Data Visualization Associate Early Internship

Week 2 - Data Transformation & Master Table Creation

by

0303 DVA TEAM 22B

 Submitted by: **Areeba Fatima (areebafatima721@gmail.com)**

 Team Members:

Sakshi Gollar (sakshigollar31@gmail.com)

Wamiq Ejaz (ejazwamiq@gmail.com)

Shivani Galande (shivanigalande2512@gmail.com)

Areeba Fatima (areebafatima721@gmail.com)

Varun D (varundevaraj1188@gmail.com)

- **Final Master Table**

Think link below gives the final master table which includes include key columns, data types, and constraints or indexes applied for efficiency.

Link:

https://docs.google.com/spreadsheets/d/1qK8fCx8kf4yKKmD1aw67uzxdUJYjca_3gER-Qz1rqtY/edit?usp=sharing

- **Table Creation Query:**

Link: https://drive.google.com/drive/folders/1FafwcCjeKQaBgQxWi_0I_EYwCNC0bTOb?usp=sharing

The SQL script that defines the schema for the Master Table, specifying all necessary constraints, indexes, and relationships to maintain data integrity.

```
---------------------------Importing all the datasets----------------------
--learner_raw data
CREATE TABLE Learner_Raw (
    learner_id TEXT PRIMARY KEY,
    country TEXT,
    degree TEXT,
    institution TEXT,
    major TEXT
);
COPY Learner_Raw FROM 'C:\Program Files\PostgreSQL\17\data\Learner_Raw.csv' WITH (FORMAT
'csv', HEADER, DELIMITER ',');

--marketing comapaigns data
CREATE TABLE Marketing_Campaigns (
    ad_account_name TEXT,
    campaign_name TEXT,
    delivery_status TEXT,
    delivery_level TEXT,
    reach BIGINT,
    outbound_clicks INT,
    landing_page_views INT,
    result_type TEXT,
    results FLOAT,
    cost_per_result FLOAT,
    amount_spent_aed FLOAT,
    cpc_cost_per_link_click FLOAT,
    reporting_starts DATE,
    reporting_ends DATE
);
```

```
COPY marketing_campaigns (
    ad_account_name, campaign_name, delivery_status, delivery_level,
    reach, outbound_clicks, landing_page_views, result_type, results,
    cost_per_result, amount_spent_aed, cpc_cost_per_link_click,
    reporting_starts, reporting_ends
)
FROM 'C:\Program Files\PostgreSQL\17\data\Cleaned_Marketing_Campaign_Data.csv'
DELIMITER ','
CSV HEADER
NULL AS 'NULL'
QUOTE '"'
ESCAPE '\';

--cognito data
CREATE TABLE Cognito_Raw (
    user_id TEXT PRIMARY KEY,
    email TEXT,
    gender TEXT,
    UserCreateDate TIMESTAMP,
    UserLastModifiedDate TIMESTAMP,
    birthdate TEXT,
    city TEXT,
    zip TEXT,
    state TEXT
);
COPY Cognito_Raw FROM 'C:\Program Files\PostgreSQL\17\data\Cleaned_Cognito_Raw2.csv' WITH
(FORMAT 'csv', HEADER, DELIMITER ',');


--opportunity raw data
CREATE TABLE Opportunity_Raw (
    opportunity_id TEXT PRIMARY KEY,
    opportunity_name TEXT,
    category TEXT,
    opportunity_code TEXT,
    tracking_questions TEXT
);

COPY Opportunity_Raw FROM 'C:\Program Files\PostgreSQL\17\data\Cleaned Opportunity_Raw
Dataset.csv' WITH (FORMAT 'csv', HEADER, DELIMITER ',');

--learneropportunity data
DROP TABLE IF EXISTS LearnerOpportunity_Raw;

CREATE TABLE temp_LearnerOpportunity_Raw (
    enrollment_id TEXT,
    learner_id TEXT,
    assigned_cohort TEXT,
    apply_date TEXT,
```

```sql
    status INTEGER
);

COPY temp_LearnerOpportunity_Raw
FROM 'C:\Program Files\PostgreSQL\17\data\Cleaned_LearnerOpportunity_Raw.csv'
WITH (FORMAT csv, HEADER, DELIMITER ',', NULL 'NULL');

SELECT enrollment_id, COUNT(*)
FROM learneropportunity_raw
GROUP BY enrollment_id
HAVING COUNT(*) > 1;


--cohort dataset
DROP TABLE IF EXISTS Cohort_Data;
CREATE TABLE Cohort_Data (
    cohort_id TEXT,
    cohort_code TEXT,
    size INTEGER,
    start_date TIMESTAMP,
    end_date TIMESTAMP
);

COPY Cohort_Data FROM 'C:\Program Files\PostgreSQL\17\data\Cleaned_Cohort_data.csv' WITH
(FORMAT 'csv', HEADER, DELIMITER ',');


-------------------viewing all the datasets------------------------

SELECT * FROM Learner_Raw LIMIT 10;
SELECT * FROM Marketing_Campaigns LIMIT 140;
SELECT * FROM LearnerOpportunity_Raw LIMIT 10;
SELECT * FROM Cognito_Raw LIMIT 10;
SELECT * FROM Cohort_Data LIMIT 10;
SELECT * FROM Opportunity_Raw LIMIT 10;

--------------Checking total row count in each column---------------------
SELECT 'Learner_Raw' AS table_name, COUNT(*) FROM Learner_Raw
SELECT 'Marketing_Campaigns' AS table_name, COUNT(*) FROM Marketing_Campaigns
SELECT 'LearnerOpportunity_Raw' AS table_name, COUNT(*) FROM LearnerOpportunity_Raw
SELECT 'Cognito_Raw' AS table_name, COUNT(*) FROM Cognito_Raw
SELECT 'Cohort_Data' AS table_name, COUNT(*) FROM Cohort_Data
SELECT 'Opportunity_Raw' AS table_name, COUNT(*) FROM Opportunity_Raw;

----------------------checking column names and their data types------------------------
SELECT column_name, data_type
FROM information_schema.columns
WHERE table_name = 'learner_raw';

SELECT column_name, data_type
```

```sql
FROM information_schema.columns
WHERE table_name = 'cognito_raw';


SELECT column_name, data_type
FROM information_schema.columns
WHERE table_name = 'marketing_campaigns';

SELECT column_name, data_type
FROM information_schema.columns
WHERE table_name = 'cohort_data';

SELECT column_name, data_type
FROM information_schema.columns
WHERE table_name = 'opportunity_raw';

SELECT column_name, data_type
FROM information_schema.columns
WHERE table_name = 'learneropportunity_raw';


------------------Checking duplicates in all the datasets----------------
--Checking duplicates in learner_raw
SELECT learner_id, COUNT(*)
FROM learner_raw
GROUP BY learner_id
HAVING COUNT(*) > 1;

--checking duplicates in cognito_raw
SELECT user_id, COUNT(*)
FROM cognito_raw
GROUP BY user_id
HAVING COUNT(*) > 1;

--checking duplicates in opportunity_raw
SELECT opportunity_id, COUNT(*)
FROM opportunity_raw
GROUP BY opportunity_id
HAVING COUNT(*) > 1;

SELECT learner_id, COUNT(*) AS occurrence_count
FROM learneropportunity_raw
GROUP BY learner_id
HAVING COUNT(*) > 1
ORDER BY occurrence_count DESC;



--------------identifying common keys and relationship--------------------
-- checking if enrollment_id and learner_id are same
```

```sql
SELECT lo.enrollment_id, lr.learner_id
FROM learneropportunity_raw lo
JOIN learner_raw lr ON lo.enrollment_id = lr.learner_id
LIMIT 1000;

-- checking if assigned_cohort and cohort_code are same
SELECT lo.assigned_cohort, cd.cohort_code
FROM learneropportunity_raw lo
JOIN cohort_data cd ON lo.assigned_cohort = cd.cohort_code
LIMIT 10;

--checking if learner_id and opportunity_id are same
SELECT lo.learner_id, orw.opportunity_id
FROM learneropportunity_raw lo
JOIN opportunity_raw orw ON lo.learner_id = orw.opportunity_id
LIMIT 10;

--checking if user_id, enrollment_id, and learner_id are same
SELECT cr.user_id, lo.enrollment_id, lr.learner_id
FROM cognito_raw cr
JOIN learneropportunity_raw lo ON cr.user_id = REPLACE(lo.enrollment_id, 'Learner#', '')
JOIN learner_raw lr ON lo.enrollment_id = lr.learner_id
LIMIT 10;

-----------------checking duplicates in the above columns-------------------
SELECT enrollment_id, COUNT(*) AS occurrence_count
FROM learneropportunity_raw
GROUP BY enrollment_id
HAVING COUNT(*) > 1
ORDER BY occurrence_count DESC;


SELECT assigned_cohort, COUNT(*) AS occurrence_count
FROM learneropportunity_raw
GROUP BY assigned_cohort
HAVING COUNT(*) > 1
ORDER BY occurrence_count DESC;

SELECT learner_id, COUNT(*) AS occurrence_count
FROM learneropportunity_raw
GROUP BY learner_id
HAVING COUNT(*) > 1
ORDER BY occurrence_count DESC;

SELECT user_id, COUNT(*) AS occurrence_count
FROM cognito_raw
GROUP BY user_id
HAVING COUNT(*) > 1
ORDER BY occurrence_count DESC;
```

```
SELECT learner_id, COUNT(*) AS occurrence_count
FROM learner_raw
GROUP BY learner_id
HAVING COUNT(*) > 1
ORDER BY occurrence_count DESC;


-----Count of each unique value in the assigned_cohort column from the learneropportunity_raw dataset:
SELECT assigned_cohort, COUNT(*) AS count
FROM learneropportunity_raw
GROUP BY assigned_cohort
ORDER BY count DESC;


--check for duplicates in the cohort_code column in the cohort_raw dataset,
SELECT cohort_code, COUNT(*) AS count
FROM cohort_data
GROUP BY cohort_code
HAVING COUNT(*) > 1
ORDER BY count DESC;


--Joining a copy of cognito_raw data and learner_raw

DROP TABLE IF EXISTS Learner_Cognito;
CREATE TABLE Learner_Cognito AS
SELECT
    mp.user_id,
    lr.learner_id,
    mp.email,
    mp.gender,
    mp.birthdate,
    lr.degree,
    lr.major,
    lr.institution,
    mp.city,
    mp.zip,
    mp.state,
    lr.country,
    mp.usercreatedate,
    mp.userlastmodifieddate
FROM cognito_raw_copy mp
LEFT JOIN learner_raw lr
ON mp.user_id = REGEXP_REPLACE(lr.learner_id, '^Learner#', '');

SELECT * FROM Learner_Cognito LIMIT 10;



--Joining Cohort_data and LearnerOpportunity_raw:
CREATE TABLE Cohort_LearnerOpportunity AS
```

```sql
SELECT
    lo.*,
    c.*
FROM learneropportunity_raw lo
LEFT JOIN cohort_data c
ON lo.assigned_cohort = c.cohort_code;

SELECT * FROM Cohort_LearnerOpportunity LIMIT 10;


----Joining Cohort_LearnerOpportunity and Opportunity_raw:
CREATE TABLE Opportunity_CohortLearnerOpp AS
SELECT
    cl.*,
    o.*
FROM Cohort_LearnerOpportunity cl
LEFT JOIN opportunity_raw o
ON cl.learner_id = o.opportunity_id;

SELECT * FROM Opportunity_CohortLearnerOpp LIMIT 10;


-------------------Creating MASTER TABLE by joining Learner_Cognito and
Opportunity_CohortLearnerOpp---------------------------------


UPDATE Opportunity_CohortLearnerOpp
SET cleaned_enrollment_id = REGEXP_REPLACE(enrollment_id, '^Learner#', '');

CREATE INDEX IF NOT EXISTS idx_cleaned_enrollment_id
ON Opportunity_CohortLearnerOpp(cleaned_enrollment_id);
DROP TABLE IF EXISTS Master_Table;
CREATE TABLE Master_Table AS
SELECT
    lc.user_id, lc.learner_id, ocl.enrollment_id,
    ocl.learner_id AS learneropp_id, lc.email, lc.gender,
    lc.birthdate, lc.degree, lc.major, lc.institution,
    lc.city, lc.zip, lc.state, lc.country, ocl.opportunity_id,
    ocl.opportunity_name, ocl.category, ocl.opportunity_code,
    ocl.cohort_id, ocl.cohort_code, ocl.assigned_cohort,
    ocl.size, ocl.apply_date, ocl.status,
    ocl.start_date, ocl.end_date, lc.usercreatedate,
    lc.userlastmodifieddate AS userlastmodifieddate
FROM Learner_Cognito lc
LEFT JOIN Opportunity_CohortLearnerOpp ocl
ON lc.user_id = ocl.cleaned_enrollment_id;

---viewing master table
SELECT * FROM master_table LIMIT 10;
```

```
--checking columns and their data types in master table
SELECT column_name, data_type
FROM information_schema.columns
WHERE table_name = 'master_table';
```

- **Stored Procedure Query**

Link: https://drive.google.com/drive/folders/1FafwcCjeKQaBgQxWi_0I_EYwCNC0bTOb?usp=sharing

```
--------------------------------Cleaning Master Table---------------------------------
CREATE TABLE final_master_table (
   user_id TEXT,
   learner_id TEXT,
   enrollment_id TEXT,
   learneropp_id TEXT,
   email TEXT,
   gender TEXT,
   birthdate DATE,
   degree TEXT,
   major TEXT,
   institution TEXT,
   city TEXT,
   zip TEXT,
   state TEXT,
   country TEXT,
   opportunity_id TEXT,
   opportunity_name TEXT,
   category TEXT,
   opportunity_code TEXT,
   cohort_id TEXT,
   cohort_code TEXT,
   assigned_cohort TEXT,
   cohort_size INT,
   apply_date TIMESTAMP,
   status TEXT,
   start_date TIMESTAMP,
   end_date TIMESTAMP,
   usercreatedate TIMESTAMP,
   userlastmodifieddate TIMESTAMP
);

CREATE TABLE master_table_temp (
   user_id TEXT, learner_id TEXT, enrollment_id TEXT, learneropp_id TEXT,
   email TEXT, gender TEXT, birthdate TEXT, degree TEXT, major TEXT,
   institution TEXT, city TEXT, zip TEXT, state TEXT, country TEXT,
   opportunity_id TEXT, opportunity_name TEXT, category TEXT,
   opportunity_code TEXT, cohort_id TEXT, cohort_code TEXT,
   assigned_cohort TEXT, cohort_size TEXT, apply_date TEXT,
    status TEXT, start_date TEXT, end_date TEXT, usercreatedate TEXT,
    userlastmodifieddate TEXT
);

COPY master_table_temp
FROM 'C:\Program Files\PostgreSQL\17\data\MASTER TABLE.csv'
WITH (FORMAT csv, HEADER, DELIMITER ',', NULL 'NULL')

SELECT column_name, data_type
FROM information_schema.columns
WHERE table_name = 'master_table' AND column_name = 'zip';



INSERT INTO final_master_table
SELECT
    user_id, learner_id, enrollment_id, learneropp_id, email, gender,
    NULLIF(birthdate, '')::DATE, degree, major, institution, city,
    NULLIF(zip, ''),  -- No type casting to INTEGER
    state, country, opportunity_id,
    opportunity_name, category, opportunity_code, cohort_id,
    cohort_code, assigned_cohort,
    NULLIF(cohort_size, '')::INTEGER,
    NULLIF(apply_date, '')::DATE, status,
    NULLIF(start_date, '')::DATE, NULLIF(end_date, '')::DATE,
    NULLIF(usercreatedate, '')::TIMESTAMP,
    NULLIF(userlastmodifieddate, '')::TIMESTAMP
FROM master_table_temp;

SELECT * FROM master_table LIMIT 100;
--------------------------------Cleaning Master Table---------------------------------

----- Updating empty string values to NULL for various columns
UPDATE final_master_table
SET cohort_id = NULL
WHERE cohort_id = '';

UPDATE final_master_table
SET degree = NULL
WHERE degree = '';

UPDATE final_master_table
SET cohort_code = NULL
WHERE cohort_code = '';

UPDATE final_master_table
SET assigned_cohort = NULL
WHERE assigned_cohort = '';

--Renaming column size as cohort size
```

```
ALTER TABLE final_master_table RENAME COLUMN "size" TO cohort_size;


UPDATE final_master_table
SET cohort_size = NULL
WHERE cohort_size = '';

UPDATE final_master_table
SET status = NULL
WHERE status= '';

UPDATE final_master_table
SET apply_date = NULL
WHERE apply_date = '';

UPDATE final_master_table
SET end_date = NULL
WHERE end_date = '';

UPDATE final_master_table
SET start_date = NULL
WHERE start_date = '';

UPDATE final_master_table
SET opportunity_id = NULL
WHERE opportunity_id= '';

UPDATE final_master_table
SET opportunity_name = NULL
WHERE opportunity_name= '';

UPDATE final_master_table
SET opportunity_code = NULL
WHERE opportunity_code= '';

UPDATE final_master_table
SET category = NULL
WHERE category= '';

-- Check distinct values in the gender column
SELECT DISTINCT gender FROM final_master_table;

-- Update NULL values in gender to 'Unknown'
UPDATE final_master_table
SET gender = 'Unknown'
WHERE gender IS NULL;

SELECT distinct degree FROM FINAL_MASTER_TABLE

UPDATE final_master_table
```

```sql
SET degree = 'Not Specified'
WHERE degree IS NULL;

SELECT distinct major FROM FINAL_MASTER_TABLE

UPDATE final_master_table
SET major = 'Not Specified'
WHERE major IS NULL;

SELECT distinct opportunity_name FROM FINAL_MASTER_TABLE

UPDATE final_master_table
SET opportunity_name = 'Not Specified'
WHERE opportunity_name IS NULL;

-- Removing leading and trailing spaces from the 'degree' column
UPDATE final_master_table
SET degree = TRIM(degree);

-- Replacing NULL or negative values in the 'size' column with 0
SELECT distinct cohort_size FROM FINAL_MASTER_TABLE

UPDATE final_master_table
SET cohort_size = 0
WHERE cohort_size IS NULL OR cohort_size < 0;

-- Replacing NULL values in the 'state', 'city', and 'country' columns with 'Not Available'
SELECT distinct count(*) as countw,state FROM final_master_table group by state order by state desc

UPDATE final_master_table
SET state = 'Not Available'
WHERE state IS NULL;

SELECT distinct count(*) as countw,city FROM final_master_table group by city order by countw desc

UPDATE final_master_table
SET city = 'Not Available'
WHERE city IS NULL;

SELECT distinct count(*) as countw,country FROM final_master_table group by country order by countw
desc

UPDATE final_master_table
SET country = 'Not Available'
WHERE country IS NULL;

-- Counting the number of users associated with each institution
SELECT institution, COUNT(user_id) AS total_users
FROM final_master_table
GROUP BY institution
```

```
ORDER BY total_users DESC;

-- Converting institution names to proper case (first letter of each word capitalized)
UPDATE final_master_table
SET institution = INITCAP(institution)
WHERE institution IS NOT NULL;

-- Replacing NULL, 'None', 'N/A', 'Na', and 'Null' values in the 'institution' column with 'Not Specified'
UPDATE final_master_table
SET institution = 'Not Specified'
WHERE institution IS NULL
   OR TRIM(INITCAP(institution)) IN ('None', 'N/A', 'Na', 'Null', '');

-- Removing leading and trailing spaces from email addresses and converting them to lowercase
SELECT email FROM final_master_table

UPDATE final_master_table
SET email = LOWER(TRIM(email));

-- Identifying duplicate user_id values and counting their occurrences
SELECT user_id, COUNT(*) AS occurrences
FROM final_master_table
GROUP BY user_id
HAVING COUNT(*) > 1;


SELECT * FROM final_master_table LIMIT 1000;
```

- <u>**Data Quality Report**</u>

A summary report detailing validation and cleaning checks implemented in the ETL process, including:

➢ **Issues Detected** (e.g., duplicates, inconsistent text formats).

- **Duplicate Records:** Identified duplicates in the user_id column and other key fields.
- **Inconsistent Text Formats:**
  - institution had mixed cases and unnecessary spaces.
  - email contained uppercase characters and leading/trailing spaces.
- **Null or Empty Values:**
  - gender, degree, major, institution, state, city, and country had missing values.
  - cohort_size, apply_date, start_date, end_date contained empty strings instead of NULLs.
- **Incorrect Data Types:**
  - birthdate, apply_date, start_date, end_date had text values instead of proper date format.
  - zip was stored as text instead of an integer.
- **Invalid Data Entries:**
  - opportunity_code contained None, N/A, and other invalid values.

- **Testing Methodology** (e.g., record counts before and after cleaning, validation queries, unit/integration testing results).

- **Cleaning Logic Applied** to resolve these issues.

- **Text Standardization:**
  - Applied INITCAP() on institution for consistent capitalization.
  - Converted email to lowercase using LOWER(TRIM(email)).
- **Handling Missing Values:**
  - Updated NULL values:
    - gender → 'Unknown'
    - degree, major, institution → 'Not Specified'
    - state, city, country → 'Not Available'
  - Converted empty strings in cohort_size, apply_date, start_date, end_date to NULL.
- **Data Type Corrections:**
  - Used NULLIF(value, '')::DATE for date fields.
  - Converted zip from text to integer using ALTER TABLE final_master_table ALTER COLUMN zip TYPE INTEGER USING NULLIF(zip, '')::INTEGER.
- **Validation Checks:**
  - Identified and replaced invalid values (None, N/A, etc.) in opportunity_code.

➢ **Testing Methodology** (e.g., record counts before and after cleaning, validation queries, unit/integration testing results).

Link: https://drive.google.com/drive/folders/1FafwcCjeKQaBgQxWi_0I_EYwCNC0bTOb?usp=sharing

Record Counts Before Cleaning:

```
------------------------Record counts before cleaning the master table--------------------------------
--Count of rows in master table before cleaning
SELECT COUNT(*) AS total_rows FROM master_table;

--Count of each column of master table before cleaning
SELECT
   COUNT(user_id) AS user_id_count,
   COUNT(learner_id) AS learner_id_count,
   COUNT(enrollment_id) AS enrollment_id_count,
   COUNT(learneropp_id) AS learneropp_id_count,
   COUNT(email) AS email_count,
   COUNT(gender) AS gender_count,
   COUNT(birthdate) AS birthdate_count,
   COUNT(degree) AS degree_count,
   COUNT(major) AS major_count,
   COUNT(institution) AS institution_count,
   COUNT(city) AS city_count,
   COUNT(zip) AS zip_count,
   COUNT(state) AS state_count,
   COUNT(country) AS country_count,
   COUNT(opportunity_id) AS opportunity_id_count,
   COUNT(opportunity_name) AS opportunity_name_count,
   COUNT(category) AS category_count,
   COUNT(opportunity_code) AS opportunity_code_count,
   COUNT(cohort_id) AS cohort_id_count,
   COUNT(cohort_code) AS cohort_code_count,
   COUNT(assigned_cohort) AS assigned_cohort_count,
   COUNT(cohort_size) AS cohort_size_count,
   COUNT(apply_date) AS apply_date_count,
   COUNT(status) AS status_count,
   COUNT(start_date) AS start_date_count,
   COUNT(end_date) AS end_date_count,
   COUNT(usercreatedate) AS usercreatedate_count,
   COUNT(userlastmodifieddate) AS userlastmodifieddate_count
FROM master_table;

--Count of null values in each column of master table before cleaning
SELECT
   COUNT(user_id) AS user_id_count,
   COUNT(learner_id) AS learner_id_count,
   COUNT(enrollment_id) AS enrollment_id_count,
   COUNT(learneropp_id) AS learneropp_id_count,
   COUNT(email) AS email_count,
   COUNT(gender) AS gender_count,
   COUNT(birthdate) AS birthdate_count,
   COUNT(degree) AS degree_count,
   COUNT(major) AS major_count,
   COUNT(institution) AS institution_count,
   COUNT(city) AS city_count,
   COUNT(zip) AS zip_count,
   COUNT(state) AS state_count,
   COUNT(country) AS country_count,
   COUNT(opportunity_id) AS opportunity_id_count,
   COUNT(opportunity_name) AS opportunity_name_count,
   COUNT(category) AS category_count,
   COUNT(opportunity_code) AS opportunity_code_count,
   COUNT(cohort_id) AS cohort_id_count,
   COUNT(cohort_code) AS cohort_code_count,
   COUNT(assigned_cohort) AS assigned_cohort_count,
   COUNT(cohort_size) AS cohort_size_count,
   COUNT(apply_date) AS apply_date_count,
   COUNT(status) AS status_count,
   COUNT(start_date) AS start_date_count,
   COUNT(end_date) AS end_date_count,
   COUNT(usercreatedate) AS usercreatedate_count,
   COUNT(userlastmodifieddate) AS userlastmodifieddate_count
FROM master_table;

--check the count of duplicates in each column of master_table before cleaning
SELECT
   'user_id' AS column_name, COUNT(user_id) - COUNT(DISTINCT user_id) AS duplicate_count FROM
master_table
```

```
UNION ALL
SELECT
    'learner_id', COUNT(learner_id) - COUNT(DISTINCT learner_id) FROM master_table
UNION ALL
SELECT
    'enrollment_id', COUNT(enrollment_id) - COUNT(DISTINCT enrollment_id) FROM master_table
UNION ALL
SELECT
    'learneropp_id', COUNT(learneropp_id) - COUNT(DISTINCT learneropp_id) FROM master_table
UNION ALL
SELECT
    'email', COUNT(email) - COUNT(DISTINCT email) FROM master_table
UNION ALL
SELECT
    'gender', COUNT(gender) - COUNT(DISTINCT gender) FROM master_table
UNION ALL
SELECT
    'birthdate', COUNT(birthdate) - COUNT(DISTINCT birthdate) FROM master_table
UNION ALL
SELECT
    'degree', COUNT(degree) - COUNT(DISTINCT degree) FROM master_table
UNION ALL
SELECT
    'major', COUNT(major) - COUNT(DISTINCT major) FROM master_table
UNION ALL
SELECT
    'institution', COUNT(institution) - COUNT(DISTINCT institution) FROM master_table
UNION ALL
SELECT
    'city', COUNT(city) - COUNT(DISTINCT city) FROM master_table
UNION ALL
SELECT
    'zip', COUNT(zip) - COUNT(DISTINCT zip) FROM master_table
UNION ALL
SELECT
    'state', COUNT(state) - COUNT(DISTINCT state) FROM master_table
UNION ALL
SELECT
    'country', COUNT(country) - COUNT(DISTINCT country) FROM master_table
UNION ALL
SELECT
    'opportunity_id', COUNT(opportunity_id) - COUNT(DISTINCT opportunity_id) FROM master_table
UNION ALL
SELECT
    'opportunity_name', COUNT(opportunity_name) - COUNT(DISTINCT opportunity_name) FROM
master_table
UNION ALL
SELECT
    'category', COUNT(category) - COUNT(DISTINCT category) FROM master_table
UNION ALL
```

```sql
SELECT
   'opportunity_code', COUNT(opportunity_code) - COUNT(DISTINCT opportunity_code) FROM
master_table
UNION ALL
SELECT
   'cohort_id', COUNT(cohort_id) - COUNT(DISTINCT cohort_id) FROM master_table
UNION ALL
SELECT
   'cohort_code', COUNT(cohort_code) - COUNT(DISTINCT cohort_code) FROM master_table
UNION ALL
SELECT
   'assigned_cohort', COUNT(assigned_cohort) - COUNT(DISTINCT assigned_cohort) FROM master_table
UNION ALL
SELECT
   'cohort_size', COUNT(cohort_size) - COUNT(DISTINCT cohort_size) FROM master_table
UNION ALL
SELECT
   'apply_date', COUNT(apply_date) - COUNT(DISTINCT apply_date) FROM master_table
UNION ALL
SELECT
   'status', COUNT(status) - COUNT(DISTINCT status) FROM master_table
UNION ALL
SELECT
   'start_date', COUNT(start_date) - COUNT(DISTINCT start_date) FROM master_table
UNION ALL
SELECT
   'end_date', COUNT(end_date) - COUNT(DISTINCT end_date) FROM master_table
UNION ALL
SELECT
   'usercreatedate', COUNT(usercreatedate) - COUNT(DISTINCT usercreatedate) FROM master_table
UNION ALL
SELECT
   'userlastmodifieddate', COUNT(userlastmodifieddate) - COUNT(DISTINCT userlastmodifieddate) FROM
master_table;

SELECT DISTINCT gender FROM master_table;
```

- **Records after cleaning**

```
-----------------------Record counts beforeafter cleaning the master table (final_final_final_master_table)-------
------------------------
--Count of each column of master table after cleaning
SELECT
   COUNT(user_id) AS user_id_count,
   COUNT(learner_id) AS learner_id_count,
   COUNT(enrollment_id) AS enrollment_id_count,
   COUNT(learneropp_id) AS learneropp_id_count,
   COUNT(email) AS email_count,
   COUNT(gender) AS gender_count,
   COUNT(birthdate) AS birthdate_count,
   COUNT(degree) AS degree_count,
   COUNT(major) AS major_count,
   COUNT(institution) AS institution_count,
   COUNT(city) AS city_count,
   COUNT(zip) AS zip_count,
   COUNT(state) AS state_count,
   COUNT(country) AS country_count,
   COUNT(opportunity_id) AS opportunity_id_count,
   COUNT(opportunity_name) AS opportunity_name_count,
   COUNT(category) AS category_count,
   COUNT(opportunity_code) AS opportunity_code_count,
   COUNT(cohort_id) AS cohort_id_count,
   COUNT(cohort_code) AS cohort_code_count,
   COUNT(assigned_cohort) AS assigned_cohort_count,
   COUNT(cohort_size) AS cohort_size_count,
   COUNT(apply_date) AS apply_date_count,
   COUNT(status) AS status_count,
   COUNT(start_date) AS start_date_count,
   COUNT(end_date) AS end_date_count,
   COUNT(usercreatedate) AS usercreatedate_count,
   COUNT(userlastmodifieddate) AS userlastmodifieddate_count
FROM final_final_master_table;

--Count of null values in each column of master table after cleaning
SELECT
   COUNT(user_id) AS user_id_count,
   COUNT(learner_id) AS learner_id_count,
   COUNT(enrollment_id) AS enrollment_id_count,
   COUNT(learneropp_id) AS learneropp_id_count,
   COUNT(email) AS email_count,
   COUNT(gender) AS gender_count,
   COUNT(birthdate) AS birthdate_count,
   COUNT(degree) AS degree_count,
   COUNT(major) AS major_count,
   COUNT(institution) AS institution_count,
   COUNT(city) AS city_count,
   COUNT(zip) AS zip_count,
   COUNT(state) AS state_count,
   COUNT(country) AS country_count,
   COUNT(opportunity_id) AS opportunity_id_count,
   COUNT(opportunity_name) AS opportunity_name_count,
   COUNT(category) AS category_count,
   COUNT(opportunity_code) AS opportunity_code_count,
   COUNT(cohort_id) AS cohort_id_count,
   COUNT(cohort_code) AS cohort_code_count,
   COUNT(assigned_cohort) AS assigned_cohort_count,
   COUNT(cohort_size) AS cohort_size_count,
   COUNT(apply_date) AS apply_date_count,
   COUNT(status) AS status_count,
   COUNT(start_date) AS start_date_count,
   COUNT(end_date) AS end_date_count,
   COUNT(usercreatedate) AS usercreatedate_count,
   COUNT(userlastmodifieddate) AS userlastmodifieddate_count
FROM final_final_master_table;

--check the count of duplicates in each column of final_final_master_table after cleaning
SELECT
   'user_id' AS column_name, COUNT(user_id) - COUNT(DISTINCT user_id) AS duplicate_count FROM
final_final_master_table
UNION ALL
SELECT
   'learner_id', COUNT(learner_id) - COUNT(DISTINCT learner_id) FROM final_final_master_table
UNION ALL
SELECT
   'enrollment_id', COUNT(enrollment_id) - COUNT(DISTINCT enrollment_id) FROM
final_final_master_table
UNION ALL
SELECT
   'learneropp_id', COUNT(learneropp_id) - COUNT(DISTINCT learneropp_id) FROM
final_final_master_table
UNION ALL
SELECT
   'email', COUNT(email) - COUNT(DISTINCT email) FROM final_final_master_table
UNION ALL
SELECT
   'gender', COUNT(gender) - COUNT(DISTINCT gender) FROM final_final_master_table
UNION ALL
SELECT
   'birthdate', COUNT(birthdate) - COUNT(DISTINCT birthdate) FROM final_final_master_table
UNION ALL
SELECT
   'degree', COUNT(degree) - COUNT(DISTINCT degree) FROM final_final_master_table
UNION ALL
SELECT
```

```
    'major', COUNT(major) - COUNT(DISTINCT major) FROM final_final_master_table
UNION ALL
SELECT
    'institution', COUNT(institution) - COUNT(DISTINCT institution) FROM final_final_master_table
UNION ALL
SELECT
    'city', COUNT(city) - COUNT(DISTINCT city) FROM final_final_master_table
UNION ALL
SELECT
    'zip', COUNT(zip) - COUNT(DISTINCT zip) FROM final_final_master_table
UNION ALL
SELECT
    'state', COUNT(state) - COUNT(DISTINCT state) FROM final_final_master_table
UNION ALL
SELECT
    'country', COUNT(country) - COUNT(DISTINCT country) FROM final_final_master_table
UNION ALL
SELECT
    'opportunity_id', COUNT(opportunity_id) - COUNT(DISTINCT opportunity_id) FROM
final_final_master_table
UNION ALL
SELECT
    'opportunity_name', COUNT(opportunity_name) - COUNT(DISTINCT opportunity_name) FROM
final_final_master_table
UNION ALL
SELECT
    'category', COUNT(category) - COUNT(DISTINCT category) FROM final_final_master_table
UNION ALL
SELECT
    'opportunity_code', COUNT(opportunity_code) - COUNT(DISTINCT opportunity_code) FROM
final_final_master_table
UNION ALL
SELECT
    'cohort_id', COUNT(cohort_id) - COUNT(DISTINCT cohort_id) FROM final_final_master_table
UNION ALL
SELECT
    'cohort_code', COUNT(cohort_code) - COUNT(DISTINCT cohort_code) FROM final_final_master_table
UNION ALL
SELECT
    'assigned_cohort', COUNT(assigned_cohort) - COUNT(DISTINCT assigned_cohort) FROM
final_final_master_table
UNION ALL
SELECT
    'cohort_size', COUNT(cohort_size) - COUNT(DISTINCT cohort_size) FROM final_final_master_table
UNION ALL
SELECT
    'apply_date', COUNT(apply_date) - COUNT(DISTINCT apply_date) FROM final_final_master_table
UNION ALL
SELECT
    'status', COUNT(status) - COUNT(DISTINCT status) FROM final_final_master_table
```

```
UNION ALL
SELECT
    'start_date', COUNT(start_date) - COUNT(DISTINCT start_date) FROM final_final_master_table
UNION ALL
SELECT
    'end_date', COUNT(end_date) - COUNT(DISTINCT end_date) FROM final_final_master_table
UNION ALL
SELECT
    'usercreatedate', COUNT(usercreatedate) - COUNT(DISTINCT usercreatedate) FROM
final_final_master_table
UNION ALL
SELECT
    'userlastmodifieddate', COUNT(userlastmodifieddate) - COUNT(DISTINCT userlastmodifieddate) FROM
final_final_master_table;
```

- **Validation Queries**

```
--------Testing Methodology__________
--Validation queries

--Check for NULL values in each column
SELECT 'user_id' AS column_name, COUNT(*) AS null_count FROM final_master_table WHERE user_id
IS NULL
UNION ALL
SELECT 'learner_id', COUNT(*) FROM final_master_table WHERE learner_id IS NULL
UNION ALL
SELECT 'enrollment_id', COUNT(*) FROM final_master_table WHERE enrollment_id IS NULL
UNION ALL
SELECT 'learneropp_id', COUNT(*) FROM final_master_table WHERE learneropp_id IS NULL
UNION ALL
SELECT 'email', COUNT(*) FROM final_master_table WHERE email IS NULL
UNION ALL
SELECT 'gender', COUNT(*) FROM final_master_table WHERE gender IS NULL
UNION ALL
SELECT 'birthdate', COUNT(*) FROM final_master_table WHERE birthdate IS NULL
UNION ALL
SELECT 'degree', COUNT(*) FROM final_master_table WHERE degree IS NULL
UNION ALL
SELECT 'major', COUNT(*) FROM final_master_table WHERE major IS NULL
UNION ALL
SELECT 'institution', COUNT(*) FROM final_master_table WHERE institution IS NULL
UNION ALL
SELECT 'city', COUNT(*) FROM final_master_table WHERE city IS NULL
UNION ALL
SELECT 'zip', COUNT(*) FROM final_master_table WHERE zip IS NULL
UNION ALL
SELECT 'state', COUNT(*) FROM final_master_table WHERE state IS NULL
UNION ALL
SELECT 'country', COUNT(*) FROM final_master_table WHERE country IS NULL
```

```
UNION ALL
SELECT 'opportunity_id', COUNT(*) FROM final_master_table WHERE opportunity_id IS NULL
UNION ALL
SELECT 'opportunity_name', COUNT(*) FROM final_master_table WHERE opportunity_name IS NULL
UNION ALL
SELECT 'category', COUNT(*) FROM final_master_table WHERE category IS NULL
UNION ALL
SELECT 'opportunity_code', COUNT(*) FROM final_master_table WHERE opportunity_code IS NULL
UNION ALL
SELECT 'cohort_id', COUNT(*) FROM final_master_table WHERE cohort_id IS NULL
UNION ALL
SELECT 'cohort_code', COUNT(*) FROM final_master_table WHERE cohort_code IS NULL
UNION ALL
SELECT 'assigned_cohort', COUNT(*) FROM final_master_table WHERE assigned_cohort IS NULL
UNION ALL
SELECT 'cohort_size', COUNT(*) FROM final_master_table WHERE cohort_size IS NULL
UNION ALL
SELECT 'apply_date', COUNT(*) FROM final_master_table WHERE apply_date IS NULL
UNION ALL
SELECT 'status', COUNT(*) FROM final_master_table WHERE status IS NULL
UNION ALL
SELECT 'start_date', COUNT(*) FROM final_master_table WHERE start_date IS NULL
UNION ALL
SELECT 'end_date', COUNT(*) FROM final_master_table WHERE end_date IS NULL
UNION ALL
SELECT 'usercreatedate', COUNT(*) FROM final_master_table WHERE usercreatedate IS NULL
UNION ALL
SELECT 'userlastmodifieddate', COUNT(*) FROM final_master_table WHERE userlastmodifieddate IS
NULL;

--Check for duplicate values in each column
SELECT 'user_id' AS column_name, COUNT(user_id) AS duplicates
FROM final_master_table GROUP BY user_id HAVING COUNT(user_id) > 1
UNION ALL
SELECT 'learner_id', COUNT(learner_id) FROM final_master_table GROUP BY learner_id HAVING
COUNT(learner_id) > 1
UNION ALL
SELECT 'email', COUNT(email) FROM final_master_table GROUP BY email HAVING COUNT(email) >
1;


--Check for duplicate rows (full row duplicates)
SELECT COUNT(*) - COUNT(DISTINCT *) AS duplicate_rows FROM final_master_table;
--validate email format
SELECT email
FROM final_master_table
WHERE email NOT LIKE '%@%.%' OR email LIKE '% %';


--Check for inconsistent capitalization in institution column
```

```
SELECT DISTINCT institution FROM final_master_table ORDER BY institution;
```

➢ Results:

Test Case	Before Cleaning	After Cleaning	Test Passed?
Total Row Count	184,569	738,276	Yes (Expected Growth)
Count of user_id (Non-Nulls)	184,569	738,276	Yes (No Nulls)
Count of learner_id (Non-Nulls)	184,569	738,276	Yes (No Nulls)
Count of email (Non-Nulls)	184,569	738,276	Yes (No Nulls)
Count of birthdate (Non-Nulls)	141,697	566,788	Yes (More Complete)
Count of zip (Non-Nulls)	141,692	566,768	Yes (More Complete)
Count of institution (Non-Nulls)	184,569	527,616	Partially Improved
Count of opportunity_id (Non-Nulls)	113,327	453,308	Yes (More Complete)
Count of cohort_id (Non-Nulls)	100,200	400,800	Yes (More Complete)
Null Values in birthdate	42,872	171,488	Data Replaced or Expanded?
Null Values in zip	42,877	171,508	Data Replaced or Expanded?
Null Values in opportunity_id	71,242	284,968	Yes (More Complete)
Null Values in cohort_id	84,369	337,476	Yes (More Complete)

➢ Data Analysis:

Link: https://drive.google.com/drive/folders/1FafwcCjeKQaBgQxWi_0I_EYwCNC0bTOb?usp=sharing

➢ Data Cleaning Limitations:

During the data processing and analysis of the master dataset, the following limitations were identified:

1. **Exclusion of Tracking Question Column**
   The tracking question column has been removed from the dataset to streamline the analysis. This decision was made to focus on core user and opportunity-related data while avoiding potential inconsistencies introduced by subjective or non-standardized responses.
2. **Exclusion of Marketing Data**
   Data related to marketing activities has not been considered in this analysis. The exclusion ensures that only direct user and opportunity interactions are evaluated, preventing potential biases introduced by marketing-driven engagements.
3. **Duplicate Entries in User ID**
   A review of the dataset revealed the presence of duplicate values in the user_id column. This indicates that multiple records exist for the same user, which may affect the accuracy of user-based insights. Further investigation and data cleansing may be required to ensure unique user identification.

Thank you