

**Final Project**

**Database Design – people Operations Database of a Small Businesses(HR database)**

**Final Project for Advanced Database Management System |**

**GROUP RED| ISM6218.004F22.95919**

**Advanced Database Management**

**Team Members**

Abbas Shaik

Srikar Pusuluri

Sai Shiva Ram Reddy Vudumula

Nithin Reddy Muduganti

Dharma Rakshak Tadi

# 

# Summary of Contents

|  |  |  |
| --- | --- | --- |
| **Serial No.** | **Title** | **Page No.** |
| 1. | Responsibilities | **3** |
| 2. | Entities to be Tracked | **4** |
| 3. | Entities with Attributes Nested | **5** |
| 5. | Data Dictionary | **6** |
| 6. | Entity Relationship Documentation (ERD) | **8** |
| 7. | Table Views | **7** |
| 8. | Data Synthesis | **11** |
| 9. | Data Visualization | **11** |
| 9. | Data Integrity | **9** |
| 10. | Test Queries/Stored Proc | **10** |
| 11. | Performance Tuning | **14** |
| 13. | DBA Scripts | **16** |

## **Responsibilities**

|  |  |  |  |
| --- | --- | --- | --- |
| **Topic Area**  **Database Design**  **Query Writing** | **Description**  This part should include a logical database design (for the relational model), using normalization to control redundancy and integrity constraints for data quality  This part is another chance to write SQL queries, explore transactions, and even do some database programming for stored procedures. | **Group Member**  Sai Shiva Ram Reddy Vudumula/ Abbas Shaik  Abbas Shaik / Srikar Pusuluri / Sai Shiva Ram Reddy Vudumula | **Weight**  28  24 |
| **Data visualization**  **Other Topics** | In this section, you can capitalize and extend your prior experiments visualization, optimizer modes, graphs, data modelling and any other techniques you want to further explore.  Here you are free to explore any other topics of interest. Suggestions include: DBA scripts, database security, interface design, data mining, and NoSQL databases | Sai Shiva Ram Reddy Vudumula  Nithin Reddy Muduganti/  Srikar Pusuluri  Abbas Shaik Srikar Pusuluri  Nithin Reddy Muduganti  Dharma Rakshak Tadi | 25  23 |

**Entities identified to be tracked**

| **Table** | **Rows** |
| --- | --- |
| Workforce2 | 40 |
| Nominee2 | 30 |
| Division2 | 11 |
| Position2 | 11 |
| Site2 | 7 |
| Country2 | 25 |
| Area2 | 4 |

**Entities with Attributes nested**

**❖** workforce2**:**

* 1. **worker\_id first\_name**
  2. **last\_name**
  3. **email**
  4. **phone\_number**
  5. **hire\_date**
  6. **position\_id**
  7. **salary**
  8. **manager\_id**
  9. **division\_id**
  10. **FOREIGN KEY**

**FOREIGN KEY (division\_id) REFERENCES division2 (division\_id) ON DELETE CASCADE,**

* **Position2:**

1. **position\_id,**
2. **position\_title,**
3. **position\_min\_salary,**
4. **position\_max\_salary**

* nominee2:

1. **nominee\_id,**
2. **first\_name,**
3. **last\_name,relation,**
4. **worker\_id**

* division2:

1. **division\_id,**
2. **division\_name,**
3. **location\_id**

* area2:

1. **area\_id,**
2. **area\_name**

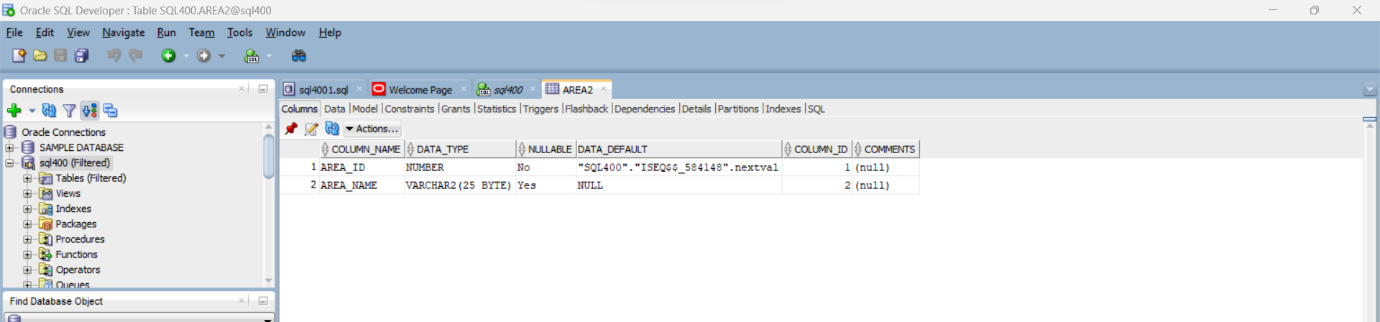
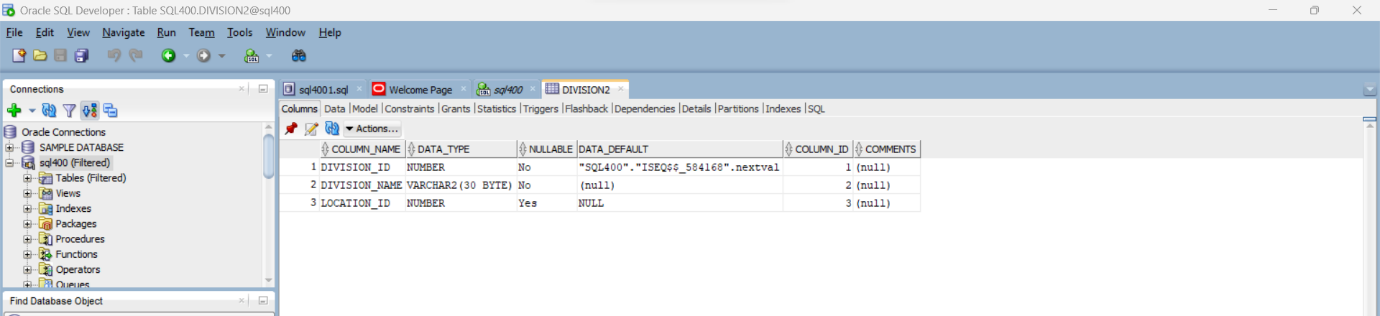
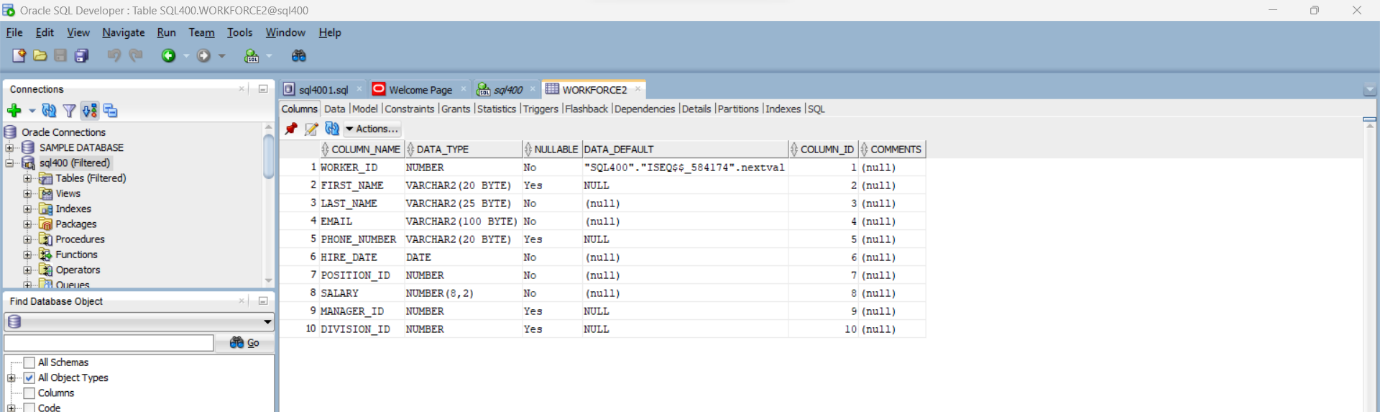
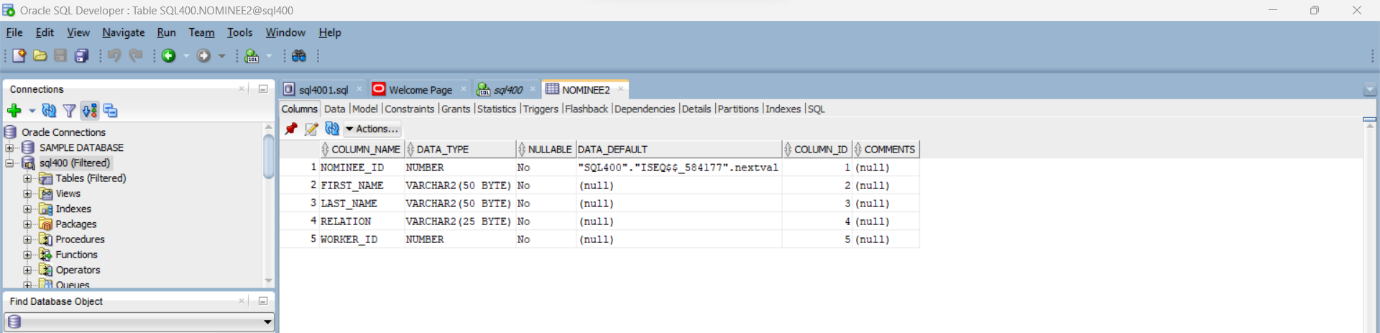
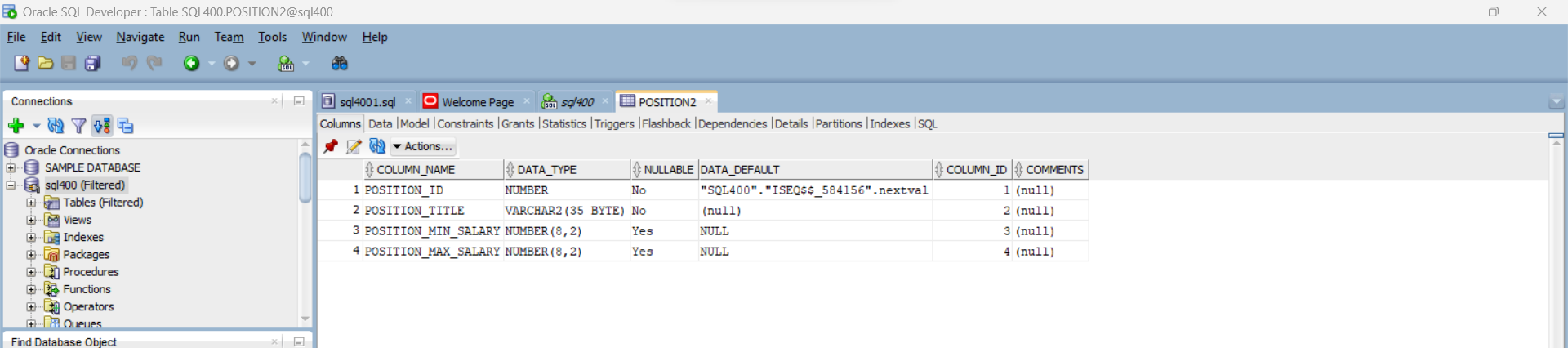
* country:

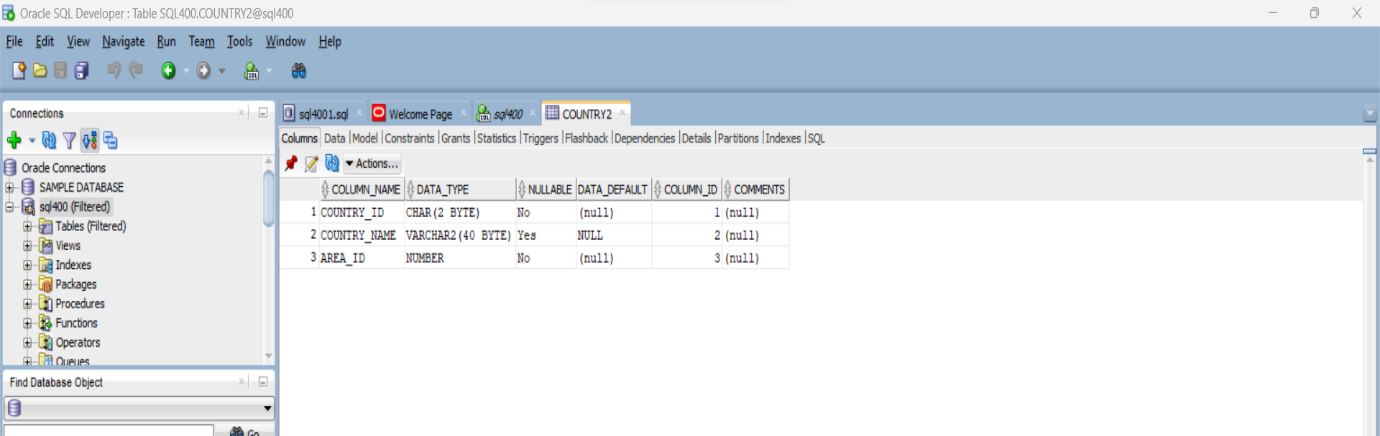
1. **(country\_id,**
2. **country\_name,**
3. **area\_id)**

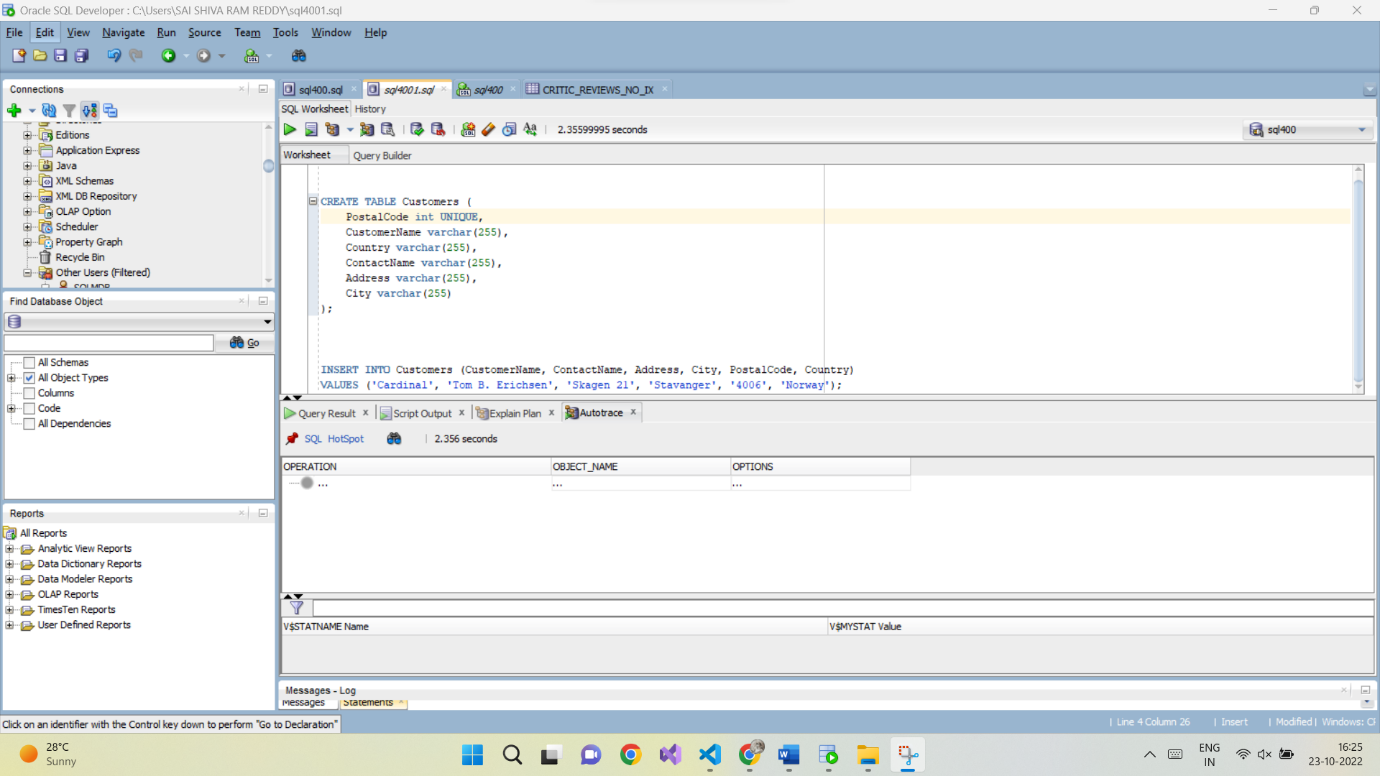
* site

1. **location\_id**
2. **street\_address,**
3. **pin\_code,**
4. **city,**
5. **state\_province,**
6. **country\_id**

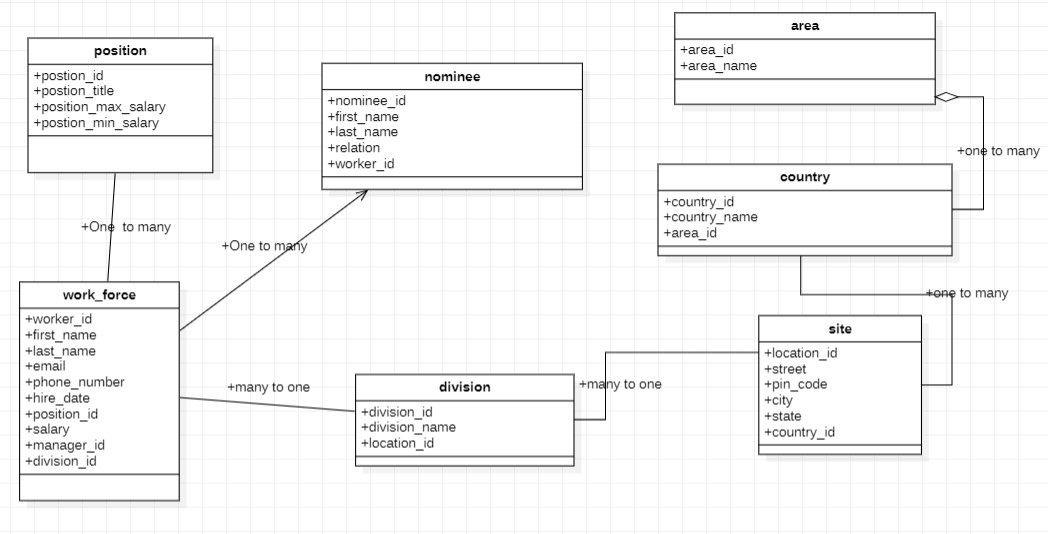
**DATA DICTIONARY**

****

****



**Entity Relationship Diagram representing Database Design**



**Data explanation:**

The people operation database has seven tables:

1. The workforce table stores the data of employees.
2. The position table stores the job data including job title and salary range.
3. The division table stores department data.
4. The nominee table stores the employee’s dependents.
5. The site table stores the location of the departments of the company.
6. The country table stores the data of countries where the company is doing business.
7. The area table stores the data of regions such as Asia, Europe, America, and the Middle East and Africa. The countries are grouped into regions

**Data Synthesis:**

The data for the project has been synthesized using a combination of an online tool named Mockaroo and Microsoft Excel. Some of the prominent functions that were used in Excel include

* INDEX
* CONCAT
* RAND

|  |  |
| --- | --- |
| Workforce2 | 40 |
| Nominee2 | 30 |
| Division2 | 11 |
| Position2 | 11 |
| Site2 | 7 |
| Country2 | 25 |
| Area2 | 4 |

* RANDBETWEEN

**Data Integrity**

Data Integrity refers to the consistency and maintenance of the data through the life cycle of the database. In a database, data integrity can be ensured through the implementation of Integrity Constraints in a table. Integrity constraints help apply business rules to the database tables. The constraints can either be at a column level or a table level. Some of the most common constraints are:

* NOT NULL – Prevents a column from having a NULL value.

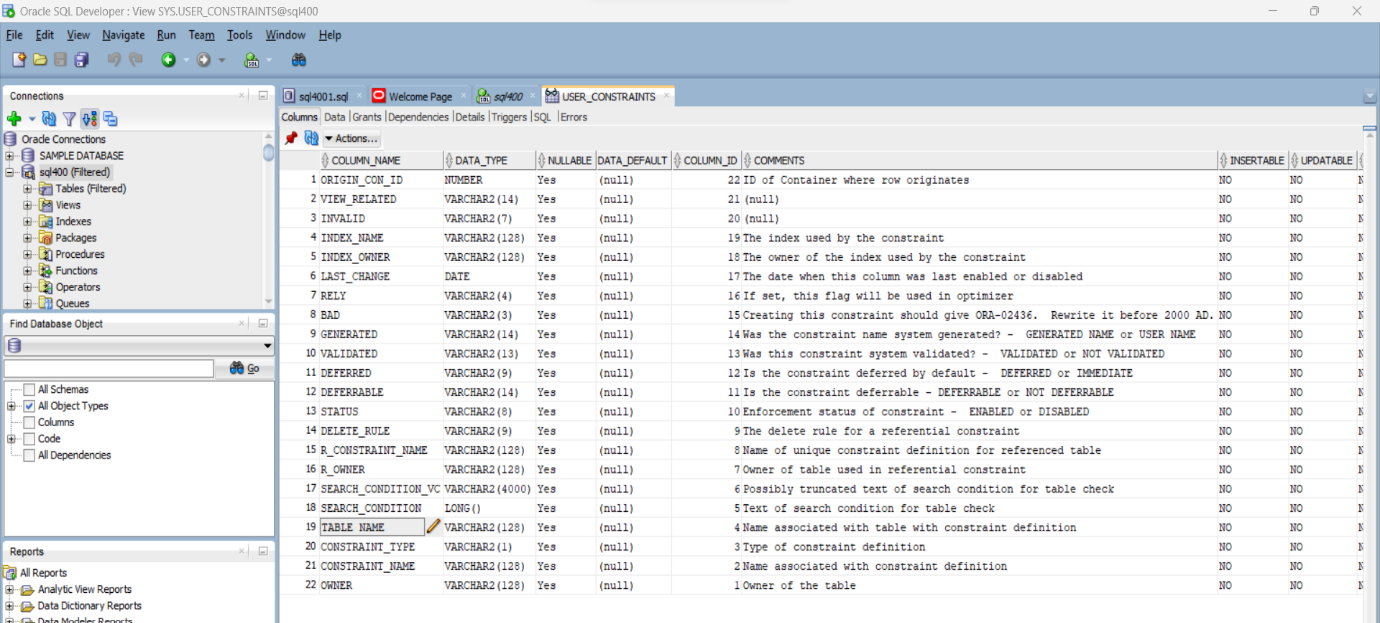
* PRIMARY KEY – Uniquely identifies each row or record in table.

* FOREIGN KEY – Uniquely identifies a column that references a PRIMARY KEY in another table.

* UNIQUE – Prevents a column from having duplicate values.

* CHECK – Checks for values that satisfy a specific condition as defined by the user.

Listed below are the constraints that were created for our database development project along with their purpose:



**Data constraints:**

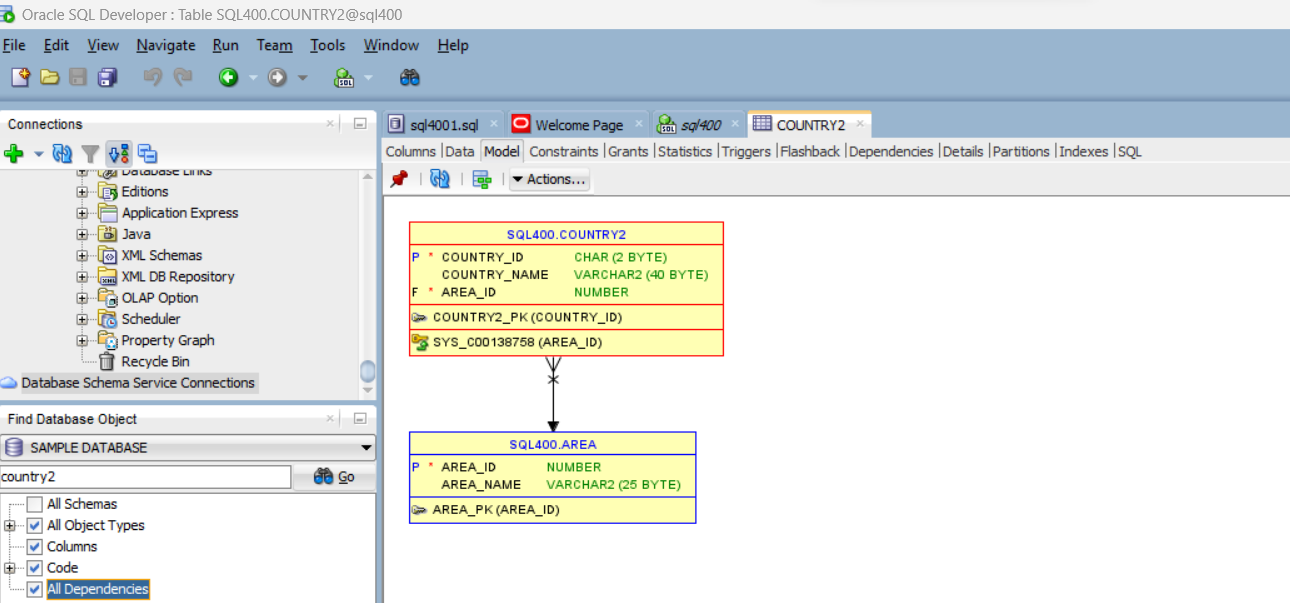
SQL constraints are used to specify rules for the data in a table.

Constraints are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the table. If there is any violation between the constraint and the data action, the action is aborted.

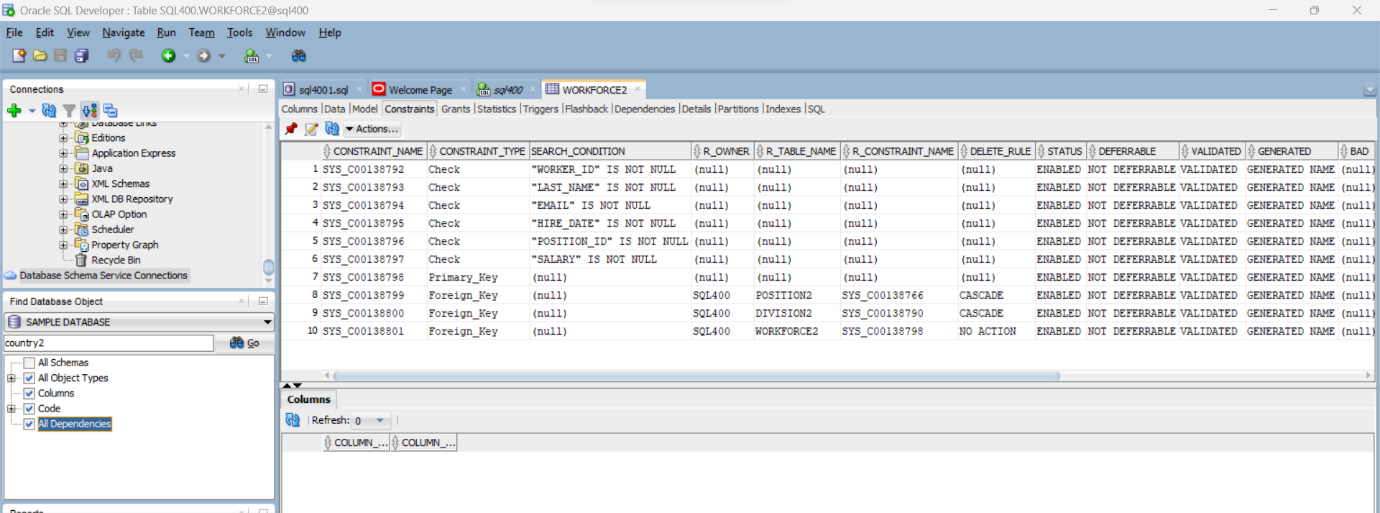
Constraints can be column level or table level. Column level constraints apply to a column, and table level constraints apply to the whole table.

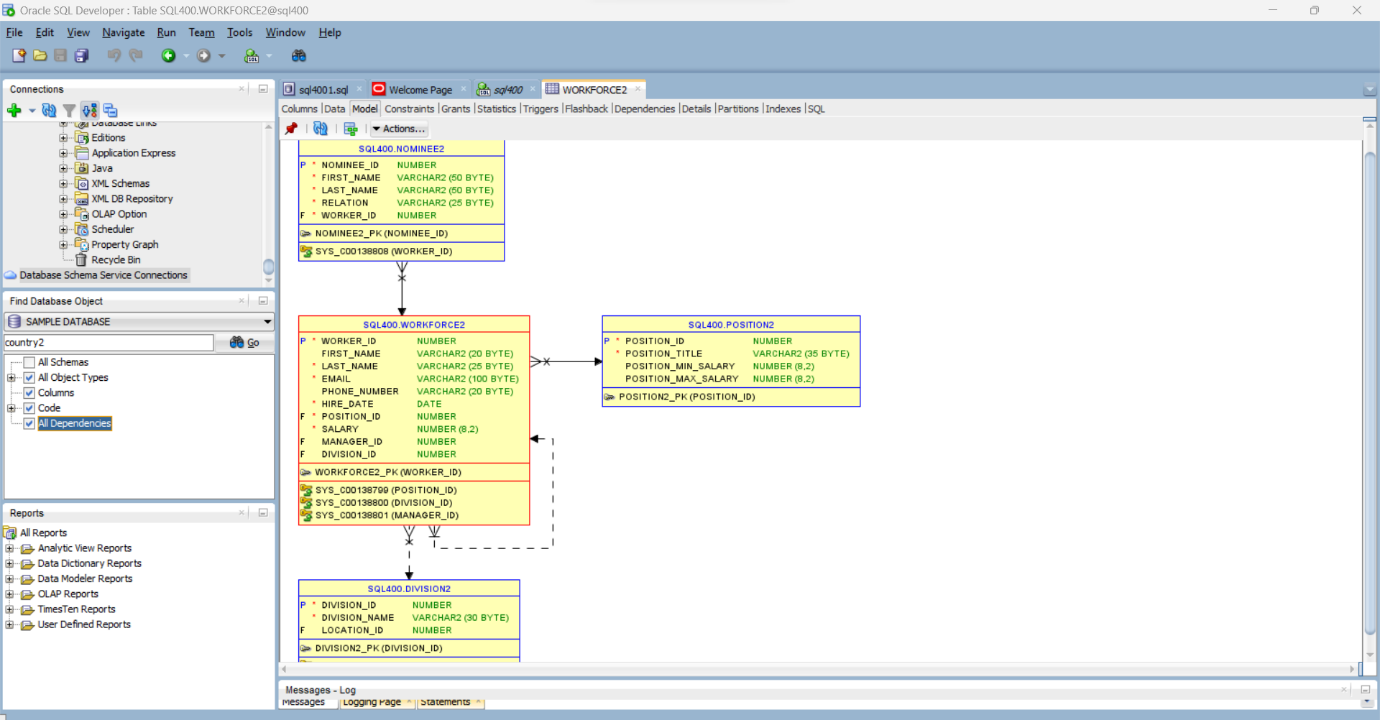
**Country2 table:**

****

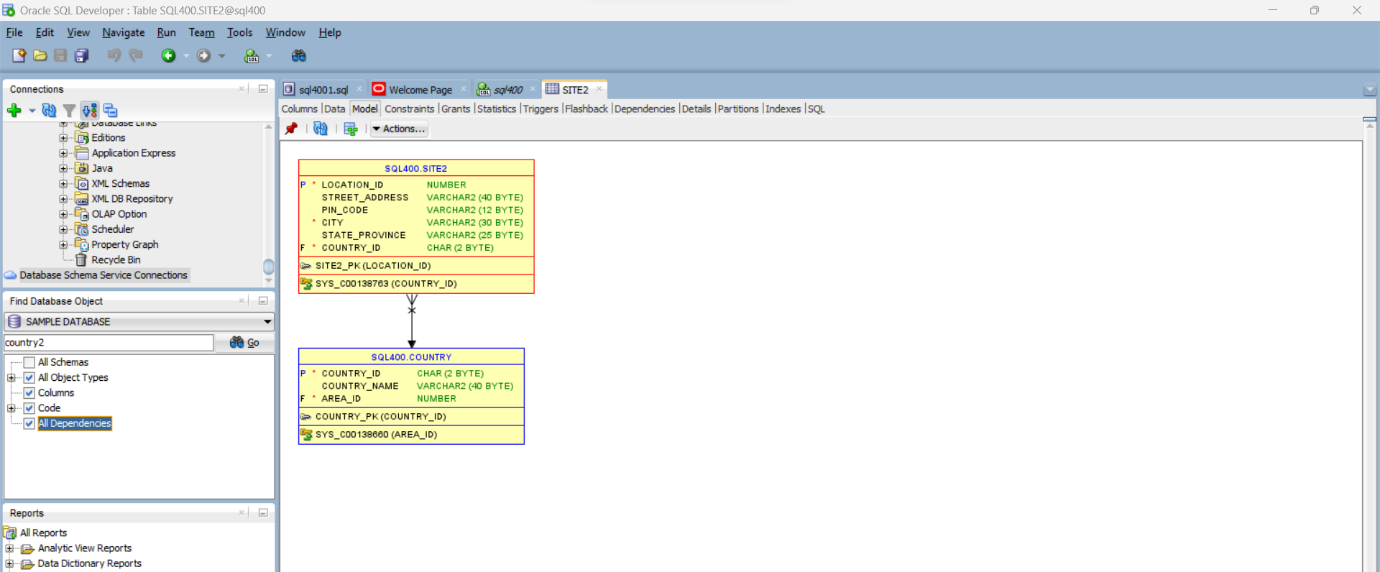
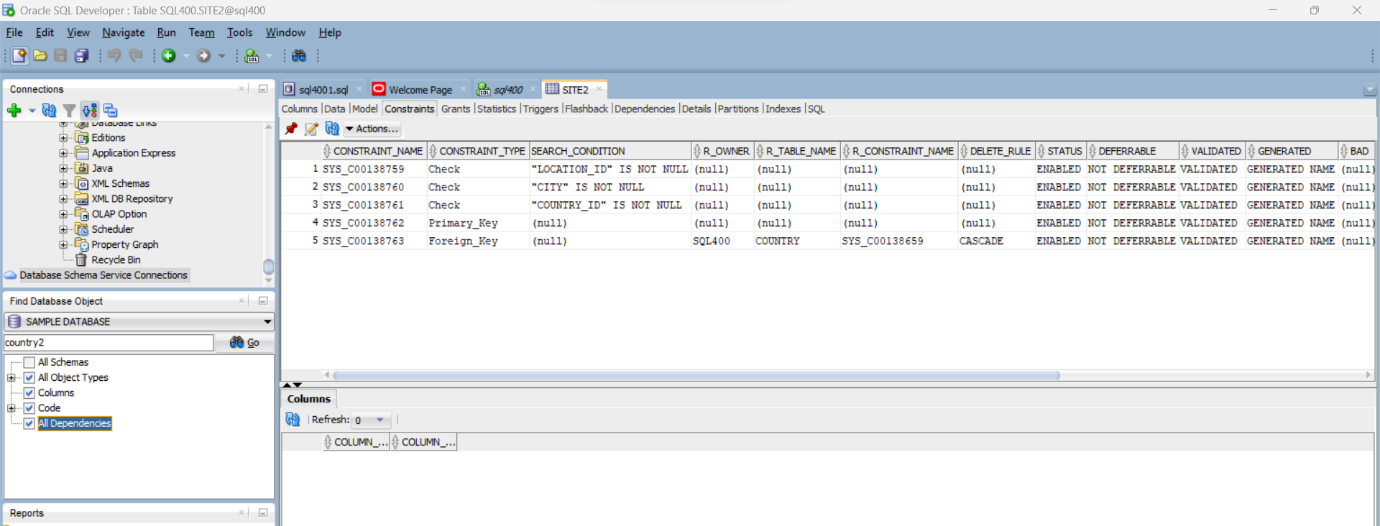
****

**Workforce2 :**

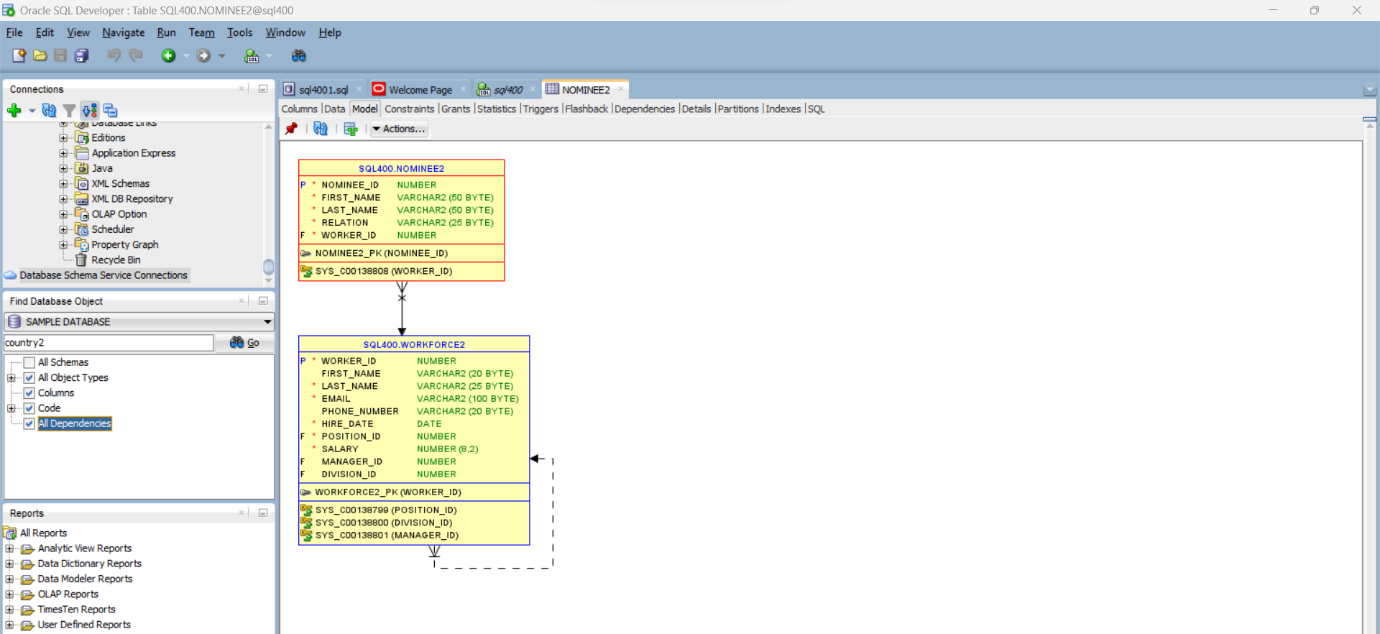
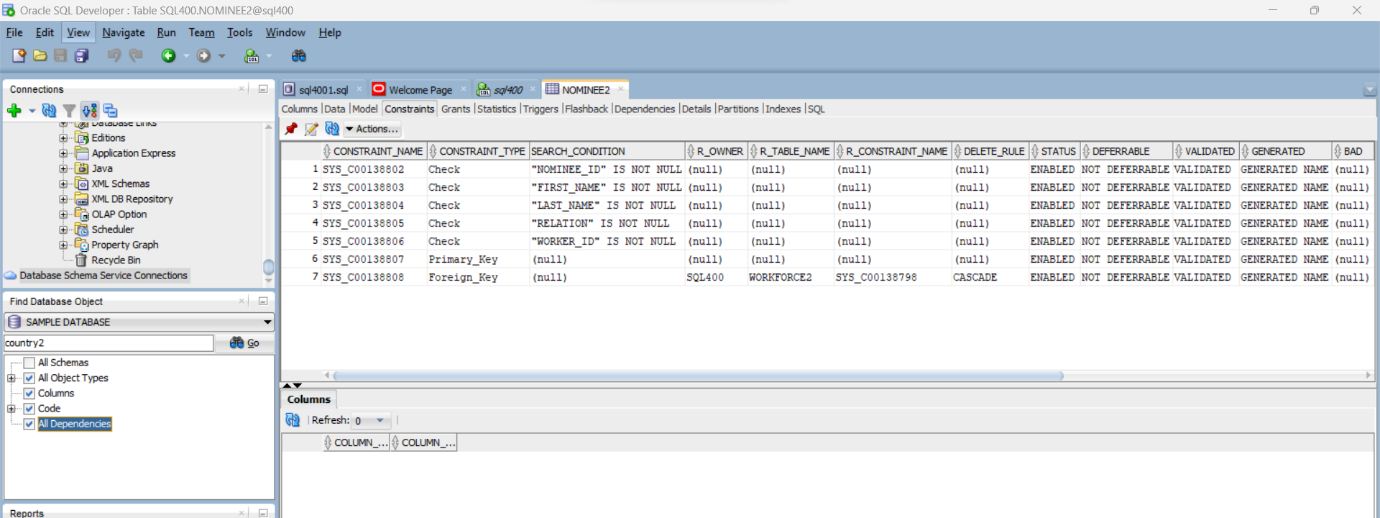
****

****

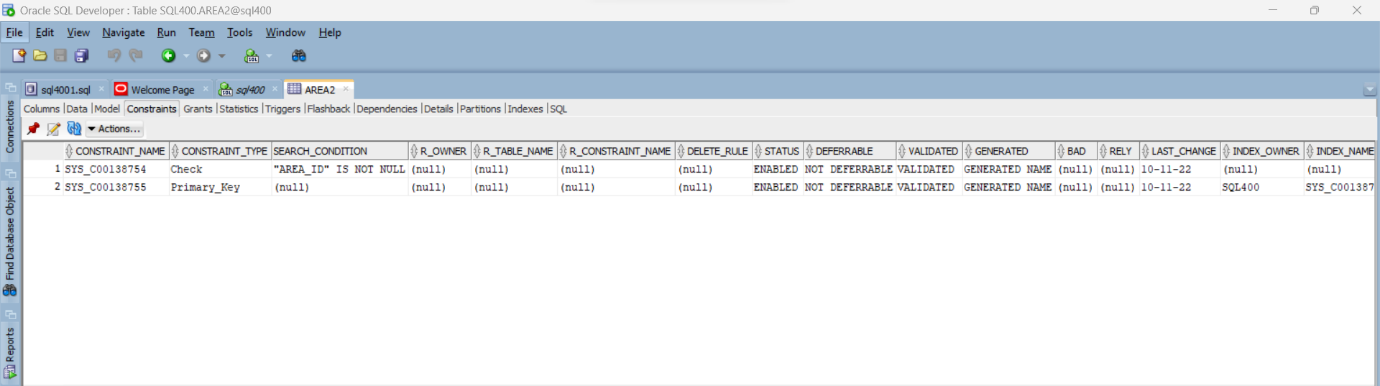
**Site2:**

****

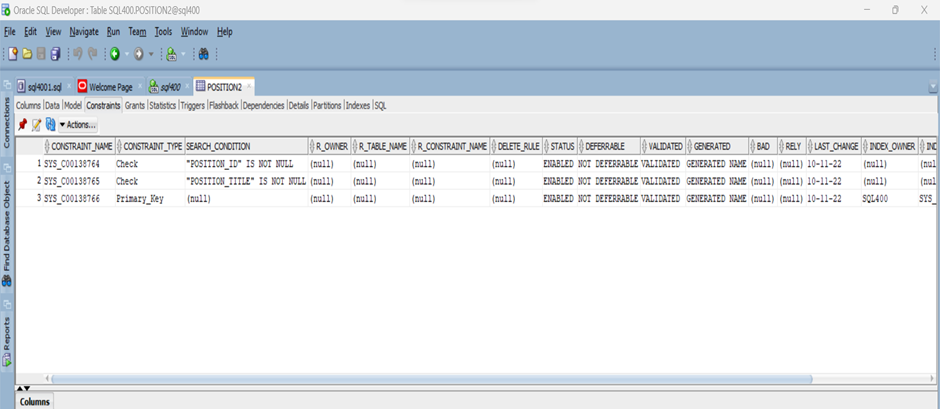
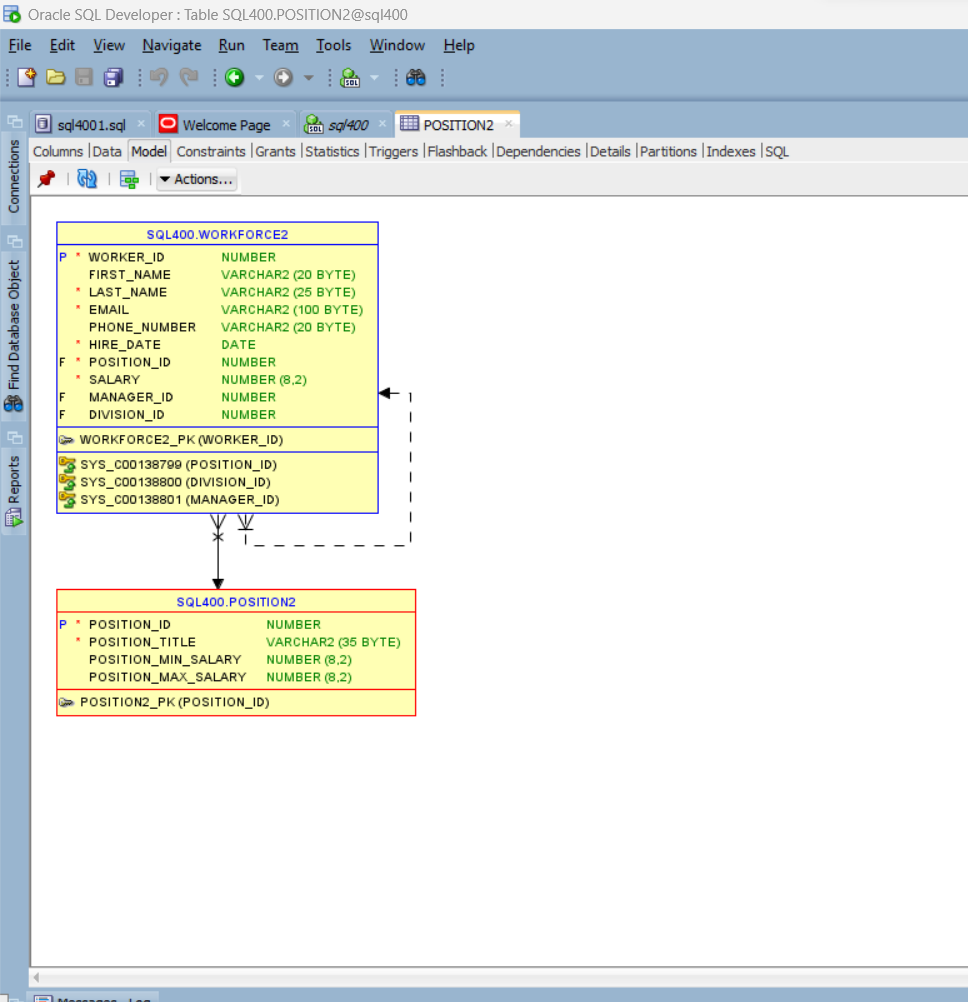
**Nominee2:**

****

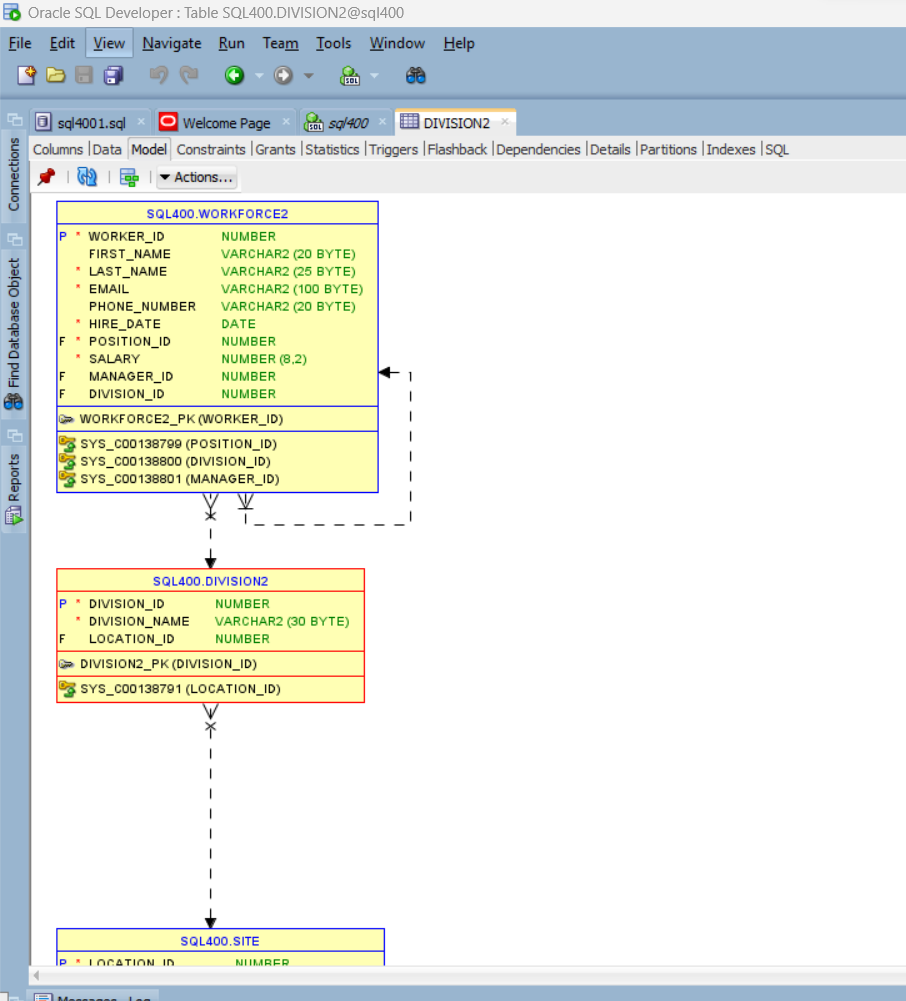
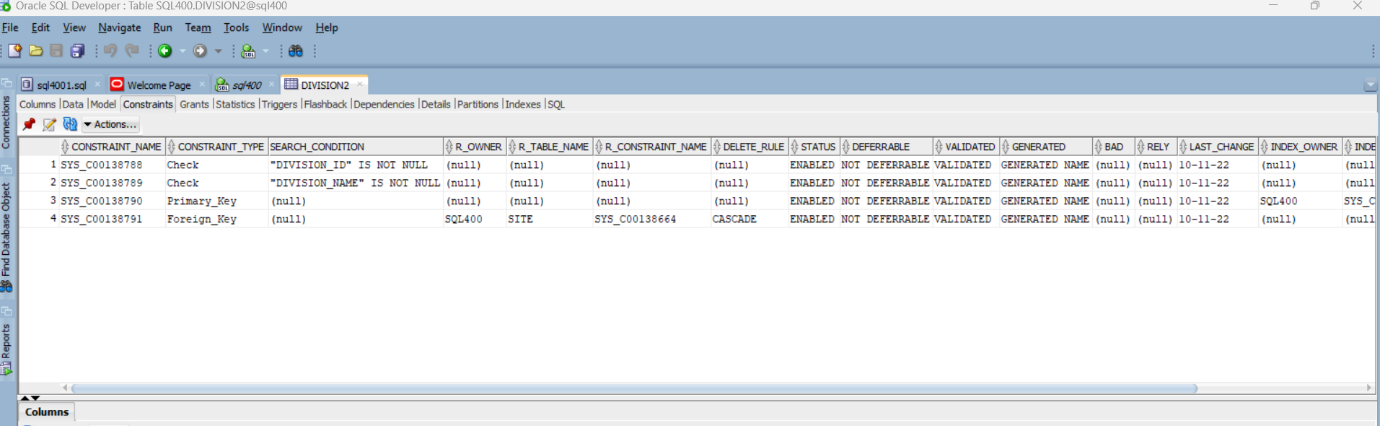
**Area2:**

****

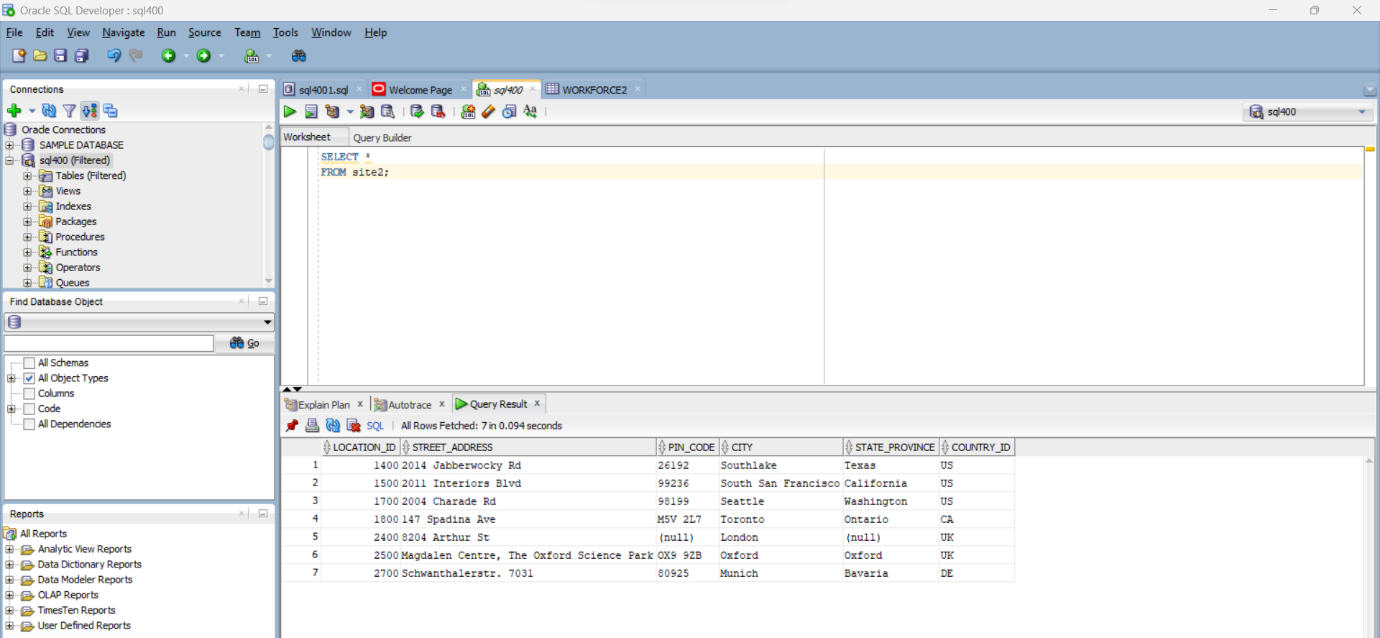
**Position2:**

****

**Division2:**

****

**QUERIES**



SELECT

division\_id,

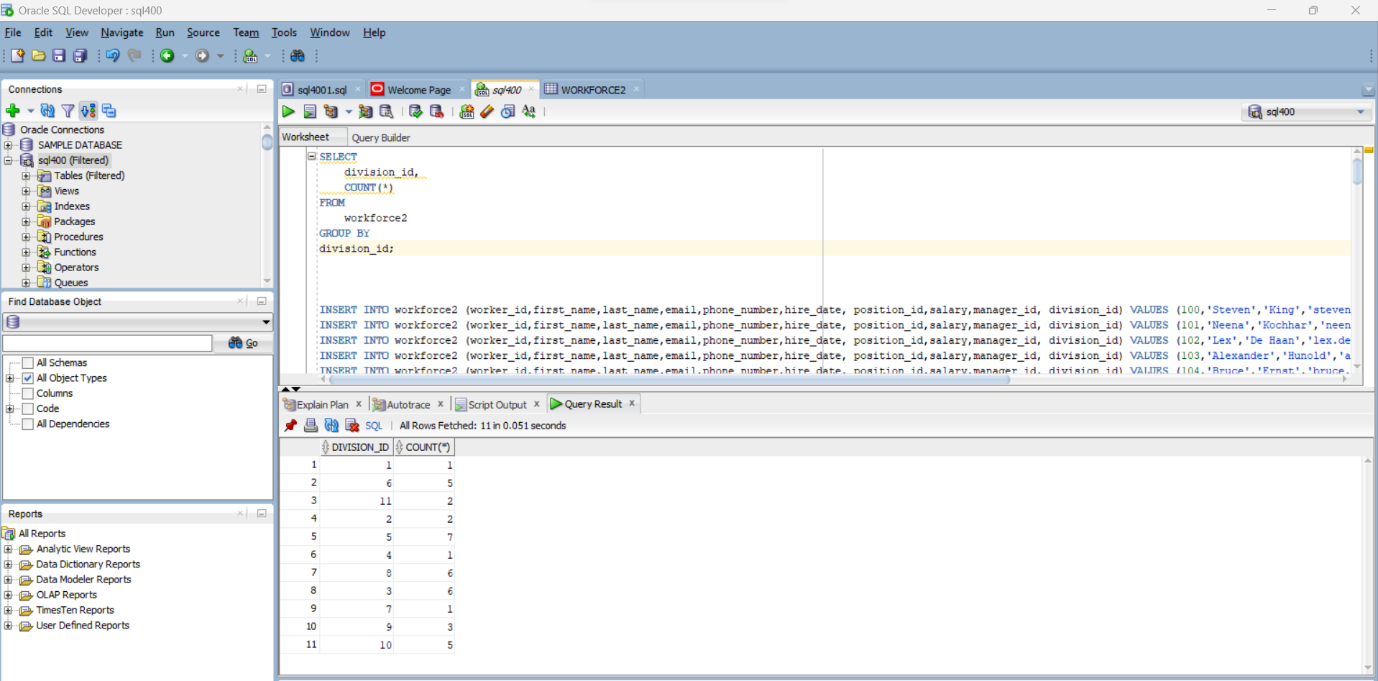
COUNT(\*)

FROM

workforce2

GROUP BY

division\_id;



SELECT

first\_name,

last\_name

FROM

workforce2

UNION

SELECT

first\_name,

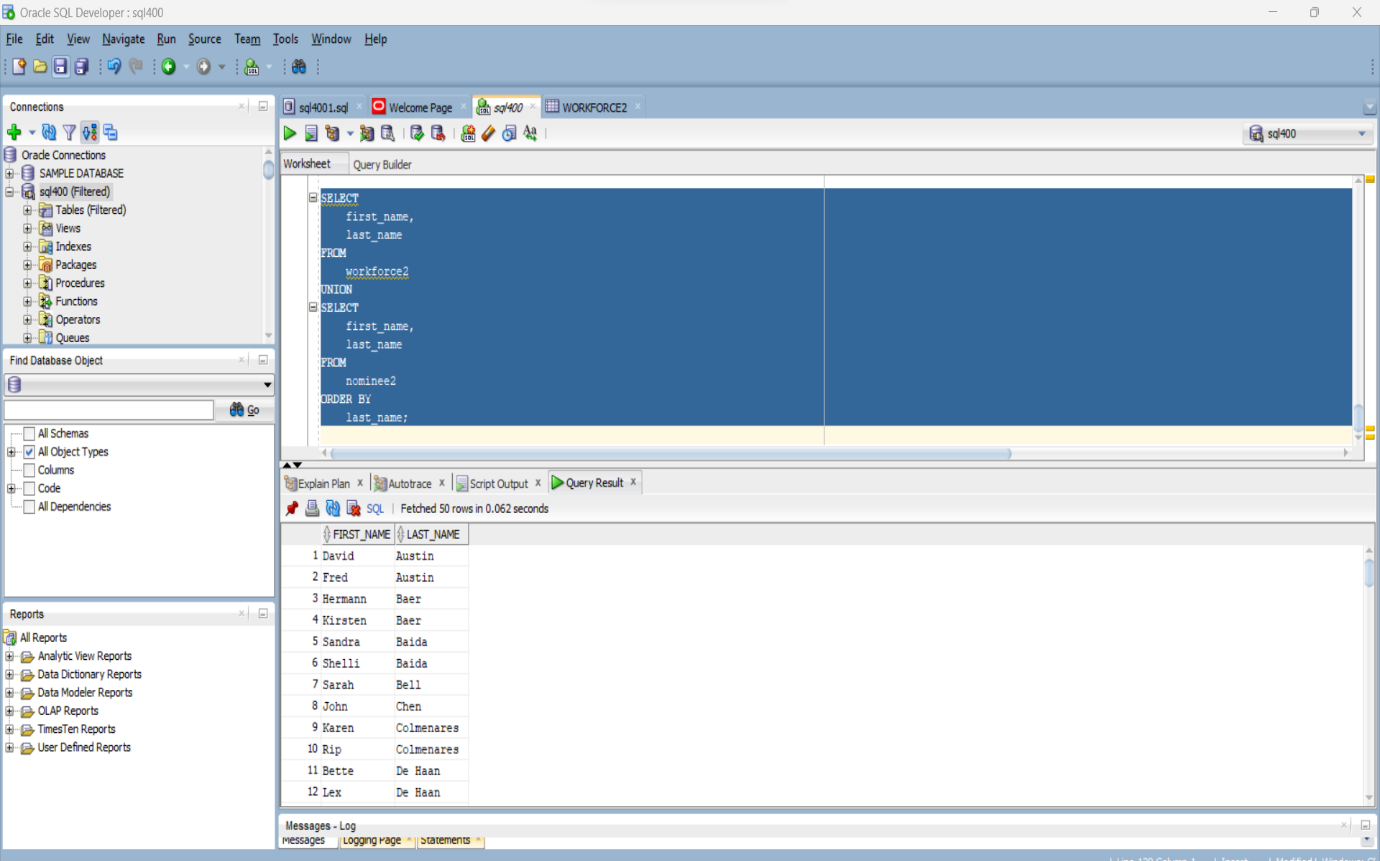
last\_name

FROM

nominee2

ORDER BY

last\_name;



SELECT

division\_name

FROM

division2 d

WHERE

EXISTS( SELECT

1

FROM

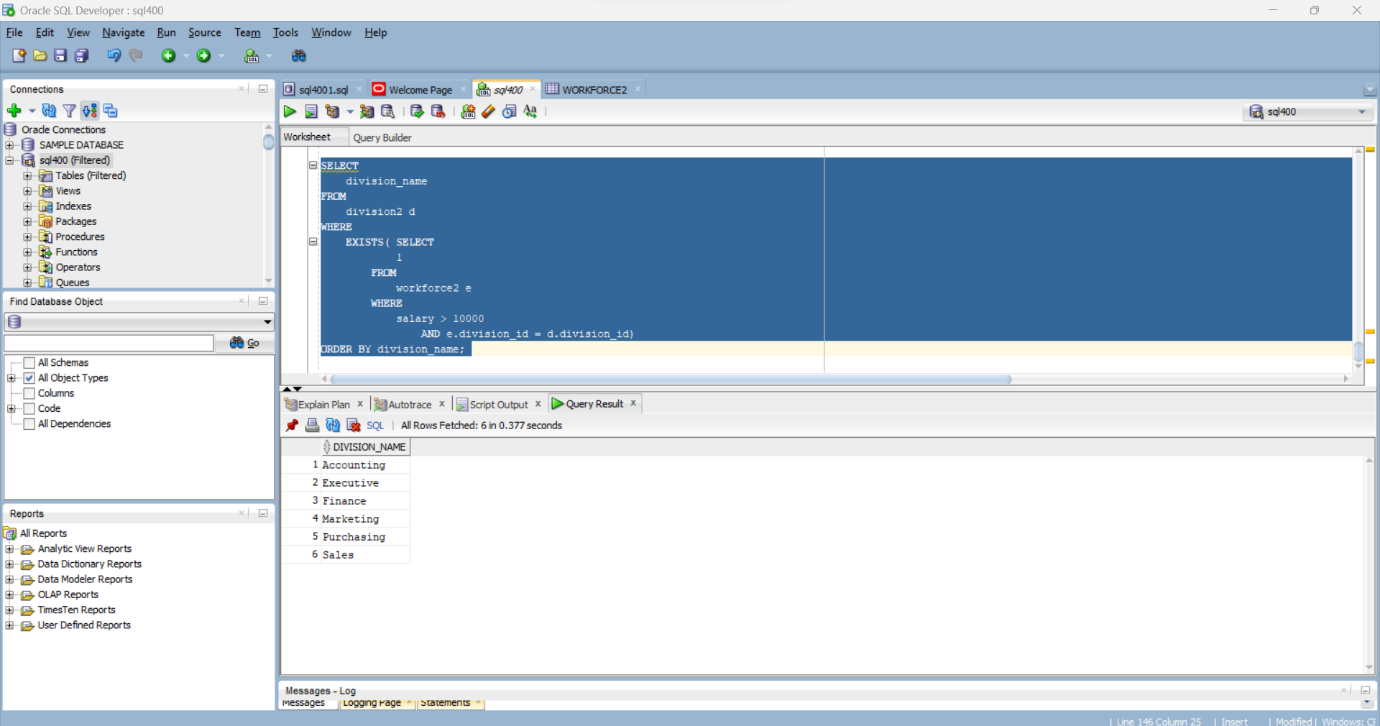
workforce2 e

WHERE

salary > 10000

AND e.division\_id = d.division\_id)

ORDER BY division\_name;



SELECT

worker\_id,

first\_name,

last\_name,

hire\_date,

salary

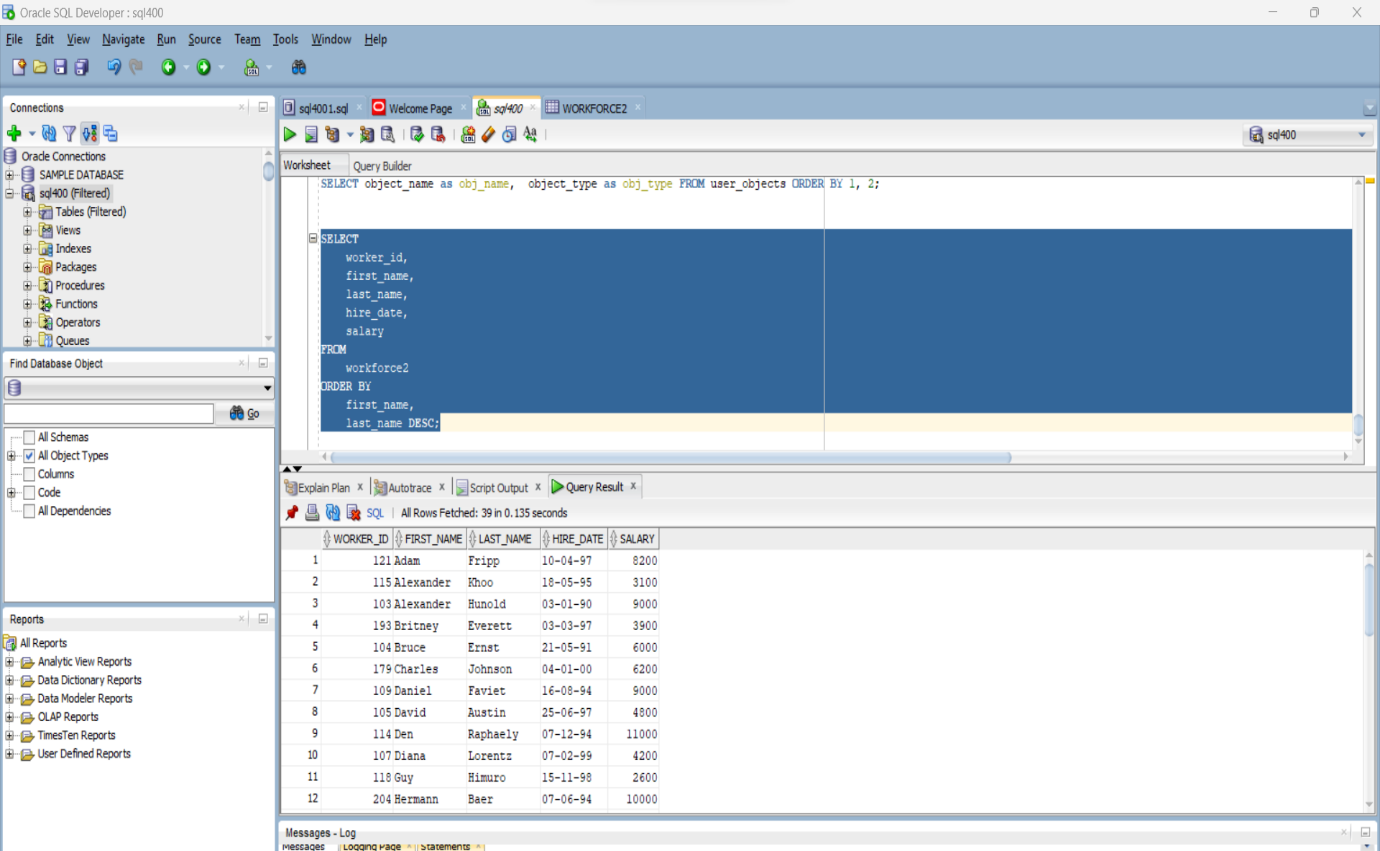
FROM

workforce2

ORDER BY

first\_name,

last\_name DESC;



SELECT

e.division\_id,

division\_name,

COUNT(\*)

FROM

workforce2 e

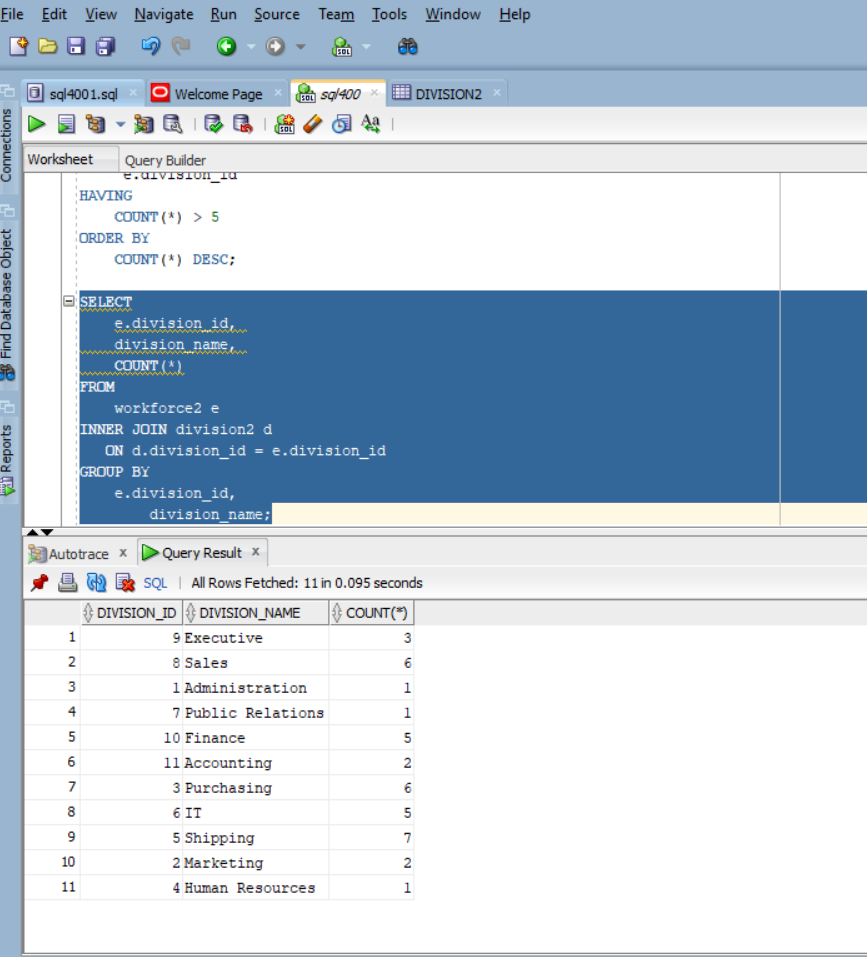
INNER JOIN division2 d

ON d.division\_id = e.division\_id

GROUP BY

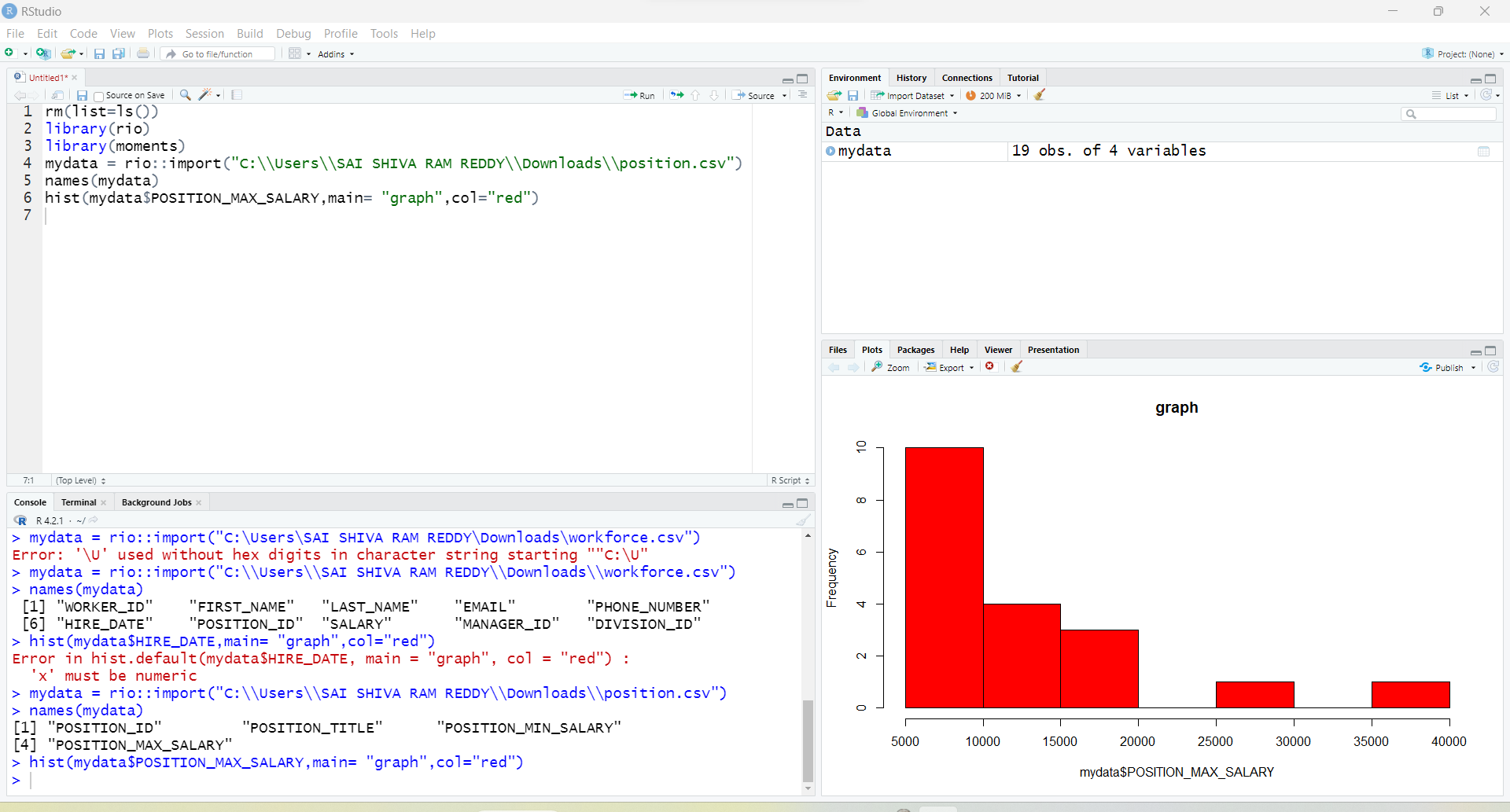
e.division\_id,

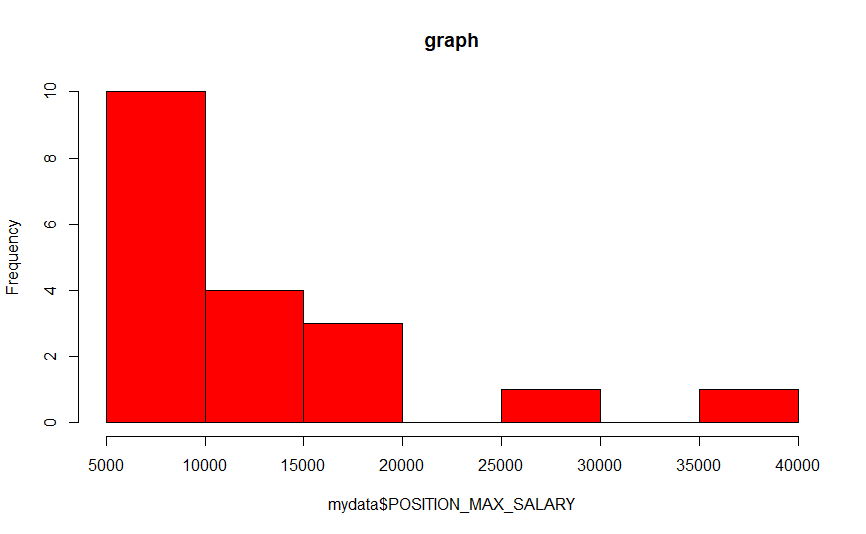
division\_name;

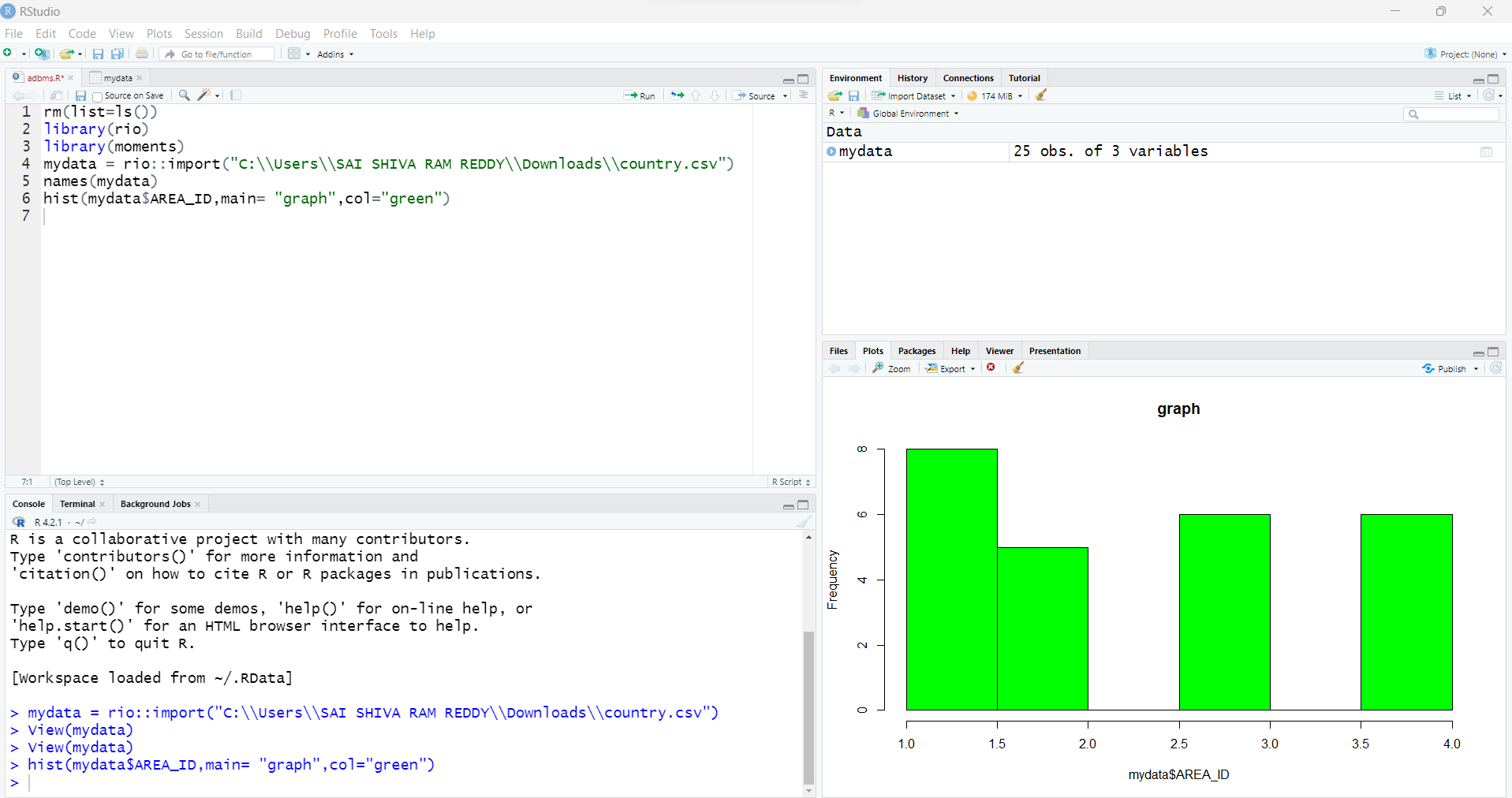


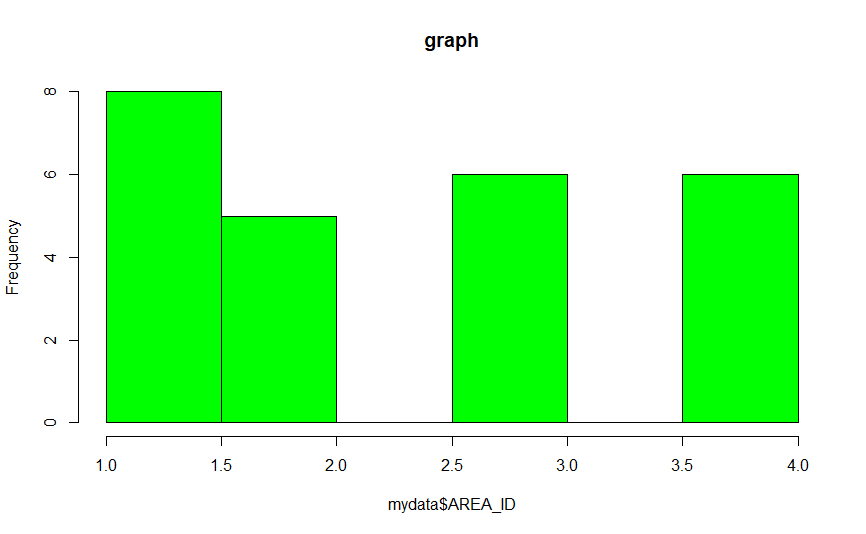
**Data Visualization**

The data was analyzed in R and Excel







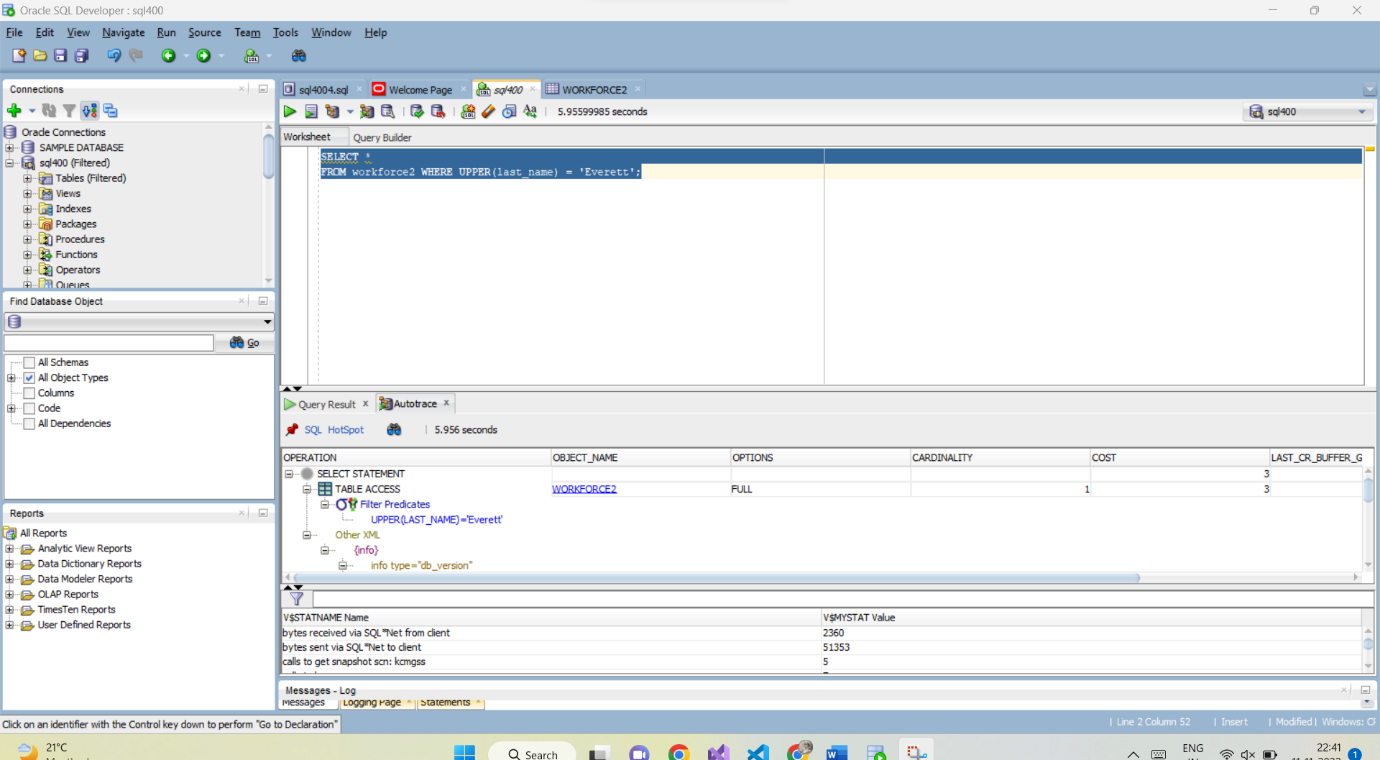


**Performance tuning and indexing:**

SELECT \*

FROM workforce2 WHERE (last\_name) = 'Everett';

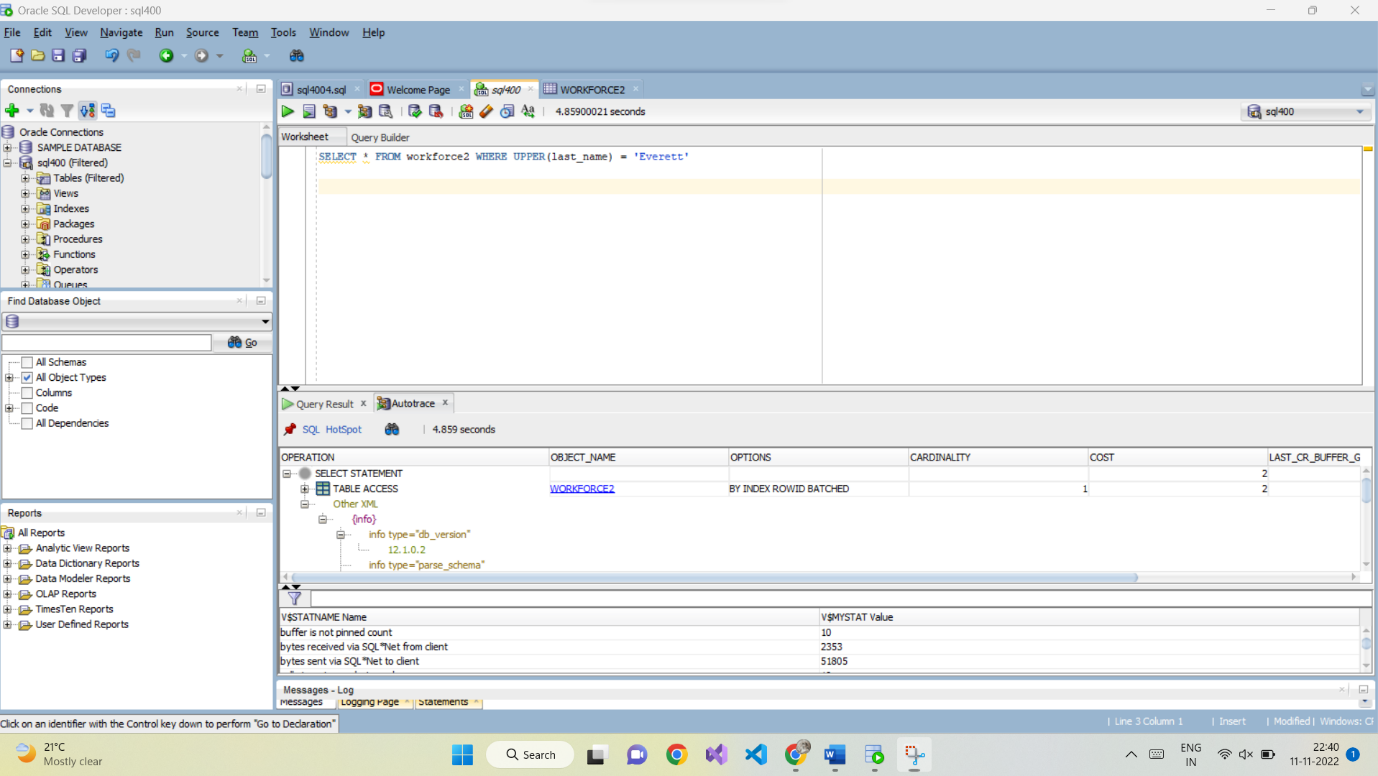
Here we can observe both cardinality and cost of the result which increases by inputing the indexing



SELECT \*

FROM workforce2 WHERE UPPER(last\_name) = 'Everett';

Now observe the same by using the indexing



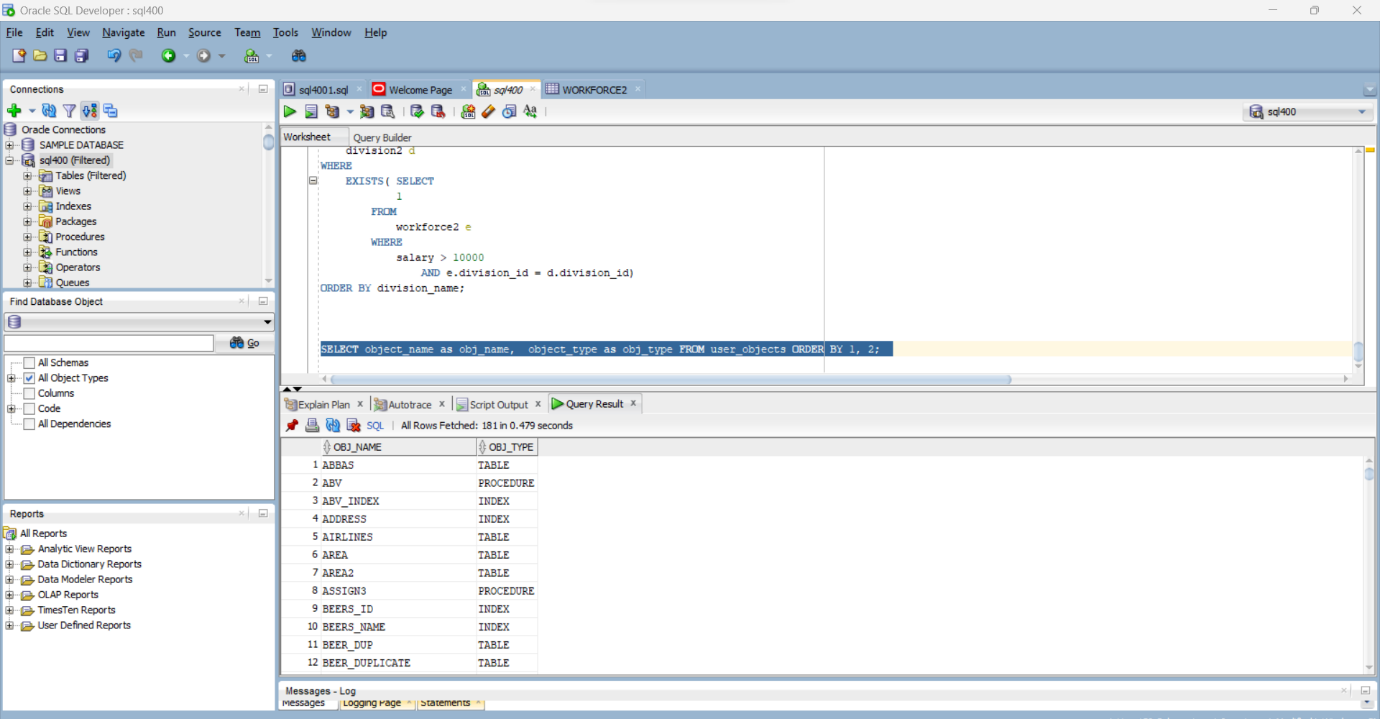
Here we can observe cost decreases and time also reduces

**DBA SCRIPTS**

1.

SELECT object\_name as obj\_name, object\_type as obj\_type FROM user\_objects ORDER BY 1, 2;

**Description**: The above sql command accesses user\_objects and displays object\_name and object type for current user.



SET SERVEROUTPUT ON

SET VERIFY OFF

SELECT

a.granted\_role, a.admin\_option

FROM

user\_role\_privs a

ORDER BY a.granted\_role;

SELECT

a.privilege,

a.admin\_option

FROM user\_sys\_privs

ORDER BY a.privilege;

SET VERIFY ON

**DBA script…**

SELECT t.table\_name,

c.constraint\_name,

c.table\_name workforce2,

worker\_id.column name

FROM all\_constraints t,

all constraints c,

all\_cons\_columns worker\_id

WHERE c.I\_constraint\_name=t.constraint\_name

AND OAND

c.table\_name

acc.table\_name

c.constraint\_name

acc.constraint\_name

AND NOT EXISTS (SELECT 1

FROM WHERE

all\_ind\_columns aid aid. table\_name

- acc.table\_name AND aid.column\_name acc.column\_name)

0RDER BY c.tabte\_name:

SELECT

privilege, grantee, admin\_option

FROM dba\_sys\_priva WHERE privilege LIKE UPPER (1) ORDER BY

privilege, grantee:

**creating the tables:**

CREATE TABLE area2 (

area\_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,

area\_name VARCHAR2 (25) DEFAULT NULL

);

CREATE TABLE country2 (

country\_id CHAR (2) PRIMARY KEY,

country\_name VARCHAR2 (40) DEFAULT NULL,

area\_id NUMBER NOT NULL,

FOREIGN KEY (area\_id) REFERENCES area (area\_id) ON DELETE CASCADE

);

CREATE TABLE site2 (

location\_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,

street\_address VARCHAR2 (40) DEFAULT NULL,

pin\_code VARCHAR2 (12) DEFAULT NULL,

city VARCHAR2 (30) NOT NULL,

state\_province VARCHAR2 (25) DEFAULT NULL,

country\_id CHAR (2) NOT NULL,

FOREIGN KEY (country\_id) REFERENCES country (country\_id) ON DELETE CASCADE

);

CREATE TABLE position2 (

position\_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,

position\_title VARCHAR2 (35) NOT NULL,

position\_min\_salary NUMBER (8, 2) DEFAULT NULL,

position\_max\_salary NUMBER (8, 2) DEFAULT NULL

);

CREATE TABLE division2 (

division\_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,

division\_name VARCHAR2 (30) NOT NULL,

location\_id NUMBER DEFAULT NULL,

FOREIGN KEY (location\_id) REFERENCES site (location\_id) ON DELETE CASCADE

);

CREATE TABLE workforce2 (

worker\_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,

first\_name VARCHAR2 (20) DEFAULT NULL,

last\_name VARCHAR2 (25) NOT NULL,

email VARCHAR2 (100) NOT NULL,

phone\_number VARCHAR2 (20) DEFAULT NULL,

hire\_date DATE NOT NULL,

position\_id NUMBER NOT NULL,

salary NUMBER (8, 2) NOT NULL,

manager\_id NUMBER DEFAULT NULL,

division\_id NUMBER DEFAULT NULL,

FOREIGN KEY (position\_id) REFERENCES position (position\_id) ON DELETE CASCADE,

FOREIGN KEY (division\_id) REFERENCES division (division\_id) ON DELETE CASCADE,

FOREIGN KEY (manager\_id) REFERENCES workforce (worker\_id)

);

CREATE TABLE nominee2 (

nominee\_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,

first\_name VARCHAR2 (50) NOT NULL,

last\_name VARCHAR2 (50) NOT NULL,

relation VARCHAR2 (25) NOT NULL,

worker\_id NUMBER NOT NULL,

FOREIGN KEY (worker\_id) REFERENCES workforce (worker\_id) ON DELETE CASCADE

);