Internship Project
Presentation
on
VERACITIZ SOLUTION
By
Group ID: G06

ARYAN MODI
[20162121011]

Under the guidance of

PROF. DHARMESH DARJI
HEAD , CSE DEPARTMENT

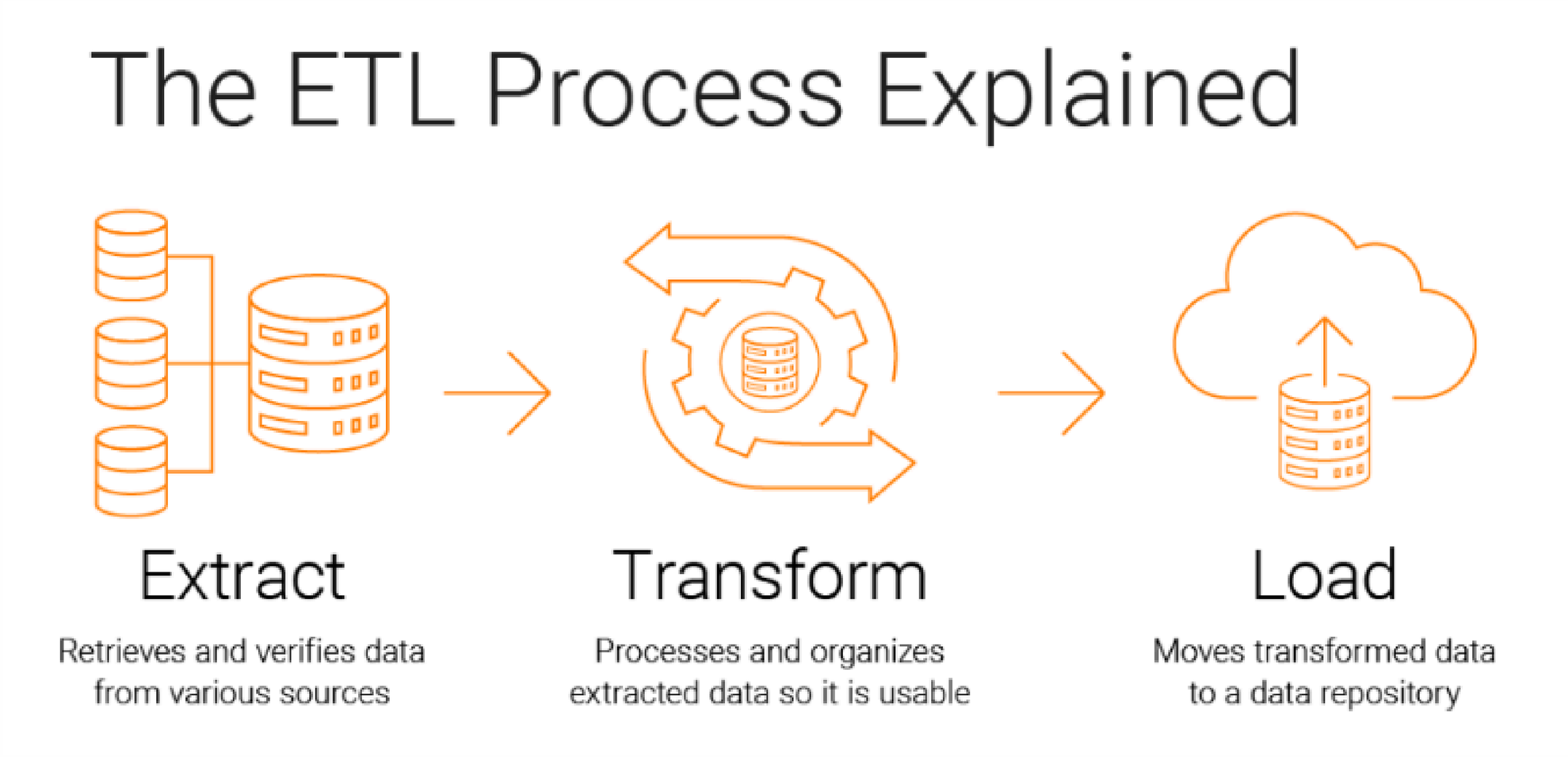
JACKY PATEL
REPORT MANAGER

Institute of Computer Technology, Ganpat University
Date: 11 May 2024

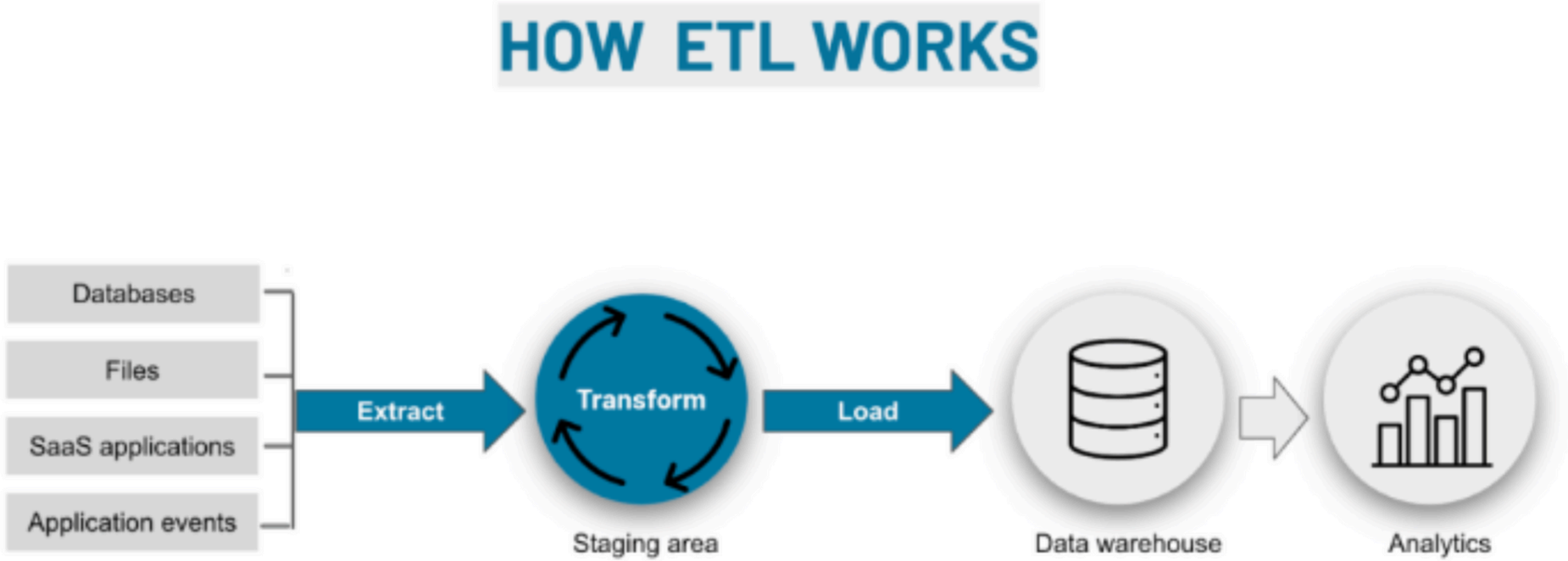
Table of Content

Week 1

15-01 to 21-01



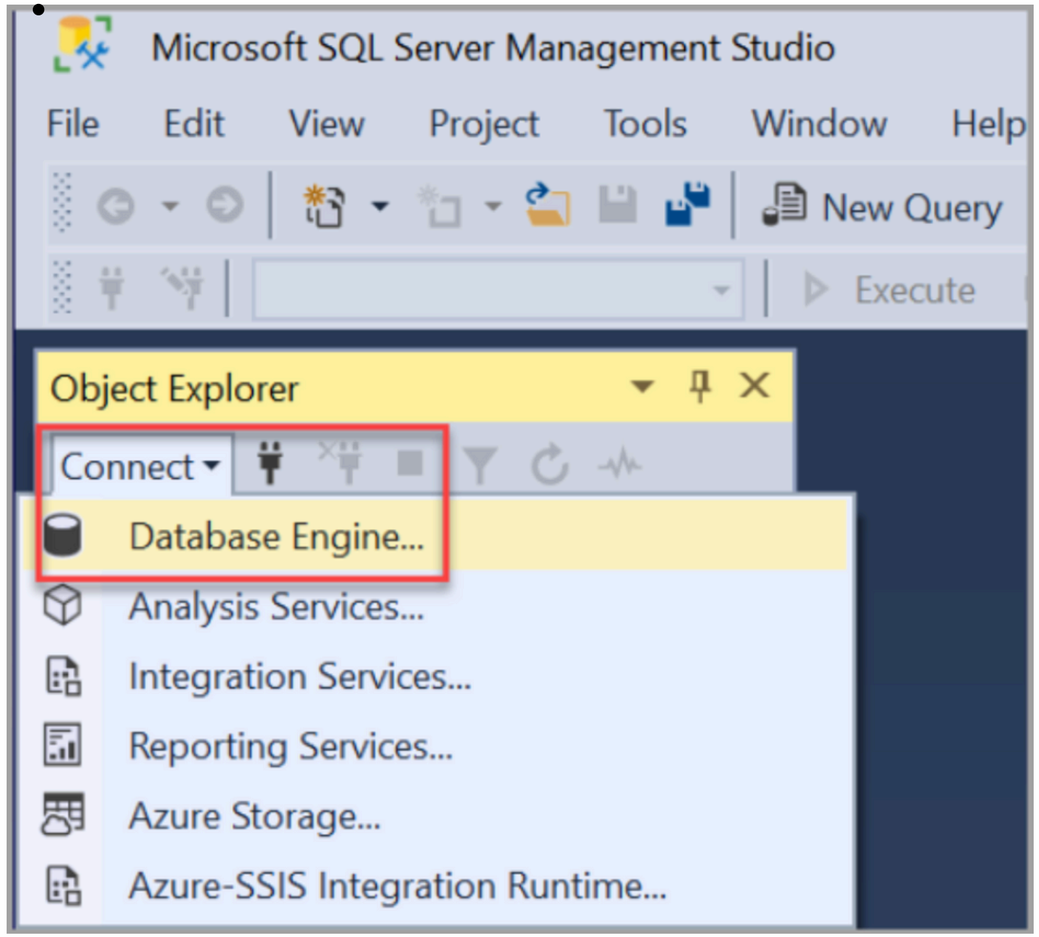
- •I have learn following concepts : brief introduction of ETL ,DATA ,Data Warehouse , SQL Concepts , ALTER , MODIFY , CHANGE , CONSTRAINT , RENAME , MIN-MAX , WHERE , AGGRIGATE functions and STRING functions , ORDER BY ,GROUP BY , JOINJS , DATE other task are mantion in the documents.

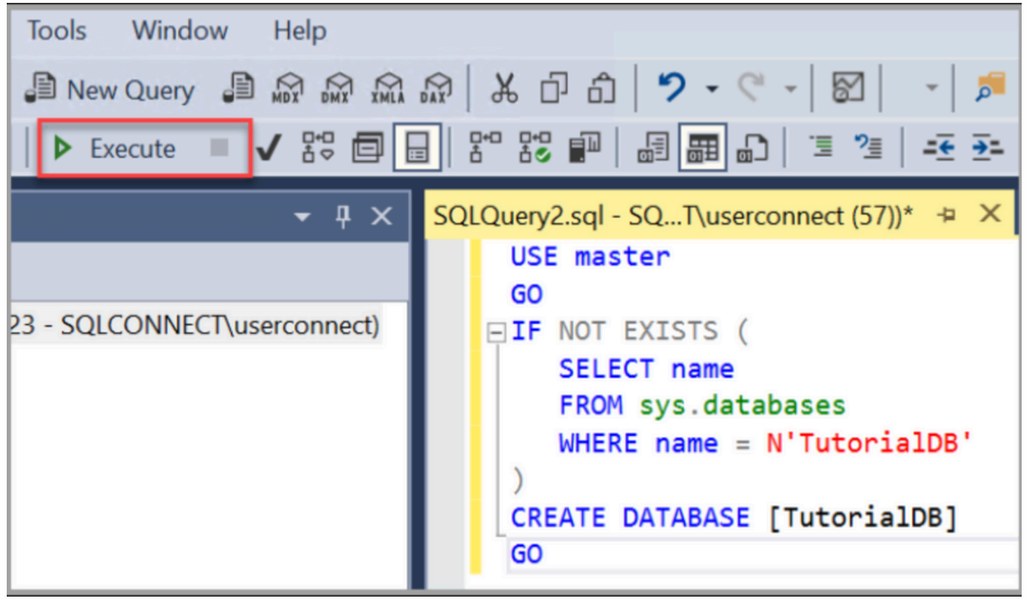


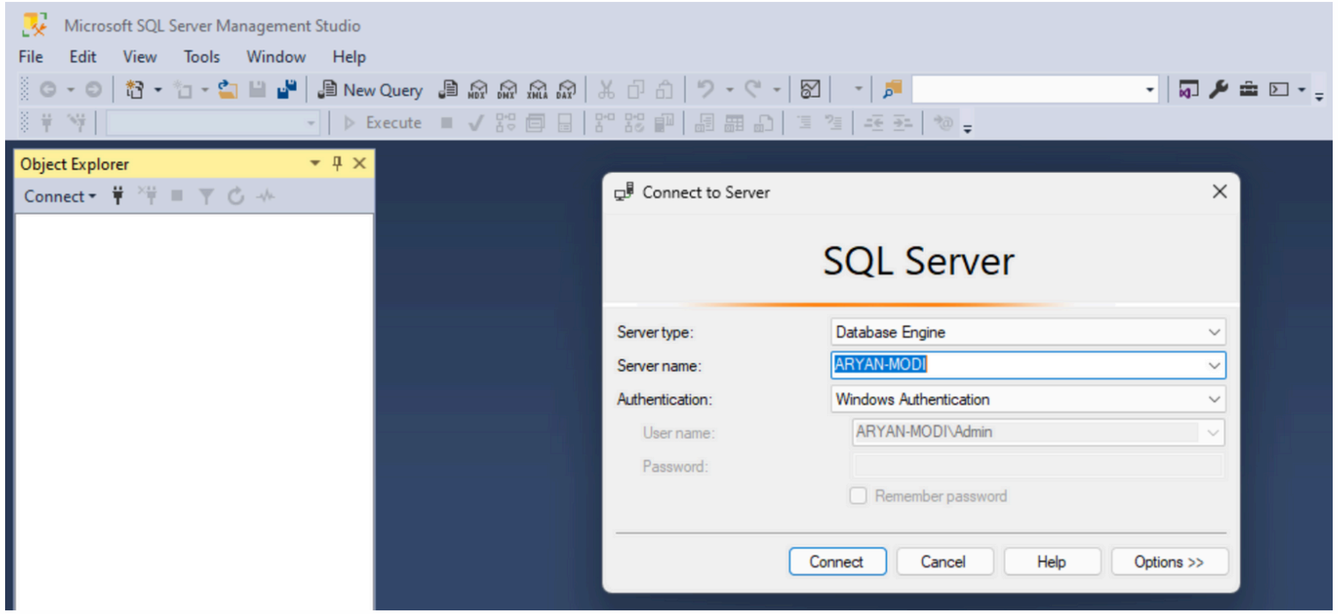
Week 2

22-01 to 28-01

- •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.
-



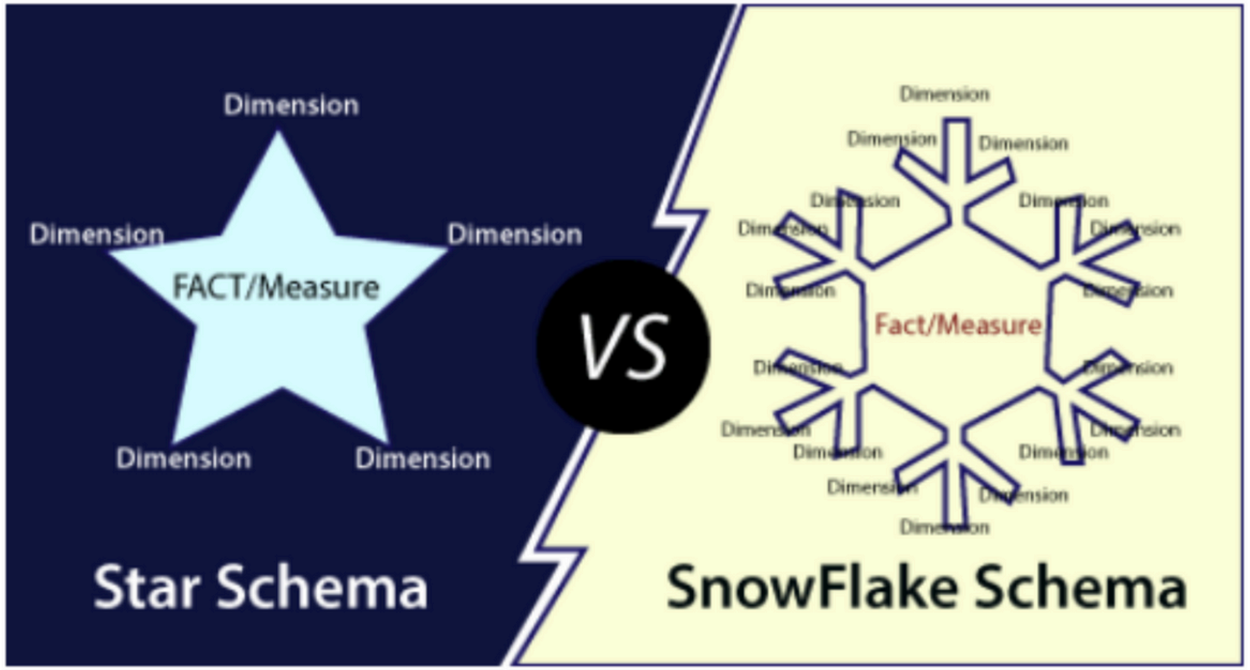


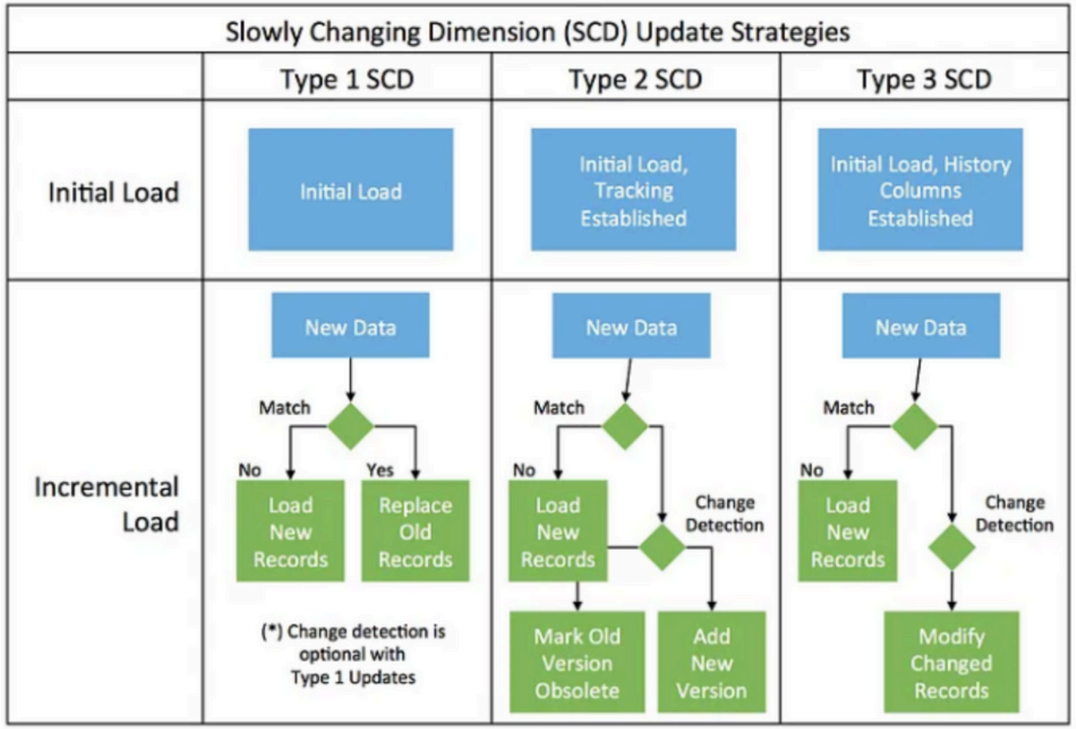


Week 3

29-01 to 04-02

- •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.





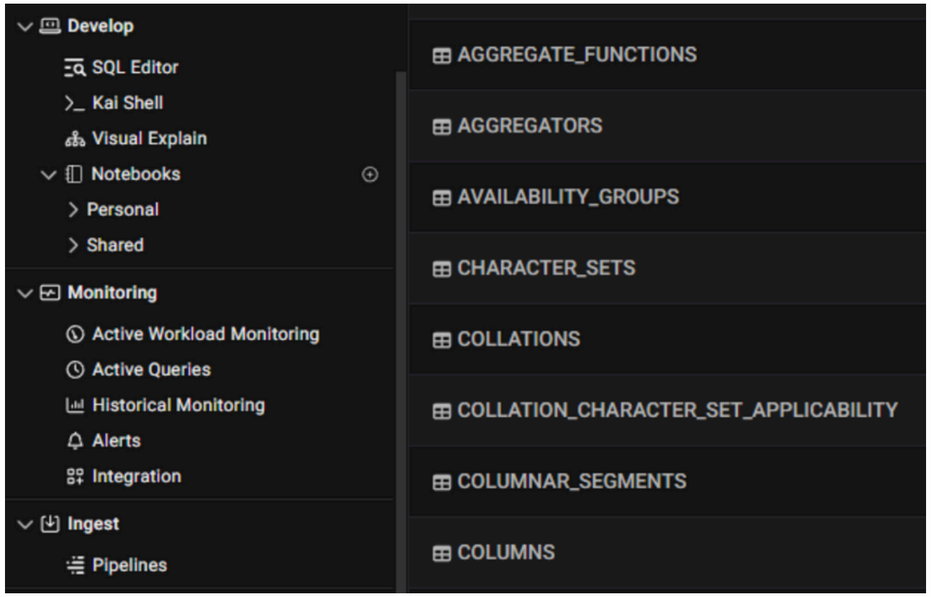
Week 4

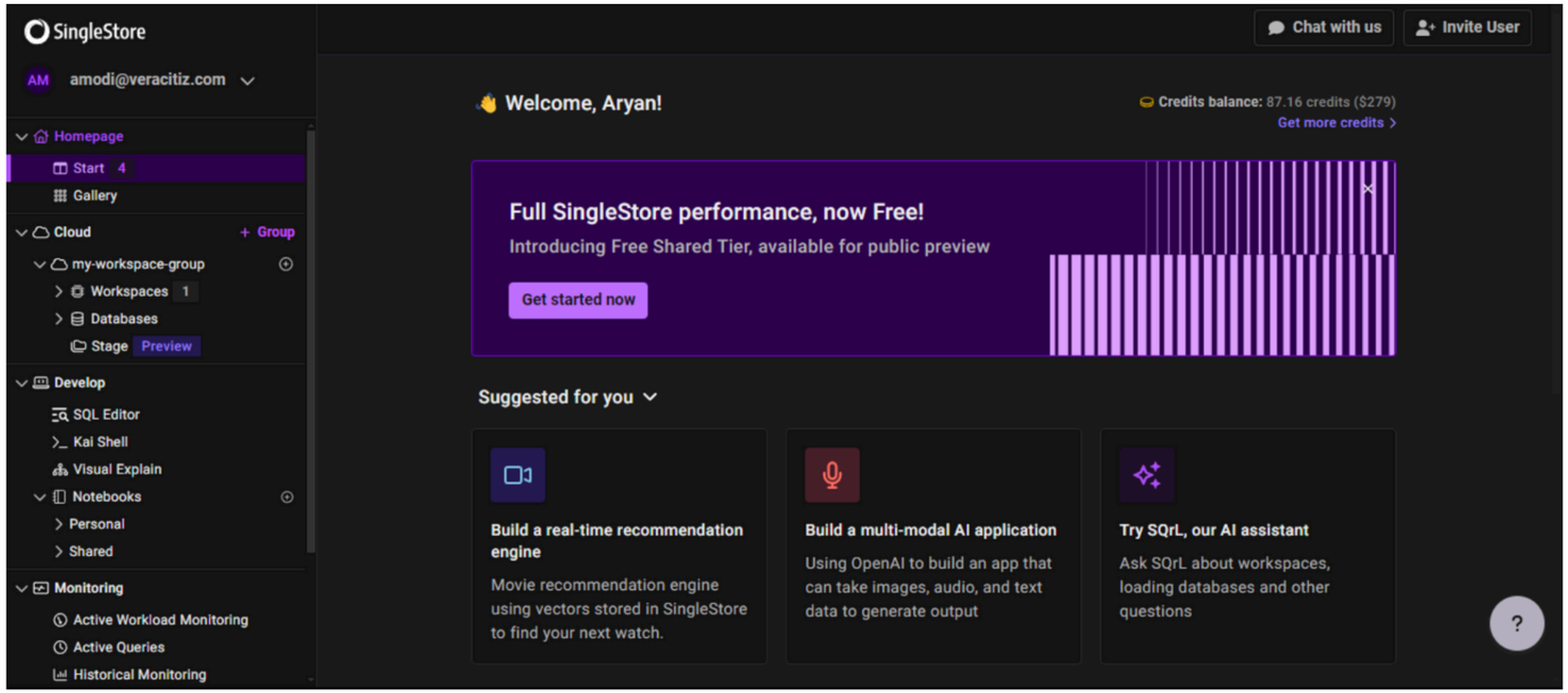
05-02 to 09-02

SingleStore

- **Hybrid Rowstore and Columnstore:**
  - **SingleStore supports both rowstore and columnstore storage formats. Rowstore is suitable for transactional workloads, while columnstore is optimized for analytical queries on large datasets.**
- **Concurrency Control:**
  - **SingleStore provides robust concurrency control mechanisms to handle multiple transactions simultaneously. This ensures data consistency and integrity in a multi-user environment.**
- **Analytics and Real-Time Processing:**
  - **SingleStore is often used for real-time analytics, allowing users to analyze and query data in real-time. This makes it suitable for applications that require immediate insights from data.**
- **JSON Support:**
  - **SingleStore supports JSON data types, allowing users to work with semi-structured data. This is especially useful in applications where flexibility in data representation is required.**
- **Integration with BI Tools:**
  - **SingleStore can be integrated with various Business Intelligence (BI) tools, enabling users to visualize and analyze data using their preferred reporting and visualization tools.**

- 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.





Week 5

12-02 to 16-02

- **I have work on HackerRank problem solving tasks.**

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.

```
    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
```

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.

```
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
```

Week 6

19-02 to 23-02

THINK THE LOGIC

IMPROVI
LOGIC
BUILDING
SKILLS IN
PROGRAMMING

- **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.**

	id	verticals	service_line	customer	product
1	1	clothing	summer	prada	shirt
2	2	clothing	summer	zara	pants
3	3	clothing	summer	zara	shirt
4	4	clothing	summer	zara	jeans
5	5	clothing	summer	gucci	jeans
6	6	clothing	summer	gucci	caps
7	7	clothing	summer	gucci	pants
8	8	clothing	summer	gucci	shades
9	9	clothing	summer	gucci	jeans
10	10	footw...	rainy	nike	flipflops
11	11	footw...	rainy	nike	sandals
12	12	footw...	rainy	nike	slipers

Query executed successfully.

```
-- Create procedure
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
```

```
EXEC RecommendProductsByCustomer @input_customer = 'nike';
```

Results Messages

	product
1	boots
2	sneakers

SQLQuery_210224.s...-MODI\Admin (54))*

89 %

Results | Messages

	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-MODI\Admin (54) | test | 00:00:00 | 18 rows

SQLQuery_220224.s...-MODI\Admin (52))*

```
7  insert into CRIC_STAT values('Australia', 'India', 'India');
8
9  select * from CRIC_STAT;
10
11 SELECT
12   Team_Name,
13   COUNT(Team_Name) AS Matches_Played,
14   SUM(CASE WHEN Winner = Team_Name THEN 1 ELSE 0 END) AS No_OF_Win,
15   SUM(CASE WHEN Winner <> Team_Name THEN 1 ELSE 0 END) AS No_OF_Loss
16 FROM(
17 SELECT Team_1 AS Team_Name, Winner FROM CRIC_STAT
18 UNION ALL
19 SELECT Team_2 AS Team_Name, Winner FROM CRIC_STAT
20 ) AS Teams
21 GROUP BY Team_Name
22 ORDER BY No_OF_Win;
23
24
```

100 %

Results | Messages

	Team_Name	Matches_Played	No_OF_Win	No_OF_Loss
1	South Africa	1	0	1
2	Sri Lanka	2	0	2
3	Australia	2	1	1
4	England	2	1	1
5	New Zealand	1	1	0
6	India	2	2	0

	Team_1	Team_2	Winner
1	India	Sri Lanka	India
2	Sri Lanka	Australia	Australia
3	South Africa	England	England
4	England	New Zealand	New Zealand
5	Australia	India	India

	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	-1920	R1
2	PA1401301	01RB0201	AC0120	PR0111	C	D	-960	-960	R1
3	PA1401301	01RB0201	AC0120	PR0111	E	F	-320	-320	R1
4	PA1401301	01RB0201	AC0120	PR0111	E	F	320	320	R1
5	PA1401301	01RB0201	AC0120	PR0111	C	D	960	960	R1
6	PA1401301	01RB0201	AC0120	PR0111	A	B	1920	1920	R1
7	PA1401303	01RB0204	AC0122	PR0113	41002	50113	-7500	-7500	R3
8	PA1401303	01RB0204	AC0122	PR0113	M	N	-1500	-1500	R3
9	PA1401303	01RB0204	AC0122	PR0113	O	P	-1000	-1000	R3
10	PA1401303	01RB0204	AC0122	PR0113	O	P	1000	1000	R3
11	PA1401303	01RB0204	AC0122	PR0113	M	N	1500	1500	R3
12	PA1401303	01RB0204	AC0122	PR0113	41002	50113	7500	7500	R3

Week 7

26-02 to 01-03

- **I have worked on SQL Logic Building sequence data and advance windows functions example , Revenue Split Level 2 task. , Revenue Split Level 1 and 2 task dynamic approach and Team Hierarchy split data , Transection and Account SCD Data , Bank DATA population. Generate DATA of bank CR/DR details.**

	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

	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

	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 sec
5	12262	Nagpur	20:55:00.0000000	15 hr 10 min 10 sec
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	Bhubaneswar	16:25:00.0000000	5 hr 35 min 35 sec

	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

	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

```
DECLARE @counter_tr INT = 1

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
END
```

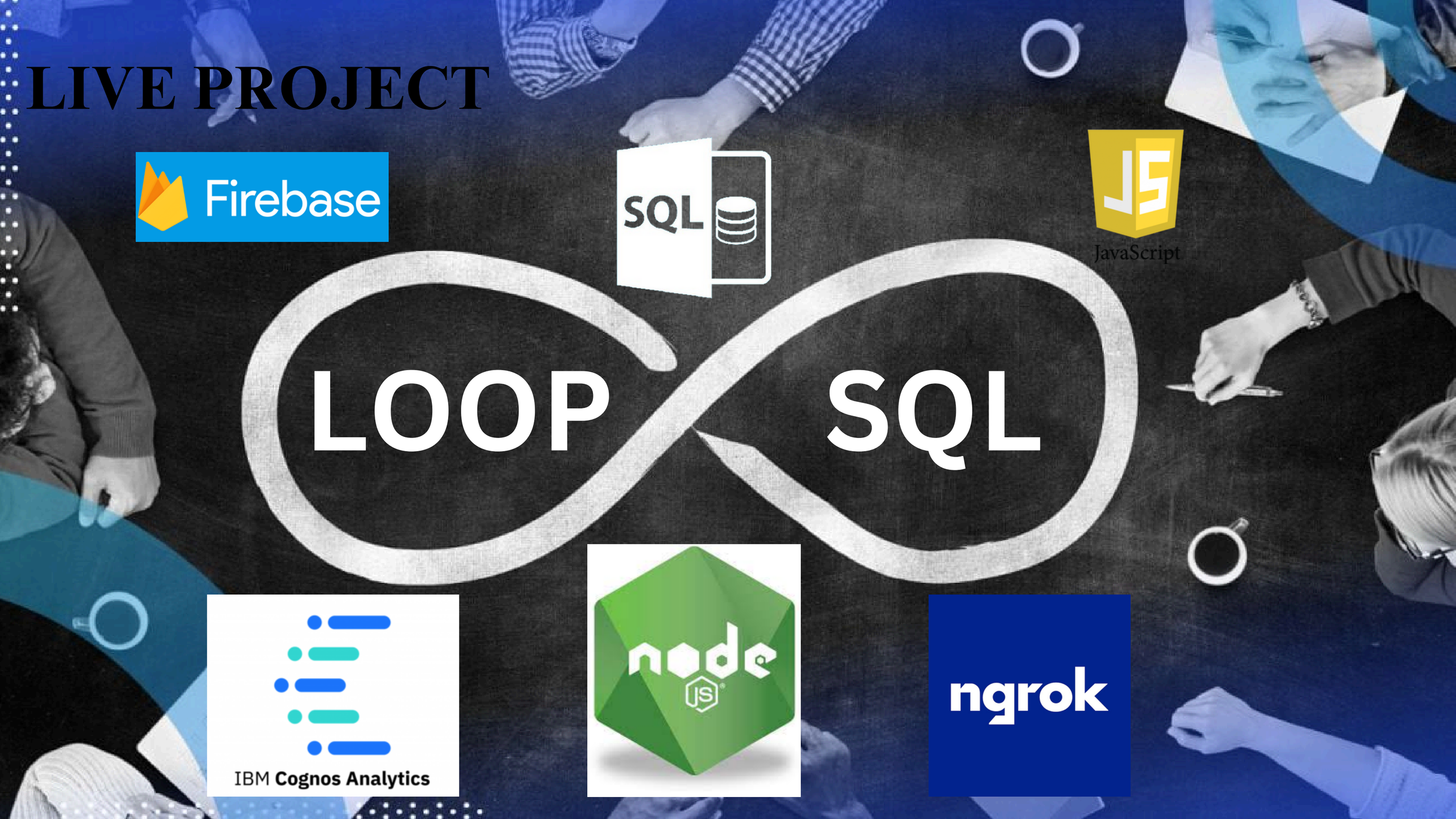
	A_ID	CUSTOMER_CODE	TRANSACTIOIN_ID	TRANSACTIOIN_DATE_ID	TRANSACTIOIN_TYPE_ID	TRANSACTIOIN_AMOUNT
1	120000236	871366511	150225112	20230412	1	4115.63
2	120000072	660907958	151238973	20230126	2	5738.63
3	120000104	572689383	832930053	20230106	1	7925.79
4	120000131	578479812	314469577	20230222	1	5626.1
5	120000205	725934031	133379672	20231212	1	7956.3
6	120000005	105497267	784431347	20230207	2	412.72
7	120000119	557804649	139803870	20230312	2	4601.93
8	120000113	213633870	859714536	20231109	1	920.14
9	120000297	988318564	435183814	20231011	1	3936.35
10	120000288	623599273	885237663	20230826	1	7085.64
11	120000152	897430719	376695831	20230109	1	9816.91
12	120000093	656281299	123226147	20240710	2	2288.07
13	120000002	183614676	383146545	20230121	1	9291.59
14	120000024	752810437	608382899	20230908	2	9683.28
15	120000173	188893570	849828751	20230404	2	2000.97
16	120000088	597044254	367511893	20230425	2	8723.01
17	120000167	420108705	474886780	20231006	2	6440.61
18	120000281	662650203	316498335	20231129	2	9778.32
19	120000060	890713376	479186908	20230918	1	228.94

Results Messages

	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	2024-06-08	7128	784893586	2024-11-06	6532	235388782
2	120000002	183614676	2024-11-07	9486	703409410	2024-07-07	9504	957119572
3	120000003	866625575	2024-07-28	4132	944404642	2024-09-26	7797	657003930
4	120000004	523621974	2024-12-06	7882	444307781	2024-11-01	9919	313183360
5	120000005	105497267	2024-07-30	6965	412055686	2024-10-24	6999	843797370
6	120000006	265344338	2024-12-07	8874	273568825	2024-12-22	3791	685841893
7	120000007	107433788	2024-07-13	3296	433144515	2024-10-30	3491	203751992
8	120000008	923956299	2024-09-01	570	325740095	2024-01-13	2973	467745520
9	120000009	860633076	2024-11-15	1796	790344388	2024-03-23	7332	179753720
10	120000010	564069273	2024-11-18	2650	532281407	2024-12-17	1306	966785774
11	120000011	253076075	2024-05-14	7521	809382102	2024-08-16	5364	598800944
12	120000012	345840107	2024-05-29	8169	370380885	2023-12-09	6725	818186121
13	120000013	872192761	2024-09-28	234	959352949	2024-10-07	8080	337604491
14	120000014	180274211	2024-12-26	7838	778845663	2024-11-21	854	712987688

Query executed successfully. ARYAN-MODI (16.0 RTM) ARYAN-MODI\Admin (52) test 00:00:00 300 rows

LIVE PROJECT



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.

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.

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.

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.

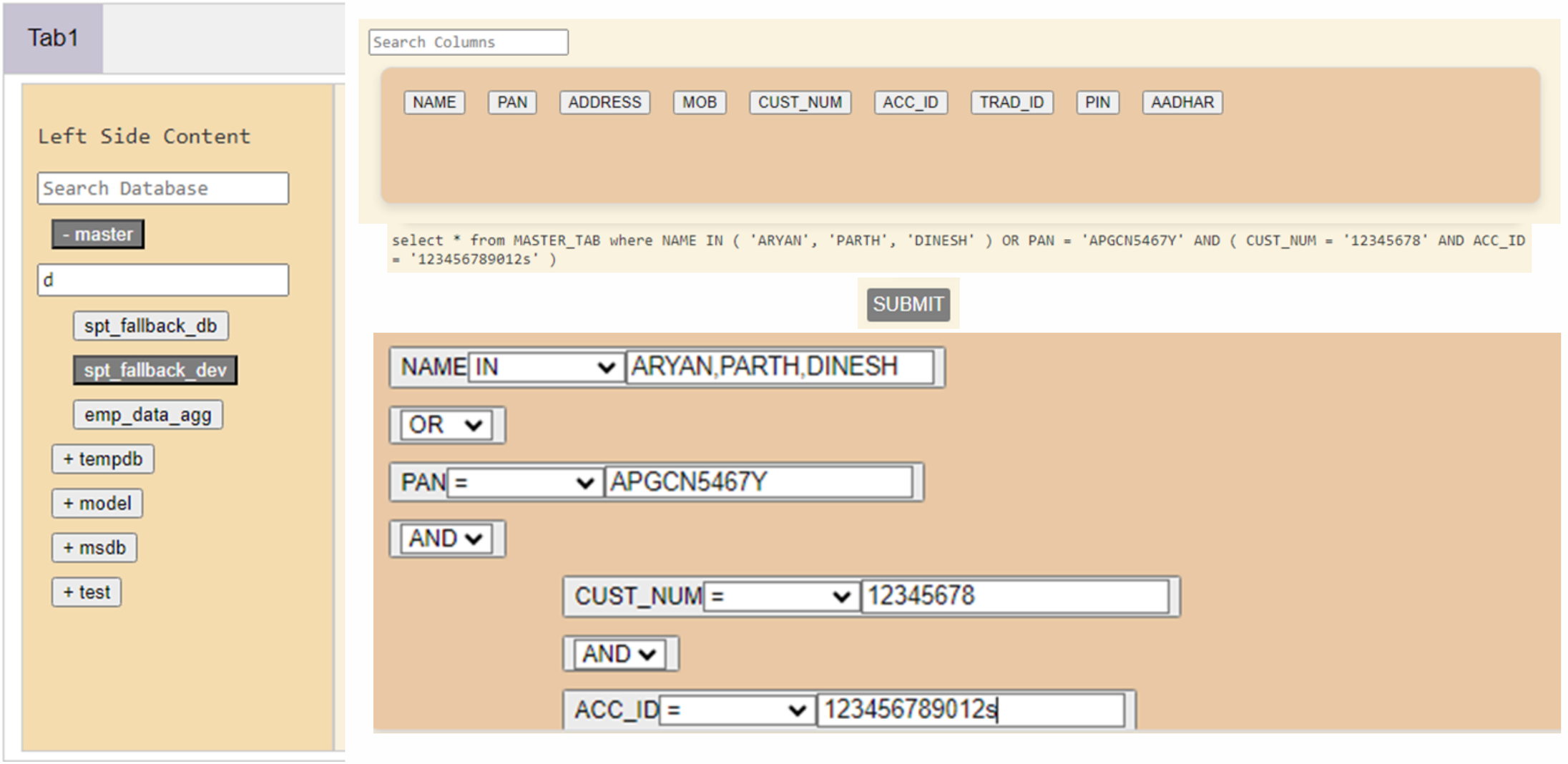
Ngrok

- 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.
  - **Integration with Development** Tools: Ngrok seamlessly integrates with various development tools and frameworks commonly used by developers, such as Node.js, Python, Ruby on Rails, and more. This integration streamlines the process of setting up tunnels and testing web applications or APIs during development.
  - **Dynamic Port Forwarding**: Ngrok's ability to dynamically assign a public URL to a local port simplifies the process of exposing multiple services running on different ports simultaneously. This flexibility is particularly useful for developers working on complex projects with multiple components.

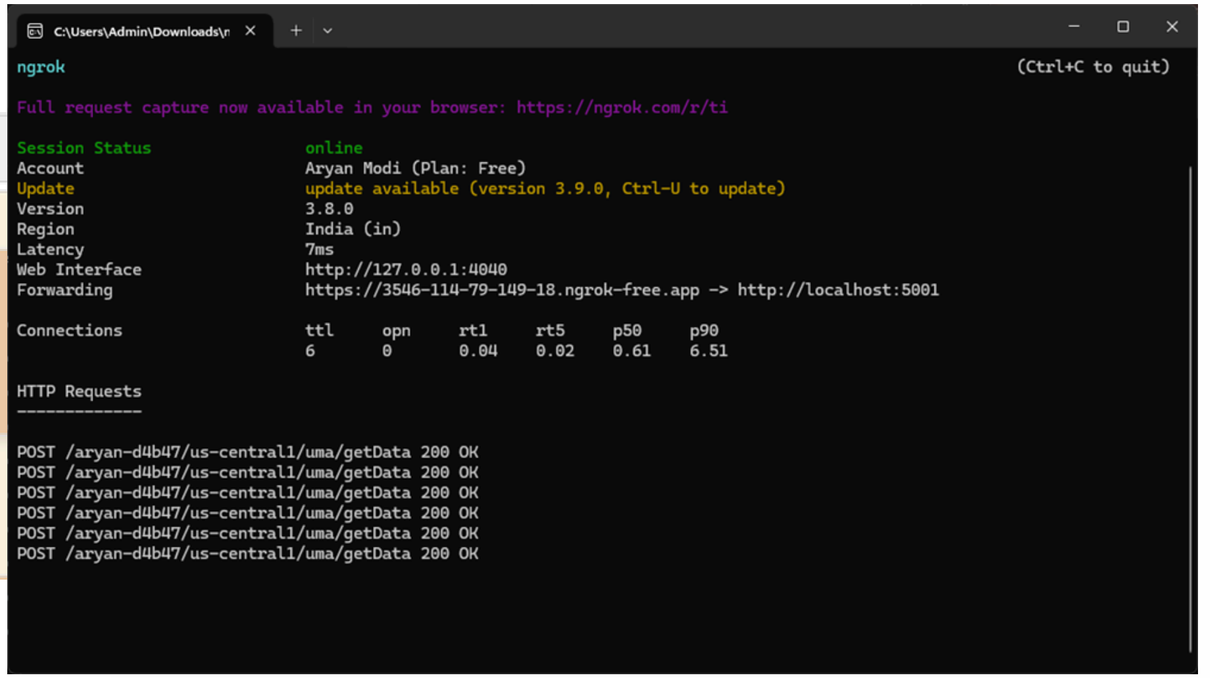
Cognos

- 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.
  - **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.
  - **Documentation and Support**: IBM provides documentation, tutorials, and forums to help developers learn how to create and integrate Custom Controls effectively.

SCREENSHOTS



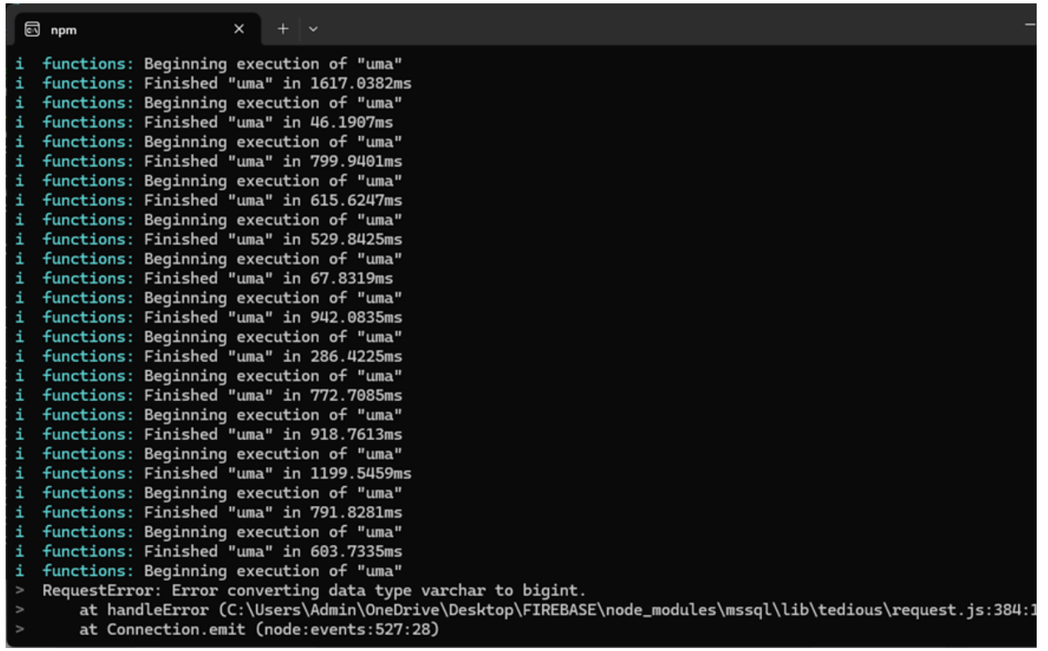
SCREENSHOTS



×

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

Download CSV



Thank you!

Feel free to reach out to us if you have any questions.

VERACITIZ

ARYAN MODI

Phone Number
+91 94293 26003

Email Address
amodi@veracitiz.com

Website
www.veracitiz.com