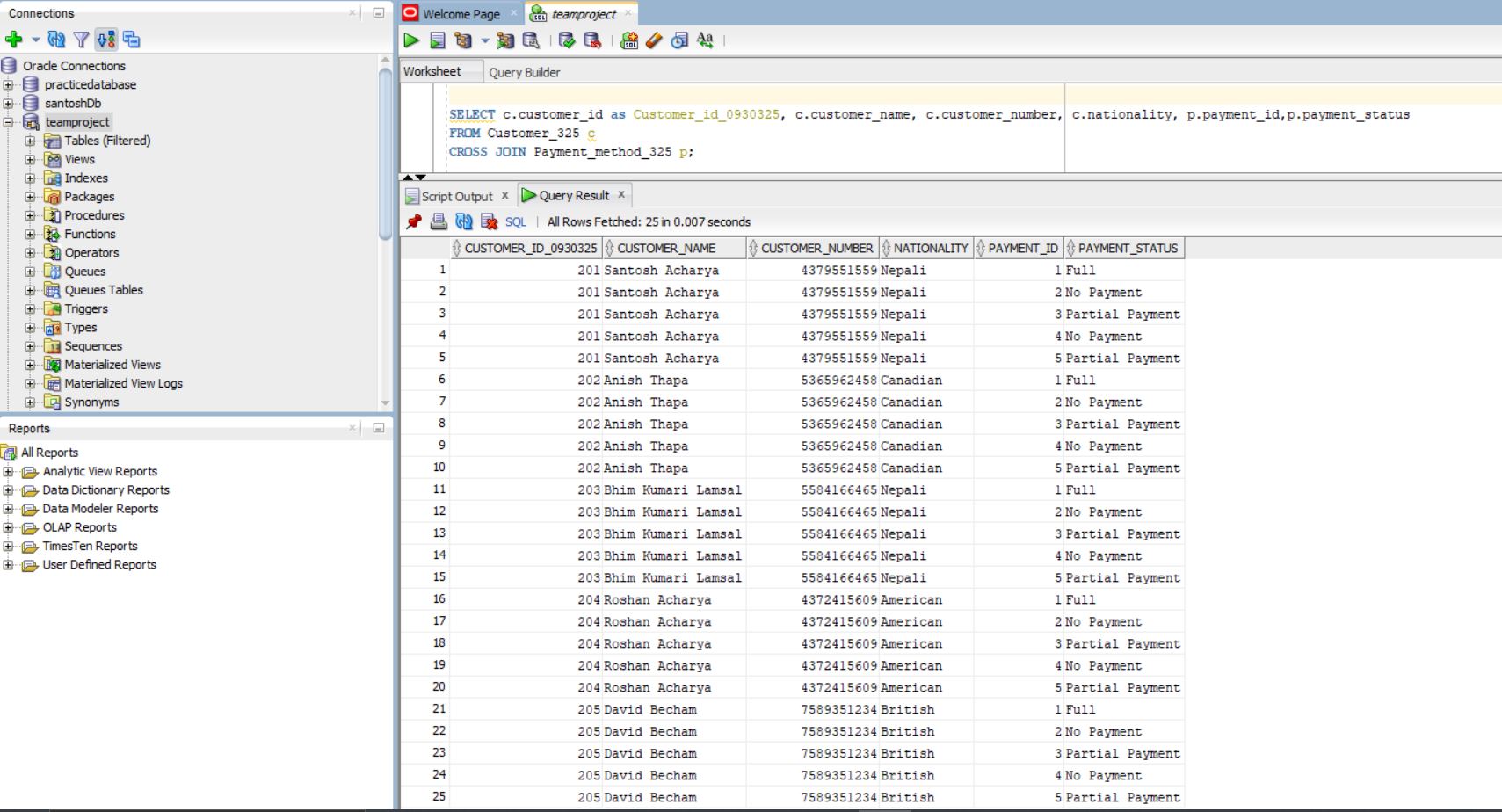
|  |
| --- |
| SELECT Supplier\_id as Supplier\_id\_0930325, product\_type, supplier\_address, supplier\_contact,Inventory\_id , Inventory\_rate,Inventory\_quantity, purchase\_date  FROM Supplier\_325  RIGHT OUTER JOIN inventory\_325 ON Supplier\_325.Supplier\_id = inventory\_325.inventory\_id; |



**4**

* Cross join is used to combine the each row of one table to each row of another table.
* It return the Cartesian product of the sets of rows.
* SQL Query

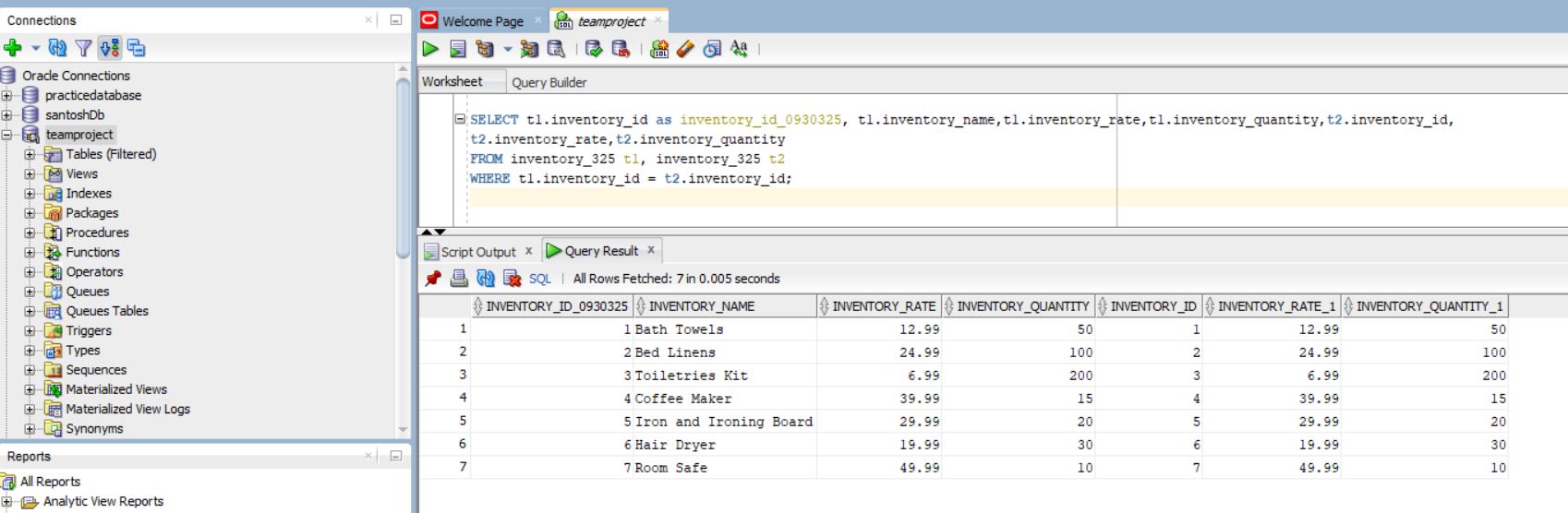
|  |
| --- |
| SELECT c.customer\_id as Customer\_id\_0930325, c.customer\_name, c.customer\_number, c.nationality, p.payment\_id,p.payment\_status FROM Customer\_325 c  CROSS JOIN Payment\_method\_325 p; |



**5**

* Self join is the query in which table is combined with itself in the sql.
* Here all the value is combined with itself in the query.
* SQL Query

|  |
| --- |
| SELECT t1.inventory\_id as inventory\_id\_0930325, t1.inventory\_name,t1.inventory\_rate,t1.inventory\_quantity,t2.inventory\_id,  t2.inventory\_rate,t2.inventory\_quantity FROM inventory\_325 t1, inventory\_325 t2  WHERE t1.inventory\_id = t2.inventory\_id; |



**Category F**

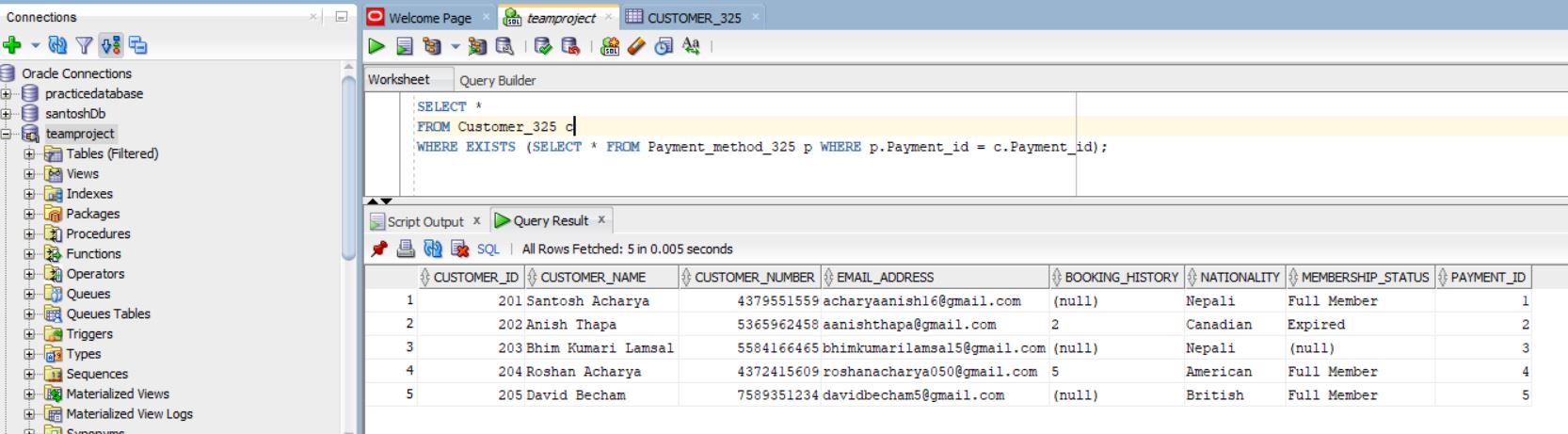
**1**

* + Nested query is the query that has query inside the other query.
  + Exists is the query in the sql which will return the value in the given condition exists in the query
  + SQL Query

|  |
| --- |
| SELECT \*  FROM Customer\_325 c  WHERE EXISTS (SELECT \* FROM Payment\_method\_325 p WHERE p.Payment\_id = c.Payment\_id); |

* SQL Explanation:

In the query the customer table will return the value if the nested query full fill the requirement where paymentid of payment method table is equal to the paymentid of customer table.



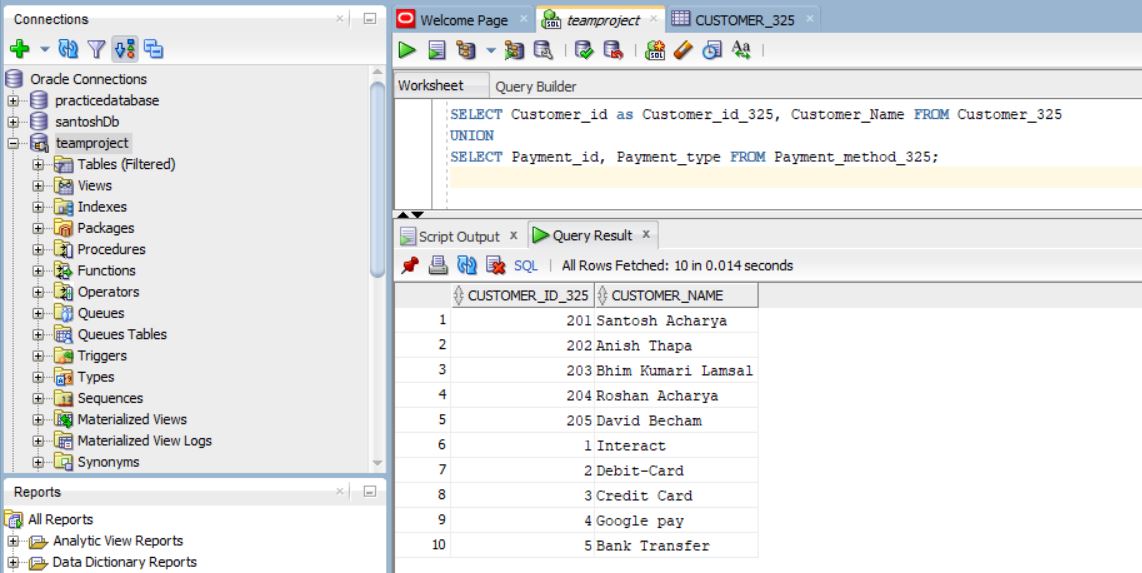
**2**

* UNION is use to combine the result set of two or more select statement.
* SQL Query

|  |
| --- |
| SELECT Customer\_id as Customer\_id\_325, Customer\_Name FROM Customer\_325  UNION  SELECT Payment\_id, Payment\_type FROM Payment\_method\_325; |

* SQL Explanation:

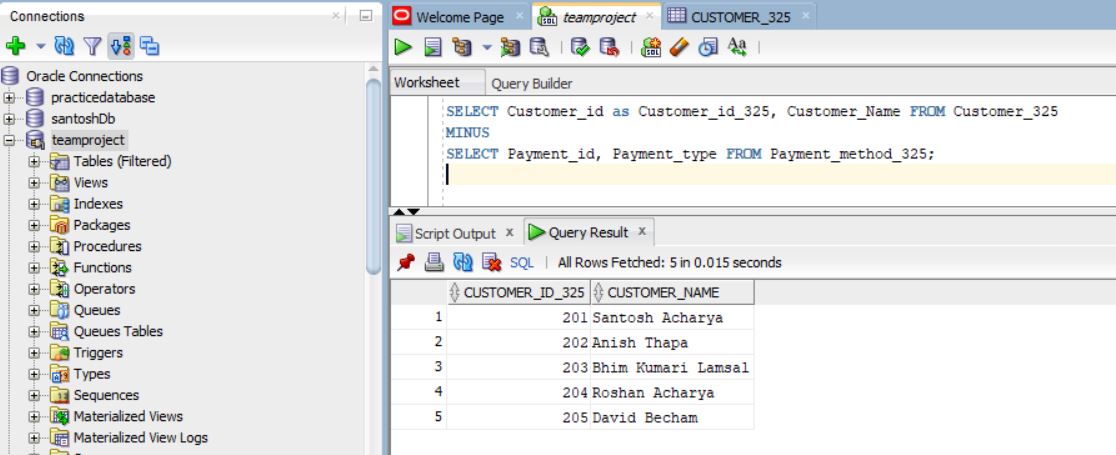
In the query the customer\_325 table is combine with the payement\_method\_325 table.



**3**

* MINUS is the function used to return all the rows in the first select statement that are not returned by the second.
* It will return the unique rows.
* SQL Query

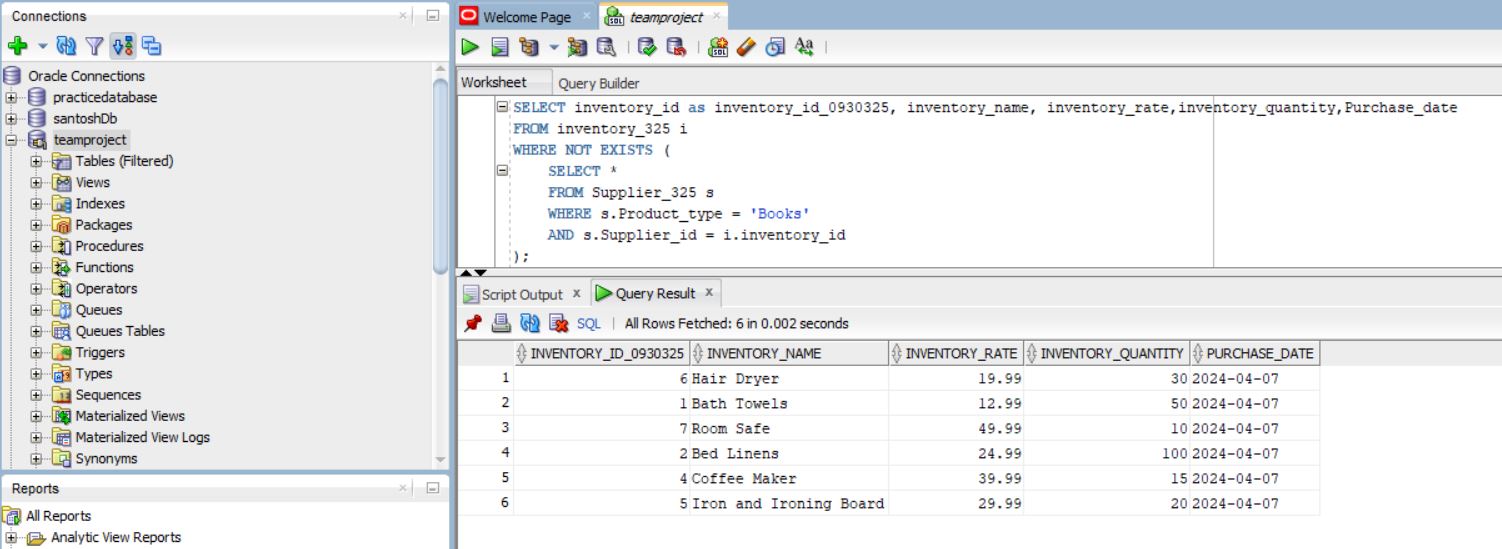
|  |
| --- |
| SELECT Customer\_id as Customer\_id\_325, Customer\_Name FROM Customer\_325  MINUS  SELECT Payment\_id, Payment\_type FROM Payment\_method\_325; |



**4**

* NOT EXISTS is the query that test the non existing value in the table.
* It is used when to ignore the logic of exists.
* SQL Query

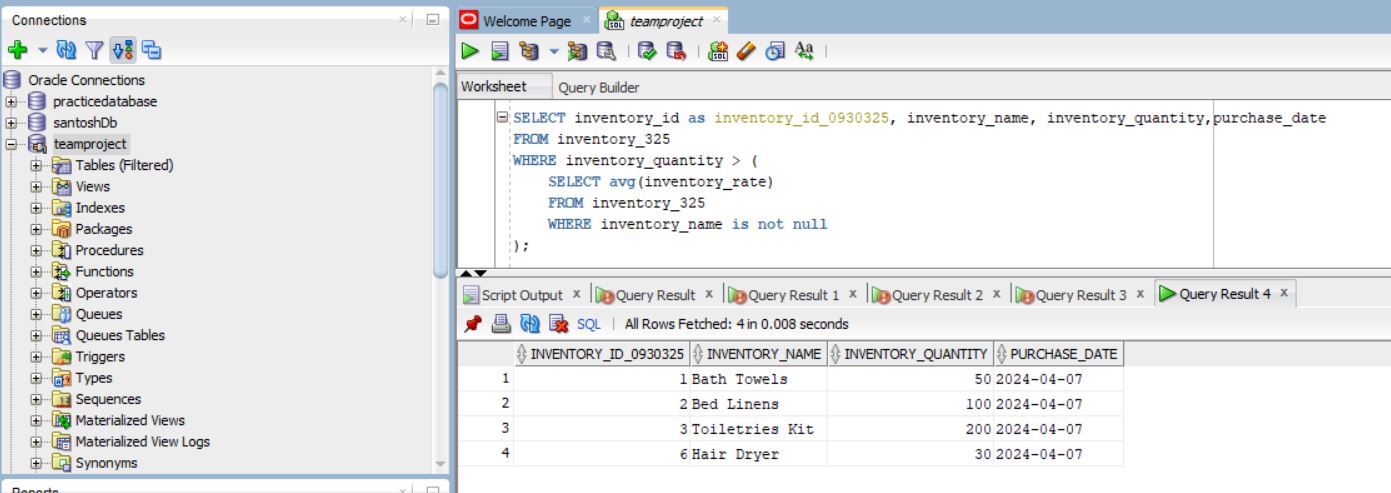
|  |
| --- |
| SELECT inventory\_id as inventory\_id\_0930325, inventory\_name, inventory\_rate,inventory\_quantity,Purchase\_date FROM inventory\_325 i  WHERE NOT EXISTS (  SELECT \* FROM Supplier\_325 s  WHERE s.Product\_type = 'Bath Towels'  AND s.Supplier\_id = i.inventory\_id  ); |



**5**

* Not null is the function that will return the value that aren’t null or have some value in the column or table.
* SQL Query

|  |
| --- |
| SELECT inventory\_id as inventory\_id\_0930325, inventory\_name, inventory\_quantity,purchase\_date FROM inventory\_325  WHERE inventory\_quantity > (  SELECT avg(inventory\_rate) FROM inventory\_325 WHERE inventory\_name is not null  ); |



**Advance SQL**

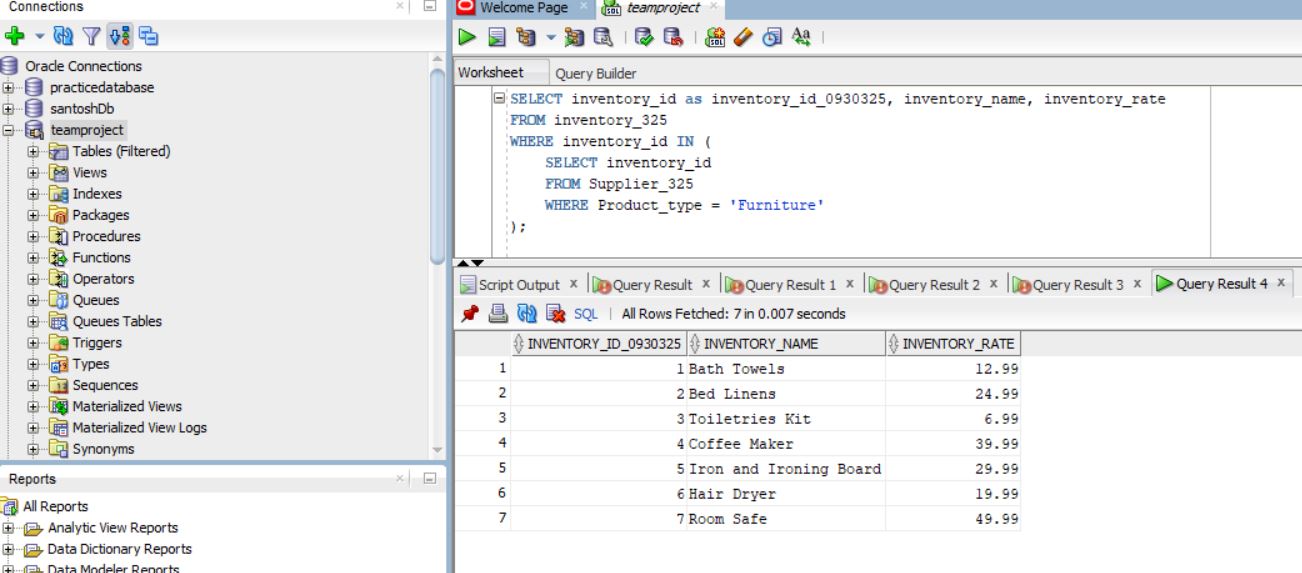
**1**

* In will allow to accept multiple value . It operate as the OR operator in the sql
* SQL Query

|  |
| --- |
| SELECT inventory\_id as inventory\_id\_0930325, inventory\_name, inventory\_rate  FROM inventory\_325 WHERE inventory\_id IN (  SELECT inventory\_id FROM Supplier\_325 WHERE Product\_type = 'Furniture'  ); |

* SQL Query Explanation:

In the query first the inventory id is selected from the suppliers table which match the furniture and then all the number that match in the inventory table is displayed.



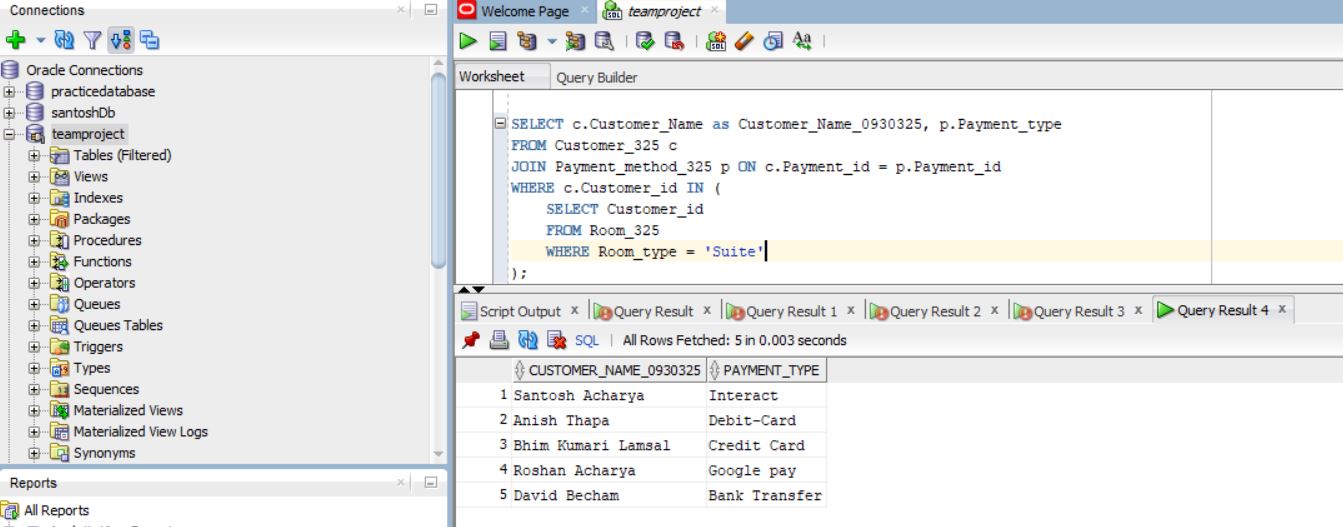
**2**

* JOIN is used to combine the two or more table related to there column.
* SQL Query

|  |
| --- |
| SELECT c.Customer\_Name as Customer\_Name\_0930325, p.Payment\_type  FROM Customer\_325 c  JOIN Payment\_method\_325 p ON c.Payment\_id = p.Payment\_id  WHERE c.Customer\_id IN ( SELECT Customer\_id FROM Room\_325 WHERE Room\_type = 'Suite'  ); |

* SQL Explanation:

In the Query the customerid is selected from the room table and according to the value the customer table and payment method table is combined.



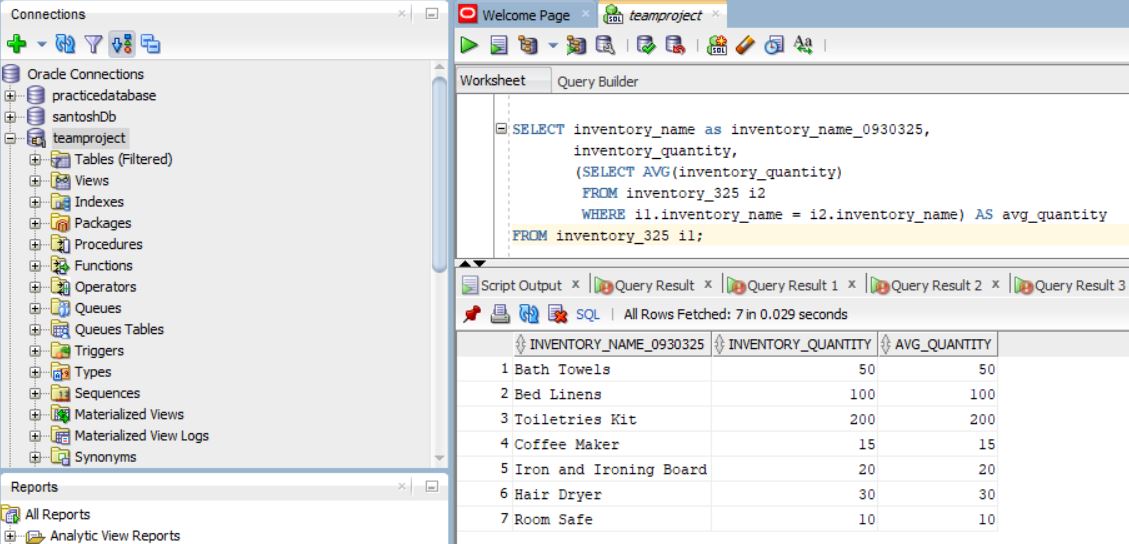
**3**

* Average function will return the average value of the column.
* SQL Query

|  |
| --- |
| SELECT inventory\_name as inventory\_name\_0930325, inventory\_quantity,  (SELECT AVG(inventory\_quantity) FROM inventory\_325 i2  WHERE i1.inventory\_name = i2.inventory\_name) AS avg\_quantity  FROM inventory\_325 i1; |

* SQL Query Explanation:

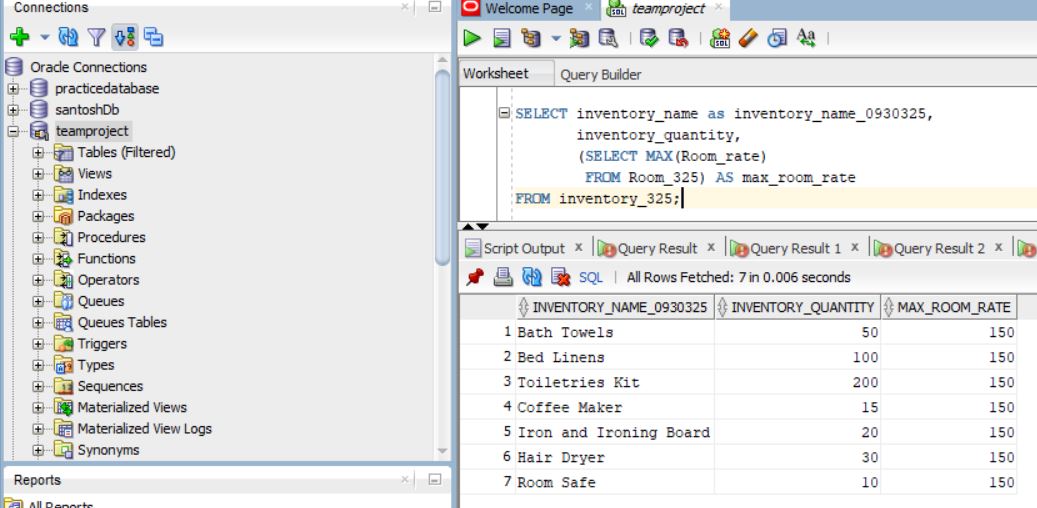
In the query the average of inventory quantity is taken when the inventory\_name of the table is equal to the inventory name of the other table and the output is displayed.



**5**

* MAX function will return the maximum value of the column.
* It is used to get the highest value with in the column.
* SQL Query

|  |
| --- |
| SELECT inventory\_name as inventory\_name\_0930325, inventory\_quantity,  (SELECT MAX(Room\_rate) FROM Room\_325) AS max\_room\_rate  FROM inventory\_325; |



**6**

* Exists function will return the value when the condition is meet.
* It works only if there is one value that will be return.
* SQL Query

|  |
| --- |
| SELECT Customer\_id as Customer\_id\_0930325, customer\_name, customer\_number,membership\_status,payment\_id  FROM Customer\_325 c WHERE EXISTS (  SELECT \* FROM Payment\_method\_325 p WHERE c.Payment\_id = p.Payment\_id  AND p.Payment\_status = 'Full'  ); |

