Industry Project Report

On

VERACITIZ SOLUTION

Developed By: -	Guided By:-
Aryan Mayurkumar Modi (20162121011)	Prof. Dharmesh Darji (Internal)

Submitted to
Faculty of Engineering and Technology
Institute of Computer Technology
Ganpat University

Year - 2024

Ganpat University
॥ विद्यया समाजोत्कर्षः ॥
Institute of Computer Technology
National Assessment and Accreditation Council NAAC A Grade Accredited University

CERTIFICATE

This is to certify that the Summer Internship report submitted along with the Weekly basis task on Data Analysis has been carried out by Aryan Modi **at VERACITIZ SOLUTION** in partial fulfillment for the degree of Bachelor of Computer Science Engineering in BDA, 8th semester of Institute of Computer Technology, Ganpat University during the academic year 2023-24. The results/findings contained in this report have not been submitted in part or full to any other University / Institute for award of any other Degree/Diploma.

Prof. Dharmesh Darji	Prof. Dharmesh Darji
Internal Guide	Head , CSE Department

ACKNOWLEDGEMENT

Summer Internship project is a golden opportunity for learning and self-development. I consider myself very lucky and honored to have so many wonderful people lead me through in completion of this project. First and foremost, I would like to thank Dr. Rohit Patel, Principal, ICT, and Prof. Dharmesh Darji, Head, ICT who gave us an opportunity to undertake this project. My grateful thanks to Prof. Dharmesh Darji & Mr. Jacky Patel for their guidance in Data Analysis, who despite being extraordinarily busy with academics, took time out to hear, guide and keep us onthe correct path. We do not know where would have been without his/her help. CSE department monitored our progress and arranged all facilities to make life easier. We choose this moment to acknowledge their contribution gratefully.

Aryan Modi (Enrollment No : 20162121011)

TABLE OF CONTENTS

Content			Page Number
	Title Page		
	College Certificate		I
	Acknowledgement		II
	Table of Contents		III
1.	**Overview of the Company**		1
	1.1	History	1
	1.2	Different product / scope of work	1
2	**Week-1 Progress**		2
	2.1	Study about ETL	2
	2.2	Extract(E)	2
	2.3	Transform(T)	3
	2.4	Load(L)	3
	2.5	Screenshot/Code Snippet	3
3	**Week-2 Progress**		6
	3.1	SQL Server Management Studio	6
	3.2	Query editor	6
	3.3	Import and Export data	6
	3.4	Database diagram	6
	3.5	Screenshot/Code Snippet	6
4	**Week-3 Progress**		11
	4.1	SQL Server	11
	4.2	Key component	11
	4.3	Key concepts	12
	4.4	Screenshot/Code Snippet	13

5	**Week-4 Progress**		17
	5.1	Lead and Lag Functions	17
	5.2	Slowly Changing Dimension	17
	5.3	Stored Procedure	17
	5.4	Screenshot/Code Snippet	18
6	**Week-5 Progress**		22
	6.1	HackerRank Tasks	22
	6.2	Screenshot/Code Snippet working on HackerRank	22
7	**Week-6 Progress**		25
	7.1	Improve Logic Building Task	25
	7.2	Screenshot/Code Snippet	26
8	**Week-7 Progress**		29
	8.1	Improve Logic Building Task	29
	8.2	Screenshot/Code Snippet	30
9	**Live Project**		35
	9.1	Loop SQL	35
	9.2	JavaScript	35
	9.3	Node JS	35
	9.4	FIREBASE	35
	9.5	Ngrok	36
	9.6	Cognos Custom Control	36
	9.7	Cognos	37
	9.8	Screenshot/Code Snippet of project	38
10	**REFERENCES**		43

CHAPTER-1

OVERVIEW OF THE COMPANY

1.1 History

- Veracitiz Solutions is a performance management consulting firm and an IBM-exclusive Premier Business Partner with a long history of experience in developing and implementing sound planning and budgeting solutions. The company has been praised for its in-depth knowledge of the Cognos Enterprise Planning software suite and its ability to understand unique business models and industry-specific requirements. Veracitiz has a strong focus on harnessing emerging technologies for impactful business results and has a track record of accomplishing preset objectives on time and on budget. The company is also recognized as an IBM Gold Business Partner and has a presence in various industries, including retail. While specific details about the history of Veracitiz Solutions are not readily available in the provided search results, the company's expertise and reputation in the field of performance management and its long-standing partnership with IBM demonstrate its established presence in the industry. For more detailed information about the history of Veracitiz Solutions, it may be beneficial to directly contact the company or refer to official press releases and publications.

1.2 Different product / scope of work

- IBM Planning Analytics : IBM Planning Analytics is an integrated planning solution that uses AI to automate planning, budgeting, and forecasting and drive more intelligent workflows.
- IBM Cognos Analytics : Cognos Analytics is an AI-fueled business intelligence platform that helps to visualize, analyze and share actionable insights about your data with anyone in your organization.
- IBM SPSS Statistics : IBM® SPSS® Statistics solves a wide range of business and research problems by providing rich statistical capabilities, ensuring high accuracy to drive quality decision-making
- IBM Decision Optimization : IBM ILOG CPLEX Optimization Studio is a data science solution that combines mathematical and AI techniques to help make complex decisions involving multiple decision-variables, constraints & trade-offs
- IBM Instana : With IBM® Observability by Instana, users can combine APM with automation capabilities and distributed tracing to deploy on premise or as a SaaS solution.
- IBM SevOne : IBM SevOne NPM solution helps you spot, address, and prevent network performance issues early with machine learning-powered analytics from a single source.

CHAPTER-2

WEEK-1 PROGRESS

2.1 Study about ETL

ETL (Extract, Transform, Load) is a data integration process that combines data from multiple sources into a single, consistent data store, such as a data warehouse or data lake. It involves three key steps: extraction, transformation, and loading. During extraction, raw data is copied from source locations to a staging area. In the transformation step, the data is cleansed and organized to address specific business intelligence needs. Finally, the transformed data is loaded into the target system. ETL is essential for data analytics, machine learning, and business intelligence, and it has been used by organizations for decades. While ETL is a time-consuming batch operation, it significantly improves data quality and is particularly suitable for building smaller data repositories.

ETL is essential for data analytics, machine learning, and business intelligence, and it has been used by organizations for decades. While ETL is a time-consuming batch operation, it significantly improves data quality and is particularly suitable for building smaller data repositories. Modern ETL solutions must cope with the accelerating volume and speed of data, as well as the ability to ingest, enrich, and manage transactions from any source, whether on-premises or in the cloud. ETL is commonly used to collect and prepare marketing data, integrate IoT data, and support data warehousing and data lake initiatives. If you are interested in learning more about ETL, there are various online courses available from top universities and industry leaders, covering topics such as data management, data warehousing, data analysis, and data visualization

.

2.2 Extract (E)

- In this phase, data is collected and extracted from various source systems, which can include databases, flat files, applications, APIs, or other data repositories.
- The goal is to gather raw data from these diverse sources, capturing relevant information for analysis.

2.3 Transform (T)

- Once data is extracted, it often needs to be transformed into a suitable format for analysis and storage. This involves cleaning, validating, and restructuring the data.
- Transformation processes may include data cleansing (fixing errors or inconsistencies), data

normalization (standardizing formats), data enrichment (adding additional information), and more.

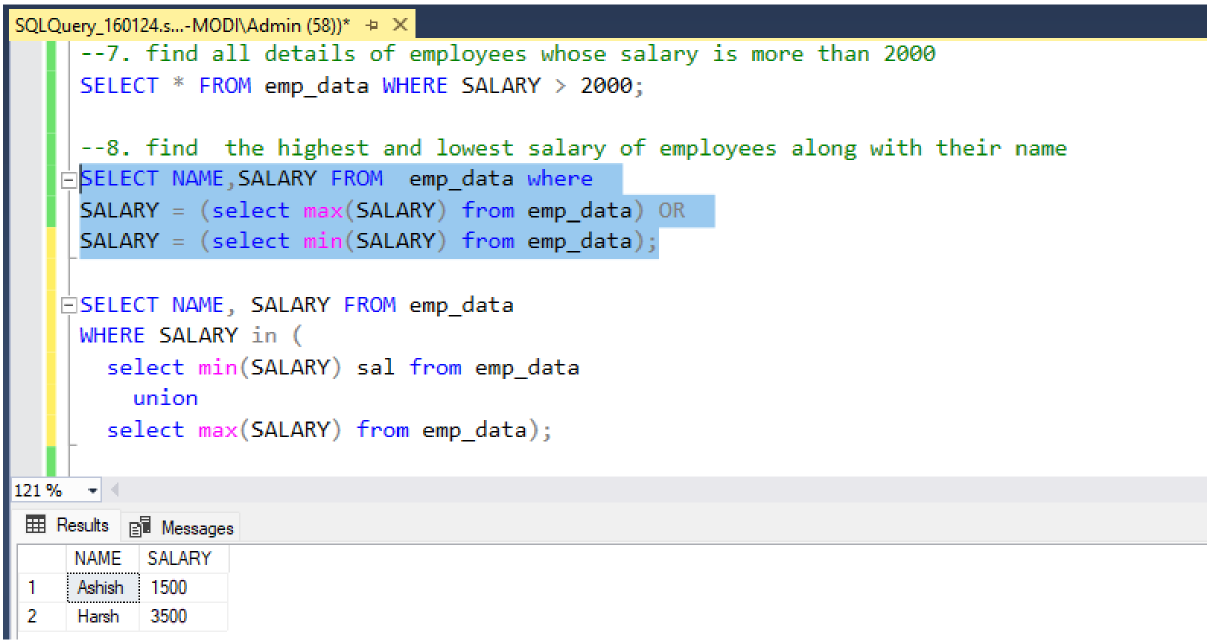
- The transformation phase ensures that the data is accurate, consistent, and ready for analysis.

2.4 Load (L)

- After the data has been extracted and transformed, it is loaded into a target system, typically a data warehouse or a data mart.
- Loading involves putting the transformed data into a structured format within the target database, making it accessible for reporting and analysis.
- The load phase may involve batch processing, real-time streaming, or a combination of both, depending on the specific requirements of the data integration.

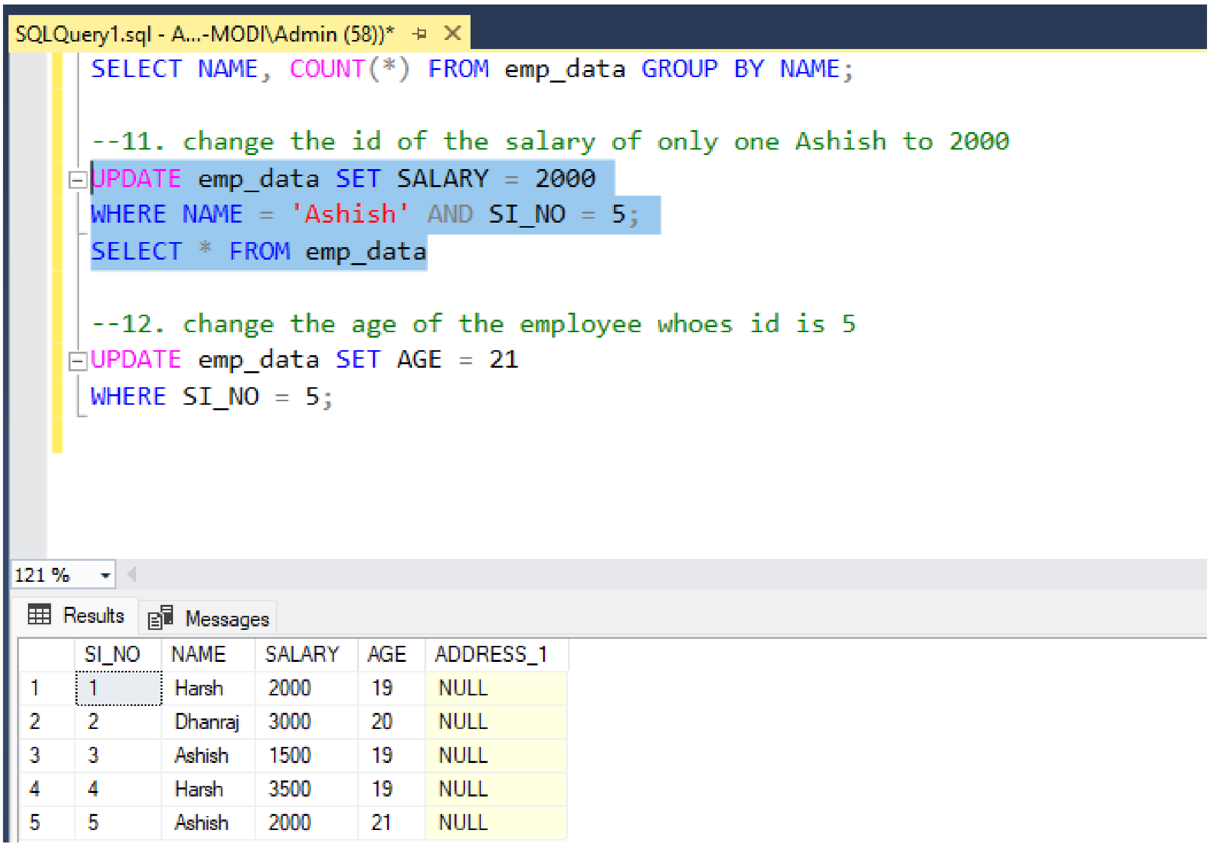
2.5 Screenshot/Code Snippet

I have learn following concepts : brief introduction of ETL ,DATA ,Data Warehouse , SQL Concepts , ALTER , MODIFY , CHANGE , CONSTRAINT , RENAME , MIN-MAX , WHERE , AGGRIGATE functions and STRING functions , ORDER BY ,GROUP BY , JOINJS , DATE other task are mantion in the documents.



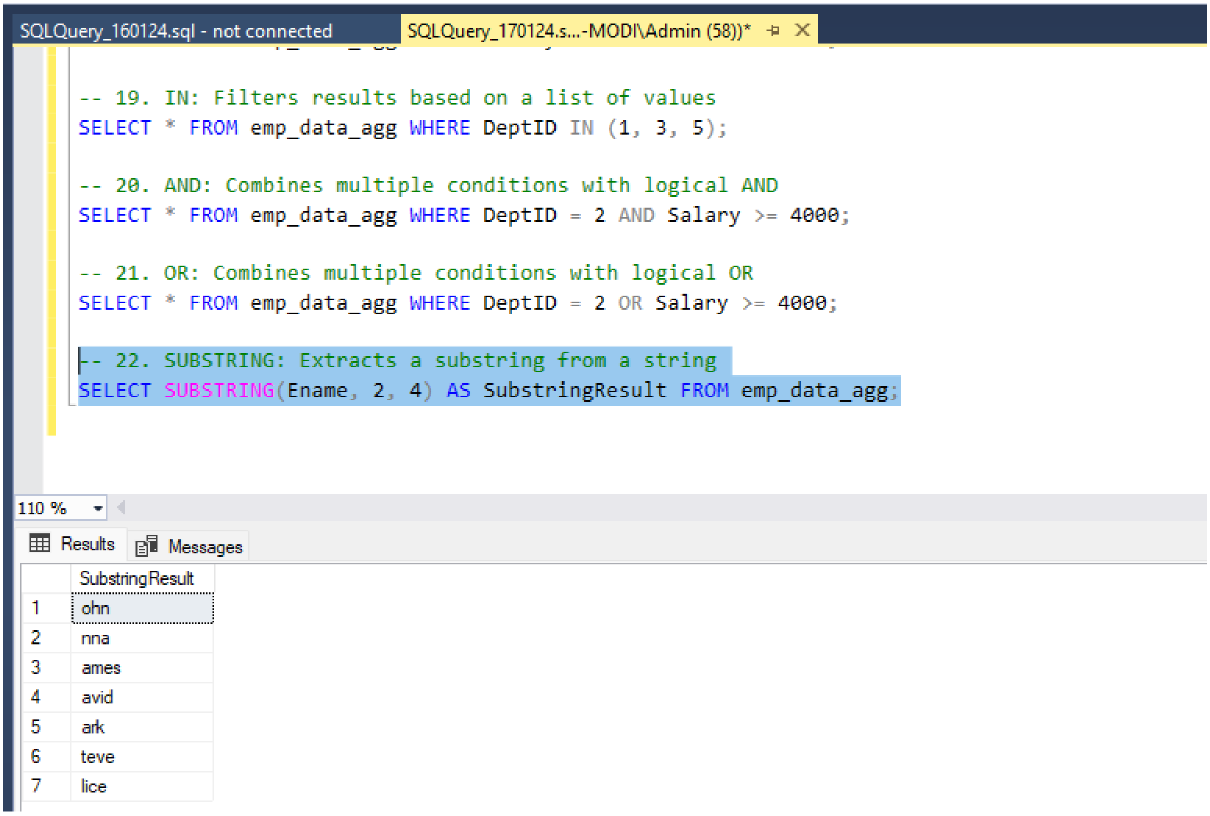
```
--7. find all details of employees whose salary is more than 2000
SELECT * FROM emp_data WHERE SALARY > 2000;

--8. find  the highest and lowest salary of employees along with their name
SELECT NAME,SALARY FROM  emp_data where
SALARY = (select max(SALARY) from emp_data) OR
SALARY = (select min(SALARY) from emp_data);

SELECT NAME, SALARY FROM emp_data
WHERE SALARY in (
  select min(SALARY) sal from emp_data
    union
  select max(SALARY) from emp_data);
```

	NAME	SALARY
1	Ashish	1500
2	Harsh	3500



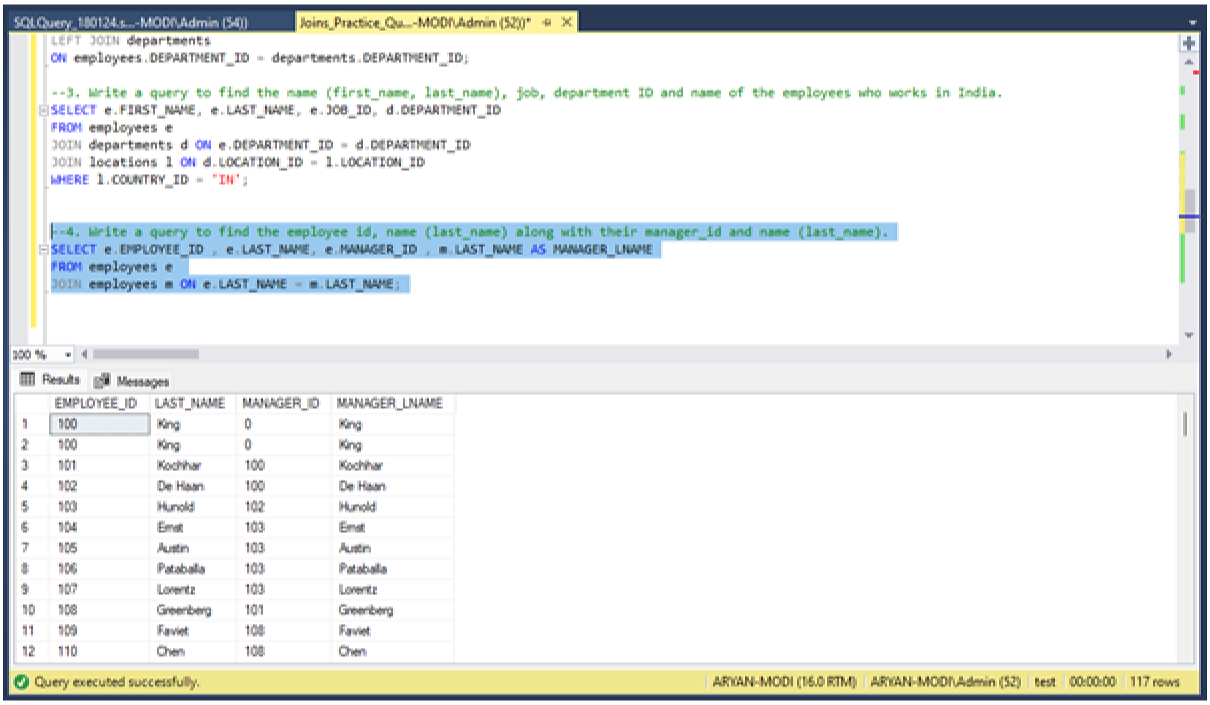
```
SELECT NAME, COUNT(*) FROM emp_data GROUP BY NAME;

--11. change the id of the salary of only one Ashish to 2000
UPDATE emp_data SET SALARY = 2000
WHERE NAME = 'Ashish' AND SI_NO = 5;
SELECT * FROM emp_data

--12. change the age of the employee whoes id is 5
UPDATE emp_data SET AGE = 21
WHERE SI_NO = 5;
```

	SI_NO	NAME	SALARY	AGE	ADDRESS_1
1	1	Harsh	2000	19	NULL
2	2	Dhanraj	3000	20	NULL
3	3	Ashish	1500	19	NULL
4	4	Harsh	3500	19	NULL
5	5	Ashish	2000	21	NULL


SQLQuery_160124.sql - not connected
SQLQuery_170124.s...-MODI\Admin (58))*
110 %
Results
Messages

```
-- 19. IN: Filters results based on a list of values
SELECT * FROM emp_data_agg WHERE DeptID IN (1, 3, 5);

-- 20. AND: Combines multiple conditions with logical AND
SELECT * FROM emp_data_agg WHERE DeptID = 2 AND Salary >= 4000;

-- 21. OR: Combines multiple conditions with logical OR
SELECT * FROM emp_data_agg WHERE DeptID = 2 OR Salary >= 4000;

-- 22. SUBSTRING: Extracts a substring from a string
SELECT SUBSTRING(Ename, 2, 4) AS SubstringResult FROM emp_data_agg;
```

	SubstringResult
1	ohn
2	nna
3	ames
4	avid
5	ark
6	teve
7	lice

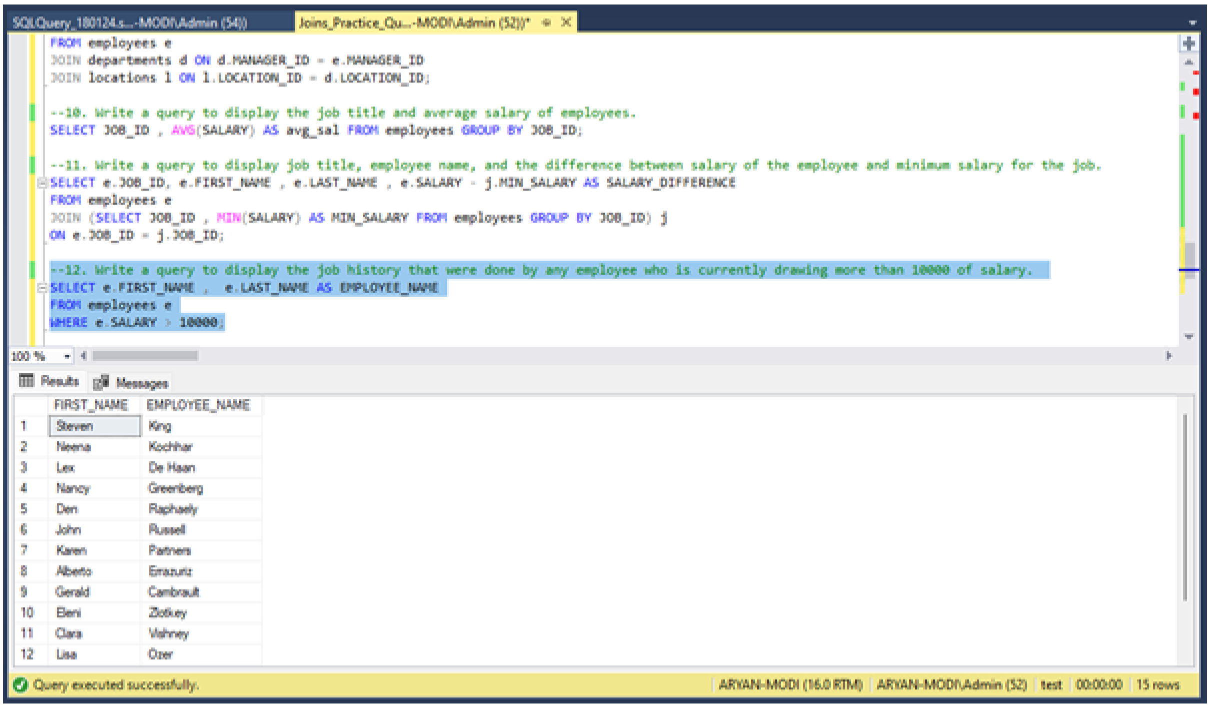


SQLQuery_180124.s...-MODI\Admin (54))
Joins_Practice_Qu...-MODI\Admin (52))*
100 %
Results
Messages

```sql
LEFT JOIN departments
ON employees.DEPARTMENT_ID = departments.DEPARTMENT_ID;

--3. Write a query to find the name (first_name, last_name), job, department ID and name of the employees who works in India.
SELECT e.FIRST_NAME, e.LAST_NAME, e.JOB_ID, d.DEPARTMENT_ID
FROM employees e
JOIN departments d ON e.DEPARTMENT_ID = d.DEPARTMENT_ID
JOIN locations l ON d.LOCATION_ID = l.LOCATION_ID
WHERE l.COUNTRY_ID = 'IN';


--4. Write a query to find the employee id, name (last_name) along with their manager_id and name (last_name).
SELECT e.EMPLOYEE_ID , e.LAST_NAME, e.MANAGER_ID , m.LAST_NAME AS MANAGER_LNAME
FROM employees e
JOIN employees m ON e.LAST_NAME = m.LAST_NAME;
```

	EMPLOYEE_ID	LAST_NAME	MANAGER_ID	MANAGER_LNAME
1	100	King	0	King
2	100	King	0	King
3	101	Kochhar	100	Kochhar
4	102	De Haan	100	De Haan
5	103	Hunold	102	Hunold
6	104	Ernst	103	Ernst
7	105	Austin	103	Austin
8	106	Pataballa	103	Pataballa
9	107	Lorentz	103	Lorentz
10	108	Greenberg	101	Greenberg
11	109	Faviet	108	Faviet
12	110	Chen	108	Chen

Query executed successfully. | ARYAN-MODI (16.0 RTM) | ARYAN-MODI\Admin (52) | test | 00:00:00 | 117 rows



SQLQuery_180124.s...-MODI\Admin (54))
Joins_Practice_Qu...-MODI\Admin (52))*
100 %
Results
Messages

```sql
FROM employees e
JOIN departments d ON d.MANAGER_ID = e.MANAGER_ID
JOIN locations l ON l.LOCATION_ID = d.LOCATION_ID;

--10. Write a query to display the job title and average salary of employees.
SELECT JOB_ID , AVG(SALARY) AS avg_sal FROM employees GROUP BY JOB_ID;

--11. Write a query to display job title, employee name, and the difference between salary of the employee and minimum salary for the job.
SELECT e.JOB_ID, e.FIRST_NAME , e.LAST_NAME , e.SALARY - j.MIN_SALARY AS SALARY_DIFFERENCE
FROM employees e
JOIN (SELECT JOB_ID , MIN(SALARY) AS MIN_SALARY FROM employees GROUP BY JOB_ID) j
ON e.JOB_ID = j.JOB_ID;

--12. Write a query to display the job history that were done by any employee who is currently drawing more than 10000 of salary.
SELECT e.FIRST_NAME ,  e.LAST_NAME AS EMPLOYEE_NAME
FROM employees e
WHERE e.SALARY > 10000;
```

	FIRST_NAME	EMPLOYEE_NAME
1	Steven	King
2	Neena	Kochhar
3	Lex	De Haan
4	Nancy	Greenberg
5	Den	Raphaely
6	John	Russell
7	Karen	Partners
8	Alberto	Errazuriz
9	Gerald	Cambrault
10	Eleni	Zlotkey
11	Clara	Vishney
12	Lisa	Ozer

Query executed successfully. | ARYAN-MODI (16.0 RTM) | ARYAN-MODI\Admin (52) | test | 00:00:00 | 15 rows

CHAPTER-3

WEEK-2 PROGRESS

3.1 SQL Server Management Studio

It is a graphical user interface (GUI) tool developed by Microsoft for managing and interacting with Microsoft SQL Server. SQL Server is a relational database management system (RDBMS) that is widely used for storing and retrieving data.

3.2 Query Editor

The Query Editor within SSMS allows users to write and execute Transact-SQL (T-SQL) queries against SQL Server databases. It provides syntax highlighting, code completion, and debugging capabilities.

3.3 Import and Export Data

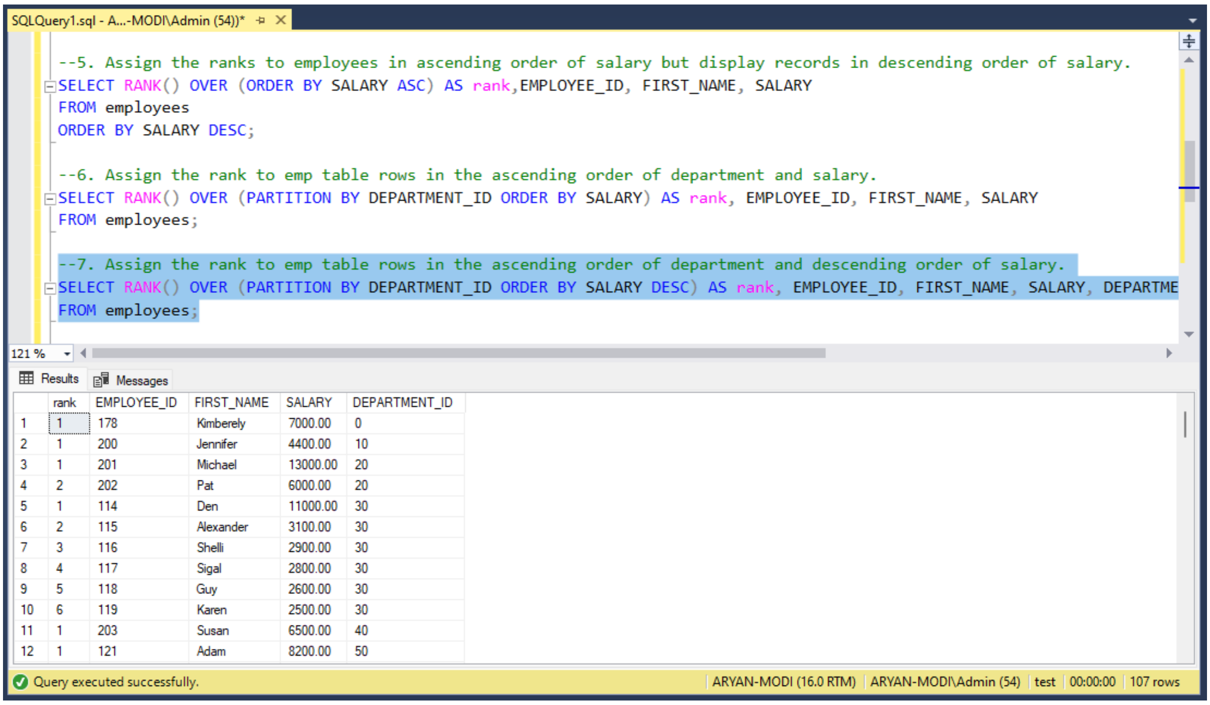
SSMS provides wizards for importing and exporting data between SQL Server and other data sources. This is useful for tasks such as data migration and integration.

3.4 Database Diagrams

Users can create visual representations of database relationships using the Database Diagrams feature in SSMS.

3.5 Screenshot/Code Snippet

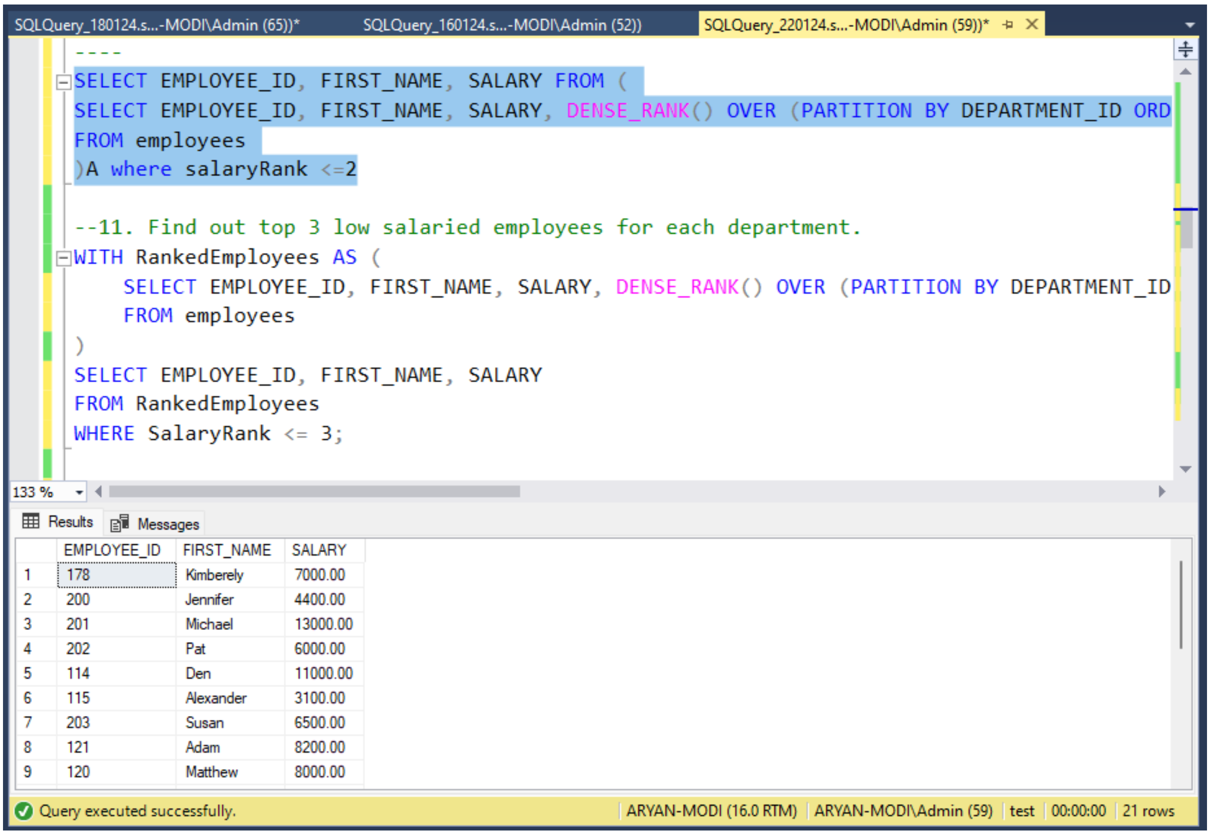
I have learn following concepts : RANK , DENSE RANK , Difference of both, Sub Quary, RANK and DENSE RANK with condition , LEAD & LAG FUNCTIONS , it's task ,STRING SPLIT functions , logic building tasks which are mentioned in Document.



```
--5. Assign the ranks to employees in ascending order of salary but display records in descending order of salary.
SELECT RANK() OVER (ORDER BY SALARY ASC) AS rank,EMPLOYEE_ID, FIRST_NAME, SALARY
FROM employees
ORDER BY SALARY DESC;

--6. Assign the rank to emp table rows in the ascending order of department and salary.
SELECT RANK() OVER (PARTITION BY DEPARTMENT_ID ORDER BY SALARY) AS rank, EMPLOYEE_ID, FIRST_NAME, SALARY
FROM employees;

--7. Assign the rank to emp table rows in the ascending order of department and descending order of salary.
SELECT RANK() OVER (PARTITION BY DEPARTMENT_ID ORDER BY SALARY DESC) AS rank, EMPLOYEE_ID, FIRST_NAME, SALARY, DEPARTME
FROM employees;
```

	rank	EMPLOYEE_ID	FIRST_NAME	SALARY	DEPARTMENT_ID
1	1	178	Kimberely	7000.00	0
2	1	200	Jennifer	4400.00	10
3	1	201	Michael	13000.00	20
4	2	202	Pat	6000.00	20
5	1	114	Den	11000.00	30
6	2	115	Alexander	3100.00	30
7	3	116	Shelli	2900.00	30
8	4	117	Sigal	2800.00	30
9	5	118	Guy	2600.00	30
10	6	119	Karen	2500.00	30
11	1	203	Susan	6500.00	40
12	1	121	Adam	8200.00	50

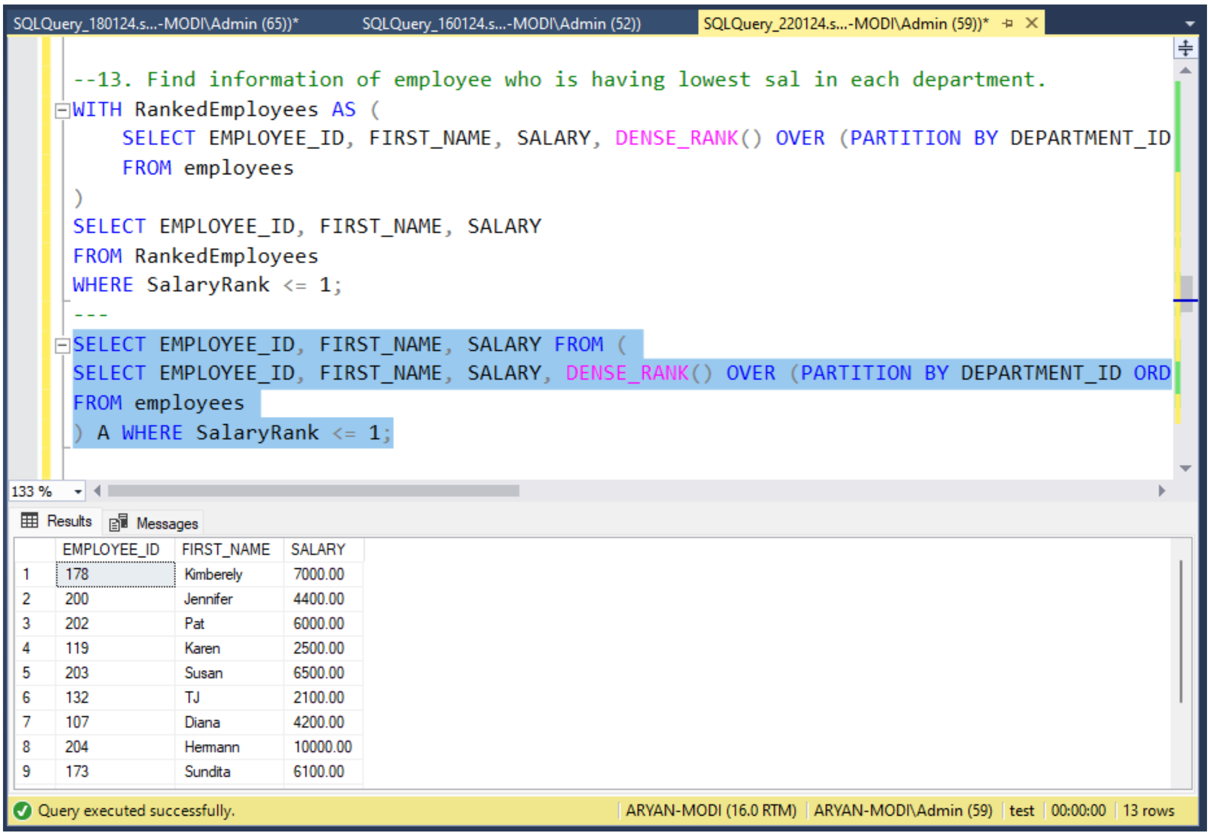
Query executed successfully. | ARYAN-MODI (16.0 RTM) | ARYAN-MODI\Admin (54) | test | 00:00:00 | 107 rows


SQLQuery_180124.s...-MODI\Admin (65))*
SQLQuery_160124.s...-MODI\Admin (52))
SQLQuery_220124.s...-MODI\Admin (59))*
133 %
Results
Messages

```
----
SELECT EMPLOYEE_ID, FIRST_NAME, SALARY FROM (
SELECT EMPLOYEE_ID, FIRST_NAME, SALARY, DENSE_RANK() OVER (PARTITION BY DEPARTMENT_ID ORD
FROM employees
)A where salaryRank <=2

--11. Find out top 3 low salaried employees for each department.
WITH RankedEmployees AS (
    SELECT EMPLOYEE_ID, FIRST_NAME, SALARY, DENSE_RANK() OVER (PARTITION BY DEPARTMENT_ID
    FROM employees
)
SELECT EMPLOYEE_ID, FIRST_NAME, SALARY
FROM RankedEmployees
WHERE SalaryRank <= 3;
```

	EMPLOYEE_ID	FIRST_NAME	SALARY
1	178	Kimberely	7000.00
2	200	Jennifer	4400.00
3	201	Michael	13000.00
4	202	Pat	6000.00
5	114	Den	11000.00
6	115	Alexander	3100.00
7	203	Susan	6500.00
8	121	Adam	8200.00
9	120	Matthew	8000.00

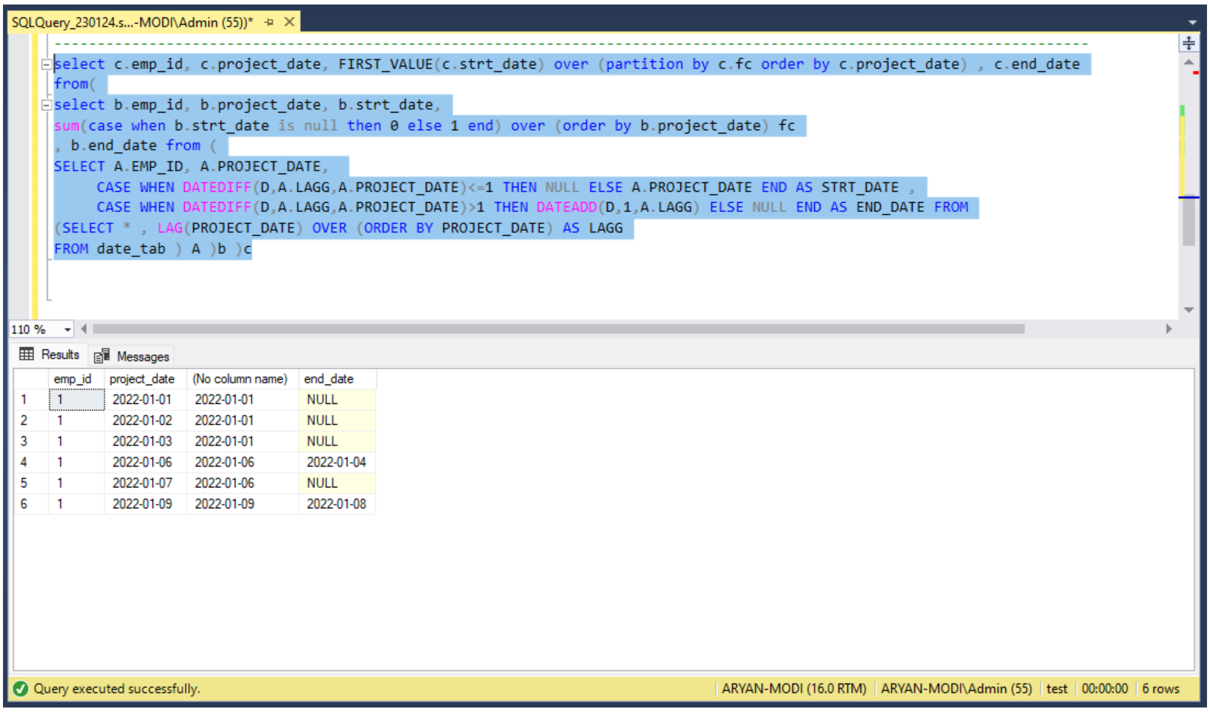
Query executed successfully. | ARYAN-MODI (16.0 RTM) | ARYAN-MODI\Admin (59) | test | 00:00:00 | 21 rows


SQLQuery_180124.s...-MODI\Admin (65))*
SQLQuery_160124.s...-MODI\Admin (52))
SQLQuery_220124.s...-MODI\Admin (59))*

```sql
--13. Find information of employee who is having lowest sal in each department.
WITH RankedEmployees AS (
    SELECT EMPLOYEE_ID, FIRST_NAME, SALARY, DENSE_RANK() OVER (PARTITION BY DEPARTMENT_ID
    FROM employees
)
SELECT EMPLOYEE_ID, FIRST_NAME, SALARY
FROM RankedEmployees
WHERE SalaryRank <= 1;
---
SELECT EMPLOYEE_ID, FIRST_NAME, SALARY FROM (
SELECT EMPLOYEE_ID, FIRST_NAME, SALARY, DENSE_RANK() OVER (PARTITION BY DEPARTMENT_ID ORD
FROM employees
) A WHERE SalaryRank <= 1;
```

	EMPLOYEE_ID	FIRST_NAME	SALARY
1	178	Kimberely	7000.00
2	200	Jennifer	4400.00
3	202	Pat	6000.00
4	119	Karen	2500.00
5	203	Susan	6500.00
6	132	TJ	2100.00
7	107	Diana	4200.00
8	204	Hermann	10000.00
9	173	Sundita	6100.00

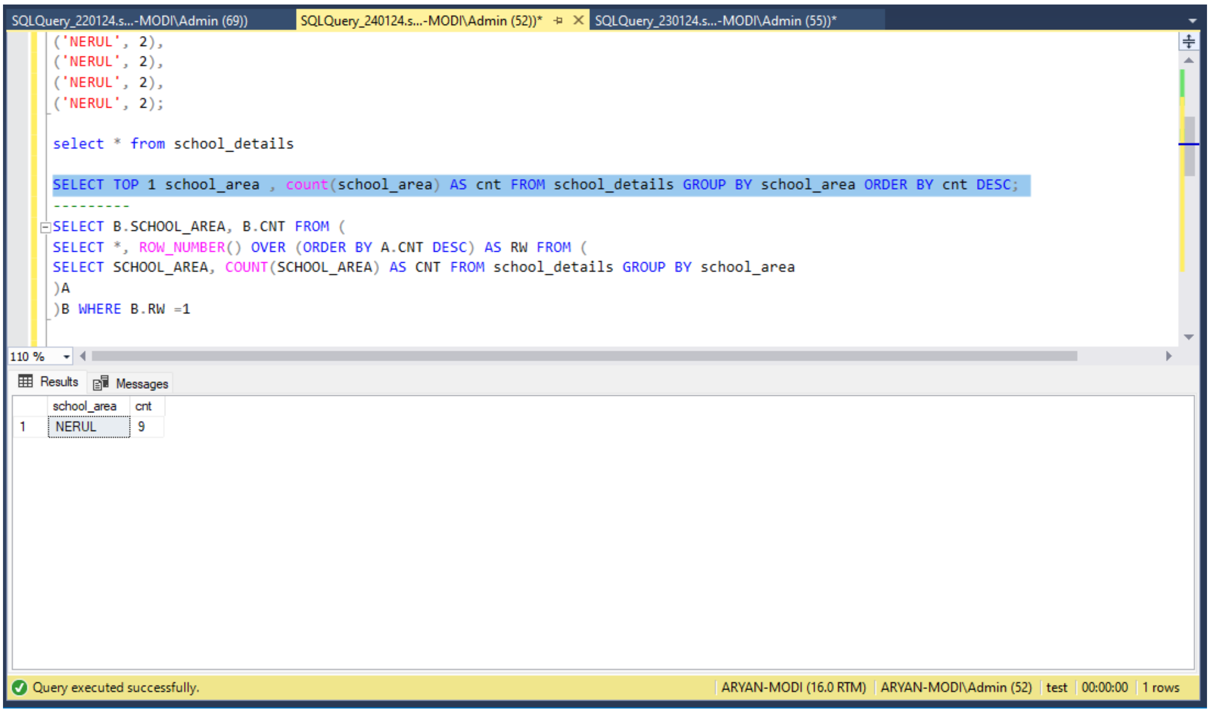
Query executed successfully. | ARYAN-MODI (16.0 RTM) | ARYAN-MODI\Admin (59) | test | 00:00:00 | 13 rows



```sql
-----------------------------------------------------------------------------------------------------------
select c.emp_id, c.project_date, FIRST_VALUE(c.strt_date) over (partition by c.fc order by c.project_date) , c.end_date
from(
select b.emp_id, b.project_date, b.strt_date,
sum(case when b.strt_date is null then 0 else 1 end) over (order by b.project_date) fc
, b.end_date from (
SELECT A.EMP_ID, A.PROJECT_DATE,
     CASE WHEN DATEDIFF(D,A.LAGG,A.PROJECT_DATE)<=1 THEN NULL ELSE A.PROJECT_DATE END AS STRT_DATE ,
     CASE WHEN DATEDIFF(D,A.LAGG,A.PROJECT_DATE)>1 THEN DATEADD(D,1,A.LAGG) ELSE NULL END AS END_DATE FROM
(SELECT * , LAG(PROJECT_DATE) OVER (ORDER BY PROJECT_DATE) AS LAGG
FROM date_tab ) A )b )c
```

	emp_id	project_date	(No column name)	end_date
1	1	2022-01-01	2022-01-01	NULL
2	1	2022-01-02	2022-01-01	NULL
3	1	2022-01-03	2022-01-01	NULL
4	1	2022-01-06	2022-01-06	2022-01-04
5	1	2022-01-07	2022-01-06	NULL
6	1	2022-01-09	2022-01-09	2022-01-08

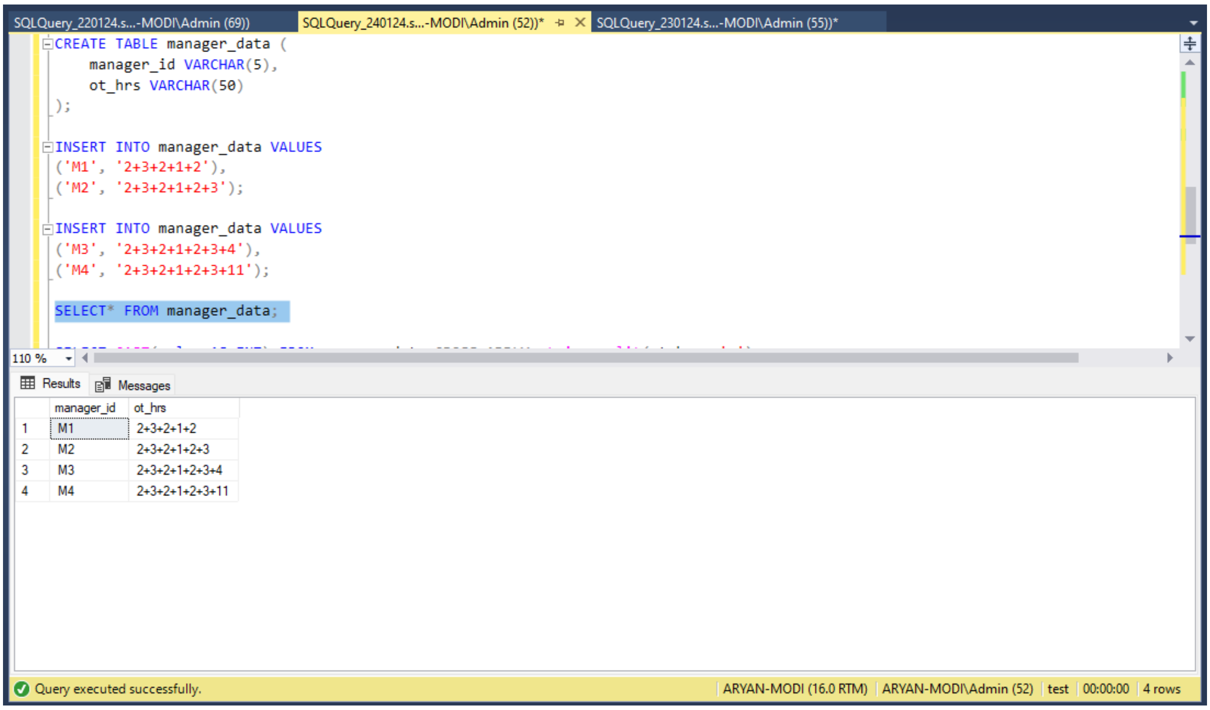
Query executed successfully. | ARYAN-MODI (16.0 RTM) | ARYAN-MODI\Admin (55) | test | 00:00:00 | 6 rows


SQLQuery_220124.s...-MODI\Admin (69))
SQLQuery_240124.s...-MODI\Admin (52))*
SQLQuery_230124.s...-MODI\Admin (55))*
110 %
Results
Messages

```
('NERUL', 2),
('NERUL', 2),
('NERUL', 2),
('NERUL', 2);

select * from school_details

SELECT TOP 1 school_area , count(school_area) AS cnt FROM school_details GROUP BY school_area ORDER BY cnt DESC;
---------
SELECT B.SCHOOL_AREA, B.CNT FROM (
SELECT *, ROW_NUMBER() OVER (ORDER BY A.CNT DESC) AS RW FROM (
SELECT SCHOOL_AREA, COUNT(SCHOOL_AREA) AS CNT FROM school_details GROUP BY school_area
)A
)B WHERE B.RW =1
```

	school_area	cnt
1	NERUL	9

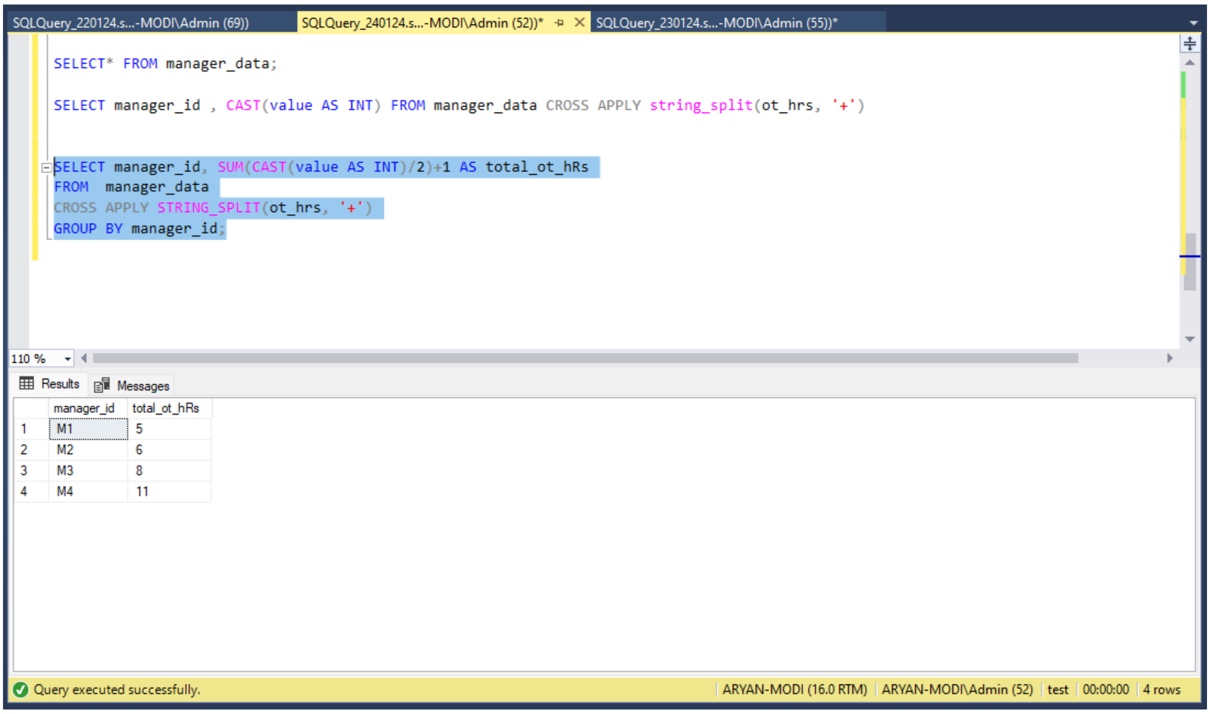
Query executed successfully. | ARYAN-MODI (16.0 RTM) | ARYAN-MODI\Admin (52) | test | 00:00:00 | 1 rows


SQLQuery_220124.s...-MODI\Admin (69))
SQLQuery_240124.s...-MODI\Admin (52))*
SQLQuery_230124.s...-MODI\Admin (55))*
110 %
Results
Messages

```
CREATE TABLE manager_data (
    manager_id VARCHAR(5),
    ot_hrs VARCHAR(50)
);

INSERT INTO manager_data VALUES
('M1', '2+3+2+1+2'),
('M2', '2+3+2+1+2+3');

INSERT INTO manager_data VALUES
('M3', '2+3+2+1+2+3+4'),
('M4', '2+3+2+1+2+3+11');

SELECT* FROM manager_data;
```

	manager_id	ot_hrs
1	M1	2+3+2+1+2
2	M2	2+3+2+1+2+3
3	M3	2+3+2+1+2+3+4
4	M4	2+3+2+1+2+3+11

Query executed successfully. | ARYAN-MODI (16.0 RTM) | ARYAN-MODI\Admin (52) | test | 00:00:00 | 4 rows



SQLQuery_220124.s...-MODI\Admin (69))
SQLQuery_240124.s...-MODI\Admin (52))*
SQLQuery_230124.s...-MODI\Admin (55))*
110 %
Results
Messages

```sql
SELECT* FROM manager_data;

SELECT manager_id , CAST(value AS INT) FROM manager_data CROSS APPLY string_split(ot_hrs, '+')


SELECT manager_id, SUM(CAST(value AS INT)/2)+1 AS total_ot_hRs
FROM  manager_data
CROSS APPLY STRING_SPLIT(ot_hrs, '+')
GROUP BY manager_id;
```

	manager_id	total_ot_hRs
1	M1	5
2	M2	6
3	M3	8
4	M4	11

Query executed successfully. | ARYAN-MODI (16.0 RTM) | ARYAN-MODI\Admin (52) | test | 00:00:00 | 4 rows

CHAPTER-4

WEEK-3 PROGRESS

4.1 SQL SERVER

SQL Server is a relational database management system (RDBMS) developed by Microsoft. It is a software product that stores and retrieves data as requested by other software applications, running either on the same computer or on another computer across a network.

4.2 Key Components

Database Engine: The core component that handles storage, processing, and security of data. It manages database objects such as tables, views, stored procedures, and triggers.

SQL Server Management Studio (SSMS): A graphical user interface (GUI) tool for managing and interacting with SQL Server. DBAs and developers use SSMS for tasks like writing queries, designing databases, and configuring server settings.

Transact-SQL (T-SQL): The SQL Server query language. It is an extension of SQL with additional programming constructs. T-SQL is used to interact with the SQL Server database engine.

Integration Services (SSIS): A tool for solving complex business problems by copying or downloading files, extracting and transforming data from different data sources, and loading data into one or more destinations.

Reporting Services (SSRS): A server-based reporting platform that provides a full range of ready-to-use tools and services to help you create, deploy, and manage reports for your organization.

Analysis Services (SSAS): Enables users to analyze and visualize data through online analytical processing (OLAP) and data mining.

4.3 Key Concepts

Database: A container that holds a set of related tables, views, stored procedures, and other database objects. It's a way to organize and store data.

Table: A fundamental storage structure in SQL Server. It consists of rows and columns, where each column has a data type, and each row represents a record.

Query: A request for data or information from a database. SQL (Structured Query Language) is used to write queries to interact with SQL Server databases.

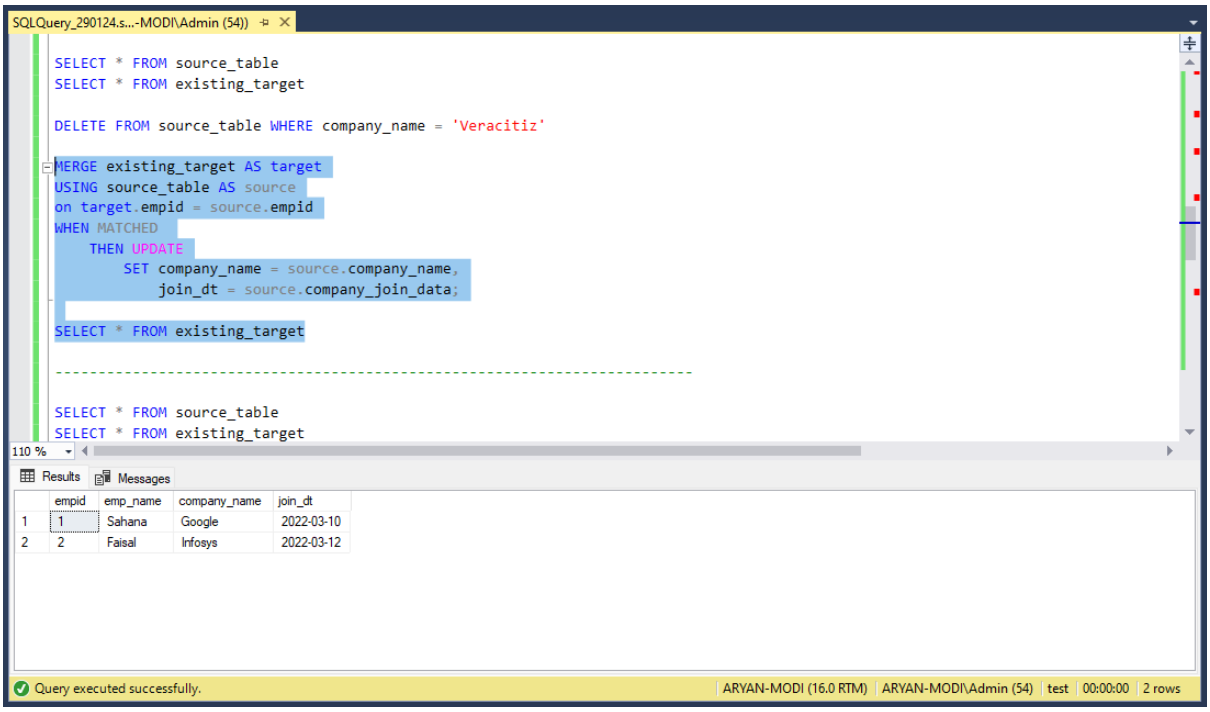
Primary Key: A unique identifier for a record in a table. It ensures data integrity by preventing duplicate or null values.

Foreign Key: A column or a set of columns in one table that refers to the primary key in another table. It establishes a link between the two tables.

Index: A structure in a database that improves the speed of data retrieval operations on a database table.

4.4 Screenshot/Code Snippet

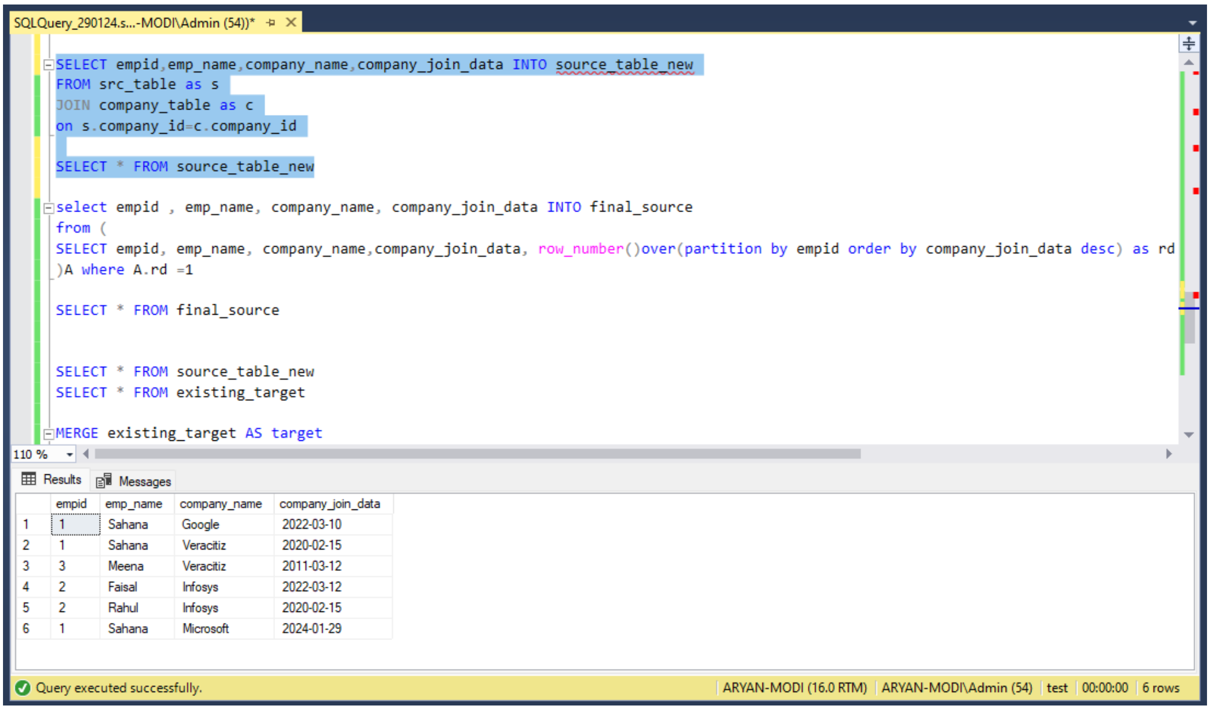
I have learn following concepts :SCD (Slowly Changing Dimension) and windows functions , Slowly Changing Dimension SCD type 1 examples with different scenarios , Slowly Changing Dimension SCD type 2 and SCD type 3 examples with different scenarios , DATABASE , DATA WAREHOUSE , STAR SCHEMA , SNOW FLAKE SCHEMA , GALAXY SCHEMA , DATA MART.



```sql
SELECT * FROM source_table
SELECT * FROM existing_target

DELETE FROM source_table WHERE company_name = 'Veracitiz'

MERGE existing_target AS target
USING source_table AS source
on target.empid = source.empid
WHEN MATCHED
    THEN UPDATE
        SET company_name = source.company_name,
            join_dt = source.company_join_data;

SELECT * FROM existing_target

--------------------------------------------------------------------------

SELECT * FROM source_table
SELECT * FROM existing_target
```

	empid	emp_name	company_name	join_dt
1	1	Sahana	Google	2022-03-10
2	2	Faisal	Infosys	2022-03-12

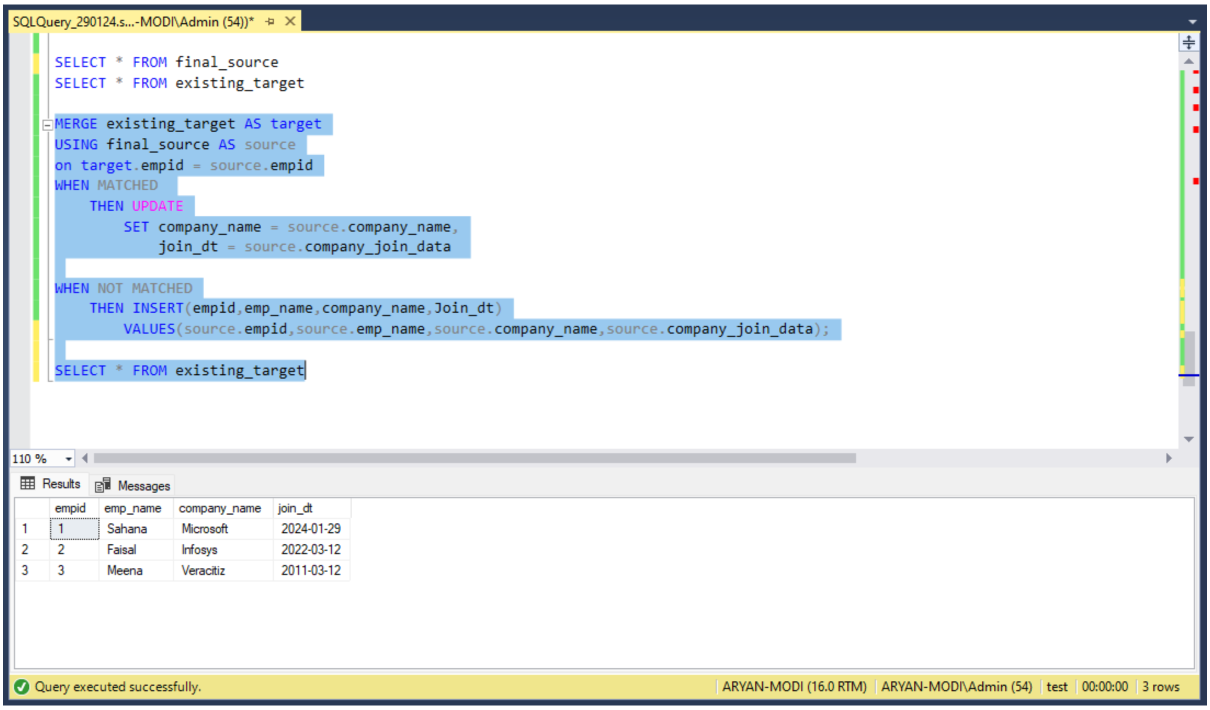
Query executed successfully. | ARYAN-MODI (16.0 RTM) | ARYAN-MODI\Admin (54) | test | 00:00:00 | 2 rows



```sql
SELECT empid,emp_name,company_name,company_join_data INTO source_table_new
FROM src_table as s
JOIN company_table as c
on s.company_id=c.company_id

SELECT * FROM source_table_new

select empid , emp_name, company_name, company_join_data INTO final_source
from (
SELECT empid, emp_name, company_name,company_join_data, row_number()over(partition by empid order by company_join_data desc) as rd
)A where A.rd =1

SELECT * FROM final_source


SELECT * FROM source_table_new
SELECT * FROM existing_target

MERGE existing_target AS target
```

	empid	emp_name	company_name	company_join_data
1	1	Sahana	Google	2022-03-10
2	1	Sahana	Veracitiz	2020-02-15
3	3	Meena	Veracitiz	2011-03-12
4	2	Faisal	Infosys	2022-03-12
5	2	Rahul	Infosys	2020-02-15
6	1	Sahana	Microsoft	2024-01-29

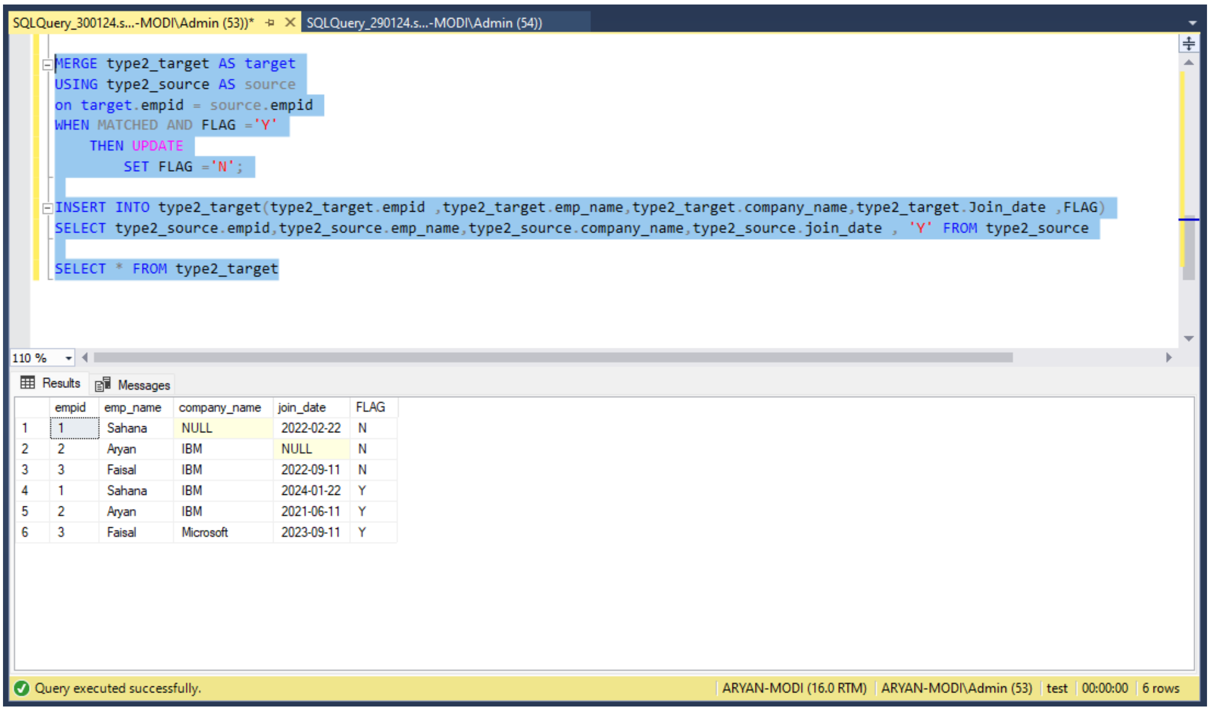
Query executed successfully. | ARYAN-MODI (16.0 RTM) | ARYAN-MODI\Admin (54) | test | 00:00:00 | 6 rows



```sql
SELECT * FROM final_source
SELECT * FROM existing_target

MERGE existing_target AS target
USING final_source AS source
on target.empid = source.empid
WHEN MATCHED
    THEN UPDATE
        SET company_name = source.company_name,
            join_dt = source.company_join_data

WHEN NOT MATCHED
    THEN INSERT(empid,emp_name,company_name,Join_dt)
        VALUES(source.empid,source.emp_name,source.company_name,source.company_join_data);

SELECT * FROM existing_target
```

	empid	emp_name	company_name	join_dt
1	1	Sahana	Microsoft	2024-01-29
2	2	Faisal	Infosys	2022-03-12
3	3	Meena	Veracitiz	2011-03-12

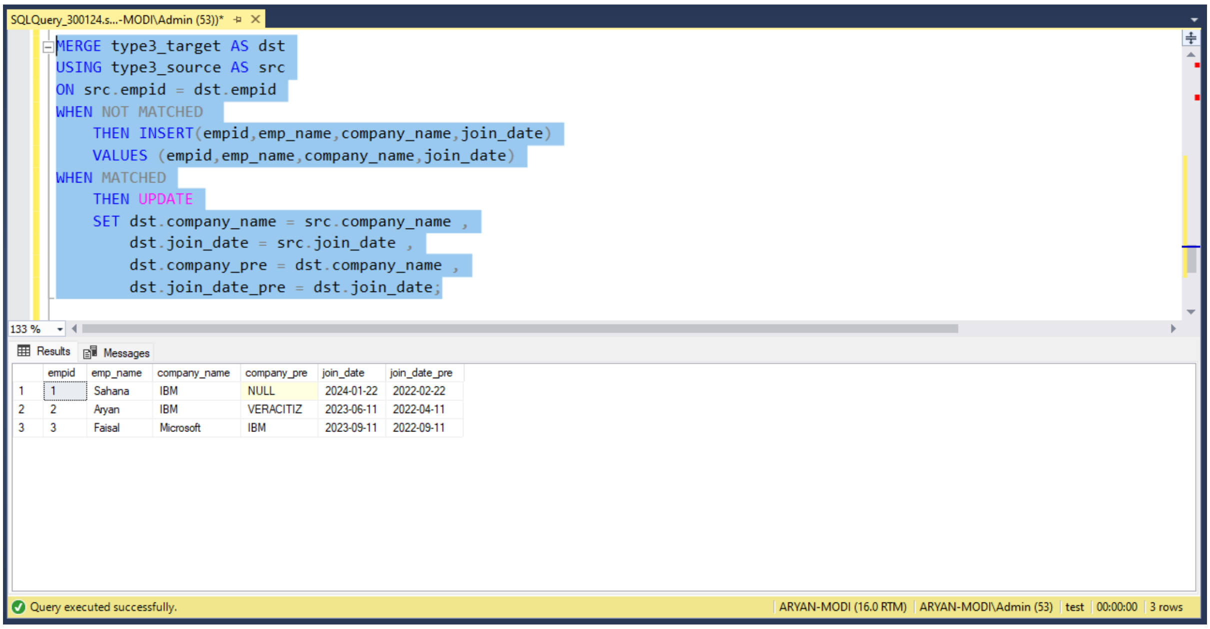
Query executed successfully. | ARYAN-MODI (16.0 RTM) | ARYAN-MODI\Admin (54) | test | 00:00:00 | 3 rows


SQLQuery_300124.s...-MODI\Admin (53))*
SQLQuery_290124.s...-MODI\Admin (54))
110 %
Results
Messages

```sql
MERGE type2_target AS target
USING type2_source AS source
on target.empid = source.empid
WHEN MATCHED AND FLAG ='Y'
    THEN UPDATE
        SET FLAG ='N';

INSERT INTO type2_target(type2_target.empid ,type2_target.emp_name,type2_target.company_name,type2_target.Join_date ,FLAG)
SELECT type2_source.empid,type2_source.emp_name,type2_source.company_name,type2_source.join_date , 'Y' FROM type2_source

SELECT * FROM type2_target
```

	empid	emp_name	company_name	join_date	FLAG
1	1	Sahana	NULL	2022-02-22	N
2	2	Aryan	IBM	NULL	N
3	3	Faisal	IBM	2022-09-11	N
4	1	Sahana	IBM	2024-01-22	Y
5	2	Aryan	IBM	2021-06-11	Y
6	3	Faisal	Microsoft	2023-09-11	Y

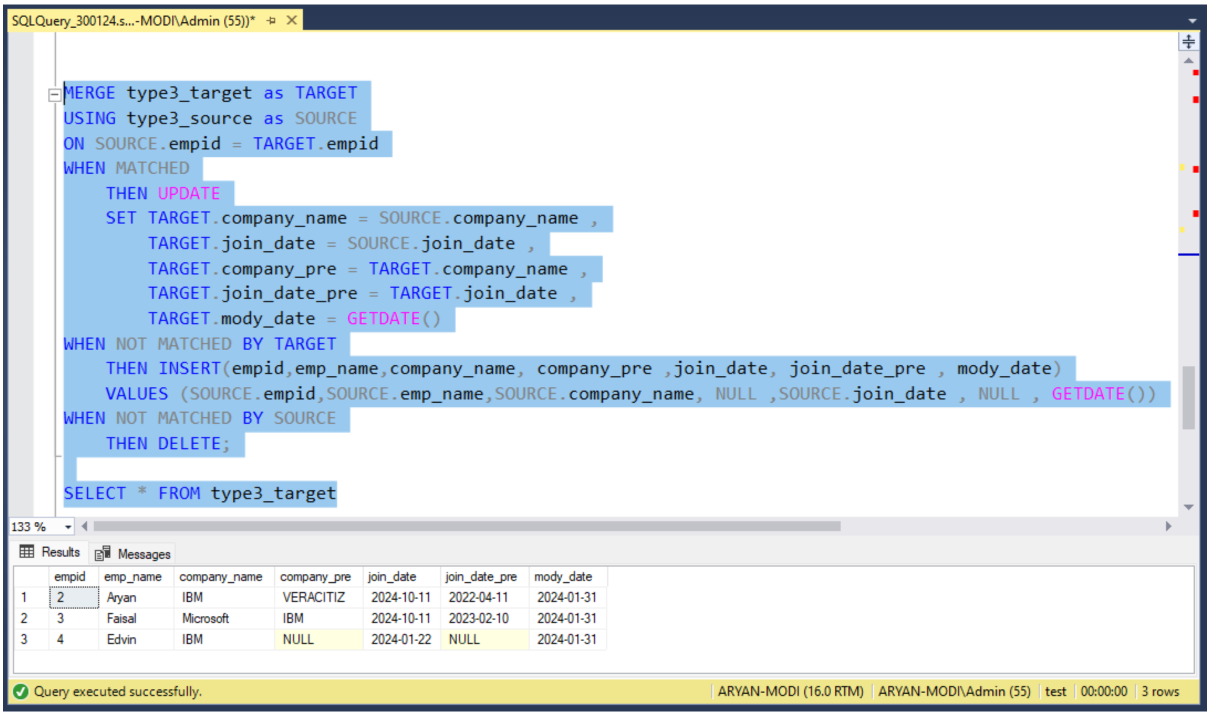
Query executed successfully. | ARYAN-MODI (16.0 RTM) | ARYAN-MODI\Admin (53) | test | 00:00:00 | 6 rows



```sql
MERGE type3_target AS dst
USING type3_source AS src
ON src.empid = dst.empid
WHEN NOT MATCHED
    THEN INSERT(empid,emp_name,company_name,join_date)
    VALUES (empid,emp_name,company_name,join_date)
WHEN MATCHED
    THEN UPDATE
    SET dst.company_name = src.company_name ,
        dst.join_date = src.join_date ,
        dst.company_pre = dst.company_name ,
        dst.join_date_pre = dst.join_date;
```

	empid	emp_name	company_name	company_pre	join_date	join_date_pre
1	1	Sahana	IBM	NULL	2024-01-22	2022-02-22
2	2	Aryan	IBM	VERACITIZ	2023-06-11	2022-04-11
3	3	Faisal	Microsoft	IBM	2023-09-11	2022-09-11

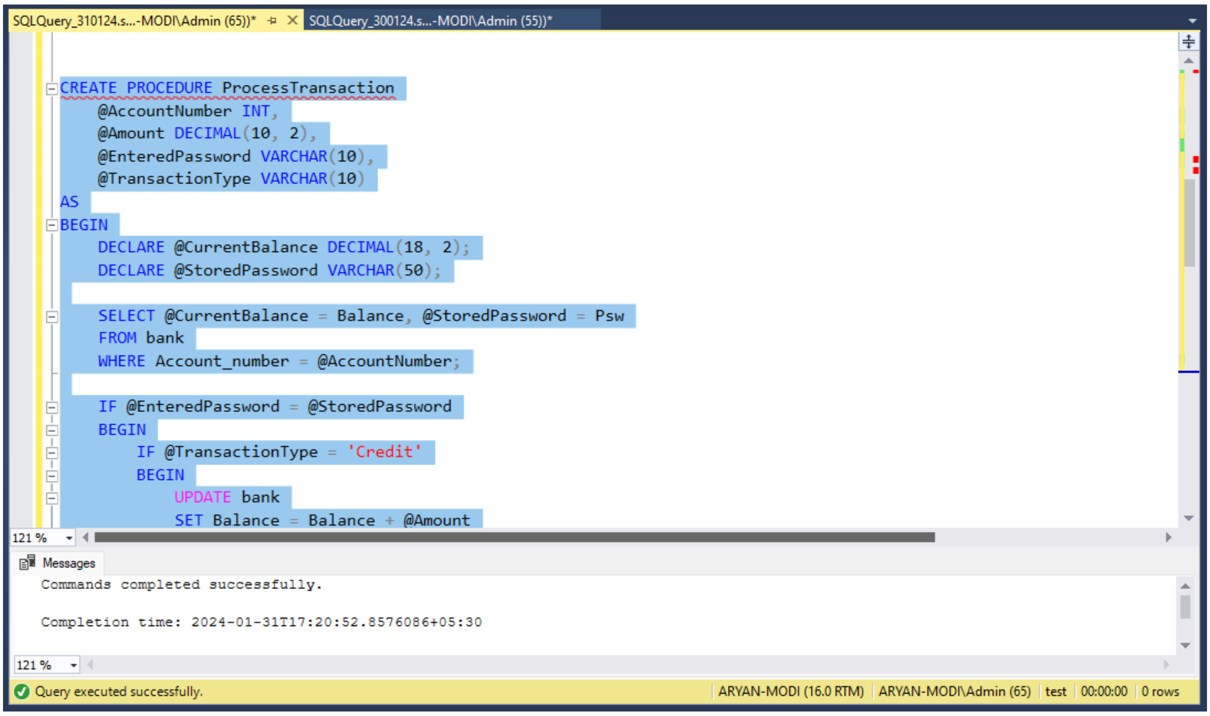
Query executed successfully. ARYAN-MODI (16.0 RTM) ARYAN-MODI\Admin (53) test 00:00:00 3 rows



```sql
MERGE type3_target as TARGET
USING type3_source as SOURCE
ON SOURCE.empid = TARGET.empid
WHEN MATCHED
    THEN UPDATE
    SET TARGET.company_name = SOURCE.company_name ,
        TARGET.join_date = SOURCE.join_date ,
        TARGET.company_pre = TARGET.company_name ,
        TARGET.join_date_pre = TARGET.join_date ,
        TARGET.mody_date = GETDATE()
WHEN NOT MATCHED BY TARGET
    THEN INSERT(empid,emp_name,company_name, company_pre ,join_date, join_date_pre , mody_date)
    VALUES (SOURCE.empid,SOURCE.emp_name,SOURCE.company_name, NULL ,SOURCE.join_date , NULL , GETDATE())
WHEN NOT MATCHED BY SOURCE
    THEN DELETE;

SELECT * FROM type3_target
```

	empid	emp_name	company_name	company_pre	join_date	join_date_pre	mody_date
1	2	Aryan	IBM	VERACITIZ	2024-10-11	2022-04-11	2024-01-31
2	3	Faisal	Microsoft	IBM	2024-10-11	2023-02-10	2024-01-31
3	4	Edvin	IBM	NULL	2024-01-22	NULL	2024-01-31

Query executed successfully. ARYAN-MODI (16.0 RTM) ARYAN-MODI\Admin (55) test 00:00:00 3 rows

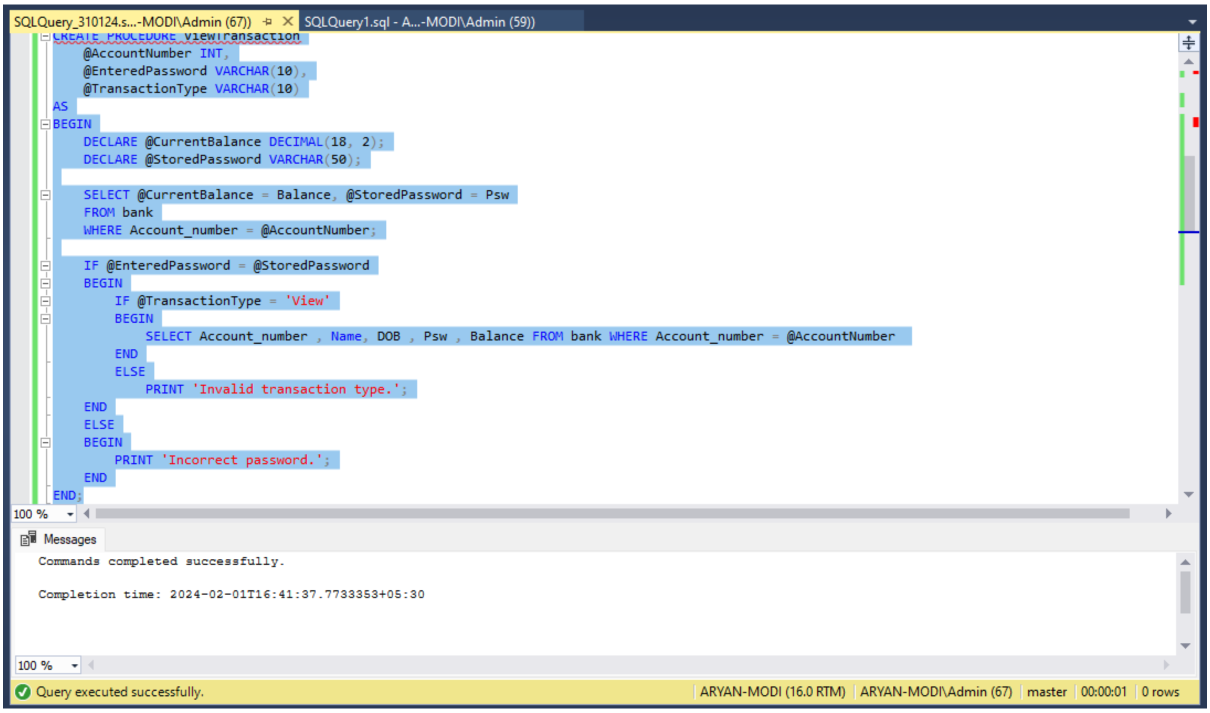


SQLQuery_310124.s...-MODI\Admin (65))*
SQLQuery_300124.s...-MODI\Admin (55))*
121 %

```sql
CREATE PROCEDURE ProcessTransaction
    @AccountNumber INT,
    @Amount DECIMAL(10, 2),
    @EnteredPassword VARCHAR(10),
    @TransactionType VARCHAR(10)
AS
BEGIN
    DECLARE @CurrentBalance DECIMAL(18, 2);
    DECLARE @StoredPassword VARCHAR(50);

    SELECT @CurrentBalance = Balance, @StoredPassword = Psw
    FROM bank
    WHERE Account_number = @AccountNumber;

    IF @EnteredPassword = @StoredPassword
    BEGIN
        IF @TransactionType = 'Credit'
        BEGIN
            UPDATE bank
            SET Balance = Balance + @Amount
```

```
Commands completed successfully.

Completion time: 2024-01-31T17:20:52.8576086+05:30
```

121 %
Query executed successfully.
ARYAN-MODI (16.0 RTM)
ARYAN-MODI\Admin (65)
test
00:00:00
0 rows



SQLQuery_310124.s...-MODI\Admin (67))
SQLQuery1.sql - A...-MODI\Admin (59))
100 %

```sql
CREATE PROCEDURE ViewTransaction
    @AccountNumber INT,
    @EnteredPassword VARCHAR(10),
    @TransactionType VARCHAR(10)
AS
BEGIN
    DECLARE @CurrentBalance DECIMAL(18, 2);
    DECLARE @StoredPassword VARCHAR(50);

    SELECT @CurrentBalance = Balance, @StoredPassword = Psw
    FROM bank
    WHERE Account_number = @AccountNumber;

    IF @EnteredPassword = @StoredPassword
    BEGIN
        IF @TransactionType = 'View'
        BEGIN
            SELECT Account_number , Name, DOB , Psw , Balance FROM bank WHERE Account_number = @AccountNumber
        END
        ELSE
            PRINT 'Invalid transaction type.';
    END
    ELSE
    BEGIN
        PRINT 'Incorrect password.';
    END
END;
```

```
Commands completed successfully.

Completion time: 2024-02-01T16:41:37.7733353+05:30
```

CHAPTER-5

WEEK-4 PROGRESS

5.1 LEAD and LAG Functions

LEAD and LAG are window functions in SQL that allow you to access data from subsequent rows (LEAD) or preceding rows (LAG) within the result set.

```
SELECT
    employee_id,
    salary,
    LEAD(salary) OVER (ORDER BY employee_id) AS next_salary,
    LAG(salary) OVER (ORDER BY employee_id) AS prev_salary
FROM
    employee;
```

5.2 Slowly Changing Dimensions (SCD)

SCD refers to the techniques used in data warehousing to manage changes in dimension attributes over time. There are three main types:

Type 1 (No History): Overwrite the existing data with the new values.

Type 2 (Add New Row): Create a new record with the updated information, maintaining historical records.

Type 3 (Add Columns): Add columns to the existing record to track changes.

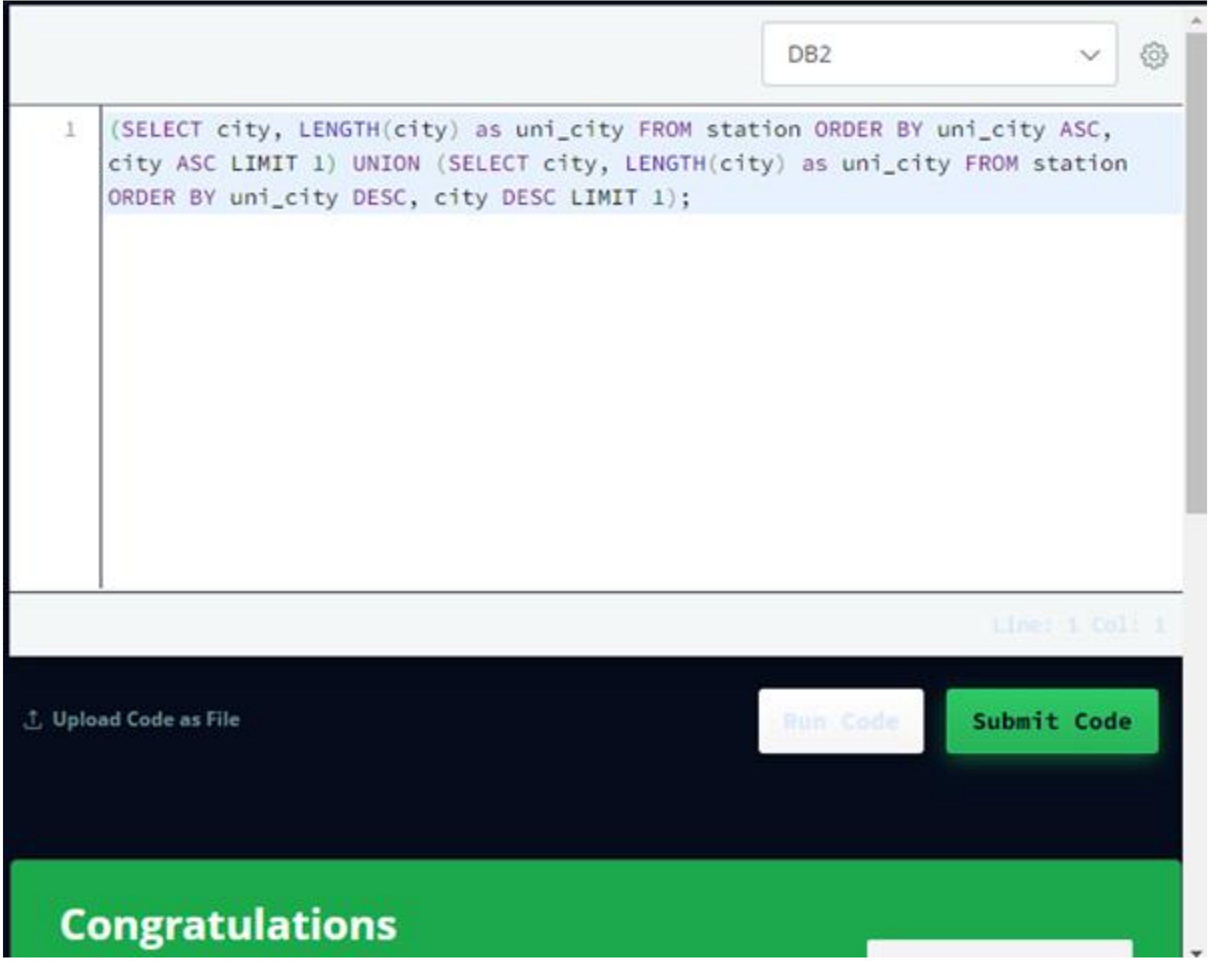
5.3 Stored Procedures

A stored procedure is a precompiled collection of one or more SQL statements that can be executed as a single unit. It is stored in the database and can be called from applications or other stored procedures.

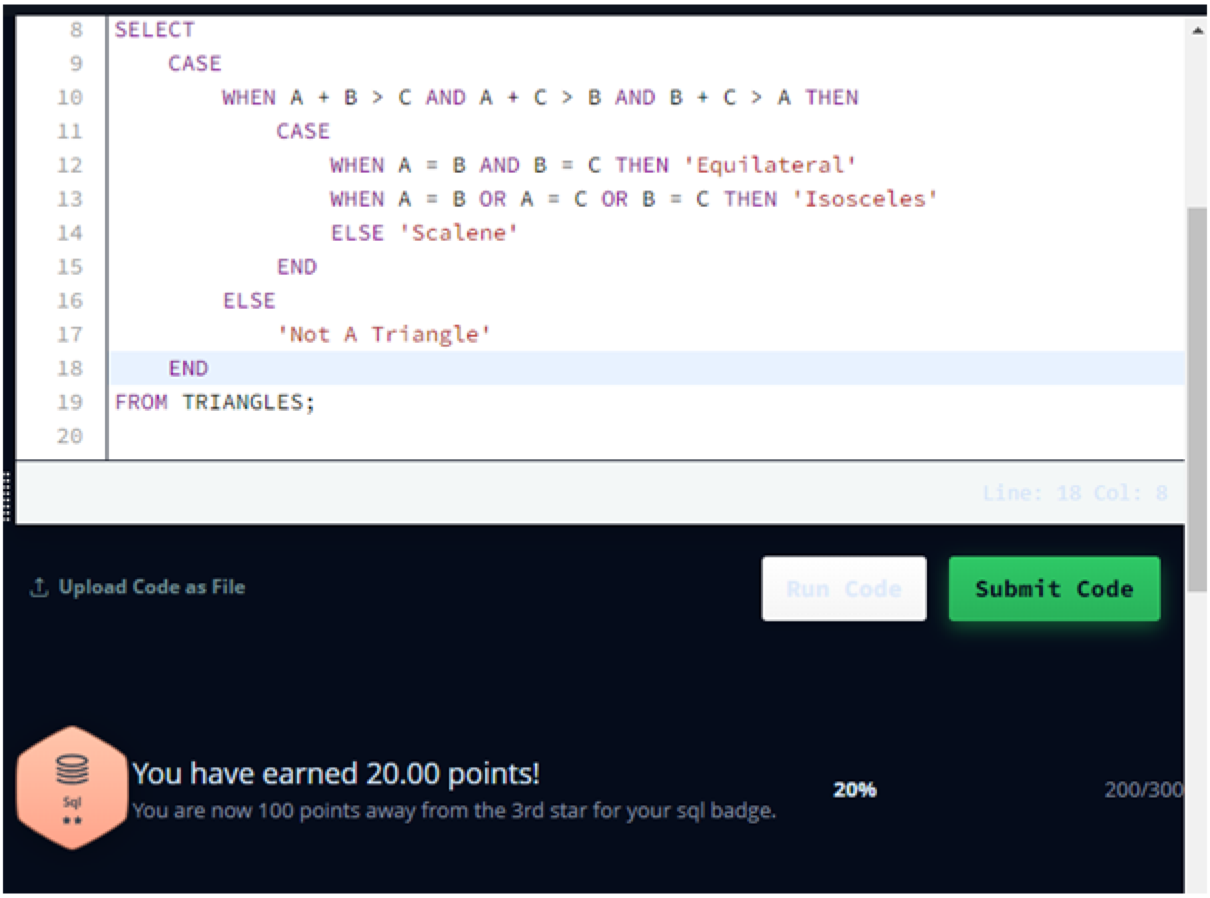
```
CREATE PROCEDURE GetEmployeeDetails
    @EmployeeID INT
AS
BEGIN
    SELECT *
    FROM Employees
    WHERE EmployeeID = @EmployeeID;
END
```

5.4 Screenshot/Code Snippet working on HackerRank

I have leant about SINGLE STORE installation process , how singlestore works , Shard Key , Data Skew , High Availability , type of Table and singlestore tools , SINGLE STORE installation process with cloud version and connect it with MySQL Workbanch , Indexing , Type of Indexing , Views , SPLIT and Store it in to new column Assignment and I have many Methods to implement dynamic way to store it , interval table with dates which are mentation in Document. In Extra , work on HackerRank problem solving tasks.



```sql
(SELECT city, LENGTH(city) as uni_city FROM station ORDER BY uni_city ASC, city ASC LIMIT 1) UNION (SELECT city, LENGTH(city) as uni_city FROM station ORDER BY uni_city DESC, city DESC LIMIT 1);
```



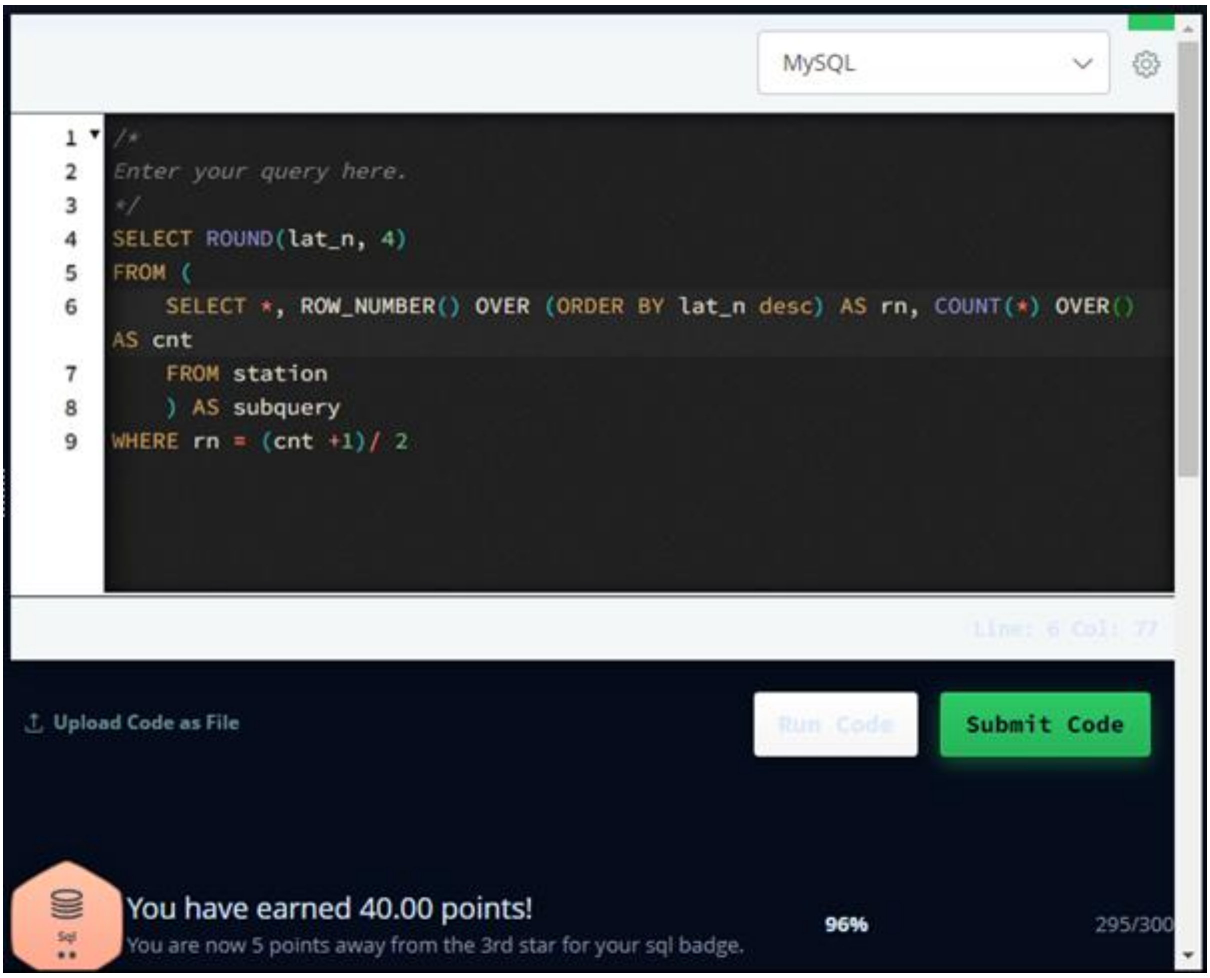
```
SELECT
    CASE
        WHEN A + B > C AND A + C > B AND B + C > A THEN
            CASE
                WHEN A = B AND B = C THEN 'Equilateral'
                WHEN A = B OR A = C OR B = C THEN 'Isosceles'
                ELSE 'Scalene'
            END
        ELSE
            'Not A Triangle'
    END
FROM TRIANGLES;
```

You have earned 20.00 points!

You are now 100 points away from the 3rd star for your sql badge.

20%

200/300



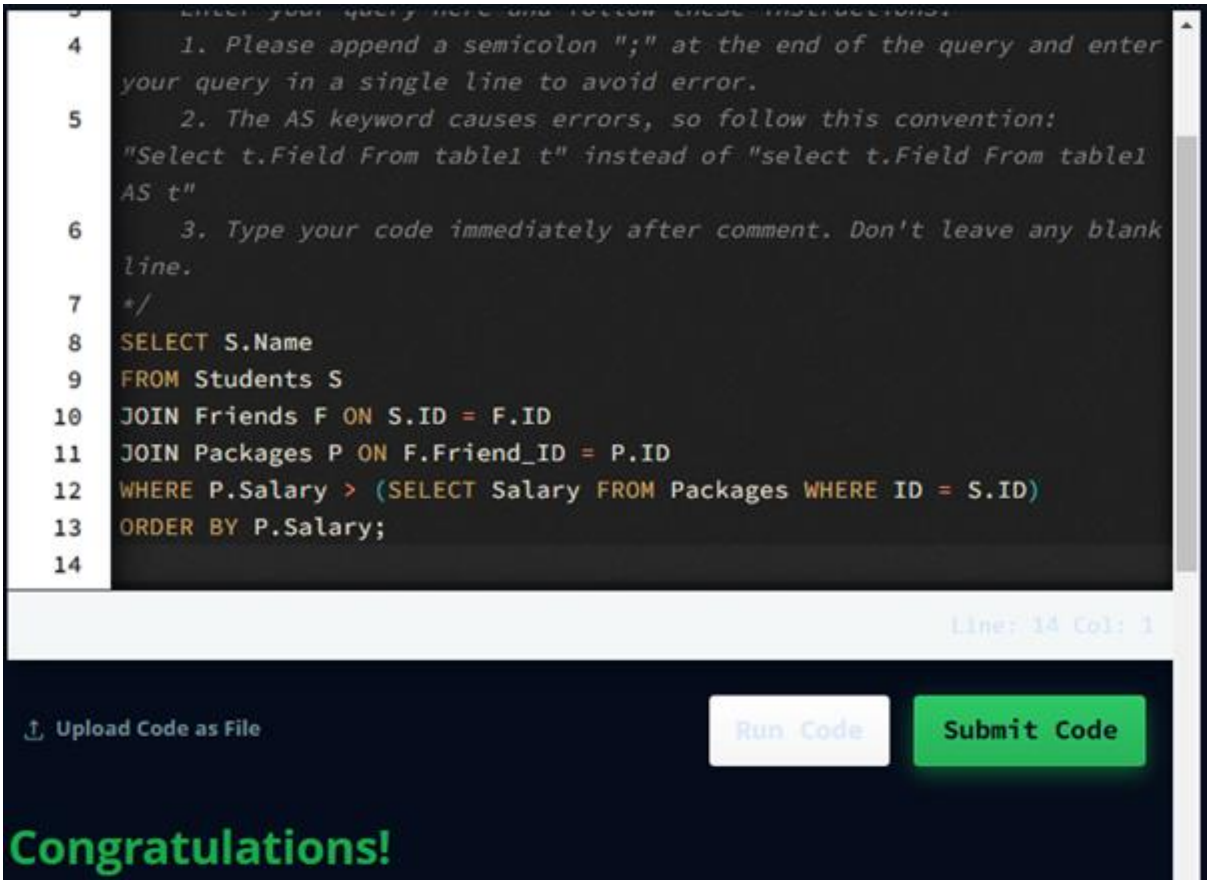
```
/*
Enter your query here.
*/
SELECT ROUND(lat_n, 4)
FROM (
    SELECT *, ROW_NUMBER() OVER (ORDER BY lat_n desc) AS rn, COUNT(*) OVER() AS cnt
    FROM station
    ) AS subquery
WHERE rn = (cnt +1)/ 2
```

You have earned 40.00 points!

You are now 5 points away from the 3rd star for your sql badge.

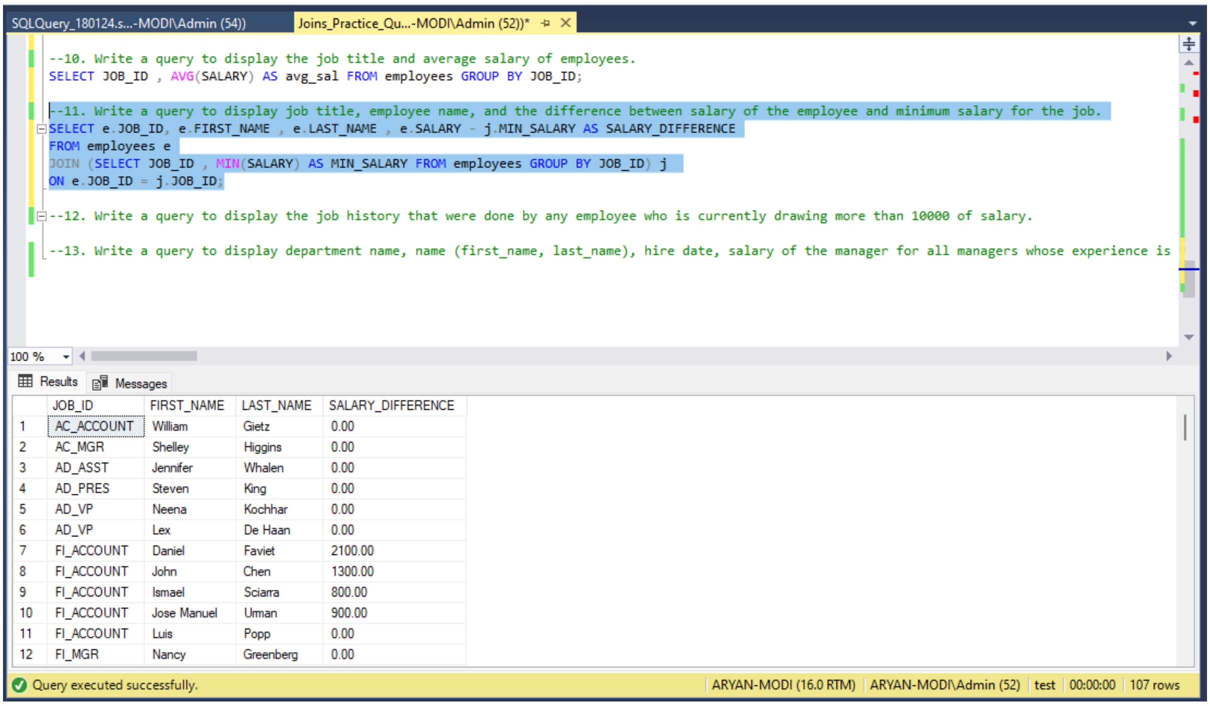
96%

295/300



```
Enter your query here and follow these instructions:
    1. Please append a semicolon ";" at the end of the query and enter your query in a single line to avoid error.
    2. The AS keyword causes errors, so follow this convention: "Select t.Field From table1 t" instead of "select t.Field From table1 AS t"
    3. Type your code immediately after comment. Don't leave any blank line.
*/
SELECT S.Name
FROM Students S
JOIN Friends F ON S.ID = F.ID
JOIN Packages P ON F.Friend_ID = P.ID
WHERE P.Salary > (SELECT Salary FROM Packages WHERE ID = S.ID)
ORDER BY P.Salary;
```

Congratulations!

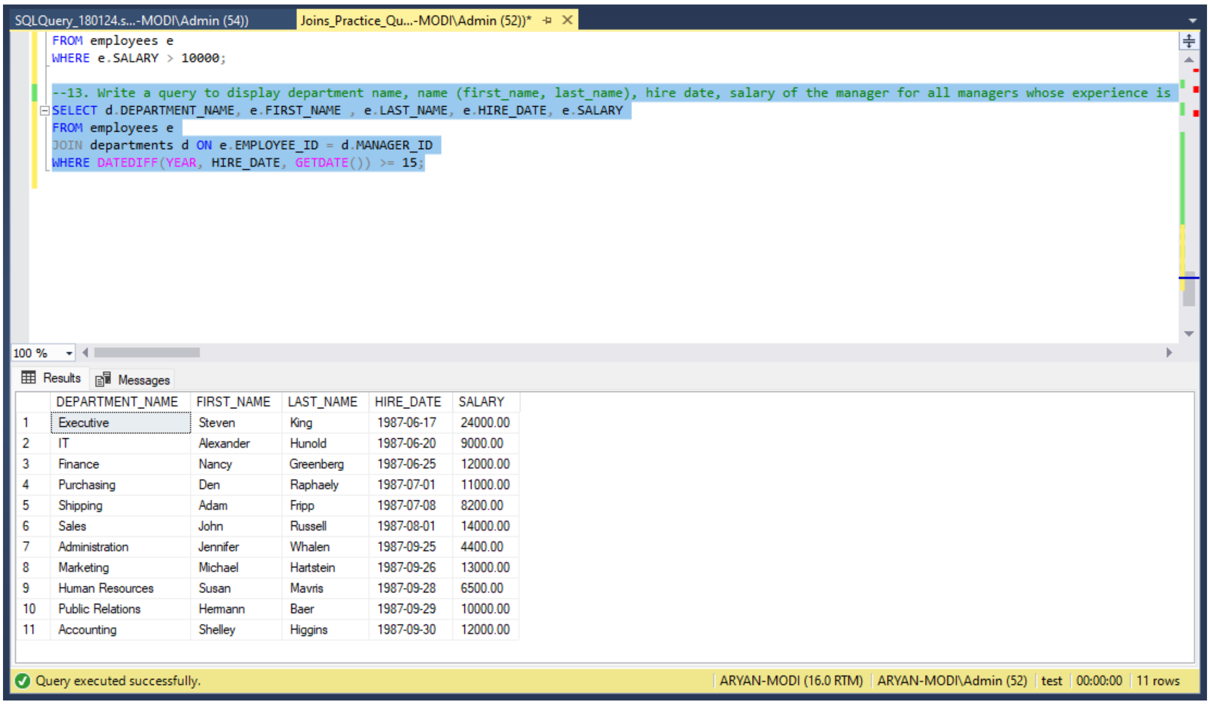


SQLQuery_180124.s...-MODI\Admin (54))
Joins_Practice_Qu...-MODI\Admin (52))*
100 %
Results
Messages

```
--10. Write a query to display the job title and average salary of employees.
SELECT JOB_ID , AVG(SALARY) AS avg_sal FROM employees GROUP BY JOB_ID;

--11. Write a query to display job title, employee name, and the difference between salary of the employee and minimum salary for the job.
SELECT e.JOB_ID, e.FIRST_NAME , e.LAST_NAME , e.SALARY - j.MIN_SALARY AS SALARY_DIFFERENCE
FROM employees e
JOIN (SELECT JOB_ID , MIN(SALARY) AS MIN_SALARY FROM employees GROUP BY JOB_ID) j
ON e.JOB_ID = j.JOB_ID;

--12. Write a query to display the job history that were done by any employee who is currently drawing more than 10000 of salary.

--13. Write a query to display department name, name (first_name, last_name), hire date, salary of the manager for all managers whose experience is
```

	JOB_ID	FIRST_NAME	LAST_NAME	SALARY_DIFFERENCE
1	AC_ACCOUNT	William	Gietz	0.00
2	AC_MGR	Shelley	Higgins	0.00
3	AD_ASST	Jennifer	Whalen	0.00
4	AD_PRES	Steven	King	0.00
5	AD_VP	Neena	Kochhar	0.00
6	AD_VP	Lex	De Haan	0.00
7	FI_ACCOUNT	Daniel	Faviet	2100.00
8	FI_ACCOUNT	John	Chen	1300.00
9	FI_ACCOUNT	Ismael	Sciarra	800.00
10	FI_ACCOUNT	Jose Manuel	Urman	900.00
11	FI_ACCOUNT	Luis	Popp	0.00
12	FI_MGR	Nancy	Greenberg	0.00

Query executed successfully. | ARYAN-MODI (16.0 RTM) | ARYAN-MODI\Admin (52) | test | 00:00:00 | 107 rows



SQLQuery_180124.s...-MODI\Admin (54))
Joins_Practice_Qu...-MODI\Admin (52))*
100 %
Results
Messages

```sql
FROM employees e
WHERE e.SALARY > 10000;

--13. Write a query to display department name, name (first_name, last_name), hire date, salary of the manager for all managers whose experience is
SELECT d.DEPARTMENT_NAME, e.FIRST_NAME , e.LAST_NAME, e.HIRE_DATE, e.SALARY
FROM employees e
JOIN departments d ON e.EMPLOYEE_ID = d.MANAGER_ID
WHERE DATEDIFF(YEAR, HIRE_DATE, GETDATE()) >= 15;
```

	DEPARTMENT_NAME	FIRST_NAME	LAST_NAME	HIRE_DATE	SALARY
1	Executive	Steven	King	1987-06-17	24000.00
2	IT	Alexander	Hunold	1987-06-20	9000.00
3	Finance	Nancy	Greenberg	1987-06-25	12000.00
4	Purchasing	Den	Raphaely	1987-07-01	11000.00
5	Shipping	Adam	Fripp	1987-07-08	8200.00
6	Sales	John	Russell	1987-08-01	14000.00
7	Administration	Jennifer	Whalen	1987-09-25	4400.00
8	Marketing	Michael	Hartstein	1987-09-26	13000.00
9	Human Resources	Susan	Mavris	1987-09-28	6500.00
10	Public Relations	Hermann	Baer	1987-09-29	10000.00
11	Accounting	Shelley	Higgins	1987-09-30	12000.00

Query executed successfully. | ARYAN-MODI (16.0 RTM) | ARYAN-MODI\Admin (52) | test | 00:00:00 | 11 rows

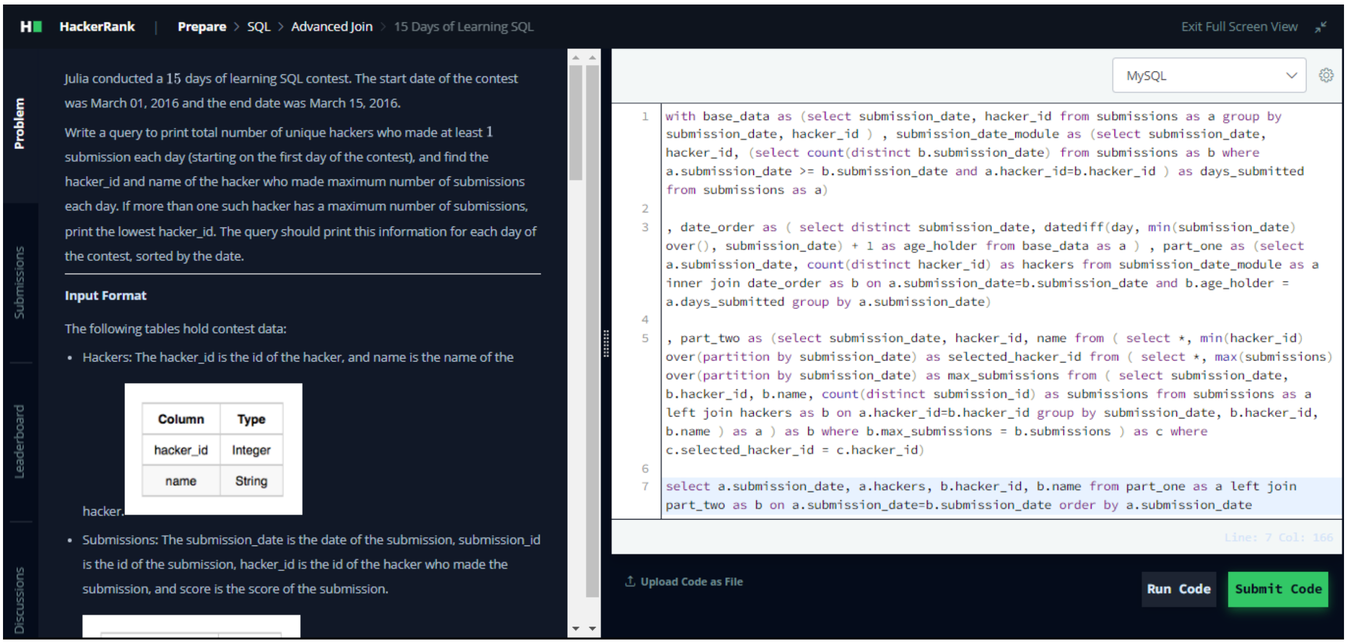
CHAPTER-6

WEEK-5 PROGRESS

6.1. HackerRank Tasks

I have work on HackerRank problem solving tasks.

6.2. Screenshot/Code Snippet working on HackerRank



Julia conducted a 15 days of learning SQL contest. The start date of the contest was March 01, 2016 and the end date was March 15, 2016.

Write a query to print total number of unique hackers who made at least 1 submission each day (starting on the first day of the contest), and find the hacker_id and name of the hacker who made maximum number of submissions each day. If more than one such hacker has a maximum number of submissions, print the lowest hacker_id. The query should print this information for each day of the contest, sorted by the date.

Input Format

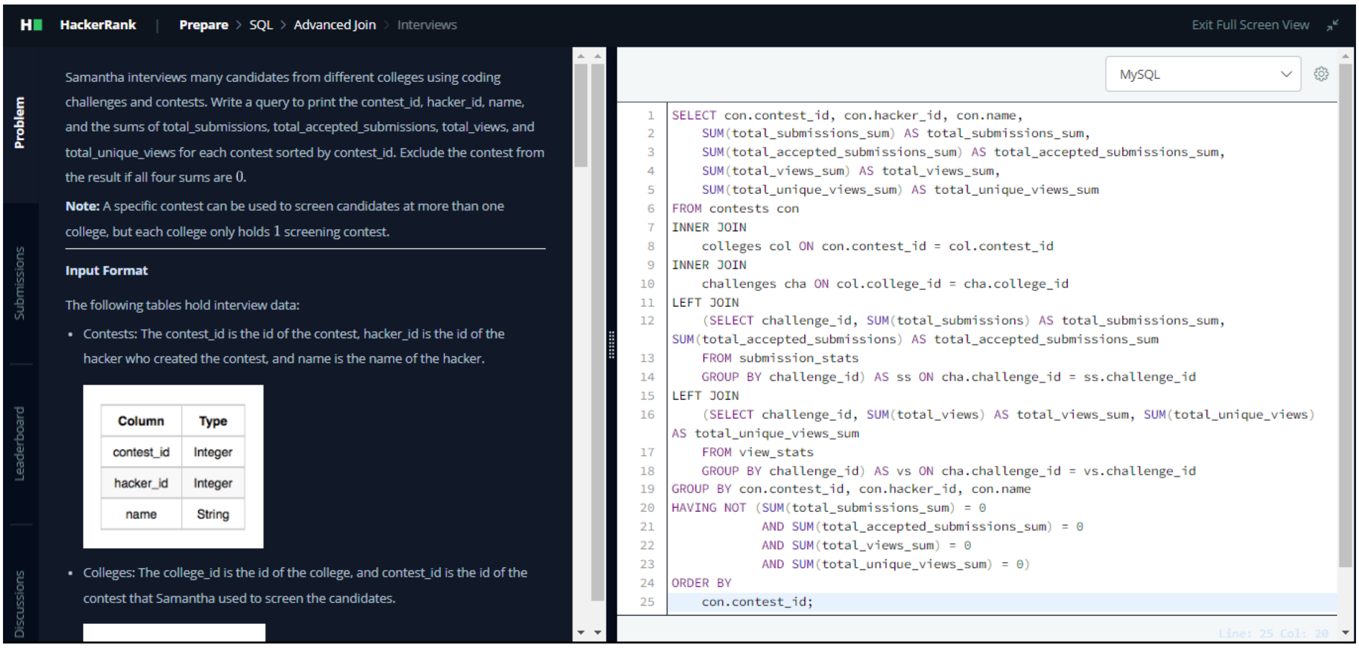
The following tables hold contest data:

- Hackers: The hacker_id is the id of the hacker, and name is the name of the hacker.

Column	Type
hacker_id	Integer
name	String

- Submissions: The submission_date is the date of the submission, submission_id is the id of the submission, hacker_id is the id of the hacker who made the submission, and score is the score of the submission.

```sql
with base_data as (select submission_date, hacker_id from submissions as a group by
submission_date, hacker_id ) , submission_date_module as (select submission_date,
hacker_id, (select count(distinct b.submission_date) from submissions as b where
a.submission_date >= b.submission_date and a.hacker_id=b.hacker_id ) as days_submitted
from submissions as a)

, date_order as ( select distinct submission_date, datediff(day, min(submission_date)
over(), submission_date) + 1 as age_holder from base_data as a ) , part_one as (select
a.submission_date, count(distinct hacker_id) as hackers from submission_date_module as a
inner join date_order as b on a.submission_date=b.submission_date and b.age_holder =
a.days_submitted group by a.submission_date)

, part_two as (select submission_date, hacker_id, name from ( select *, min(hacker_id)
over(partition by submission_date) as selected_hacker_id from ( select *, max(submissions)
over(partition by submission_date) as max_submissions from ( select submission_date,
b.hacker_id, b.name, count(distinct submission_id) as submissions from submissions as a
left join hackers as b on a.hacker_id=b.hacker_id group by submission_date, b.hacker_id,
b.name ) as a ) as b where b.max_submissions = b.submissions ) as c where
c.selected_hacker_id = c.hacker_id)

select a.submission_date, a.hackers, b.hacker_id, b.name from part_one as a left join
part_two as b on a.submission_date=b.submission_date order by a.submission_date
```



Samantha interviews many candidates from different colleges using coding challenges and contests. Write a query to print the contest_id, hacker_id, name, and the sums of total_submissions, total_accepted_submissions, total_views, and total_unique_views for each contest sorted by contest_id. Exclude the contest from the result if all four sums are 0.

Note: A specific contest can be used to screen candidates at more than one college, but each college only holds 1 screening contest.

Input Format

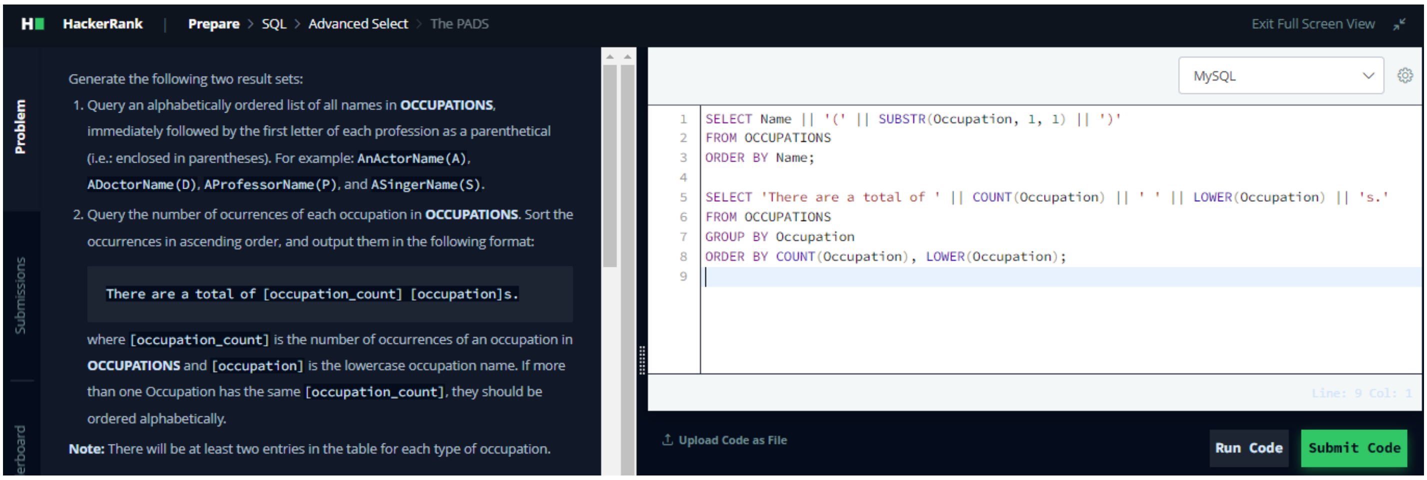
The following tables hold interview data:

- Contests: The contest_id is the id of the contest, hacker_id is the id of the hacker who created the contest, and name is the name of the hacker.

Column	Type
contest_id	Integer
hacker_id	Integer
name	String

- Colleges: The college_id is the id of the college, and contest_id is the id of the contest that Samantha used to screen the candidates.

```sql
SELECT con.contest_id, con.hacker_id, con.name,
    SUM(total_submissions_sum) AS total_submissions_sum,
    SUM(total_accepted_submissions_sum) AS total_accepted_submissions_sum,
    SUM(total_views_sum) AS total_views_sum,
    SUM(total_unique_views_sum) AS total_unique_views_sum
FROM contests con
INNER JOIN
    colleges col ON con.contest_id = col.contest_id
INNER JOIN
    challenges cha ON col.college_id = cha.college_id
LEFT JOIN
    (SELECT challenge_id, SUM(total_submissions) AS total_submissions_sum,
SUM(total_accepted_submissions) AS total_accepted_submissions_sum
    FROM submission_stats
    GROUP BY challenge_id) AS ss ON cha.challenge_id = ss.challenge_id
LEFT JOIN
    (SELECT challenge_id, SUM(total_views) AS total_views_sum, SUM(total_unique_views)
AS total_unique_views_sum
    FROM view_stats
    GROUP BY challenge_id) AS vs ON cha.challenge_id = vs.challenge_id
GROUP BY con.contest_id, con.hacker_id, con.name
HAVING NOT (SUM(total_submissions_sum) = 0
            AND SUM(total_accepted_submissions_sum) = 0
            AND SUM(total_views_sum) = 0
            AND SUM(total_unique_views_sum) = 0)
ORDER BY
    con.contest_id;
```



Generate the following two result sets:

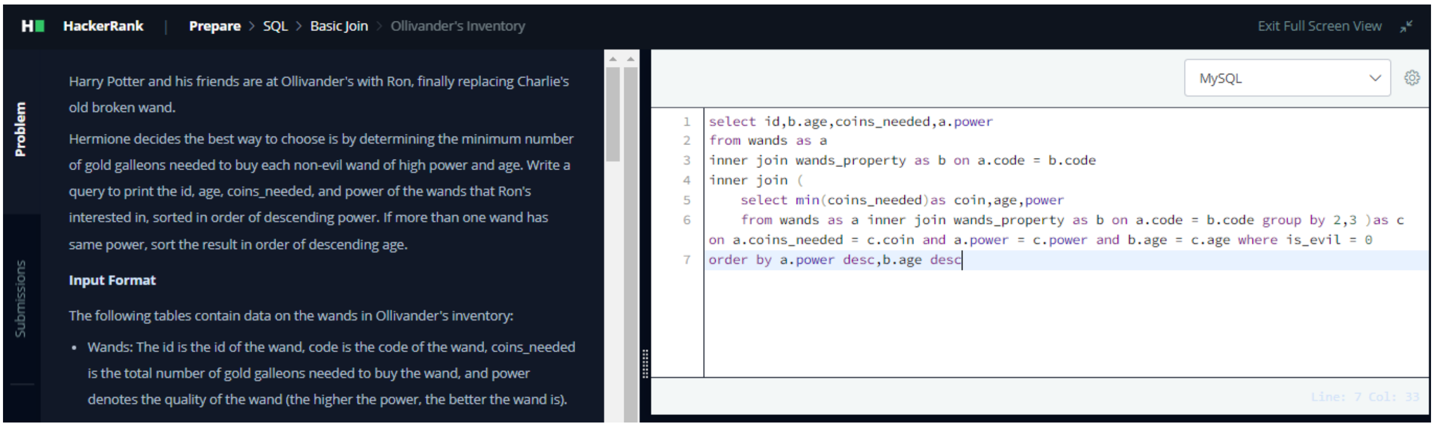
1. Query an alphabetically ordered list of all names in **OCCUPATIONS**, immediately followed by the first letter of each profession as a parenthetical (i.e.: enclosed in parentheses). For example: `AnActorName(A)`, `ADoctorName(D)`, `AProfessorName(P)`, and `ASingerName(S)`.
2. Query the number of ocurrences of each occupation in **OCCUPATIONS**. Sort the occurrences in ascending order, and output them in the following format:

```
There are a total of [occupation_count] [occupation]s.
```

where `[occupation_count]` is the number of occurrences of an occupation in **OCCUPATIONS** and `[occupation]` is the lowercase occupation name. If more than one Occupation has the same `[occupation_count]`, they should be ordered alphabetically.

Note: There will be at least two entries in the table for each type of occupation.

```sql
SELECT Name || '(' || SUBSTR(Occupation, 1, 1) || ')'
FROM OCCUPATIONS
ORDER BY Name;

SELECT 'There are a total of ' || COUNT(Occupation) || ' ' || LOWER(Occupation) || 's.'
FROM OCCUPATIONS
GROUP BY Occupation
ORDER BY COUNT(Occupation), LOWER(Occupation);

```



Harry Potter and his friends are at Ollivander's with Ron, finally replacing Charlie's old broken wand.

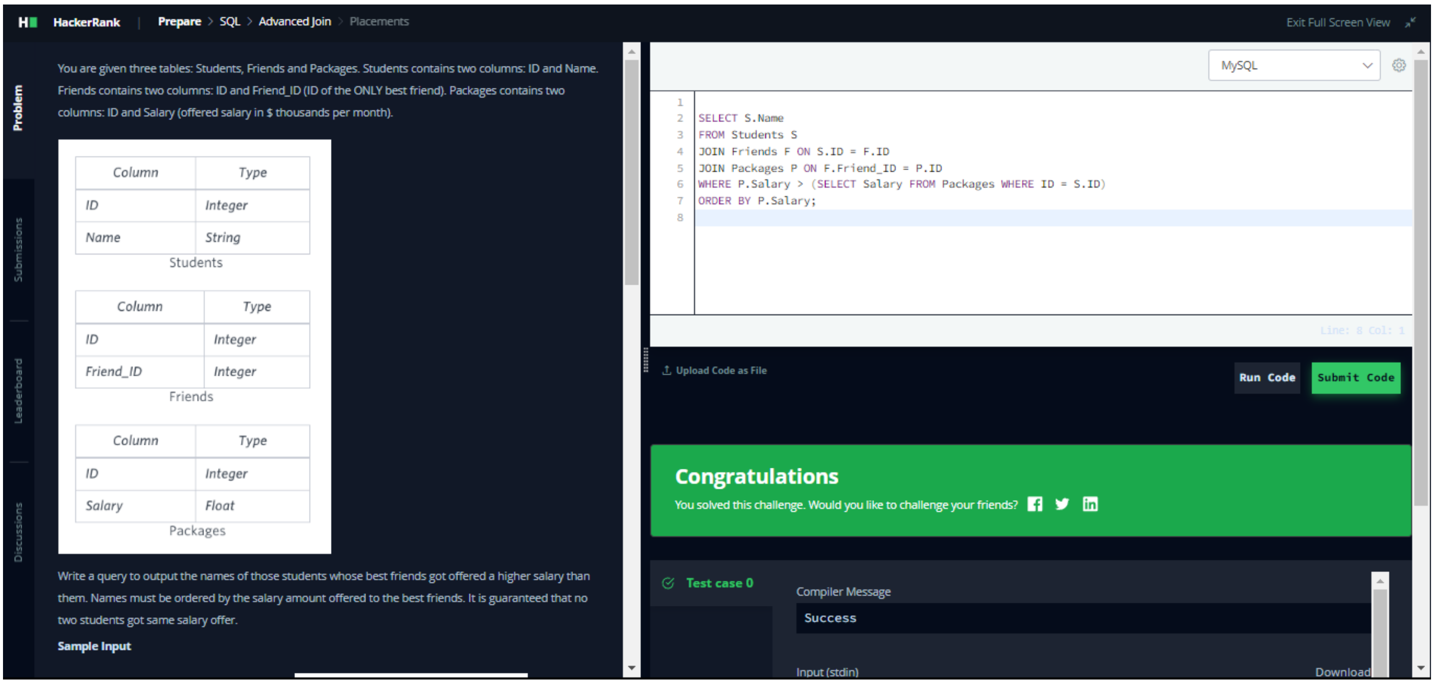
Hermione decides the best way to choose is by determining the minimum number of gold galleons needed to buy each non-evil wand of high power and age. Write a query to print the id, age, coins_needed, and power of the wands that Ron's interested in, sorted in order of descending power. If more than one wand has same power, sort the result in order of descending age.

Input Format

The following tables contain data on the wands in Ollivander's inventory:

- Wands: The id is the id of the wand, code is the code of the wand, coins_needed is the total number of gold galleons needed to buy the wand, and power denotes the quality of the wand (the higher the power, the better the wand is).

```sql
select id,b.age,coins_needed,a.power
from wands as a
inner join wands_property as b on a.code = b.code
inner join (
    select min(coins_needed)as coin,age,power
    from wands as a inner join wands_property as b on a.code = b.code group by 2,3 )as c on a.coins_needed = c.coin and a.power = c.power and b.age = c.age where is_evil = 0
order by a.power desc,b.age desc
```



You are given three tables: Students, Friends and Packages. Students contains two columns: ID and Name. Friends contains two columns: ID and Friend_ID (ID of the ONLY best friend). Packages contains two columns: ID and Salary (offered salary in $ thousands per month).

Column	Type
ID	Integer
Name	String

Students

Column	Type
ID	Integer
Friend_ID	Integer

Friends

Column	Type
ID	Integer
Salary	Float

Packages

Write a query to output the names of those students whose best friends got offered a higher salary than them. Names must be ordered by the salary amount offered to the best friends. It is guaranteed that no two students got same salary offer.

Sample Input

```sql

SELECT S.Name
FROM Students S
JOIN Friends F ON S.ID = F.ID
JOIN Packages P ON F.Friend_ID = P.ID
WHERE P.Salary > (SELECT Salary FROM Packages WHERE ID = S.ID)
ORDER BY P.Salary;

```

Congratulations

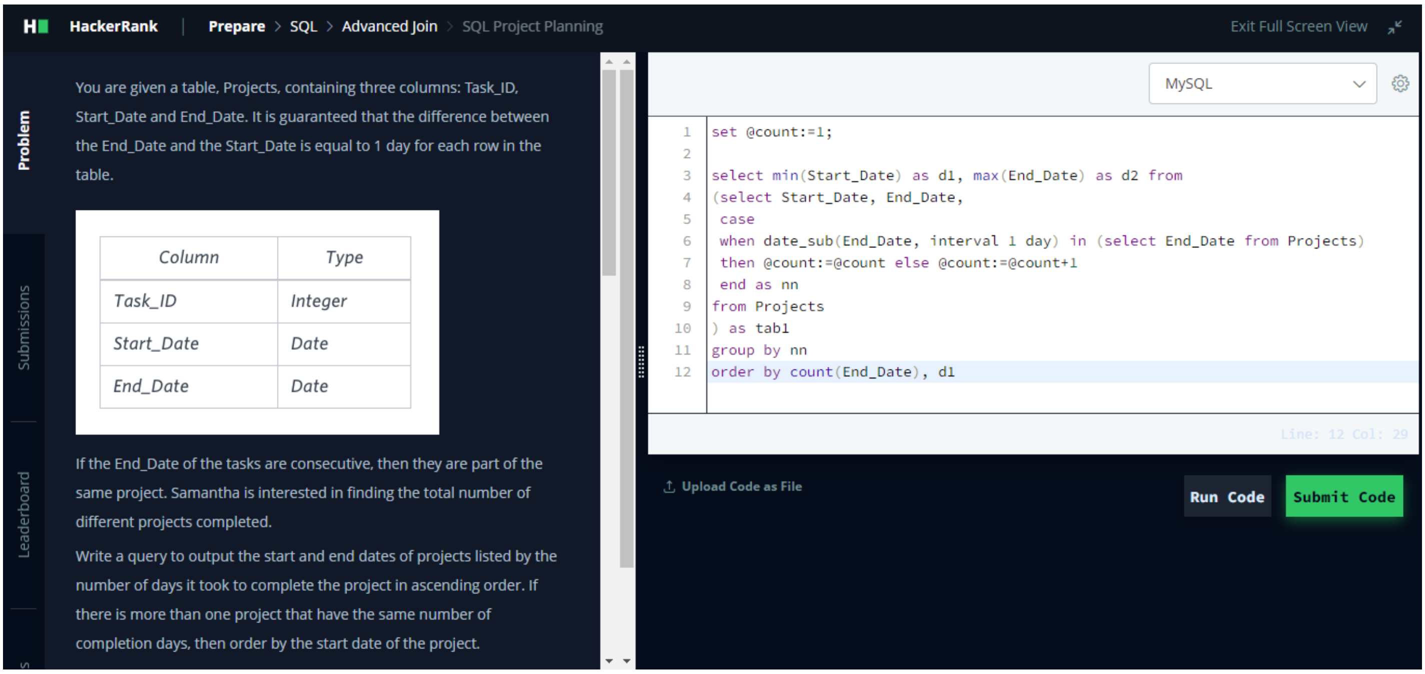
You solved this challenge. Would you like to challenge your friends?

✓ Test case 0

Compiler Message

```
Success
```

Input (stdin) Download



You are given a table, Projects, containing three columns: Task_ID, Start_Date and End_Date. It is guaranteed that the difference between the End_Date and the Start_Date is equal to 1 day for each row in the table.

Column	Type
Task_ID	Integer
Start_Date	Date
End_Date	Date

If the End_Date of the tasks are consecutive, then they are part of the same project. Samantha is interested in finding the total number of different projects completed.

Write a query to output the start and end dates of projects listed by the number of days it took to complete the project in ascending order. If there is more than one project that have the same number of completion days, then order by the start date of the project.

```sql
set @count:=1;

select min(Start_Date) as d1, max(End_Date) as d2 from
(select Start_Date, End_Date,
 case
 when date_sub(End_Date, interval 1 day) in (select End_Date from Projects)
 then @count:=@count else @count:=@count+1
 end as nn
from Projects
) as tab1
group by nn
order by count(End_Date), d1
```

CHAPTER-7

WEEK-6 PROGRESS

7.1 Improve Logic Building skill

7.1.1 **Stored Procedures:** Practice creating and optimizing stored procedures for common tasks such as data manipulation, reporting, and business logic implementation.Explore advanced features of stored procedures such as parameterization, error handling, transactions, and dynamic SQL.

7.1.2 **Subqueries**: Deepen your understanding of subqueries by practicing various types including scalar subqueries, correlated subqueries, nested subqueries, and common table expressions (CTEs).Work on tasks that require you to use subqueries for filtering, joining, and aggregating data from multiple tables.

7.1.3 **Logic Building Tasks**: Challenge yourself with complex logic building tasks that involve multiple SQL operations such as filtering, sorting, grouping, and joining data to achieve specific outcomes.Practice breaking down complex problems into smaller, manageable steps and implementing the logic using SQL queries.

7.1.4 **SingleStore Architecture**: Continue exploring SingleStore architecture and its features by working on real-world scenarios and use cases.Experiment with optimizing query performance in SingleStore by leveraging its distributed architecture, indexing strategies, and query optimization techniques.

7.1.5 **Data Manipulation Tasks**: Work with real datasets such as cricket data or revenue data to perform various data manipulation tasks using SQL.Practice tasks like data cleaning, transformation, aggregation, and analysis to gain practical experience in handling real-world data scenarios.

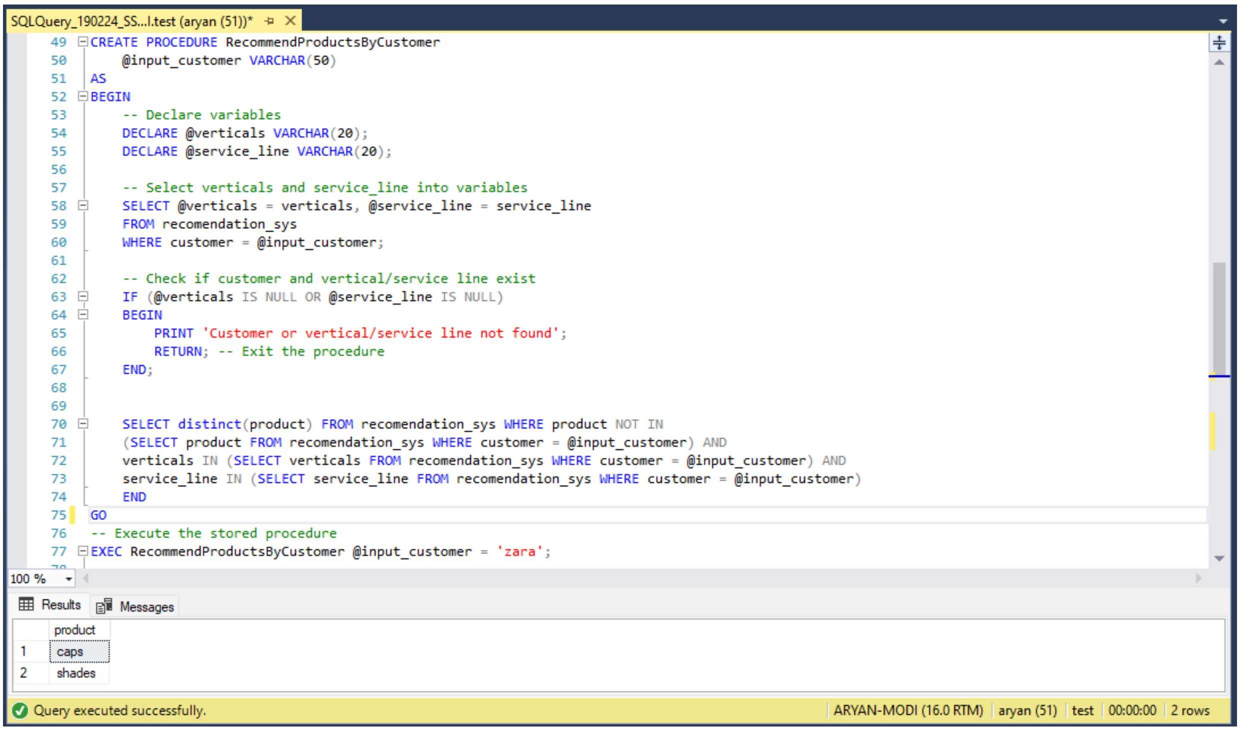
7.1.6 **Advanced SQL Functions**: Explore advanced SQL functions and features such as window functions, recursive queries, pivot tables, string manipulation functions, and date/time functions.Experiment with using these functions in combination with other SQL operations to solve complex problems efficiently.

7.1.7 **Performance Tuning and Optimization**: Learn about performance tuning and optimization techniques in SQL, including indexing, query optimization, partitioning, and caching strategies.Practice identifying and resolving performance bottlenecks in SQL queries and procedures to improve overall system performance.

7.1.8 **Continuous Learning and Practice**: Stay updated with the latest advancements and best practices in SQL by reading articles, tutorials, and documentation.Regularly participate in online coding platforms, forums, and communities to solve SQL challenges and share knowledge with peers.

7.2. Screenshot/Code Snippet working on Task

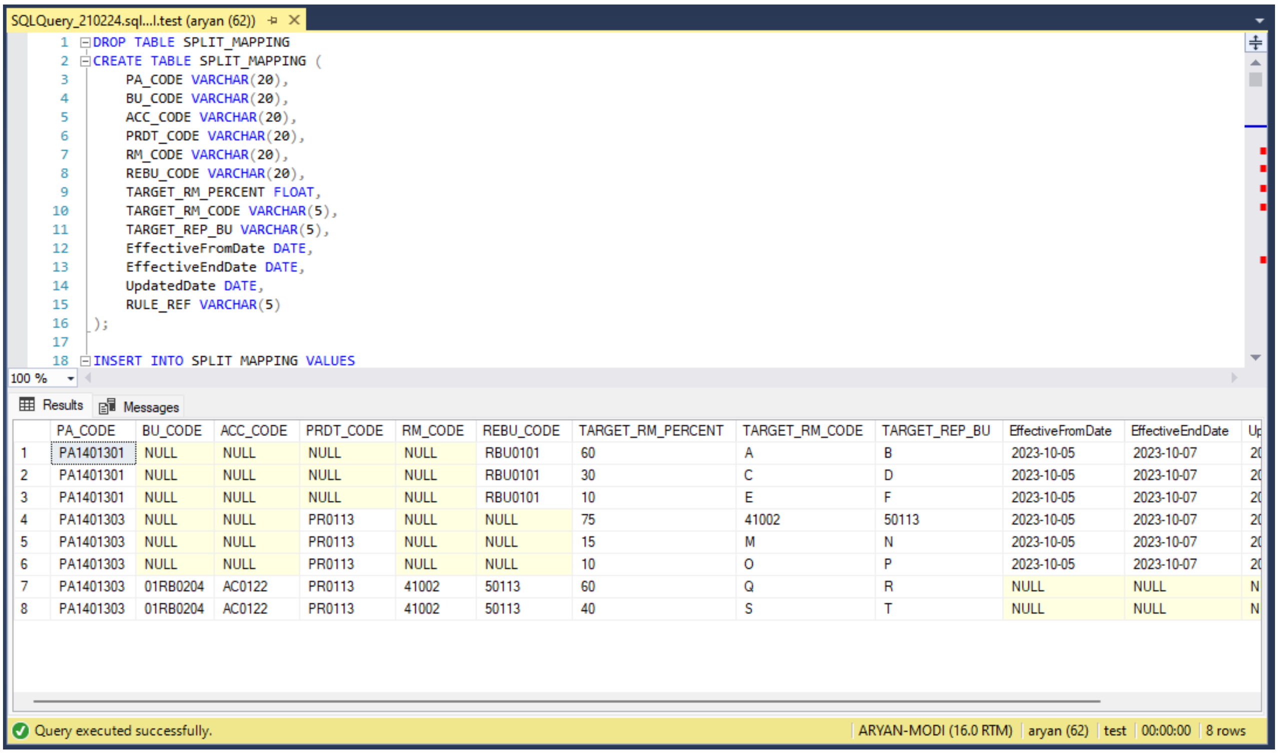
I have work on task regarding procedure , subquery , logic building task which mentation in document. I have leant about singlestore architecture , Demo: Recognizing Faces Using a SQL Database , SingleStore Architecture Overview and Product Demo , Demo: Recognizing Faces Using a SQL Database , Revenue Split Level 1 and learn about mapping and join ,cricket data and explore Singlestore and perform PIVOTE and SPLIT STRING functions , worked on SQL Logic Building which mention in document.



```sql
CREATE PROCEDURE RecommendProductsByCustomer
    @input_customer VARCHAR(50)
AS
BEGIN
    -- Declare variables
    DECLARE @verticals VARCHAR(20);
    DECLARE @service_line VARCHAR(20);

    -- Select verticals and service_line into variables
    SELECT @verticals = verticals, @service_line = service_line
    FROM recomendation_sys
    WHERE customer = @input_customer;

    -- Check if customer and vertical/service line exist
    IF (@verticals IS NULL OR @service_line IS NULL)
    BEGIN
        PRINT 'Customer or vertical/service line not found';
        RETURN; -- Exit the procedure
    END;


    SELECT distinct(product) FROM recomendation_sys WHERE product NOT IN
    (SELECT product FROM recomendation_sys WHERE customer = @input_customer) AND
    verticals IN (SELECT verticals FROM recomendation_sys WHERE customer = @input_customer) AND
    service_line IN (SELECT service_line FROM recomendation_sys WHERE customer = @input_customer)
    END
GO
-- Execute the stored procedure
EXEC RecommendProductsByCustomer @input_customer = 'zara';
```

	product
1	caps
2	shades

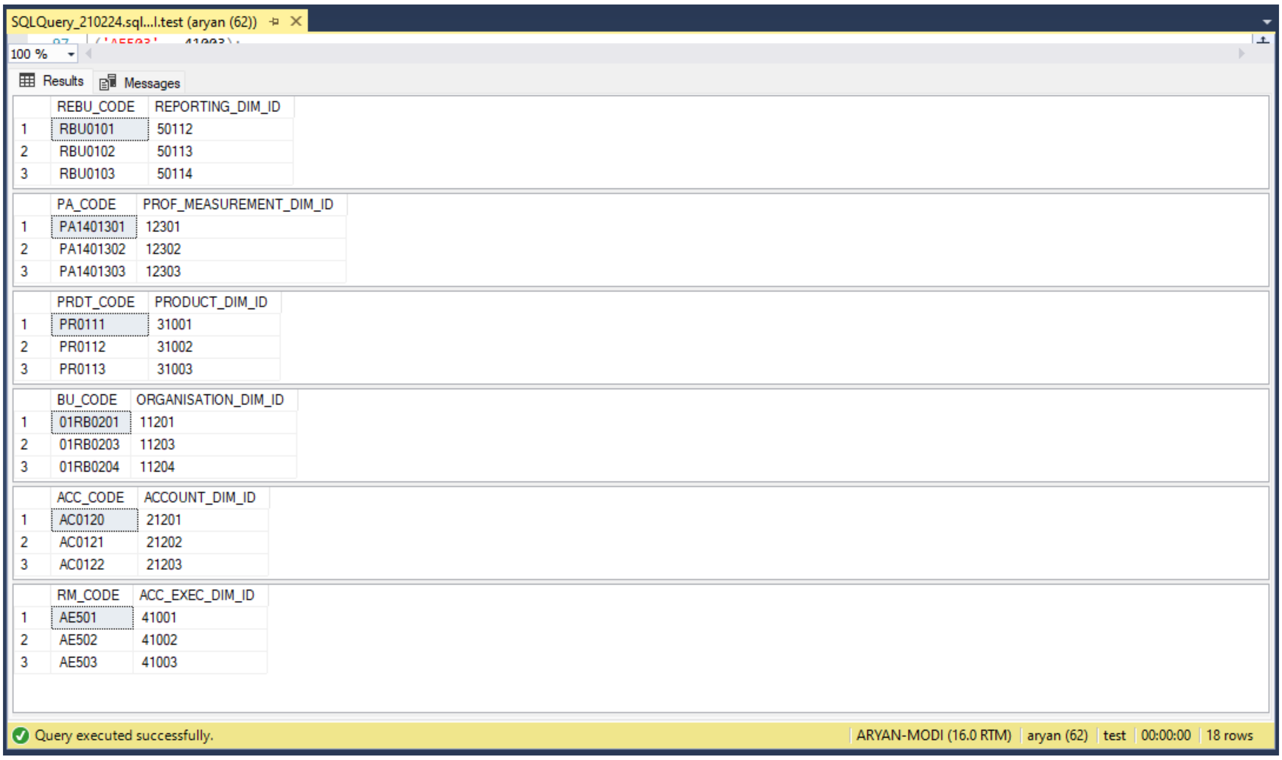
Query executed successfully. | ARYAN-MODI (16.0 RTM) | aryan (51) | test | 00:00:00 | 2 rows



```sql
DROP TABLE SPLIT_MAPPING
CREATE TABLE SPLIT_MAPPING (
    PA_CODE VARCHAR(20),
    BU_CODE VARCHAR(20),
    ACC_CODE VARCHAR(20),
    PRDT_CODE VARCHAR(20),
    RM_CODE VARCHAR(20),
    REBU_CODE VARCHAR(20),
    TARGET_RM_PERCENT FLOAT,
    TARGET_RM_CODE VARCHAR(5),
    TARGET_REP_BU VARCHAR(5),
    EffectiveFromDate DATE,
    EffectiveEndDate DATE,
    UpdatedDate DATE,
    RULE_REF VARCHAR(5)
);

INSERT INTO SPLIT MAPPING VALUES
```

	PA_CODE	BU_CODE	ACC_CODE	PRDT_CODE	RM_CODE	REBU_CODE	TARGET_RM_PERCENT	TARGET_RM_CODE	TARGET_REP_BU	EffectiveFromDate	EffectiveEndDate	Up
1	PA1401301	NULL	NULL	NULL	NULL	RBU0101	60	A	B	2023-10-05	2023-10-07	20
2	PA1401301	NULL	NULL	NULL	NULL	RBU0101	30	C	D	2023-10-05	2023-10-07	20
3	PA1401301	NULL	NULL	NULL	NULL	RBU0101	10	E	F	2023-10-05	2023-10-07	20
4	PA1401303	NULL	NULL	PR0113	NULL	NULL	75	41002	50113	2023-10-05	2023-10-07	20
5	PA1401303	NULL	NULL	PR0113	NULL	NULL	15	M	N	2023-10-05	2023-10-07	20
6	PA1401303	NULL	NULL	PR0113	NULL	NULL	10	O	P	2023-10-05	2023-10-07	20
7	PA1401303	01RB0204	AC0122	PR0113	41002	50113	60	Q	R	NULL	NULL	N
8	PA1401303	01RB0204	AC0122	PR0113	41002	50113	40	S	T	NULL	NULL	N

Query executed successfully. | ARYAN-MODI (16.0 RTM) | aryan (62) | test | 00:00:00 | 8 rows



```
('AE503', 41003);
```

	REBU_CODE	REPORTING_DIM_ID
1	RBU0101	50112
2	RBU0102	50113
3	RBU0103	50114

	PA_CODE	PROF_MEASUREMENT_DIM_ID
1	PA1401301	12301
2	PA1401302	12302
3	PA1401303	12303

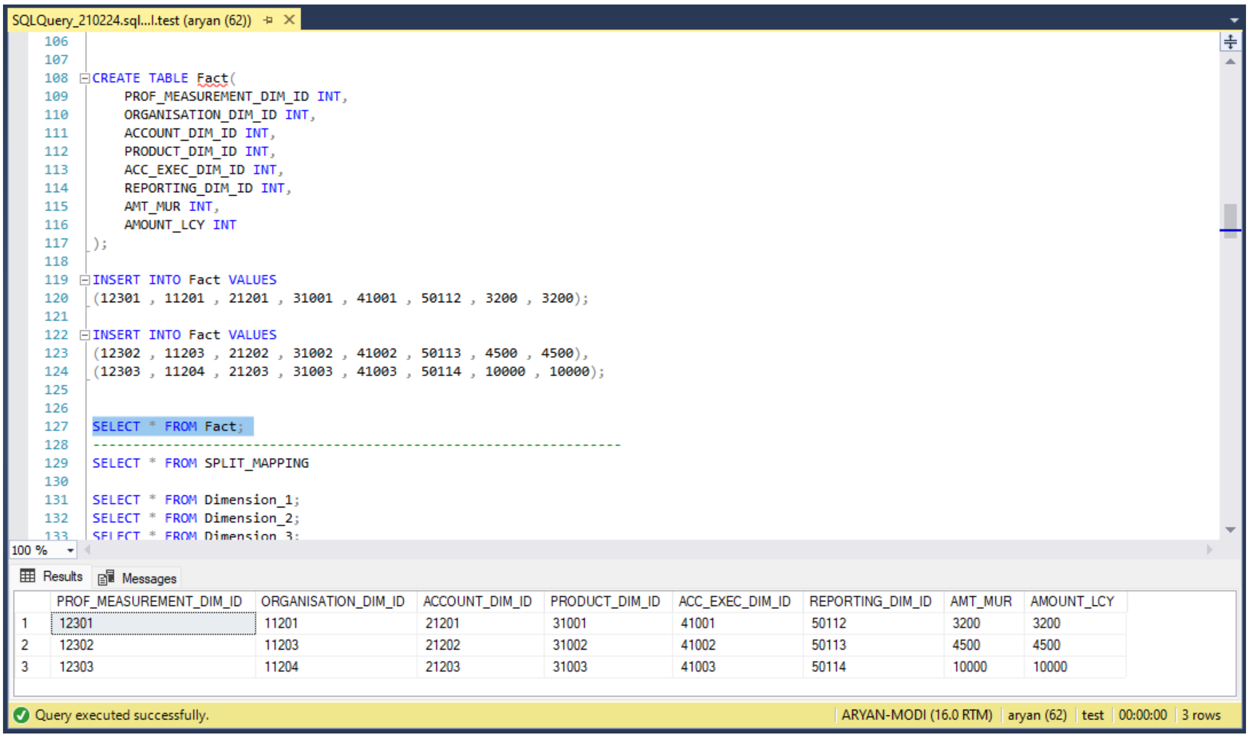
	PRDT_CODE	PRODUCT_DIM_ID
1	PR0111	31001
2	PR0112	31002
3	PR0113	31003

	BU_CODE	ORGANISATION_DIM_ID
1	01RB0201	11201
2	01RB0203	11203
3	01RB0204	11204

	ACC_CODE	ACCOUNT_DIM_ID
1	AC0120	21201
2	AC0121	21202
3	AC0122	21203

	RM_CODE	ACC_EXEC_DIM_ID
1	AE501	41001
2	AE502	41002
3	AE503	41003

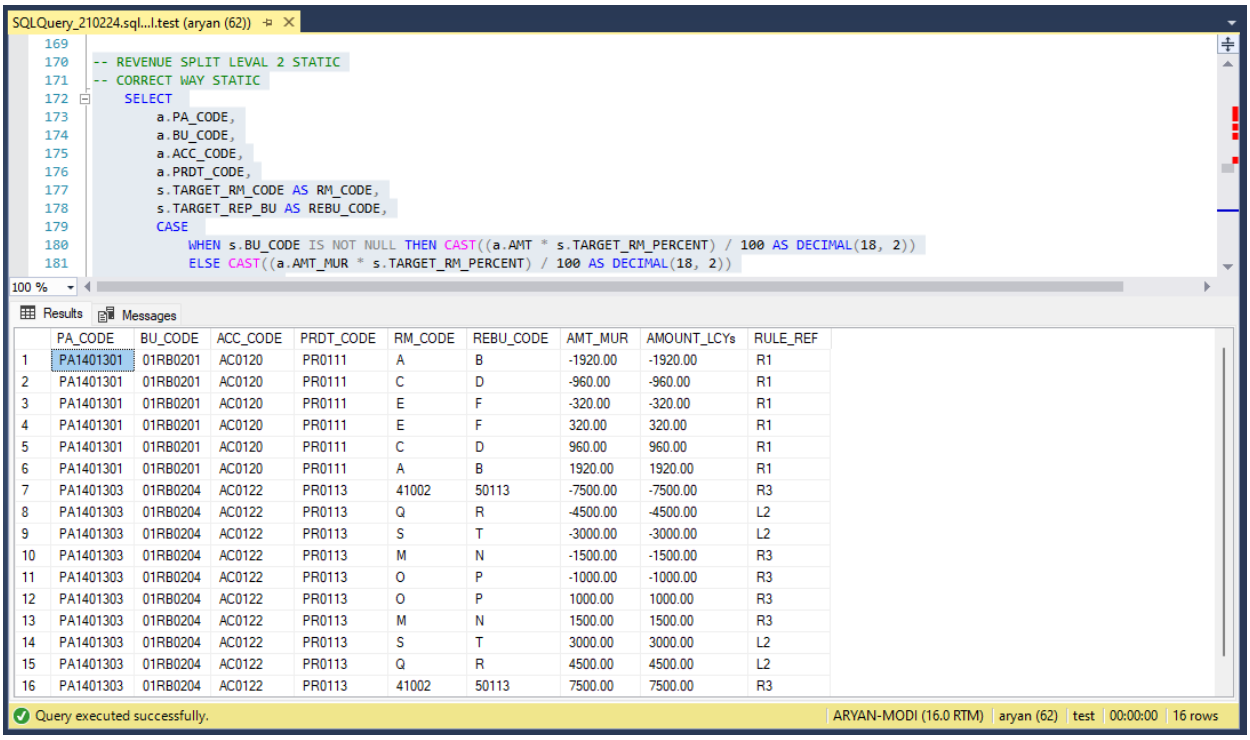
Query executed successfully. | ARYAN-MODI (16.0 RTM) | aryan (62) | test | 00:00:00 | 18 rows



```
CREATE TABLE Fact(
    PROF_MEASUREMENT_DIM_ID INT,
    ORGANISATION_DIM_ID INT,
    ACCOUNT_DIM_ID INT,
    PRODUCT_DIM_ID INT,
    ACC_EXEC_DIM_ID INT,
    REPORTING_DIM_ID INT,
    AMT_MUR INT,
    AMOUNT_LCY INT
);

INSERT INTO Fact VALUES
(12301 , 11201 , 21201 , 31001 , 41001 , 50112 , 3200 , 3200);

INSERT INTO Fact VALUES
(12302 , 11203 , 21202 , 31002 , 41002 , 50113 , 4500 , 4500),
(12303 , 11204 , 21203 , 31003 , 41003 , 50114 , 10000 , 10000);


SELECT * FROM Fact;
-------------------------------------------------------------------
SELECT * FROM SPLIT_MAPPING

SELECT * FROM Dimension_1;
SELECT * FROM Dimension_2;
SELECT * FROM Dimension_3;
```

	PROF_MEASUREMENT_DIM_ID	ORGANISATION_DIM_ID	ACCOUNT_DIM_ID	PRODUCT_DIM_ID	ACC_EXEC_DIM_ID	REPORTING_DIM_ID	AMT_MUR	AMOUNT_LCY
1	12301	11201	21201	31001	41001	50112	3200	3200
2	12302	11203	21202	31002	41002	50113	4500	4500
3	12303	11204	21203	31003	41003	50114	10000	10000

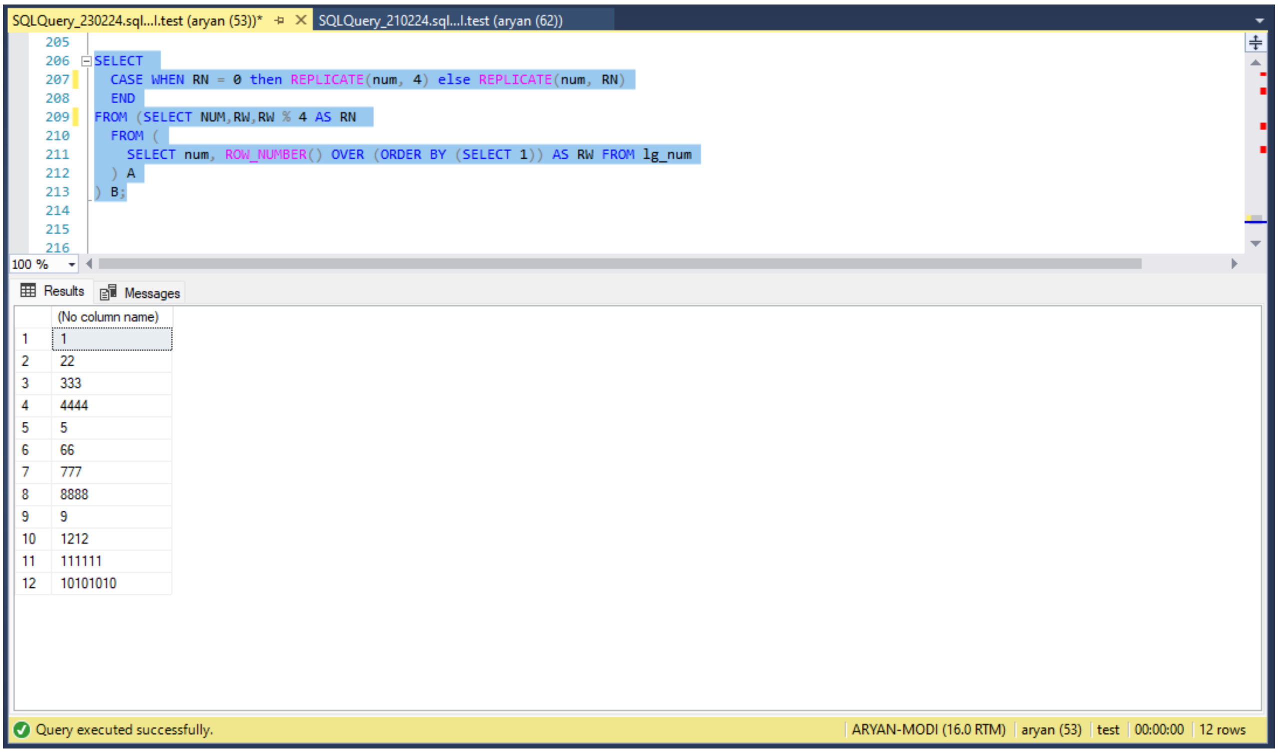
Query executed successfully. | ARYAN-MODI (16.0 RTM) | aryan (62) | test | 00:00:00 | 3 rows



```sql
-- REVENUE SPLIT LEVAL 2 STATIC
-- CORRECT WAY STATIC
    SELECT
        a.PA_CODE,
        a.BU_CODE,
        a.ACC_CODE,
        a.PRDT_CODE,
        s.TARGET_RM_CODE AS RM_CODE,
        s.TARGET_REP_BU AS REBU_CODE,
        CASE
            WHEN s.BU_CODE IS NOT NULL THEN CAST((a.AMT * s.TARGET_RM_PERCENT) / 100 AS DECIMAL(18, 2))
            ELSE CAST((a.AMT_MUR * s.TARGET_RM_PERCENT) / 100 AS DECIMAL(18, 2))
```

	PA_CODE	BU_CODE	ACC_CODE	PRDT_CODE	RM_CODE	REBU_CODE	AMT_MUR	AMOUNT_LCYs	RULE_REF
1	PA1401301	01RB0201	AC0120	PR0111	A	B	-1920.00	-1920.00	R1
2	PA1401301	01RB0201	AC0120	PR0111	C	D	-960.00	-960.00	R1
3	PA1401301	01RB0201	AC0120	PR0111	E	F	-320.00	-320.00	R1
4	PA1401301	01RB0201	AC0120	PR0111	E	F	320.00	320.00	R1
5	PA1401301	01RB0201	AC0120	PR0111	C	D	960.00	960.00	R1
6	PA1401301	01RB0201	AC0120	PR0111	A	B	1920.00	1920.00	R1
7	PA1401303	01RB0204	AC0122	PR0113	41002	50113	-7500.00	-7500.00	R3
8	PA1401303	01RB0204	AC0122	PR0113	Q	R	-4500.00	-4500.00	L2
9	PA1401303	01RB0204	AC0122	PR0113	S	T	-3000.00	-3000.00	L2
10	PA1401303	01RB0204	AC0122	PR0113	M	N	-1500.00	-1500.00	R3
11	PA1401303	01RB0204	AC0122	PR0113	O	P	-1000.00	-1000.00	R3
12	PA1401303	01RB0204	AC0122	PR0113	O	P	1000.00	1000.00	R3
13	PA1401303	01RB0204	AC0122	PR0113	M	N	1500.00	1500.00	R3
14	PA1401303	01RB0204	AC0122	PR0113	S	T	3000.00	3000.00	L2
15	PA1401303	01RB0204	AC0122	PR0113	Q	R	4500.00	4500.00	L2
16	PA1401303	01RB0204	AC0122	PR0113	41002	50113	7500.00	7500.00	R3

Query executed successfully. | ARYAN-MODI (16.0 RTM) | aryan (62) | test | 00:00:00 | 16 rows


SQLQuery_230224.sql...l.test (aryan (53))*
SQLQuery_210224.sql...l.test (aryan (62))
100 %
Results
Messages

```sql
SELECT
  CASE WHEN RN = 0 then REPLICATE(num, 4) else REPLICATE(num, RN)
  END
FROM (SELECT NUM,RW,RW % 4 AS RN
  FROM (
    SELECT num, ROW_NUMBER() OVER (ORDER BY (SELECT 1)) AS RW FROM lg_num
  ) A
) B;
```

	(No column name)
1	1
2	22
3	333
4	4444
5	5
6	66
7	777
8	8888
9	9
10	1212
11	111111
12	10101010

Query executed successfully. | ARYAN-MODI (16.0 RTM) | aryan (53) | test | 00:00:00 | 12 rows

CHAPTER-8

WEEK-7 PROGRESS

8.1. Improve Logic Building skill

8.1.1 **Advanced Window Functions:-** Explore advanced window functions such as ROW_NUMBER(), RANK(), DENSE_RANK(), LEAD(), LAG(), NTILE(), and PERCENT_RANK().Practice using window functions in combination with partitioning, ordering, and filtering to solve complex analytical problems.

8.1.2 **Dynamic Approaches for Revenue Splitting:-** Experiment with dynamic SQL techniques to implement flexible revenue splitting logic that can adapt to changing business requirements or conditions.Explore the use of temporary tables, dynamic pivot queries, or CASE statements to dynamically calculate revenue splits based on various factors.

8.1.3 **Team Hierarchy Split Data:-** Dive deeper into hierarchical data structures and practice manipulating hierarchical data using common table expressions (CTEs) or recursive queries.Explore different methods for representing and querying hierarchical data such as adjacency lists, nested sets, and path enumeration.

8.1.4 **Transaction Management and Account SCD Data:-** Gain a deeper understanding of transaction management concepts in SQL, including ACID properties (Atomicity, Consistency, Isolation, Durability), transaction isolation levels, and locking mechanisms.Practice implementing transactional logic for handling complex operations involving multiple data updates and ensuring data integrity.Explore techniques for managing slowly changing dimensions (SCD) in a data warehouse environment, including Type 1, Type 2, and Type 3 SCD strategies.

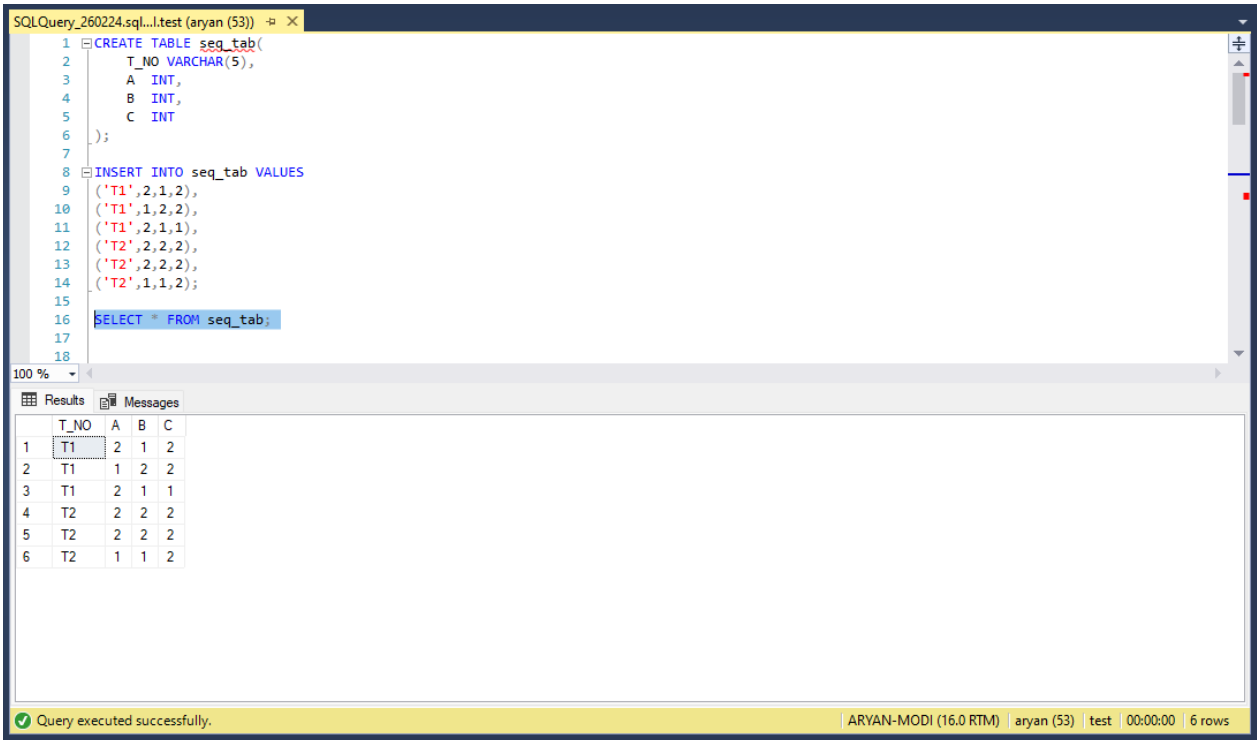
8.1.5 **Bank Data Population and CR/DR Details:-** Continue working with financial data and practice generating simulated bank transaction data for credit (CR) and debit (DR) details.Explore techniques for generating realistic transaction data such as random number generation, distribution sampling, and temporal patterns.Experiment with aggregating and analyzing bank transaction data to extract meaningful insights and perform financial analysis.

8.1.6 **Optimization and Performance Tuning:-** Focus on optimizing SQL queries and procedures for performance by analyzing query execution plans, indexing strategies, and query optimization techniques.Practice identifying and resolving performance bottlenecks in SQL code to improve overall system efficiency and scalability.

8.1.7 **Documentation and Best Practices:-** Document your SQL solutions, including logic, assumptions, and design decisions, to improve readability and maintainability.Follow best practices for SQL development, such as using meaningful naming conventions, writing clear and concise code, and properly commenting your code for future reference.

8.2. Screenshot/Code Snippet working on Task

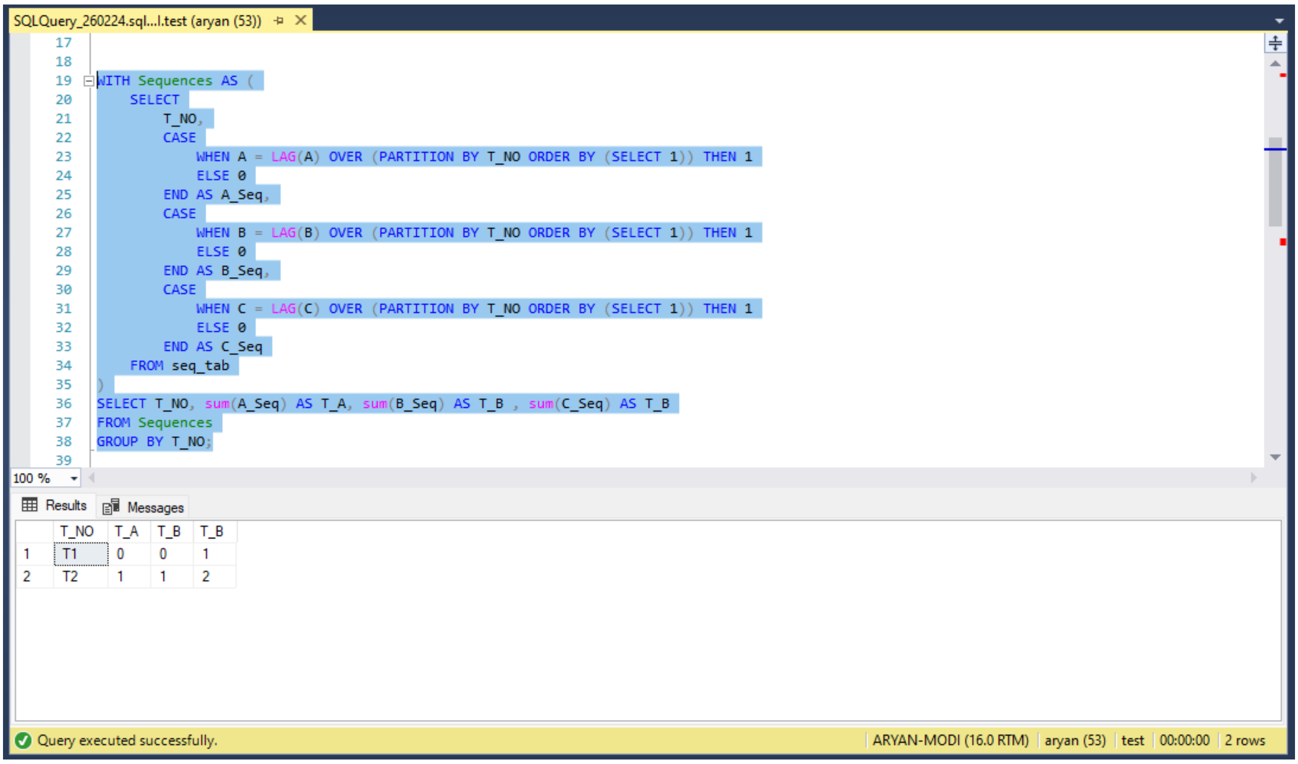
I have work on task regarding procedure , subquery , logic building task which mentation in document. I have leant about singlestore architecture , Demo: Recognizing Faces Using a SQL Database , SingleStore Architecture Overview and Product Demo , Demo: Recognizing Faces Using a SQL Database , Revenue Split Level 1 and learn about mapping and join ,cricket data and explore Singlestore and perform PIVOTE and SPLIT STRING functions , worked on SQL Logic Building which mention in document.



```
CREATE TABLE seq_tab(
    T_NO VARCHAR(5),
    A  INT,
    B  INT,
    C  INT
);

INSERT INTO seq_tab VALUES
('T1',2,1,2),
('T1',1,2,2),
('T1',2,1,1),
('T2',2,2,2),
('T2',2,2,2),
('T2',1,1,2);

SELECT * FROM seq_tab;
```

	T_NO	A	B	C
1	T1	2	1	2
2	T1	1	2	2
3	T1	2	1	1
4	T2	2	2	2
5	T2	2	2	2
6	T2	1	1	2

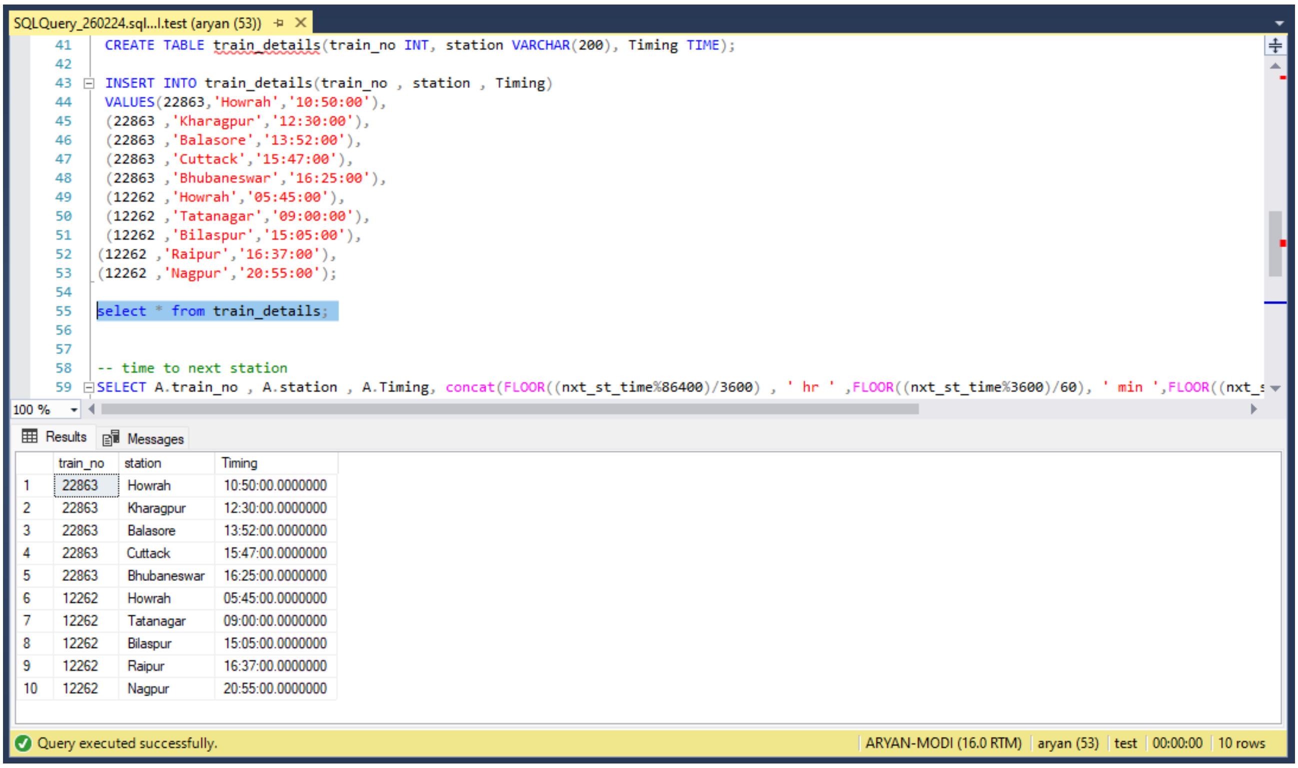
Query executed successfully. | ARYAN-MODI (16.0 RTM) | aryan (53) | test | 00:00:00 | 6 rows



```sql


WITH Sequences AS (
    SELECT
        T_NO,
        CASE
            WHEN A = LAG(A) OVER (PARTITION BY T_NO ORDER BY (SELECT 1)) THEN 1
            ELSE 0
        END AS A_Seq,
        CASE
            WHEN B = LAG(B) OVER (PARTITION BY T_NO ORDER BY (SELECT 1)) THEN 1
            ELSE 0
        END AS B_Seq,
        CASE
            WHEN C = LAG(C) OVER (PARTITION BY T_NO ORDER BY (SELECT 1)) THEN 1
            ELSE 0
        END AS C_Seq
    FROM seq_tab
)
SELECT T_NO, sum(A_Seq) AS T_A, sum(B_Seq) AS T_B , sum(C_Seq) AS T_B
FROM Sequences
GROUP BY T_NO;

```

	T_NO	T_A	T_B	T_B
1	T1	0	0	1
2	T2	1	1	2

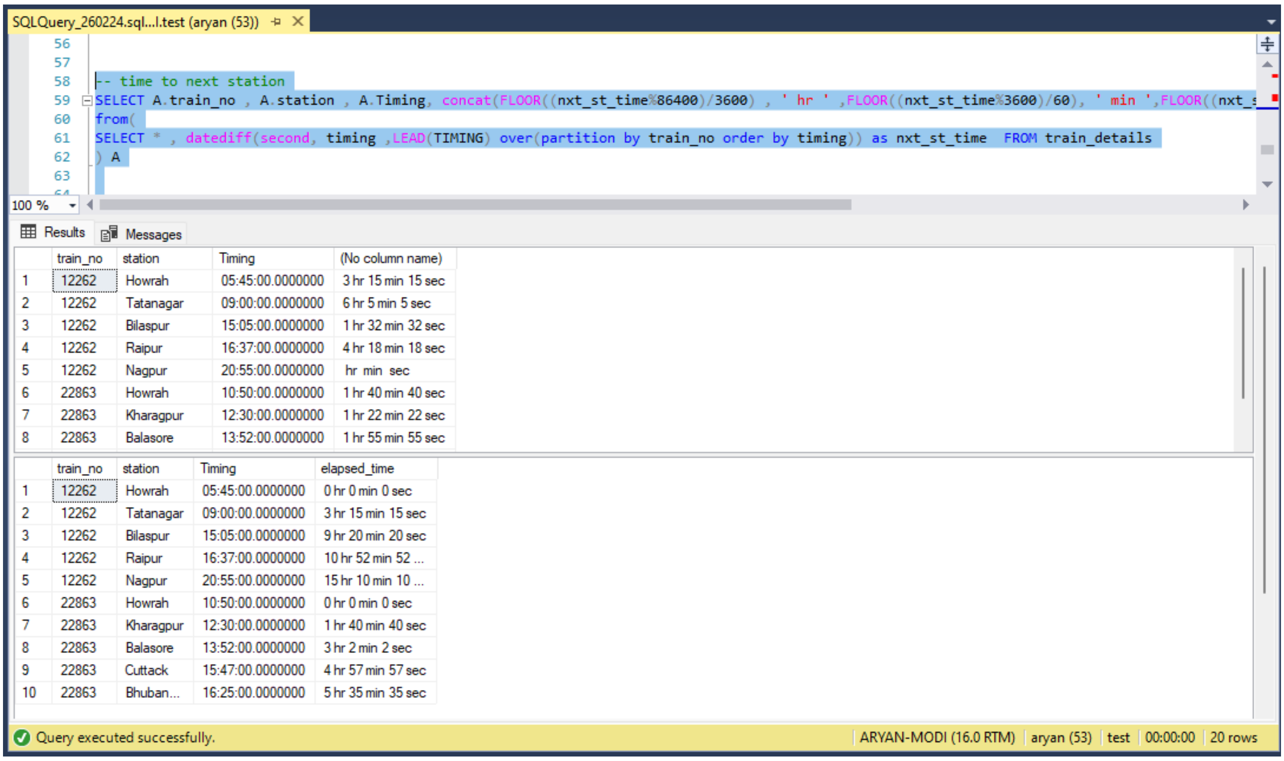
Query executed successfully. ARYAN-MODI (16.0 RTM) aryan (53) test 00:00:00 2 rows



```sql
 CREATE TABLE train_details(train_no INT, station VARCHAR(200), Timing TIME);

 INSERT INTO train_details(train_no , station , Timing)
 VALUES(22863,'Howrah','10:50:00'),
 (22863 ,'Kharagpur','12:30:00'),
 (22863 ,'Balasore','13:52:00'),
 (22863 ,'Cuttack','15:47:00'),
 (22863 ,'Bhubaneswar','16:25:00'),
 (12262 ,'Howrah','05:45:00'),
 (12262 ,'Tatanagar','09:00:00'),
 (12262 ,'Bilaspur','15:05:00'),
(12262 ,'Raipur','16:37:00'),
(12262 ,'Nagpur','20:55:00');

select * from train_details;


-- time to next station
SELECT A.train_no , A.station , A.Timing, concat(FLOOR((nxt_st_time%86400)/3600) , ' hr ' ,FLOOR((nxt_st_time%3600)/60), ' min ',FLOOR((nxt_s
```

	train_no	station	Timing
1	22863	Howrah	10:50:00.0000000
2	22863	Kharagpur	12:30:00.0000000
3	22863	Balasore	13:52:00.0000000
4	22863	Cuttack	15:47:00.0000000
5	22863	Bhubaneswar	16:25:00.0000000
6	12262	Howrah	05:45:00.0000000
7	12262	Tatanagar	09:00:00.0000000
8	12262	Bilaspur	15:05:00.0000000
9	12262	Raipur	16:37:00.0000000
10	12262	Nagpur	20:55:00.0000000

Query executed successfully. ARYAN-MODI (16.0 RTM) aryan (53) test 00:00:00 10 rows

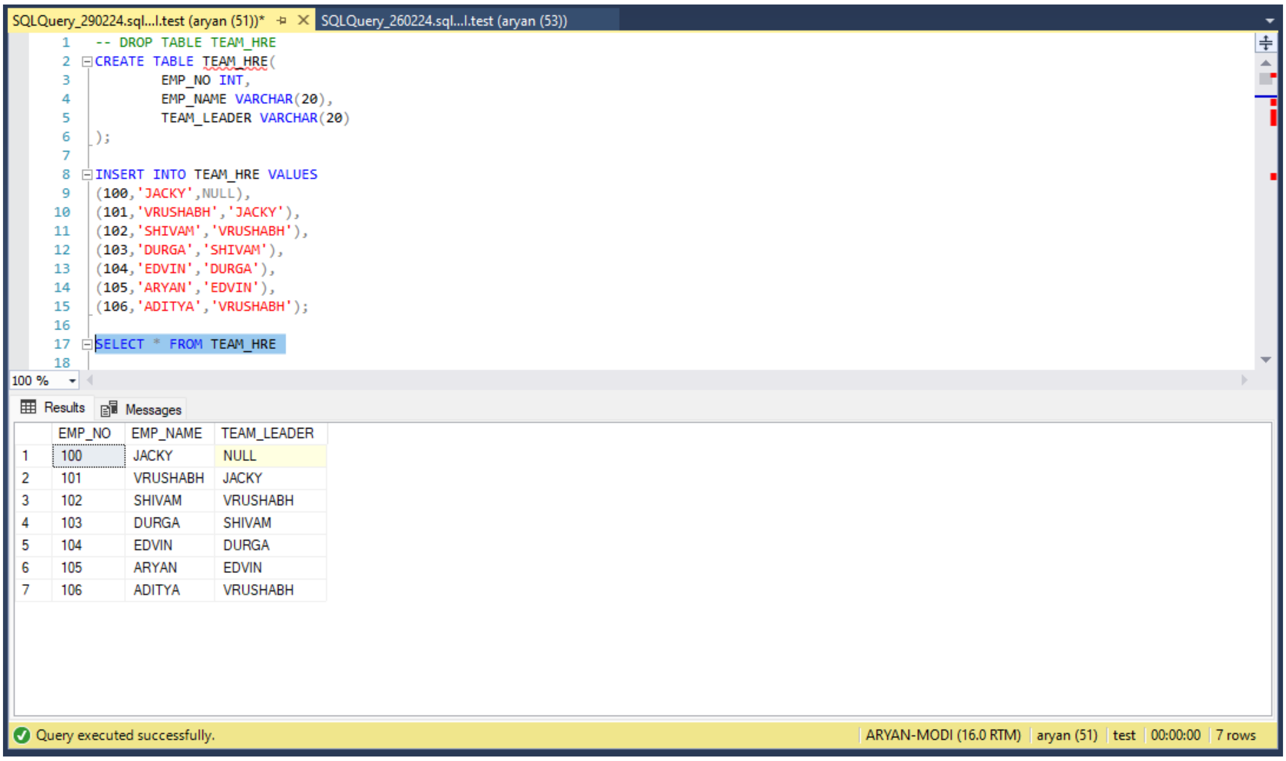


```sql
-- time to next station
SELECT A.train_no , A.station , A.Timing, concat(FLOOR((nxt_st_time%86400)/3600) , ' hr ' ,FLOOR((nxt_st_time%3600)/60), ' min ',FLOOR((nxt_s
from(
SELECT * , datediff(second, timing ,LEAD(TIMING) over(partition by train_no order by timing)) as nxt_st_time  FROM train_details
) A
```

	train_no	station	Timing	(No column name)
1	12262	Howrah	05:45:00.0000000	3 hr 15 min 15 sec
2	12262	Tatanagar	09:00:00.0000000	6 hr 5 min 5 sec
3	12262	Bilaspur	15:05:00.0000000	1 hr 32 min 32 sec
4	12262	Raipur	16:37:00.0000000	4 hr 18 min 18 sec
5	12262	Nagpur	20:55:00.0000000	hr min sec
6	22863	Howrah	10:50:00.0000000	1 hr 40 min 40 sec
7	22863	Kharagpur	12:30:00.0000000	1 hr 22 min 22 sec
8	22863	Balasore	13:52:00.0000000	1 hr 55 min 55 sec

	train_no	station	Timing	elapsed_time
1	12262	Howrah	05:45:00.0000000	0 hr 0 min 0 sec
2	12262	Tatanagar	09:00:00.0000000	3 hr 15 min 15 sec
3	12262	Bilaspur	15:05:00.0000000	9 hr 20 min 20 sec
4	12262	Raipur	16:37:00.0000000	10 hr 52 min 52 ...
5	12262	Nagpur	20:55:00.0000000	15 hr 10 min 10 ...
6	22863	Howrah	10:50:00.0000000	0 hr 0 min 0 sec
7	22863	Kharagpur	12:30:00.0000000	1 hr 40 min 40 sec
8	22863	Balasore	13:52:00.0000000	3 hr 2 min 2 sec
9	22863	Cuttack	15:47:00.0000000	4 hr 57 min 57 sec
10	22863	Bhuban...	16:25:00.0000000	5 hr 35 min 35 sec

Query executed successfully. | ARYAN-MODI (16.0 RTM) | aryan (53) | test | 00:00:00 | 20 rows

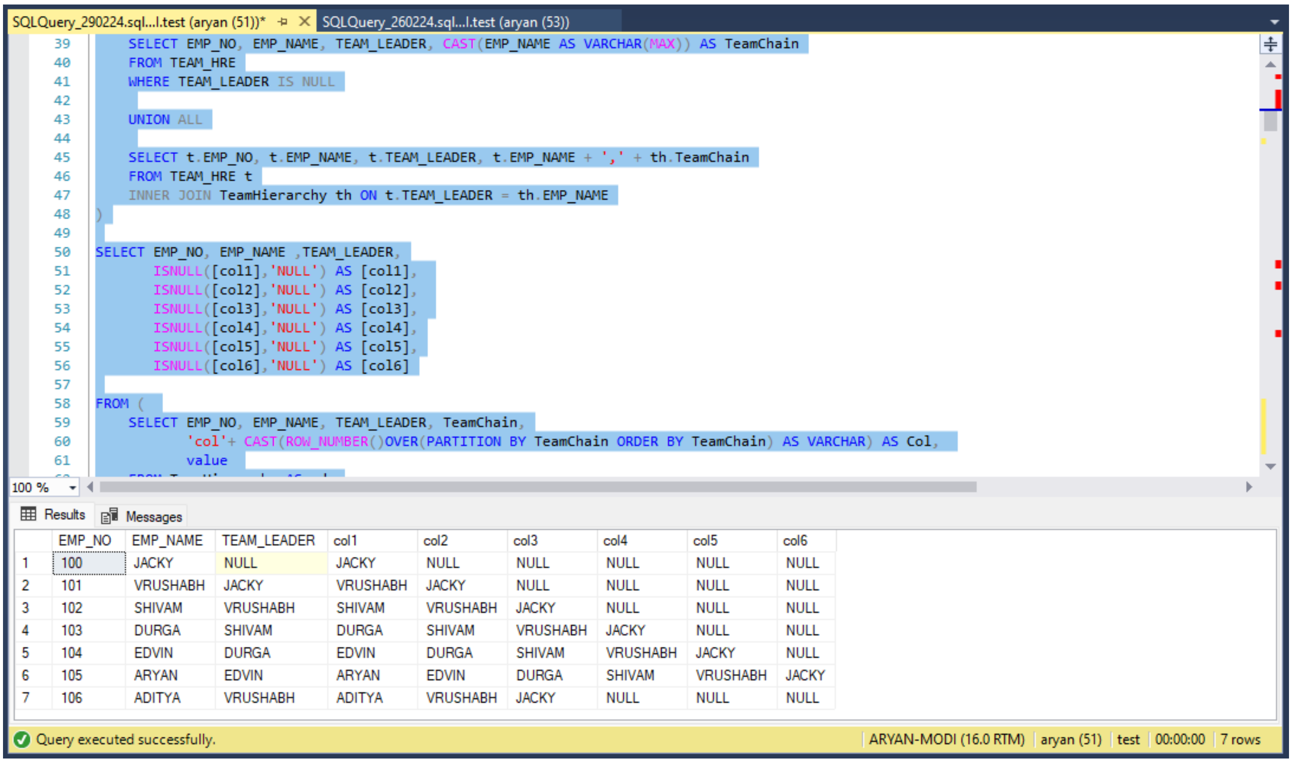


SQLQuery_290224.sql...l.test (aryan (51))*
SQLQuery_260224.sql...l.test (aryan (53))

```sql
 -- DROP TABLE TEAM_HRE
CREATE TABLE TEAM_HRE(
         EMP_NO INT,
         EMP_NAME VARCHAR(20),
         TEAM_LEADER VARCHAR(20)
);

INSERT INTO TEAM_HRE VALUES
(100,'JACKY',NULL),
(101,'VRUSHABH','JACKY'),
(102,'SHIVAM','VRUSHABH'),
(103,'DURGA','SHIVAM'),
(104,'EDVIN','DURGA'),
(105,'ARYAN','EDVIN'),
(106,'ADITYA','VRUSHABH');

SELECT * FROM TEAM_HRE
```

	EMP_NO	EMP_NAME	TEAM_LEADER
1	100	JACKY	NULL
2	101	VRUSHABH	JACKY
3	102	SHIVAM	VRUSHABH
4	103	DURGA	SHIVAM
5	104	EDVIN	DURGA
6	105	ARYAN	EDVIN
7	106	ADITYA	VRUSHABH

Query executed successfully. | ARYAN-MODI (16.0 RTM) | aryan (51) | test | 00:00:00 | 7 rows

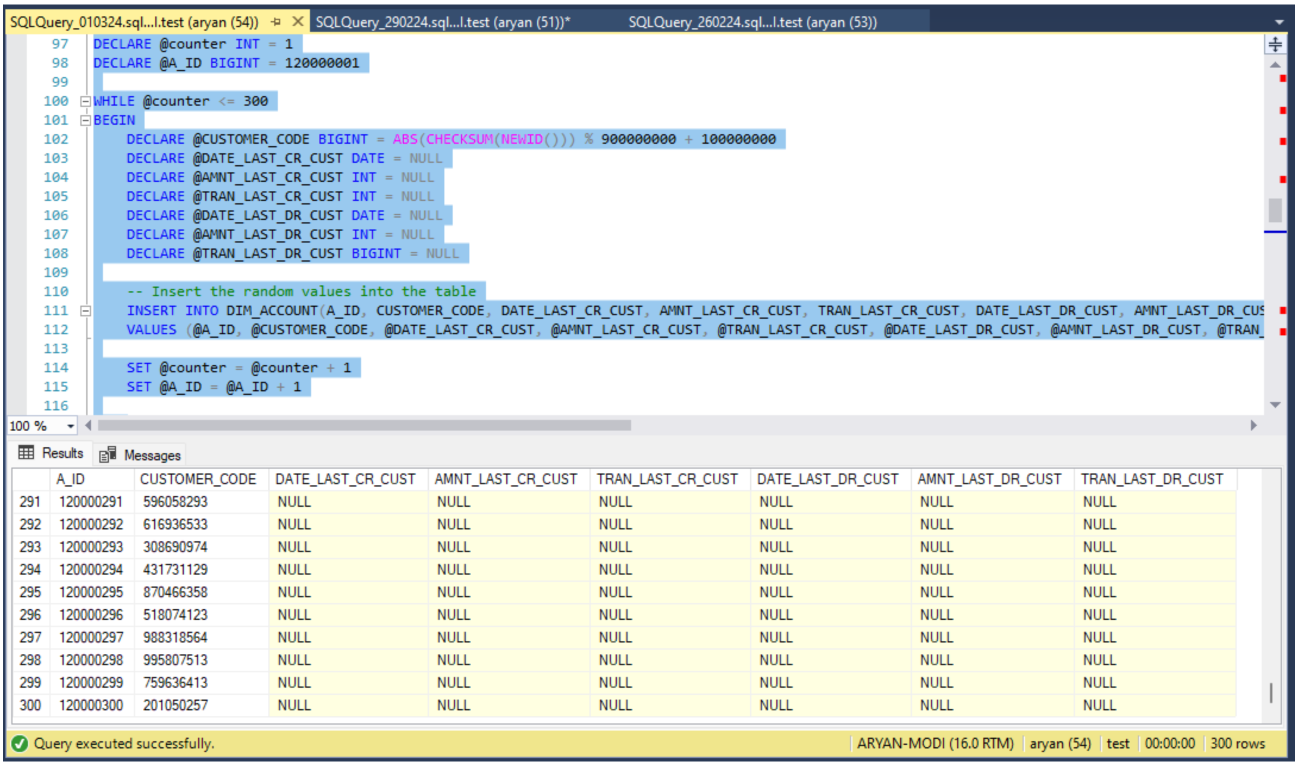


SQLQuery_290224.sql...l.test (aryan (51))*
SQLQuery_260224.sql...l.test (aryan (53))
100 %
Results
Messages

```sql
    SELECT EMP_NO, EMP_NAME, TEAM_LEADER, CAST(EMP_NAME AS VARCHAR(MAX)) AS TeamChain
    FROM TEAM_HRE
    WHERE TEAM_LEADER IS NULL

    UNION ALL

    SELECT t.EMP_NO, t.EMP_NAME, t.TEAM_LEADER, t.EMP_NAME + ',' + th.TeamChain
    FROM TEAM_HRE t
    INNER JOIN TeamHierarchy th ON t.TEAM_LEADER = th.EMP_NAME
)

SELECT EMP_NO, EMP_NAME ,TEAM_LEADER,
       ISNULL([col1],'NULL') AS [col1],
       ISNULL([col2],'NULL') AS [col2],
       ISNULL([col3],'NULL') AS [col3],
       ISNULL([col4],'NULL') AS [col4],
       ISNULL([col5],'NULL') AS [col5],
       ISNULL([col6],'NULL') AS [col6]

FROM (
    SELECT EMP_NO, EMP_NAME, TEAM_LEADER, TeamChain,
           'col'+ CAST(ROW_NUMBER()OVER(PARTITION BY TeamChain ORDER BY TeamChain) AS VARCHAR) AS Col,
           value
```

	EMP_NO	EMP_NAME	TEAM_LEADER	col1	col2	col3	col4	col5	col6
1	100	JACKY	NULL	JACKY	NULL	NULL	NULL	NULL	NULL
2	101	VRUSHABH	JACKY	VRUSHABH	JACKY	NULL	NULL	NULL	NULL
3	102	SHIVAM	VRUSHABH	SHIVAM	VRUSHABH	JACKY	NULL	NULL	NULL
4	103	DURGA	SHIVAM	DURGA	SHIVAM	VRUSHABH	JACKY	NULL	NULL
5	104	EDVIN	DURGA	EDVIN	DURGA	SHIVAM	VRUSHABH	JACKY	NULL
6	105	ARYAN	EDVIN	ARYAN	EDVIN	DURGA	SHIVAM	VRUSHABH	JACKY
7	106	ADITYA	VRUSHABH	ADITYA	VRUSHABH	JACKY	NULL	NULL	NULL

Query executed successfully. | ARYAN-MODI (16.0 RTM) | aryan (51) | test | 00:00:00 | 7 rows

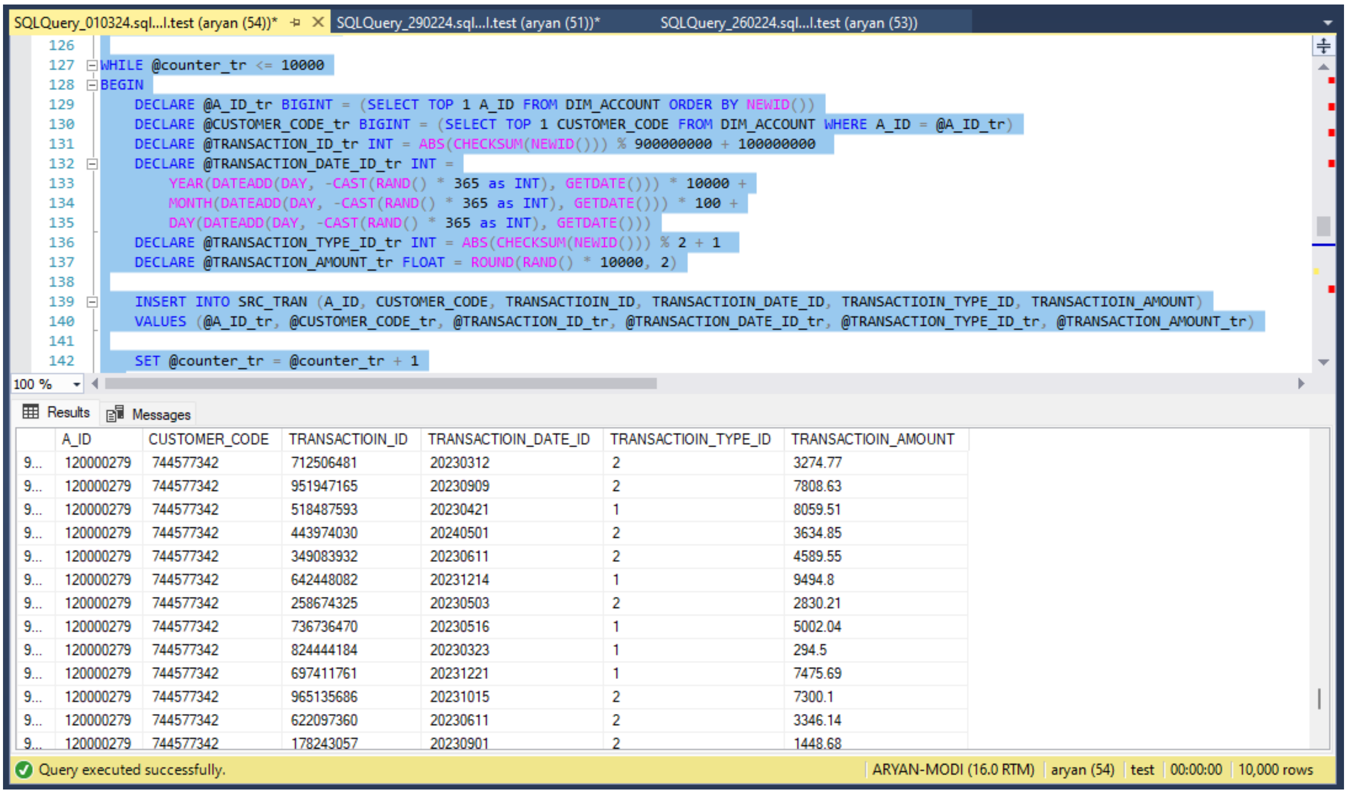


SQLQuery_010324.sql...l.test (aryan (54))
SQLQuery_290224.sql...l.test (aryan (51))*
SQLQuery_260224.sql...l.test (aryan (53))
100 %
Results
Messages

```sql
DECLARE @counter INT = 1
DECLARE @A_ID BIGINT = 120000001

WHILE @counter <= 300
BEGIN
    DECLARE @CUSTOMER_CODE BIGINT = ABS(CHECKSUM(NEWID())) % 900000000 + 100000000
    DECLARE @DATE_LAST_CR_CUST DATE = NULL
    DECLARE @AMNT_LAST_CR_CUST INT = NULL
    DECLARE @TRAN_LAST_CR_CUST INT = NULL
    DECLARE @DATE_LAST_DR_CUST DATE = NULL
    DECLARE @AMNT_LAST_DR_CUST INT = NULL
    DECLARE @TRAN_LAST_DR_CUST BIGINT = NULL

    -- Insert the random values into the table
    INSERT INTO DIM_ACCOUNT(A_ID, CUSTOMER_CODE, DATE_LAST_CR_CUST, AMNT_LAST_CR_CUST, TRAN_LAST_CR_CUST, DATE_LAST_DR_CUST, AMNT_LAST_DR_CUS
    VALUES (@A_ID, @CUSTOMER_CODE, @DATE_LAST_CR_CUST, @AMNT_LAST_CR_CUST, @TRAN_LAST_CR_CUST, @DATE_LAST_DR_CUST, @AMNT_LAST_DR_CUST, @TRAN_

    SET @counter = @counter + 1
    SET @A_ID = @A_ID + 1
```

	A_ID	CUSTOMER_CODE	DATE_LAST_CR_CUST	AMNT_LAST_CR_CUST	TRAN_LAST_CR_CUST	DATE_LAST_DR_CUST	AMNT_LAST_DR_CUST	TRAN_LAST_DR_CUST
291	120000291	596058293	NULL	NULL	NULL	NULL	NULL	NULL
292	120000292	616936533	NULL	NULL	NULL	NULL	NULL	NULL
293	120000293	308690974	NULL	NULL	NULL	NULL	NULL	NULL
294	120000294	431731129	NULL	NULL	NULL	NULL	NULL	NULL
295	120000295	870466358	NULL	NULL	NULL	NULL	NULL	NULL
296	120000296	518074123	NULL	NULL	NULL	NULL	NULL	NULL
297	120000297	988318564	NULL	NULL	NULL	NULL	NULL	NULL
298	120000298	995807513	NULL	NULL	NULL	NULL	NULL	NULL
299	120000299	759636413	NULL	NULL	NULL	NULL	NULL	NULL
300	120000300	201050257	NULL	NULL	NULL	NULL	NULL	NULL

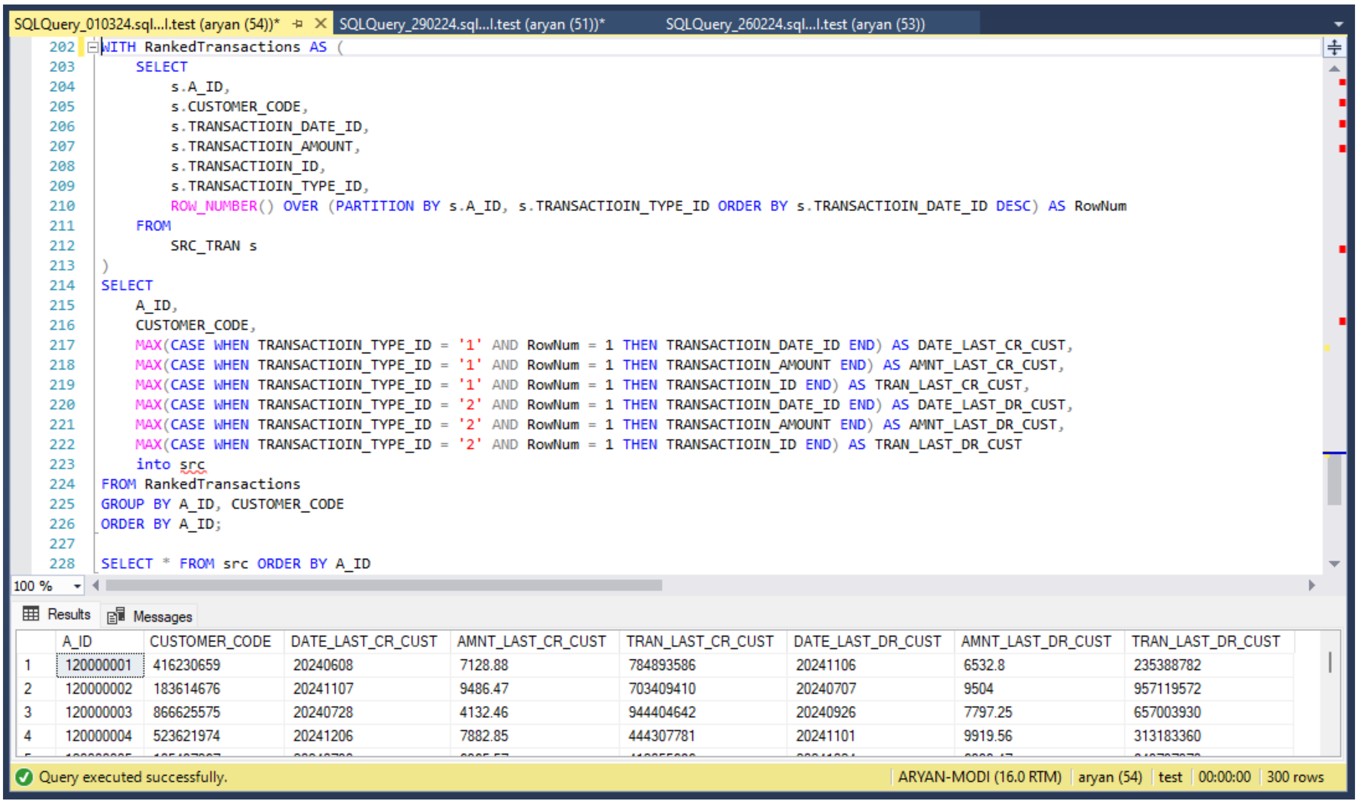
Query executed successfully. | ARYAN-MODI (16.0 RTM) | aryan (54) | test | 00:00:00 | 300 rows


SQLQuery_010324.sql...l.test (aryan (54))*
SQLQuery_290224.sql...l.test (aryan (51))*
SQLQuery_260224.sql...l.test (aryan (53))
100 %
Results
Messages

```sql

WHILE @counter_tr <= 10000
BEGIN
    DECLARE @A_ID_tr BIGINT = (SELECT TOP 1 A_ID FROM DIM_ACCOUNT ORDER BY NEWID())
    DECLARE @CUSTOMER_CODE_tr BIGINT = (SELECT TOP 1 CUSTOMER_CODE FROM DIM_ACCOUNT WHERE A_ID = @A_ID_tr)
    DECLARE @TRANSACTION_ID_tr INT = ABS(CHECKSUM(NEWID())) % 900000000 + 100000000
    DECLARE @TRANSACTION_DATE_ID_tr INT =
        YEAR(DATEADD(DAY, -CAST(RAND() * 365 as INT), GETDATE())) * 10000 +
        MONTH(DATEADD(DAY, -CAST(RAND() * 365 as INT), GETDATE())) * 100 +
        DAY(DATEADD(DAY, -CAST(RAND() * 365 as INT), GETDATE()))
    DECLARE @TRANSACTION_TYPE_ID_tr INT = ABS(CHECKSUM(NEWID())) % 2 + 1
    DECLARE @TRANSACTION_AMOUNT_tr FLOAT = ROUND(RAND() * 10000, 2)

    INSERT INTO SRC_TRAN (A_ID, CUSTOMER_CODE, TRANSACTIOIN_ID, TRANSACTIOIN_DATE_ID, TRANSACTIOIN_TYPE_ID, TRANSACTIOIN_AMOUNT)
    VALUES (@A_ID_tr, @CUSTOMER_CODE_tr, @TRANSACTION_ID_tr, @TRANSACTION_DATE_ID_tr, @TRANSACTION_TYPE_ID_tr, @TRANSACTION_AMOUNT_tr)

    SET @counter_tr = @counter_tr + 1
```

	A_ID	CUSTOMER_CODE	TRANSACTIOIN_ID	TRANSACTIOIN_DATE_ID	TRANSACTIOIN_TYPE_ID	TRANSACTIOIN_AMOUNT
9...	120000279	744577342	712506481	20230312	2	3274.77
9...	120000279	744577342	951947165	20230909	2	7808.63
9...	120000279	744577342	518487593	20230421	1	8059.51
9...	120000279	744577342	443974030	20240501	2	3634.85
9...	120000279	744577342	349083932	20230611	2	4589.55
9...	120000279	744577342	642448082	20231214	1	9494.8
9...	120000279	744577342	258674325	20230503	2	2830.21
9...	120000279	744577342	736736470	20230516	1	5002.04
9...	120000279	744577342	824444184	20230323	1	294.5
9...	120000279	744577342	697411761	20231221	1	7475.69
9...	120000279	744577342	965135686	20231015	2	7300.1
9...	120000279	744577342	622097360	20230611	2	3346.14
9...	120000279	744577342	178243057	20230901	2	1448.68

✓ Query executed successfully. | ARYAN-MODI (16.0 RTM) | aryan (54) | test | 00:00:00 | 10,000 rows


SQLQuery_010324.sql...l.test (aryan (54))*
SQLQuery_290224.sql...l.test (aryan (51))*
SQLQuery_260224.sql...l.test (aryan (53))
100 %
Results
Messages

```sql
WITH RankedTransactions AS (
    SELECT
        s.A_ID,
        s.CUSTOMER_CODE,
        s.TRANSACTIOIN_DATE_ID,
        s.TRANSACTIOIN_AMOUNT,
        s.TRANSACTIOIN_ID,
        s.TRANSACTIOIN_TYPE_ID,
        ROW_NUMBER() OVER (PARTITION BY s.A_ID, s.TRANSACTIOIN_TYPE_ID ORDER BY s.TRANSACTIOIN_DATE_ID DESC) AS RowNum
    FROM
        SRC_TRAN s
)
SELECT
    A_ID,
    CUSTOMER_CODE,
    MAX(CASE WHEN TRANSACTIOIN_TYPE_ID = '1' AND RowNum = 1 THEN TRANSACTIOIN_DATE_ID END) AS DATE_LAST_CR_CUST,
    MAX(CASE WHEN TRANSACTIOIN_TYPE_ID = '1' AND RowNum = 1 THEN TRANSACTIOIN_AMOUNT END) AS AMNT_LAST_CR_CUST,
    MAX(CASE WHEN TRANSACTIOIN_TYPE_ID = '1' AND RowNum = 1 THEN TRANSACTIOIN_ID END) AS TRAN_LAST_CR_CUST,
    MAX(CASE WHEN TRANSACTIOIN_TYPE_ID = '2' AND RowNum = 1 THEN TRANSACTIOIN_DATE_ID END) AS DATE_LAST_DR_CUST,
    MAX(CASE WHEN TRANSACTIOIN_TYPE_ID = '2' AND RowNum = 1 THEN TRANSACTIOIN_AMOUNT END) AS AMNT_LAST_DR_CUST,
    MAX(CASE WHEN TRANSACTIOIN_TYPE_ID = '2' AND RowNum = 1 THEN TRANSACTIOIN_ID END) AS TRAN_LAST_DR_CUST
    into src
FROM RankedTransactions
GROUP BY A_ID, CUSTOMER_CODE
ORDER BY A_ID;

SELECT * FROM src ORDER BY A_ID
```

	A_ID	CUSTOMER_CODE	DATE_LAST_CR_CUST	AMNT_LAST_CR_CUST	TRAN_LAST_CR_CUST	DATE_LAST_DR_CUST	AMNT_LAST_DR_CUST	TRAN_LAST_DR_CUST
1	120000001	416230659	20240608	7128.88	784893586	20241106	6532.8	235388782
2	120000002	183614676	20241107	9486.47	703409410	20240707	9504	957119572
3	120000003	866625575	20240728	4132.46	944404642	20240926	7797.25	657003930
4	120000004	523621974	20241206	7882.85	444307781	20241101	9919.56	313183360

✓ Query executed successfully. | ARYAN-MODI (16.0 RTM) | aryan (54) | test | 00:00:00 | 300 rows

CHAPTER-9

LIVE PROJECT

9.1. LOOP SQL

- In SQL Server, a loop is the technique where a set of SQL statements are executed repeatedly until a condition is met.
- SQL (Structured Query Language), loops are constructs used to iterate over a set of data or perform repetitive tasks.
- Unlike traditional programming languages like Java or Python, SQL doesn't have built-in looping constructs like "for" or "while" loops.
- However, there are methods available in SQL to achieve similar iterative functionality, such as cursors and recursive queries.

9.2.JavaScript

- JavaScript is a versatile programming language commonly used for building interactive web applications.
- Event Handling:- JavaScript is frequently used to handle user interactions such as clicks, key presses, form submissions, etc. You can define event listeners that trigger functions when specific events occur on HTML elements.
- DOM Manipulation:- JavaScript allows you to dynamically update the content and styles of HTML elements on a webpage. You can manipulate the Document Object Model (DOM) using functions to add, remove, or modify elements.
- AJAX Requests:- JavaScript enables asynchronous communication with a server using AJAX (Asynchronous JavaScript and XML) requests. You can define functions to send HTTP requests to a server and handle the responses without reloading the entire page.
- Error Handling:- JavaScript allows you to handle runtime errors and exceptions gracefully using try-catch blocks. You can define functions to catch and handle errors to prevent them from crashing the entire application.

9.3.node JS

- Node.js is widely used for building web servers due to its non-blocking, event-driven architecture. Frameworks like Express.js make it easy to create robust and scalable web applications.
- API Development:- Node js is ideal for building RESTful APIs to provide data and services to client-side applications. You can define routes, handle requests, and interact with databases using libraries like Mongoose or SQL databases.
- Real-time Applications:- Node js, along with frameworks like Socket.io, is used for building real-time web applications that require bidirectional communication between clients and servers.
- Microservices:- Node js is well-suited for building microservices architecture, where small, independent services communicate with each other to perform specific tasks. This allows for better scalability, maintainability, and deployment flexibility.

9.4.FIREVASE

- Firebase is a comprehensive platform provided by Google for building web and mobile applications. It offers a variety of services that can be used individually or together to streamline the development process.
- Realtime Database:- Firebase Realtime Database is a NoSQL cloud database that stores and syncs data between your users in real-time. It's ideal for building collaborative and real-time applications such as chat apps, collaborative editing tools, etc.
- Hosting:- Firebase Hosting provides fast and secure hosting for your web app, including SSL encryption, CDN integration, and automatic deployment from a Git repository. It's ideal for hosting static websites, single-page apps, and progressive web apps.

- Cloud Functions:- Firebase Cloud Functions allows you to run server-side code in response to events triggered by Firebase features and HTTPS requests. You can use it to extend your app's functionality, integrate with third-party services, and automate tasks.

9.5. Ngrok

Ngrok is a service that creates secure tunnels to your localhost, exposing it to the internet. It's often used by developers during the development and testing phase of web applications or APIs. This allows them to share their work with others or test functionality across different devices without deploying it to a public server.

- Tunneling: Ngrok establishes secure tunnels from a public endpoint (e.g., ngrok.io) to a port on your local machine. This means you can expose a local server running on your machine to the internet without having to deploy it to a public server.
- Security: Ngrok tunnels are secure, utilizing TLS encryption for data transfer. This means that data transmitted between your local machine and the ngrok server is encrypted.
- Subdomains: Ngrok allows you to reserve a subdomain under ngrok.io for your account, making it easier to remember the URL for your tunnels.
- Inspecting traffic: Ngrok provides a web interface where you can inspect the HTTP traffic passing through the tunnel in real-time. This can be helpful for debugging and monitoring purposes.
- Authentication and custom domains: Ngrok offers features like HTTP basic authentication for added security, and you can also use custom domains if you’re on one of their paid plans.
- Pricing: Ngrok offers both free and paid plans. The free plan is limited in terms of features and concurrent connections, while the paid plans offer more features and higher limits.

9.6. Cognos Custom Cntrol

In IBM Cognos, Custom Controls are a way to extend the functionality of the report authoring environment by allowing developers to create custom user interface components. These components can be integrated into Cognos reports to enhance interactivity, visualization, or data manipulation capabilities beyond what's provided by default.

- Purpose: Custom Controls allow developers to create custom user interface elements or components that can be embedded within Cognos reports. These components can range from simple input fields to complex interactive visualizations.
- Development: Custom Controls are typically developed using web technologies such as HTML, CSS, and JavaScript. Developers can leverage libraries like jQuery or D3.js to create rich and interactive components.
- Integration: Once developed, Custom Controls can be integrated into Cognos reports using the Report Studio authoring tool. They are added to reports just like any other report item, such as tables or charts.
- Functionality: Custom Controls can provide a wide range of functionality, including custom input forms, interactive charts and graphs, data filters, and more. They can interact with Cognos data sources and respond to user actions within the report.
- Flexibility: Custom Controls offer flexibility in terms of design and functionality, allowing developers to tailor them to specific reporting requirements or user preferences.
- Deployment: Custom Controls need to be deployed to the Cognos server environment so that they can be used within reports. This typically involves uploading the control files to the appropriate directories on the Cognos server.
- Security: Like any custom code integrated into a software environment, Custom Controls should be developed and deployed with security considerations in mind to prevent vulnerabilities or exploits.
- Architecture: Custom Controls are typically built using a combination of HTML, CSS, and JavaScript.

They leverage the Cognos JavaScript API (Cognos Mashup Services) to interact with the Cognos report environment. This API provides methods for accessing report data, parameters, and other properties, as well as for responding to user interactions.

- Types of Custom Controls:
  - Input Controls: These controls allow users to input data or make selections that can affect the behavior of the report. Examples include custom dropdown lists, date pickers, sliders, etc.
  - Visualization Controls: These controls are used to enhance the visual representation of data within the report. Examples include custom charts, graphs, maps, etc.
  - Interactive Controls: These controls enable interactive features within the report, such as collapsible sections, expandable tables, tooltips, etc.
- Integration with Cognos Reports:
  - Custom Controls are integrated into Cognos reports through the use of Report Studio, which is the report authoring tool in IBM Cognos.
  - Developers can add Custom Controls to reports by dragging and dropping them onto the report canvas just like any other report item.
  - Once added, Custom Controls can be configured to interact with report data, parameters, and other elements.
- Data Interaction:
  - Custom Controls can interact with Cognos report data by fetching data from data sources, filtering data based on user input, and updating report content dynamically.
  - They can also respond to events such as data updates, user interactions, or changes in report state.
- Extensibility:
  - Custom Controls can be extended and customized further by integrating third-party libraries or frameworks.
  - Developers can create reusable components or templates to streamline development and maintain consistency across reports.
- Documentation and Support:
  - IBM provides documentation, tutorials, and forums to help developers learn how to create and integrate Custom Controls effectively.
  - Developers can also find community-contributed resources, such as code samples and plugins, to accelerate development.

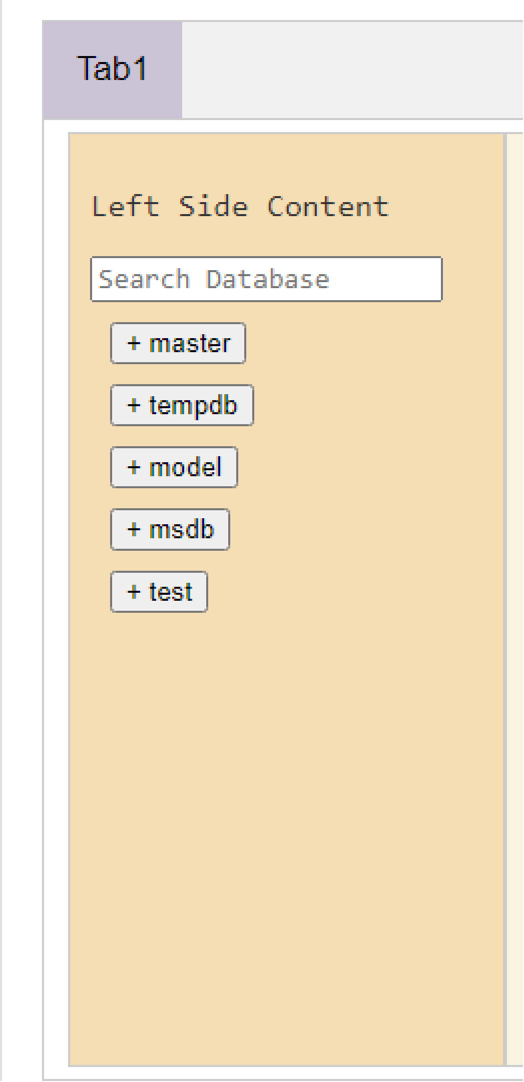
9.7. Cognos

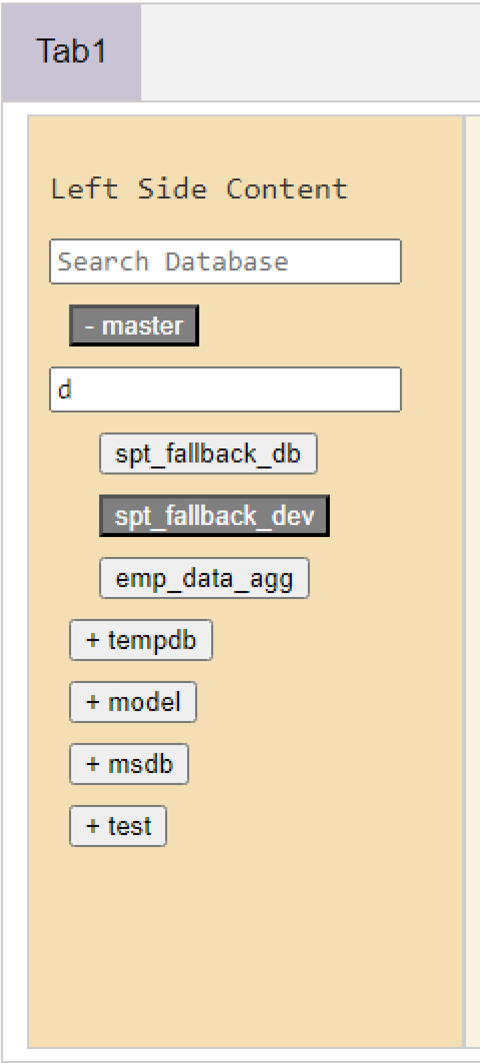
IBM Cognos is a suite of business intelligence (BI) and performance management software products. It's designed to help businesses extract insights from their data and make informed decisions to improve performance and achieve their goals.

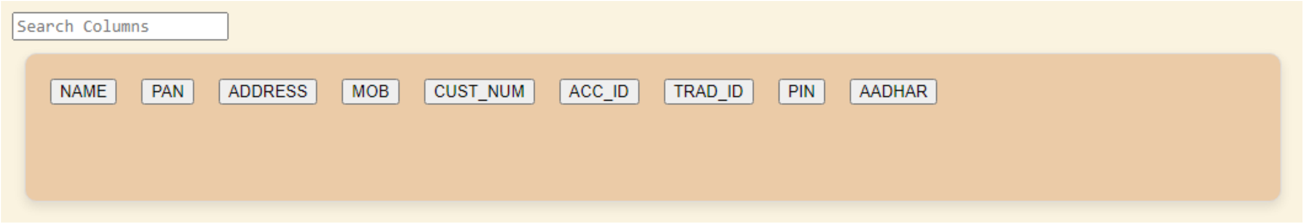
- Reporting: Cognos provides robust reporting capabilities, allowing users to create and distribute a wide range of reports, including operational reports, financial statements, dashboards, and scorecards. Reports can be highly customizable, with options for formatting, filtering, and interactive features.
- Analysis: With Cognos, users can perform multidimensional analysis of data to uncover trends, patterns, and correlations. It supports ad-hoc querying, slicing and dicing, and drill-down capabilities to explore data from different angles.
- Dashboarding: Cognos offers dashboarding functionality to visualize key performance indicators (KPIs) and metrics in a centralized and interactive manner. Users can create personalized dashboards with charts, graphs, and other visualizations to monitor performance and track progress towards goals.
- Data Integration: Cognos can integrate with a variety of data sources, including relational databases, data warehouses, OLAP cubes, spreadsheets, and cloud-based data sources. It supports both structured and unstructured data, allowing users to access and analyze data from multiple sources in a unified environment.

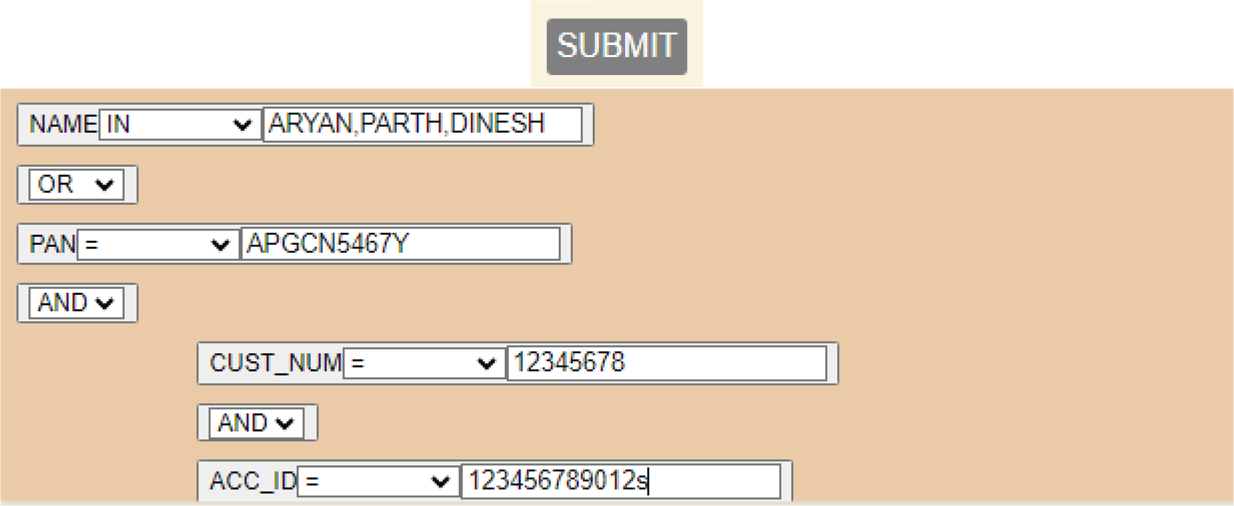
- Planning and Budgeting: Cognos includes capabilities for budgeting, planning, and forecasting, enabling organizations to create, manage, and analyze budgets and financial plans. It supports collaborative planning processes, workflow automation, and what-if scenario analysis.
- Predictive Analytics: In addition to traditional BI and reporting features, Cognos offers advanced analytics capabilities, including predictive modeling, data mining, and statistical analysis. This allows users to uncover insights and make predictions based on historical data and predictive algorithms.
- Mobile Access: Cognos provides mobile access to reports, dashboards, and analytics, allowing users to access insights and make decisions on the go. It supports responsive design and native mobile apps for iOS and Android devices.
- Security and Governance: Cognos includes features for data security, access control, and governance to ensure that sensitive information is protected and regulatory compliance requirements are met. It supports role-based access control, encryption, auditing, and data masking.

9.8. Screenshot/Code Snippet working of project









NAME IN ARYAN,PARTH,DINESH

OR

PAN = APGCN5467Y

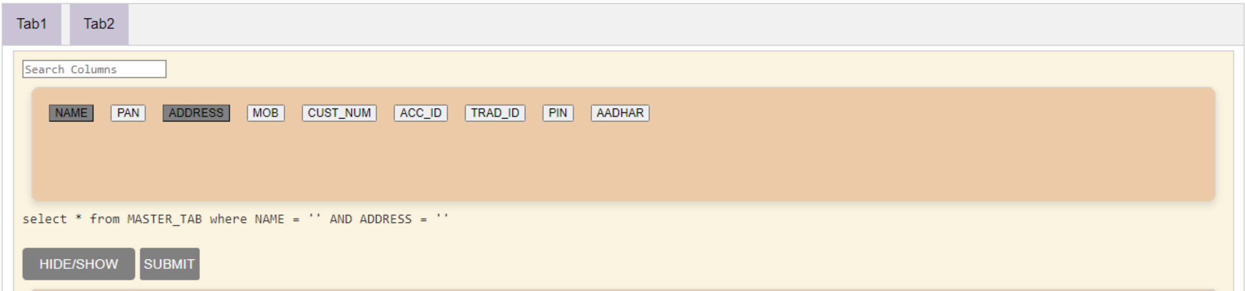
AND

CUST_NUM = 12345678

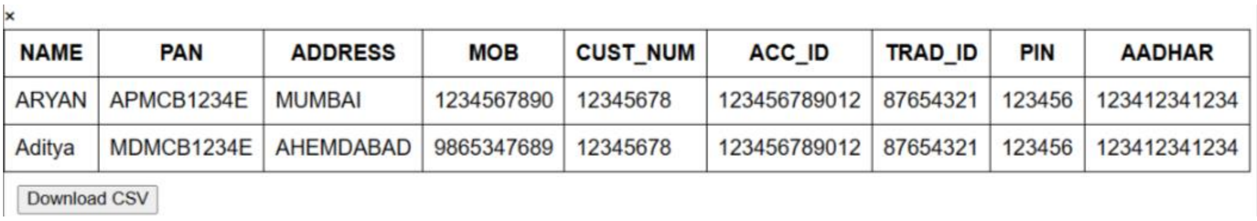
AND

ACC_ID = 123456789012s

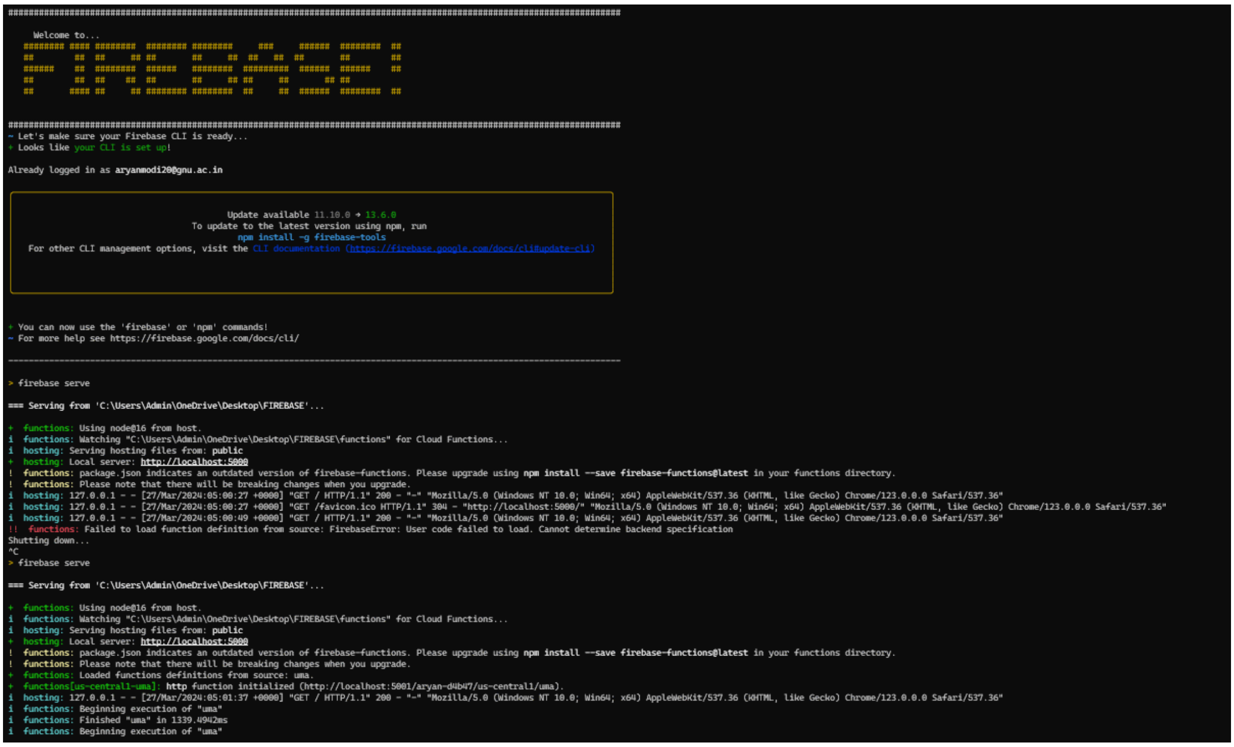
```
select * from MASTER_TAB where NAME IN ( 'ARYAN', 'PARTH', 'DINESH' ) OR PAN = 'APGCN5467Y' AND ( CUST_NUM = '12345678' AND ACC_ID
= '123456789012s' )
```



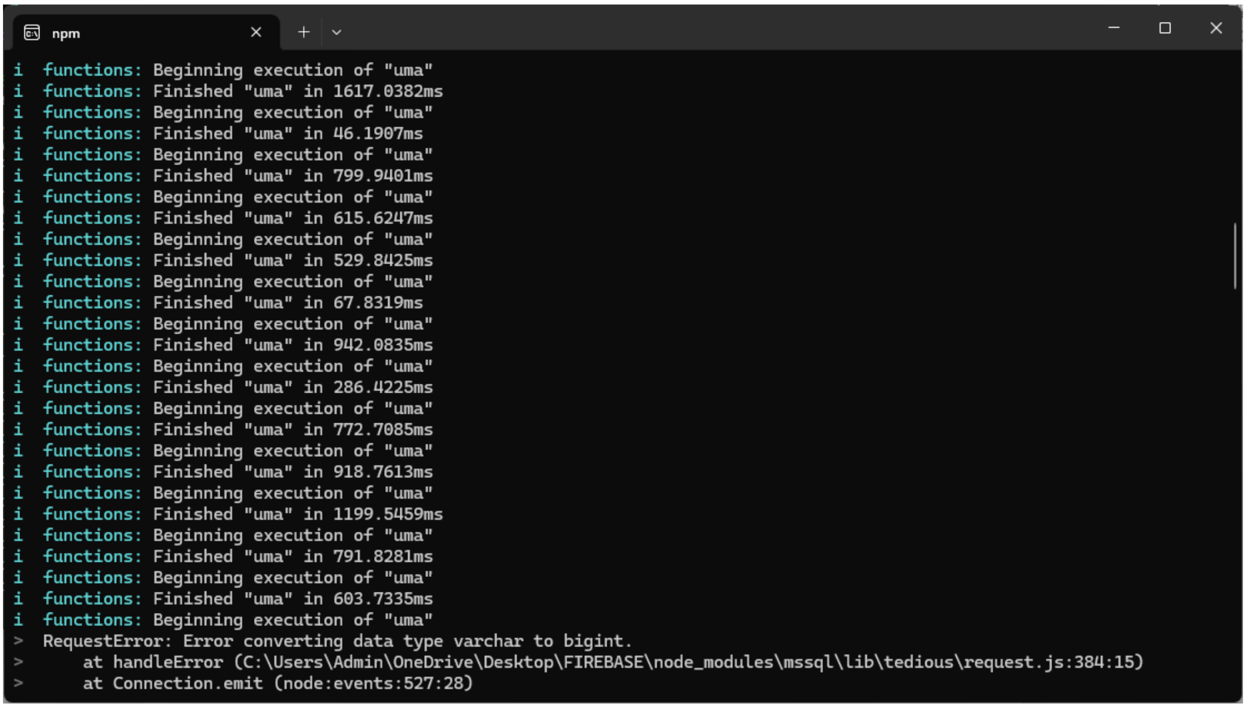
```
select * from MASTER_TAB where NAME = '' AND ADDRESS = ''
```



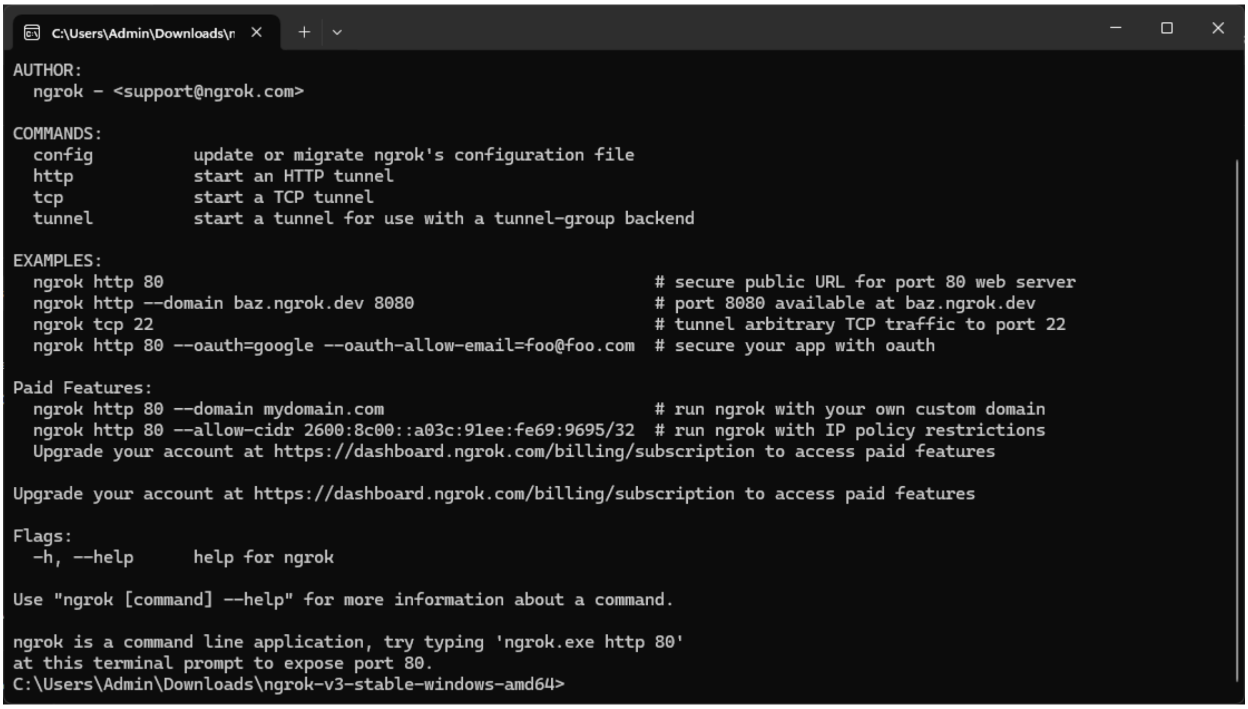
NAME	PAN	ADDRESS	MOB	CUST_NUM	ACC_ID	TRAD_ID	PIN	AADHAR
ARYAN	APMCB1234E	MUMBAI	1234567890	12345678	123456789012	87654321	123456	123412341234
Aditya	MDMCB1234E	AHEMDABAD	9865347689	12345678	123456789012	87654321	123456	123412341234



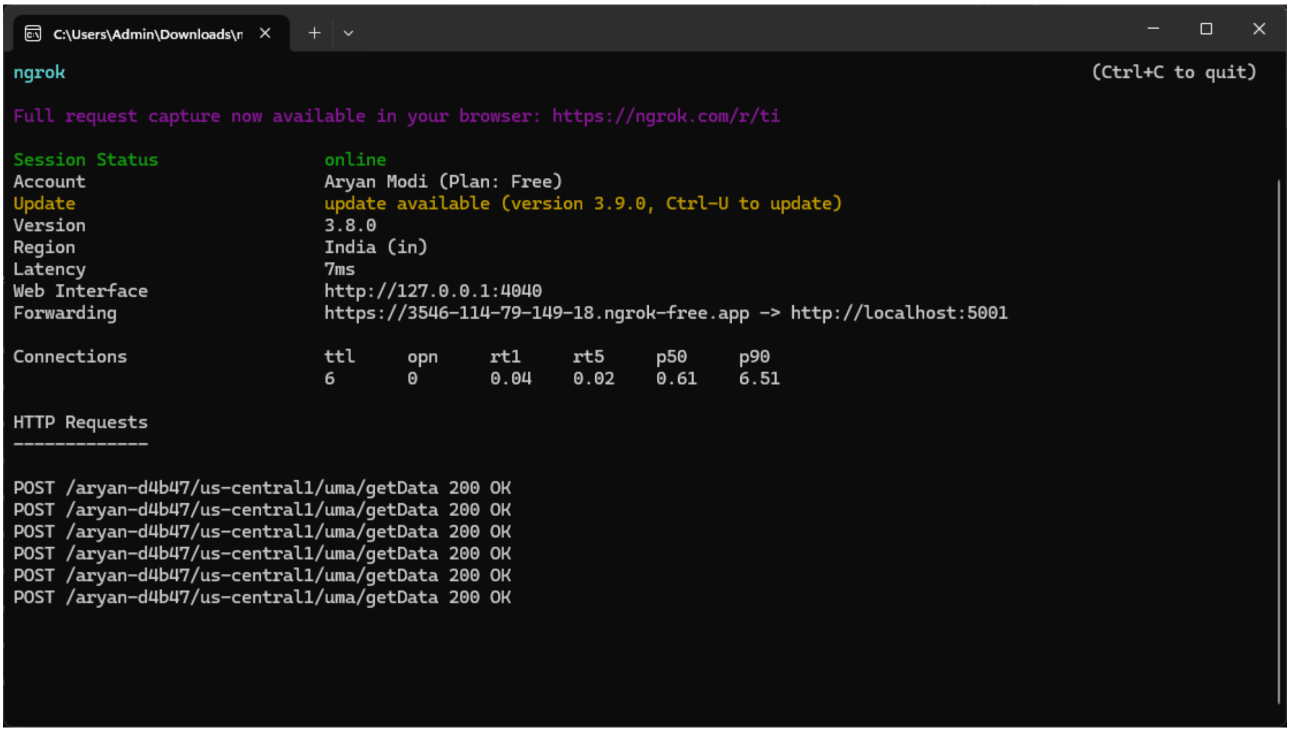
```
########################################################################################################

     Welcome to...
   ######## #### ########  ######## ########     ###     ######  ######## ##
   ##        ##  ##     ## ##       ##     ##  ##   ##  ##       ##       ##
   ######    ##  ########  ######   ########  #########  ######  ######   ##
   ##        ##  ##    ##  ##       ##     ## ##     ##       ## ##
   ##       #### ##     ## ######## ########  ##     ##  ######  ######## ##


########################################################################################################
~ Let's make sure your Firebase CLI is ready...
+ Looks like your CLI is set up!

Already logged in as aryanmodi20@gnu.ac.in

 ┌──────────────────────────────────────────────────────────────────────────────────────────────────┐
 │                                                                                                  │
 │                              Update available 11.10.0 → 13.6.0                                   │
 │                       To update to the latest version using npm, run                             │
 │                                npm install -g firebase-tools                                     │
 │   For other CLI management options, visit the CLI documentation (https://firebase.google.com/docs/cli#update-cli) │
 │                                                                                                  │
 │                                                                                                  │
 └──────────────────────────────────────────────────────────────────────────────────────────────────┘


+ You can now use the 'firebase' or 'npm' commands!
~ For more help see https://firebase.google.com/docs/cli/

--------------------------------------------------------------------------------------------------------

> firebase serve

=== Serving from 'C:\Users\Admin\OneDrive\Desktop\FIREBASE'...

+  functions: Using node@16 from host.
i  functions: Watching "C:\Users\Admin\OneDrive\Desktop\FIREBASE\functions" for Cloud Functions...
i  hosting: Serving hosting files from: public
+  hosting: Local server: http://localhost:5000
!  functions: package.json indicates an outdated version of firebase-functions. Please upgrade using npm install --save firebase-functions@latest in your functions directory.
!  functions: Please note that there will be breaking changes when you upgrade.
i  hosting: 127.0.0.1 - - [27/Mar/2024:05:00:27 +0000] "GET / HTTP/1.1" 200 - "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/123.0.0.0 Safari/537.36"
i  hosting: 127.0.0.1 - - [27/Mar/2024:05:00:27 +0000] "GET /favicon.ico HTTP/1.1" 304 - "http://localhost:5000/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/123.0.0.0 Safari/537.36"
i  hosting: 127.0.0.1 - - [27/Mar/2024:05:00:49 +0000] "GET / HTTP/1.1" 200 - "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/123.0.0.0 Safari/537.36"
!!  functions: Failed to load function definition from source: FirebaseError: User code failed to load. Cannot determine backend specification
Shutting down...
^C
> firebase serve

=== Serving from 'C:\Users\Admin\OneDrive\Desktop\FIREBASE'...

+  functions: Using node@16 from host.
i  functions: Watching "C:\Users\Admin\OneDrive\Desktop\FIREBASE\functions" for Cloud Functions...
i  hosting: Serving hosting files from: public
+  hosting: Local server: http://localhost:5000
!  functions: package.json indicates an outdated version of firebase-functions. Please upgrade using npm install --save firebase-functions@latest in your functions directory.
!  functions: Please note that there will be breaking changes when you upgrade.
+  functions: Loaded functions definitions from source: uma.
+  functions[us-central1-uma]: http function initialized (http://localhost:5001/aryan-d4b47/us-central1/uma).
i  hosting: 127.0.0.1 - - [27/Mar/2024:05:01:37 +0000] "GET / HTTP/1.1" 200 - "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/123.0.0.0 Safari/537.36"
i  functions: Beginning execution of "uma"
i  functions: Finished "uma" in 1339.4942ms
i  functions: Beginning execution of "uma"
```



```
i  functions: Beginning execution of "uma"
i  functions: Finished "uma" in 1617.0382ms
i  functions: Beginning execution of "uma"
i  functions: Finished "uma" in 46.1907ms
i  functions: Beginning execution of "uma"
i  functions: Finished "uma" in 799.9401ms
i  functions: Beginning execution of "uma"
i  functions: Finished "uma" in 615.6247ms
i  functions: Beginning execution of "uma"
i  functions: Finished "uma" in 529.8425ms
i  functions: Beginning execution of "uma"
i  functions: Finished "uma" in 67.8319ms
i  functions: Beginning execution of "uma"
i  functions: Finished "uma" in 942.0835ms
i  functions: Beginning execution of "uma"
i  functions: Finished "uma" in 286.4225ms
i  functions: Beginning execution of "uma"
i  functions: Finished "uma" in 772.7085ms
i  functions: Beginning execution of "uma"
i  functions: Finished "uma" in 918.7613ms
i  functions: Beginning execution of "uma"
i  functions: Finished "uma" in 1199.5459ms
i  functions: Beginning execution of "uma"
i  functions: Finished "uma" in 791.8281ms
i  functions: Beginning execution of "uma"
i  functions: Finished "uma" in 603.7335ms
i  functions: Beginning execution of "uma"
>  RequestError: Error converting data type varchar to bigint.
>      at handleError (C:\Users\Admin\OneDrive\Desktop\FIREBASE\node_modules\mssql\lib\tedious\request.js:384:15)
>      at Connection.emit (node:events:527:28)
```



```
AUTHOR:
  ngrok - <support@ngrok.com>

COMMANDS:
  config          update or migrate ngrok's configuration file
  http            start an HTTP tunnel
  tcp             start a TCP tunnel
  tunnel          start a tunnel for use with a tunnel-group backend

EXAMPLES:
  ngrok http 80                                                 # secure public URL for port 80 web server
  ngrok http --domain baz.ngrok.dev 8080                        # port 8080 available at baz.ngrok.dev
  ngrok tcp 22                                                  # tunnel arbitrary TCP traffic to port 22
  ngrok http 80 --oauth=google --oauth-allow-email=foo@foo.com  # secure your app with oauth

Paid Features:
  ngrok http 80 --domain mydomain.com                           # run ngrok with your own custom domain
  ngrok http 80 --allow-cidr 2600:8c00::a03c:91ee:fe69:9695/32  # run ngrok with IP policy restrictions
  Upgrade your account at https://dashboard.ngrok.com/billing/subscription to access paid features

Upgrade your account at https://dashboard.ngrok.com/billing/subscription to access paid features

Flags:
  -h, --help      help for ngrok

Use "ngrok [command] --help" for more information about a command.

ngrok is a command line application, try typing 'ngrok.exe http 80'
at this terminal prompt to expose port 80.
C:\Users\Admin\Downloads\ngrok-v3-stable-windows-amd64>
```



```
ngrok                                                                                        (Ctrl+C to quit)

Full request capture now available in your browser: https://ngrok.com/r/ti

Session Status                online
Account                       Aryan Modi (Plan: Free)
Update                        update available (version 3.9.0, Ctrl-U to update)
Version                       3.8.0
Region                        India (in)
Latency                       7ms
Web Interface                 http://127.0.0.1:4040
Forwarding                    https://3546-114-79-149-18.ngrok-free.app -> http://localhost:5001

Connections                   ttl     opn     rt1     rt5     p50     p90
                              6       0       0.04    0.02    0.61    6.51

HTTP Requests
-------------

POST /aryan-d4b47/us-central1/uma/getData 200 OK
POST /aryan-d4b47/us-central1/uma/getData 200 OK
POST /aryan-d4b47/us-central1/uma/getData 200 OK
POST /aryan-d4b47/us-central1/uma/getData 200 OK
POST /aryan-d4b47/us-central1/uma/getData 200 OK
POST /aryan-d4b47/us-central1/uma/getData 200 OK
```

REFERENCES

10.1 https://stackoverflow.com/

10.2 https://learn.microsoft.com/en-in/

10.3 https://www.sqlservertutorial.net/

10.4 https://www.tutorialspoint.com/index.htm

10.5 https://hub.docker.com/

10.6 https://portal.singlestore.com/organizations/5a2441ce-87e9-44e5-a6c1-83b759f33913/homepage

10.7 https://medium.com/

10.8 https://nodejs.org/en

10.9 https://console.firebase.google.com/u/0/