Week 2
Master Table & ETL Process

Submitted By :

Hima Sameera Munjampally
Muhammed Shamil PP
Bashil Shrestha
Johar Ali Shaik
Shivam Sharma
Hetal Tharani

Report Overview

This report focuses on conducting a comprehensive Data Quality Assessment across six raw datasets provided during the internship. It outlines the data structure, examines key relationships, and identifies potential quality issues such as:

• Missing or null values
• Duplicate records
• Inconsistent data formats
• Orphan records (broken relationships)

Each check is executed using SQL queries in PostgreSQL, and the outputs are documented. The findings in this stage will inform decisions in the subsequent ETL planning, guiding how to clean, standardize, and integrate the datasets into a unified master structure.

The report is structured into the following steps:

1. Dataset Structure (columns, data types)
2. Identification of Key Columns
3. Relationship Mapping
4. Dataset Purpose and Contribution
5. Data Quality Checks (with queries)
6. SQL Query Summary
7. Recommended Solutions for ETL
8. Documented Findings Summary

Problem Statement

Although the datasets appear structurally complete at first glance, real-world data is rarely perfect. Duplicates, missing fields, formatting inconsistencies, and improperly linked records can severely degrade the quality of analysis. In our Week 1 exploration, we discovered hints of such issues— especially within enrollment and opportunity mappings. The core problem to solve in Week 2 is:

Objective

- To examine the structure and schema of each raw dataset used in the project.
- To identify key columns that establish relationships across datasets.

- To assess the quality of the data by checking for missing values, duplicates, inconsistencies, and orphan records using SQL.
- To determine which datasets are ready for integration and which need transformations.
- To document findings that will guide the design of the ETL process in Week
  2.
- To ensure all data quality checks are done without modifying any raw dataset.

Building Master Table

Below table gives information of **Primary Keys** from all the 6 datasets to use for the Master table.

Dataset Name	**Primary key Column**
Marketing Campaign Data (marketing_campaign_data.csv)	No primary key column to match with reset of the tables so matching with Date table
Cognito Data (cognito_raw.csv)	Provides user contact info and demographic data
Learner Data (learner_raw.csv)	Learners_id is the primary key in learner_raw table and this table contains academic background or education details in which learner has enrolled
Learner Opportunity Data (learner_opportunity_raw.csv)	Central table for joining all datasets

Cohort Data (cohort_raw.csv)	**Cohort_id** is suitable for the primary key in the cohort table, and the table learner_opportunity contains **cohort code** as **assigned_cohort** that is referenced from the Cohort table itself.
Opportunity Data (opportunity_raw.csv)	opportunity_id is the Primary Key in the opportunity table, and in the learner_opportunity table the column opportunity_id is a Foreign Key that references it
Calendar table (Calculated Date Table)	Contains dates from first cohort start date to cohort last end date.

Why Create a Calculated Date Table

A calculated date table serves as a common time dimension, enabling connections between datasets that lack a direct primary key or shared column — for example, linking the Marketing Campaign table with other datasets.

By standardizing and centralizing date-related information, it allows for more comprehensive analysis, such as:

- Comparing cohort joinings during periods with active marketing campaigns versus periods without.
- Measuring the impact of specific campaigns on learner enrollment.
- Identifying which marketing campaigns have been most effective in driving registrations.

This unified time-based structure ensures consistent filtering, trend analysis, and cross-dataset insights that would otherwise be difficult to achieve.

SQL Codes

For Datasets and Calendar Table

Marketing_Campaign

```
CREATE TABLE Marketing_Campaign(
     "Ad Account Name" TEXT, Campaign name"
     TEXT,
     "Category" TEXT,
"Delivery status" TEXT,
```

```
"Delivery level" TEXT,
        "Reach" INT,
        "Outbound clicks" INT,
        "Outbound type" INT,
        "Result type" TEXT,
        "Results" INT,
"Cost per result" FLOAT,
"Amount spent (AED)" FLOAT,
        "CPC (cost per link click)" FLOAT,
        "dates" DATE
   );
```

Calendar Table

```
CREATE TABLE Calendar(
     "D
     a
     t
     e
     "
     D
     A
     T
     E
     ,
     "
     D
     a
     y
     "
     I
     N
     T
     ,
     "Month" INT,
     "Year" INT,
     "Day Name" TEXT,
     "Month Name" TEXT
);
```

Cognito_Raw

```
create table
     cognito_d
     ata(
     user_id
     varchar(5
     0), email
     varchar(1
```

```
00),
gender
varchar(5
0),
UserCreateDate TIMESTAMP WITH
TIME ZONE,
UserLastModifiedDate
TIMESTAMP WITH TIME ZONE,
birthdate date, city
varchar(100), zip
varchar(100), state
varchar(100)
);
```

Cohort_Raw

```
CREATE TABLE Cohort (
cohort_id VARCHAR(50)
PRIMARY KEY,
cohort_code
VARCHAR(50),
start_date DATE,
end_date DATE,
size INT
);
```

Learner_Raw -> User_Table

```
CREATE TABLE IF NOT EXISTS public.learner_raw
(
    learner_id text COLLATE
pg_catalog."default" NOT NULL,     country
text COLLATE pg_catalog."default" NOT NULL,
degree text COLLATE pg_catalog."default" NOT
NULL,     institution text COLLATE
pg_catalog."default" NOT NULL,     major text
COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT "Learner_pkey1" PRIMARY
KEY (learner_id) )
```

LearnerOpportunity_Raw

```
CREATE TABLE
learner_opportun
ity(
enrollment_id
varchar(100),--
fk learner_id
varchar(100),--
fk
assigned_cohort
varchar(50),--fk
apply_date
timestamp,
status integer
);
```

Opportunity_Raw

```
DROP TABLE if exists
opportunity_data;
CREATE TABLE
opportunity_data(
opportunity_id
varchar(100)PRIMARY
KEY, opportunity_name
varchar(100), category
varchar(50),
opportunity_code
varchar(20),
tracking_questions
text
);
```

Final Master Table Creation

Name: master_table_db

Purpose: one-stop, analysis-ready table that links enrollments to learner profile, cohort, and opportunity.

Row count: 50,764 (from your CSV).

Column Name	Information
enrollment_id	**TEXT** (PK)
learner_id	**TEXT** → FK to "User"(learner_id)
assigned_cohort	**TEXT** → FK to Cohort(cohort_code)
apply_date	**TIMESTAMP WITH TIME ZONE**
status	**INT**
country	**TEXT**
degree	**TEXT**
institution	**TEXT**
major	**TEXT**
user_id	**TEXT** → FK to cognito_data(user_id)
email	**TEXT**
gender	**TEXT**
usercreatedate	**TIMESTAMP WITH TIME ZONE**
userlastmodifieddate	**TIMESTAMP WITH TIME ZONE**

birthdate	**DATE**
city	**TEXT**
zip	**TEXT**
cohort_code	**TEXT**
start_date	**DATE**
end_date	**DATE**
size	**INT**
opportunity_id	**TEXT** → FK to opportunity_data(opportunity_id
opportunity_name	**TEXT**
category	**TEXT**
opportunity_code	**TEXT**
rn	**INT** (should always be 1 in the final table)

Constraints

- PRIMARY KEY (enrollment_id)
- FKs as noted above
- CHECK (assigned_cohort = cohort_code) (keeps the time-dimension mapping consistent)

Indexes (performance)

- idx_master_apply_date on (apply_date)
- idx_master_email on (email)

- idx_master_assigned_cohort on (assigned_cohort)
- idx_master_opportunity_id on (opportunity_id)
- idx_master_learner_id on (learner_id)

Table Creation Query

```
CREATE TABLE
master_table_raw AS
SELECT
 lo.enrollment_id, lo.learner_id,
lo.assigned_cohort, lo.apply_date, lo.status,
 l.country, l.degree, l.institution, l.major,
 c.user_id, c.email, c.gender, c.usercreatedate,
c.userlastmodifieddate, c.birthdate, c.city, c.zip,
co.cohort_code, co.start_date, co.end_date,
co.size,
 o.opportunity_id, o.opportunity_name, o.category,
o.opportunity_code
FROM learner_opportunity lo
LEFT JOIN user_data l ON lo.enrollment_id =
l.learner_id
LEFT JOIN cognito_data c ON lo.enrollment_id =
c.user_id
LEFT JOIN cohort_data co ON lo.assigned_cohort =
co.cohort_code
LEFT JOIN opportunity_data o ON lo.learner_id =
o.opportunity_id;

select
enrollment_id,count(*) as
duplicate from
```

```
master_table_raw group by
1 having count(*)>1

create table final_ as(
SELECT *
FROM (
 SELECT *, ROW_NUMBER() OVER (PARTITION BY
enrollment_id ORDER BY enrollment_id) AS rn
 FROM master_table_raw
) t
WHERE rn = 1)

SELECT* from final where   ---final
create table final__master_table as(
select * from final_ where apply_date
is not null and email is not null and
enrollment_id is not null
)  create table
master_table_db
as( select* from
final__master_tab
le where
assigned_cohort=c
ohort_code)

select*from master_table_db
```

Please find the Master Table here: [master_table_db.csv]

Stored Procedure Query

```
DROP TABLE IF EXISTS master_table_db CASCADE;
DROP TABLE IF EXISTS final__master_table CASCADE;
DROP TABLE IF EXISTS final_ CASCADE;
DROP TABLE IF EXISTS master_table_raw CASCADE;

CREATE TABLE
master_table_ra
w AS   SELECT
lo.enrollment_i
d,
lo.learner_id,
lo.assigned_coh
ort,
lo.apply_date,
lo.status,

      l.country,
      l.degree,
```

```
      l.institution,
      l.major,

      c.user_id,
      c.email,
      c.gender,
      c.usercreatedate,
      c.userlastmodifieddate,
      c.birthdate,
      c.city,

c.zi
p,
co.c
ohor
t_co
de,
co.s
tart
_dat
e,
co.e
nd_d
ate,
co.s
ize,

      o.opportunity_id,
      o.opportunity_name,
      o.category,
      o.opportunity_code
  FROM learner_opportunity   lo
  LEFT JOIN user_data        l  ON lo.learner_id
= l.learner_id
  LEFT JOIN cognito_data     c  ON lo.learner_id
= c.user_id   LEFT JOIN cohort_data      co ON
lo.assigned_cohort = co.cohort_code
  LEFT JOIN opportunity_data o  ON
lo.opportunity_id  = o.opportunity_id;


-------------------------------------------------
---------------
-
  -- 2) Dedupe on enrollment_id (keep most recently
modified/created)

-------------------------------------------------
---------------
```

```
-
  CREATE TABLE final_ AS
  SELECT *
  FROM (

S
E
L
E
C
T

r
.
*
,
      ROW_NUMBER() OVER (
        PARTITION BY r.enrollment_id
        ORDER BY COALESCE(r.userlastmodifieddate,
r.usercreatedate) DESC NULLS LAST
      ) AS rn
    FROM master_table_raw r
  ) t
  WHERE rn = 1;


-------------------------------------------------
---------------
-
  -- 3) Business-rule filters (non-null key fields)

-------------------------------------------------
---------------
-
  CREATE TABLE final__master_table AS
  SELECT *
  FROM final_
  WHERE apply_date    IS NOT NULL
    AND email         IS NOT NULL
    AND enrollment_id IS NOT NULL;


-------------------------------------------------
---------------
-
  -- 4) Cohort integrity check and final table
  --    (only keep rows where assigned_cohort
matches resolved cohort_code)
```

```
-----------------------------------------------------
---------------
-
  CREATE TABLE master_table_db AS
  SELECT *
  FROM final__master_table
  WHERE assigned_cohort = cohort_code;


-----------------------------------------------------
---------------
-
  -- 5) Indexes for performance (match your usage
patterns)

-----------------------------------------------------
---------------
-
  CREATE INDEX IF NOT EXISTS idx_master_apply_date
ON master_table_db(apply_date);
  CREATE INDEX IF NOT EXISTS idx_master_email
ON master_table_db(email);
  CREATE INDEX IF NOT EXISTS
idx_master_assigned_cohort  ON
master_table_db(assigned_cohort);
  CREATE INDEX IF NOT EXISTS
idx_master_opportunity_id   ON
master_table_db(opportunity_id);
  CREATE INDEX IF NOT EXISTS idx_master_learner_id
ON master_table_db(learner_id);
END;
```

Data Quality Report

Marketing_Campaign:

Issues Detected

- **Blank Campaign Names**

Two records are missing values in the *Campaign Name* field while still containing data in all other columns. Since a campaign name is a critical identifier, these rows cannot be reliably used in analysis and were removed. ● **Results Column – Potential Outliers**

At first glance, the *Results* column contains unusually high values. However, when cross-checked against the *Reach* metric, these values align with expected performance. This occurs when the *Result Type* is **"Reach"** or **"ThruPlay"**, confirming the values are valid.

Cleaning Logic Applied

- Removed two rows with blank *Campaign Name* values.

Testing Methodology

- Initial row count: 141
- Final row count after cleaning: 139
- Manual cross-check of *Results* vs *Reach* confirmed consistency in cases flagged as potential outliers.

Cognito_Raw:

Issues Detected

- No missing values found
- No duplicate rows

Cleaning Logic Applied

- This data set is data is accurate

Testing Methodology

- Row count is 129178

Cohort_Raw:

Issues Detected

- No missing values were found across all columns.
- No duplicate rows detected.
- Data types appear consistent:

  - cohort_id, cohort_code → identifiers (string)
  - start_date, end_date → stored as text but convertible to DATE
  - size → numeric (integer)

Cleaning Logic Applied

Verified absence of nulls and duplicates.

Testing Methodology

- Initial row count: 639
- Final row count after cleaning: 639 (no rows removed).
- Null check confirmed 0 missing values across all fields.
- Duplicate check confirmed 0 duplicates.

Learner_Raw(User_Raw):

Issues Detected

- There are several missing values detected and data inconsistency detected in the table.

Cleaning Logic Applied

- Handled missing values and data validation

Testing Methodology

- Initial row count: 129,259 Rows
- Final row count: 76,562 Rows
- No duplicates found in raw data 0 duplicates confirmed

LearnerOpportunity_Raw:

Issues Detected

- There are several duplicates detected in the column of enrollment_id

Cleaning Logic Applied

- Verified absence of nulls and duplicates.

Testing Methodology

- This is a parent table
- 2 null rows found in the column of apply_date

Opportunity_Raw:

Issues Detected

- Most of values are null in the column of tracking_question

Cleaning Logic Applied

- Verified absence of nulls and duplicates.

Testing Methodology

- Initial row count: 187
- Final row count after cleaning: 187(no rows removed).
- Duplicate check confirmed 0 duplicates.

Conclusion

Objective

During Week 2 of the Data Visualization Internship Project, our main objective was to design, build, and validate a centralized Master Table that integrates data from multiple raw datasets. This consolidated structure was intended to serve as the foundation for all future analysis and visualization efforts.

We began by outlining the structure of the Master Table, identifying essential fields, and defining relationships across datasets. To ensure reliability, primary and foreign key constraints were carefully planned to enable consistent linkage between records.

ETL Process

A complete ETL (Extract, Transform, Load) pipeline was implemented:

- Extraction: Data was sourced from cleaned raw tables, including *learneropp1, learner1, cognito2, cohort1, and opportunity1*.

- Transformation: The datasets were joined into a staging table (master_table_raw) using logical key mappings. We applied transformations such as:
    - Removing duplicate records based on enrollment_id.
    - Handling missing or null values in critical fields like email, learner_id, and assigned_cohort.
    - Standardizing inconsistent formats (e.g., dates, zip codes).
- Loading: The cleaned and filtered records were inserted into the final table (master_table_db), which is now optimized for performance and downstream analysis.

Data Validation

To confirm the accuracy of the final dataset, multiple validation steps were carried out:

- Record Count Comparison: Approximately 50% of records were dropped during cleaning. This reduction was expected due to duplicates, null values, and foreign key mismatches.
- Completeness Checks: Critical fields were validated for missing values, and the cleaned dataset showed no gaps in these fields.

- Referential Integrity: Foreign key relationships were verified. For instance, the assigned_cohort field was validated against the Cohort table, leaving only a small number of unmatched (orphan) records, which were documented for reference.

Outcome

By the end of Week 2, we successfully delivered a cleaned and reliable Master Table (master_table_db) that ensures both data consistency and integrity. This final table:

- Provides a robust structure for creating joins, filters, and aggregations.
- Reduces noise from duplicate or incomplete records.
- Establishes a trustworthy foundation for all upcoming visualizations and dashboards.

Next Steps

With the Master Table in place, the project is now positioned to transition into Week 3, where the focus will shift to building interactive dashboards. These dashboards will leverage the centralized dataset to deliver actionable insights and meaningful data stories.

Opportunity data

Learning or course opportunities linked to learners.

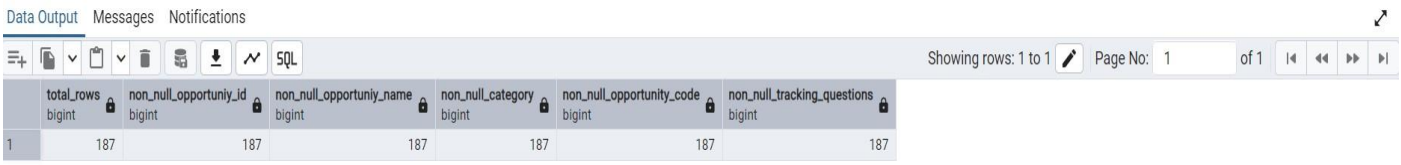
Column Name	Data Type	Description
opportunity_id	INTEGER	Primary key
opportunity_name	TEXT	Name of the opportunity/course
category	TEXT	Category of the opportunity
opportunity_code	TEXT	Internal code
tracking_questions	JSON	object with progress/completion

Data Quality Checks

We performed key data validation checks on each dataset to ensure integrity before ETL. These include checking for missing values, duplicates, inconsistent formats, and orphan records.

Check for Missing Values

```
SELECT

COUNT(*) AS total_rows,

COUNT(opportunity_id) AS non_null_opportuniy_id,

COUNT(opportunity_name) AS non_null_opportuniy_name,

COUNT(category) AS non_null_category,

COUNT(opportunity_code) AS non_null_opportunity_code,

COUNT(tracking_questions) AS

non_null_tracking_questions  from opportunity_data;
```



Check for Duplicate Values

```
select opportunity_code,Count(*)

from opportunity_data group by

1

HAVING count(*)>1
```

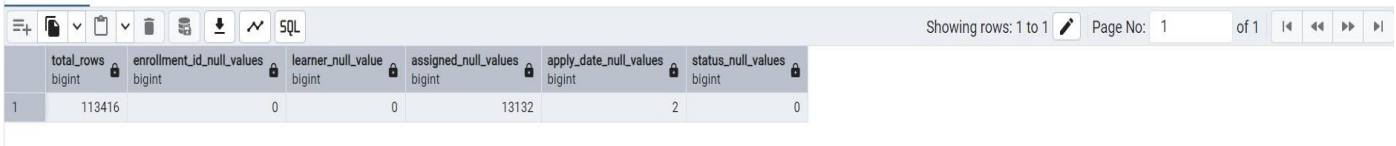
Learner Opportunity data

This table captures learner enrollment data linked to opportunities

Column Name	Data Type	Description
enrollment_id	TEXT	Unique ID for each enrolment
learner_id	TEXT	Learner ID (links to opportunity_id)
assigned_cohort	TEXT	Foreign key linking to cohort_code
apply_date	TIMESTAMP	Timestamp of learner application
status	INTEGER	Numeric status code of application

Check Missing Values

```
select Count(*)as total_rows, sum(case when enrollment_id is null then 1 else 0
end)as enrollment_id_null_values, sum(case when learner_id is null then 1 else 0
end)as learner_null_value, sum(case when assigned_cohort is null then 1 else 0 end
)as assigned_null_values, sum(case when apply_date is null then 1 else 0 end)as
apply_date_null_values, sum(case when status is null then 1 else 0 end )as
status_null_values from learner_opportunity
```

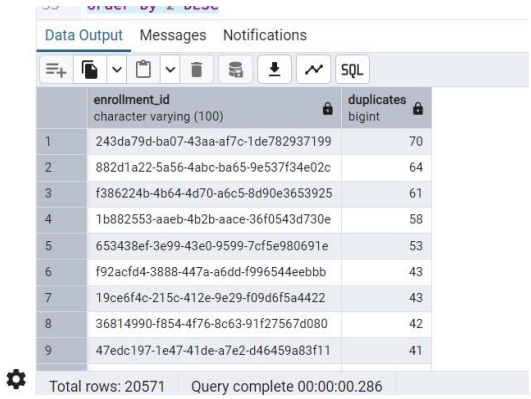


Showing rows: 1 to 1 Page No: 1 of 1

	total_rows bigint	enrollment_id_null_values bigint	learner_null_value bigint	assigned_null_values bigint	apply_date_null_values bigint	status_null_values bigint
1	113416	0	0	13132	2	0

Check for Duplicates

```
select enrollment_id,count(*) as Duplicates
from learner_opportunity group by 1
having count(*)>1
order by 2 DESC
```



Data Output Messages Notifications

	enrollment_id character varying (100)	duplicates bigint
1	243da79d-ba07-43aa-af7c-1de782937199	70
2	882d1a22-5a56-4abc-ba65-9e537f34e02c	64
3	f386224b-4b64-4d70-a6c5-8d90e3653925	61
4	1b882553-aaeb-4b2b-aace-36f0543d730e	58
5	653438ef-3e99-43e0-9599-7cf5e980691e	53
6	f92acfd4-3888-447a-a6dd-f996544eebbb	43
7	19ce6f4c-215c-412e-9e29-f09d6f5a4422	43
8	36814990-f854-4f76-8c63-91f27567d080	42
9	47edc197-1e47-41de-a7e2-d46459a83f11	41

Total rows: 20571 Query complete 00:00:00.286

Relationships Between Dataset

learneropp1.learner_id --> opportunity1.opportunity_id **SQL**

Query for Extraction

DROP TABLE IF EXISTS master_table

CREATE TABLE master_table as SELECT

lp.enrollment_id,lp.learner_id,lp.assigned_cohort,lp.apply_date,lp.status,

o.opportunity_id,o.opportunity_name,o.category,o.opportunity_code,o.tracking_questions

FROM learner_opportunity lp left join opportunity_data o on

o.opportunity_id=lp.learner_id; **Count of row after join the table** select count(*)from

master_table

Create Initial Clean Table from Raw

Purpose:

To begin the cleaning process, we first take a copy of the raw master table (master_table_raw) into a working table so the original remains untouched.

Why:

This maintains data traceability and ensures that we can always refer back to the raw source if needed.

SQL Query:

DROP THE TABLE IF EXISTS cleaned_master_data

CREATE TABLE cleaned_master_data AS

SELECT * FROM master_table

Remove Duplicate Records Purpose:

To eliminate duplicate entries based on the primary identifier enrollment_id.

Why: Duplicates can distort analysis and cause inflated counts.

We assume one record per learner's enrollment is Expected

How:

We use ROW_NUMBER() to rank duplicates and keep only the first occurrence.

SQL Query:

```
DROP TABLE IF EXISTS non_duplicates

CREATE TABLE non_duplicates AS

SELECT * FROM

(SELECT *,ROW_NUMBER()OVER (PARTITION BY enrollment_id order by enrollment_id)as rn

from cleaned_master_table)t  where rn=1
```

After remove duplicates :

```
SELECT COUNT(*)FROM non_duplicates
```

Remove Rows with NULL in Critical Fields

Purpose:

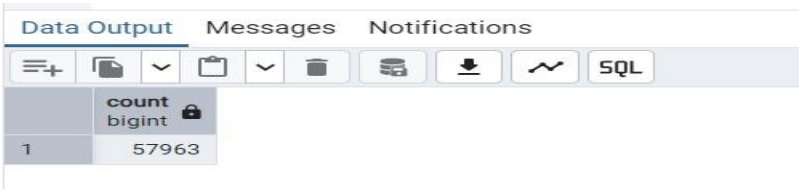
To remove records that are missing key information,

especially:

- enrollment_id (primary key)
- email (for user contact)

SQL Query:

```
DROP TABLE IF EXISTS master_table_non_null;

CREATE TABLE master_table_non_null as

SELECT *FROM non_duplicates

WHERE enrollment_id is not null and

apply_date is not null;

SELECT COUNT(*) FROM master_table_non_null;
```



Standardize and Fix Date Formats

Purpose:

To ensure all date fields are stored in proper formats, particularly:

- Apply_date

Why:

Inconsistent date formats or strings (e.g., 7/5/2001 or NULL) can break filters, sorting, and timebased analysis.

Select applydate

- Keep apply_date as is if it's already in correct format.

Final Cleaned Table

Purpose:

To consolidate all previous steps into the final, cleaned table ready for analysis.

Why: This is the table that will be used for **visualization, reporting, and further validation**.

SQL Query

```
DROP TABLE IF EXISTS final_master_table

CREATE TABLE final_master_table AS SELECT

* FROM master_table_non_null
```

VALIDATION & REFINEMENT

Observation:

There is a clear difference between the number of records in the raw table and the cleaned master table:

• Raw: 113,416 rows

• Cleaned: 57,963 rows

• Difference: 57453 rows removed

Cohort Data

Introduction

The **Cohort dataset** serves as a foundational reference table within the Master Table schema. Each cohort represents a structured learning group, defined by its unique identifier, associated code, start and end dates, and total size (number of learners enrolled). This dataset provides temporal boundaries for learning programs and is critical for linking learners and opportunities to specific training periods.

By integrating the Cohort dataset with learner opportunity data, the Master Table can accurately reflect **which learners participated in which programs, when they started, when they ended, and the cohort capacity during that period**. This ensures analytical processes, such as performance tracking, attrition analysis, and scheduling optimization, are supported with accurate contextual data.

Key attributes of the Cohort dataset include:

- **cohort_id** → Primary key; uniquely identifies each cohort.
- **cohort_code** → An alternate identifier used for joining with other datasets.
- **start_date / end_date** → Define the duration of the cohort; critical for linking with the calendar table.
- **size** → Represents the number of learners assigned, useful for capacity and utilization analysis.

The dataset has undergone cleaning and validation checks to ensure data consistency:

- **No missing values** across all columns.
- **No duplicate rows**.
- **Unique identifiers** for both `cohort_id` and `cohort_code`.

Column Name	Data Type	Description
cohort_id	text	Primary Key
cohort_code	text	Unique code assigned to the cohort
Start_date	date	Start date for the cohort
end_date	date	End date for the cohort
size	integer	Number of learners in the cohort

Table 1 table description for the cohort table.

Data quality checks

Validation checks were performed on the Cohort dataset to ensure integrity before ETL. This included checking for missing values, duplicates, inconsistent formats, and invalid date ranges.

Checking missing values

```
SELECT
    COUNT(*) AS total_rows,
    SUM(CASE WHEN cohort_id IS NULL THEN 1 ELSE 0 END) AS
cohort_id_nulls,
    SUM(CASE WHEN cohort_code IS NULL THEN 1 ELSE 0 END) AS
cohort_code_nulls,
    SUM(CASE WHEN start_date IS NULL THEN 1 ELSE 0 END) AS
start_date_nulls,
    SUM(CASE WHEN end_date IS NULL THEN 1 ELSE 0 END) AS end_date_nulls,
    SUM(CASE WHEN size IS NULL THEN 1 ELSE 0 END) AS size_nulls
FROM cohort;
```

```
cohort_db=# SELECT
cohort_db-#     COUNT(*) AS total_rows,
cohort_db-#     SUM(CASE WHEN cohort_id IS NULL THEN 1 ELSE 0 END) AS cohort_id_nulls,
cohort_db-#     SUM(CASE WHEN cohort_code IS NULL THEN 1 ELSE 0 END) AS cohort_code_nulls,
cohort_db-#     SUM(CASE WHEN start_date IS NULL THEN 1 ELSE 0 END) AS start_date_nulls,
cohort_db-#     SUM(CASE WHEN end_date IS NULL THEN 1 ELSE 0 END) AS end_date_nulls,
cohort_db-#     SUM(CASE WHEN size IS NULL THEN 1 ELSE 0 END) AS size_nulls
cohort_db-# FROM cohort;
 total_rows | cohort_id_nulls | cohort_code_nulls | start_date_nulls | end_date_nulls | size_nulls
------------+-----------------+-------------------+------------------+----------------+------------
        639 |               0 |                 0 |                0 |              0 |          0
(1 row)
```

Figure 1 testing the query in the psql for missing values

Checking for duplicate values

```
SELECT cohort_id, COUNT(*) AS duplicates

FROM cohort

GROUP BY cohort_id

HAVING COUNT(*) > 1;
```

```
cohort_db=# SELECT cohort_id, COUNT(*) AS duplicates
cohort_db-# FROM cohort
cohort_db-# GROUP BY cohort_id
cohort_db-# HAVING COUNT(*) > 1;
 cohort_id | duplicates
-----------+------------
(0 rows)
```

Figure 2 testing for duplicate values for cohort_id

```
SELECT cohort_code, COUNT(*) AS duplicates

FROM cohort

GROUP BY cohort_code HAVING

COUNT(*) > 1;
```

```
cohort_db=# SELECT cohort_code, COUNT(*) AS duplicates
cohort_db-# FROM cohort
cohort_db-# GROUP BY cohort_code
cohort_db-# HAVING COUNT(*) > 1;
 cohort_code | duplicates
-------------+------------
(0 rows)
```

Figure 3 testing for duplicate values for cohort_code

Learner Opportunity Data

```
CREATE TABLE learner_opportunity (
enrollment_id TEXT PRIMARY KEY,
learner_id TEXT,
assigned_cohort TEXT     apply_date
TIMESTAMP,     status INTEGER
);
```

Column Name`	Data Type	Description
Enrollment id	Text	Unique ID
Learner ID	Text	Learner ID links to opportunity id
Assigned_id	Text	Foreign key referenced from cohort id
Apply_date	Date	Timestamp of learner application
Status	Integer	Numeric status code of the application

Table 2 table discription of learner opportunity

Checking the missing values: -

```
SELECT
COUNT(*) AS total_rows,
SUM(CASE WHEN enrollment_id IS NULL THEN 1 ELSE 0 END) AS
enrollment_id_null_values,
SUM(CASE WHEN learner_id IS NULL THEN 1 ELSE 0 END) AS
learner_id_null_values,
SUM(CASE WHEN assigned_cohort IS NULL THEN 1 ELSE 0 END) AS
assigned_cohort_null_values,
SUM(CASE WHEN apply_date IS NULL THEN 1 ELSE 0 END) AS
apply_date_null_values,
SUM(CASE WHEN status IS NULL THEN 1 ELSE 0 END) AS status_null_values
FROM learner_opportunity;
```

	total_rows bigint	enrollment_id_null_values bigint	learner_id_null_values bigint	assigned_cohort_null_values bigint	apply_date_null_values bigint	status_null_values bigint
1	57966	0	0	6623	3	1

Figure 4 checking missing values in the learner opportunity table

Checking for the duplicate values: -

```
 SELECT
enrollment_id,
COUNT(*) AS duplicate_count
FROM learner_opportunity
GROUP BY enrollment_id
HAVING COUNT(*) > 1
ORDER BY duplicate_count DESC;
```

```
cohort_db=# SELECT
cohort_db-#     enrollment_id,
cohort_db-#     COUNT(*) AS duplicate_count
cohort_db-# FROM learner_opportunity
cohort_db-# GROUP BY enrollment_id
cohort_db-# HAVING COUNT(*) > 1
cohort_db-# ORDER BY duplicate_count DESC;
 enrollment_id | duplicate_count
---------------+-----------------
(0 rows)
```

Figure 5 checking duplicate values for the learner opportunity table

Relationship between Cohort and Learner Opportunity table

learner_opportunity.assigned_cohort → cohort.cohort_code

```
CREATE TABLE master_table AS
SELECT
lo.enrollment_id,
lo.learner_id,
lo.assigned_cohort,
```

```
lo.apply_date,
lo.status,
    c.cohort_id,
    c.cohort_code,
    c.start_date,
    c.end_date,
    c.size
FROM learner_opportunity lo
LEFT JOIN cohort c
    ON lo.assigned_cohort = c.cohort_id;
```

```
cohort_db=# SELECT COUNT(*) FROM master_table;
 count
-------
 57966
(1 row)
```

Figure 6 checking the row count for the master table

Creating initial clean copy

```
DROP TABLE IF EXISTS cohort_cleaned_stage;
CREATE TABLE cohort_cleaned_stage AS
SELECT * FROM master_table;
```

Purpose: Create a working copy for cleaning.

Why: Keeps raw joined data (master_table) untouched for traceability and rollback if needed.

Removing Duplicates

```
DROP TABLE IF EXISTS cohort_no_duplicates;

CREATE TABLE cohort_no_duplicates AS
SELECT *
```

```
FROM (
    SELECT *,
           ROW_NUMBER() OVER (PARTITION BY enrollment_id ORDER BY
enrollment_id) AS rn
    FROM cohort_cleaned_stage
) t
WHERE rn = 1;
```

Purpose: Eliminate duplicate learner enrollments.

Why: Each learner should only have one unique enrollment record. Duplicates can inflate metrics and distort insights.

```
cohort_db=# Select count(*) from cohort_no_duplicates;
 count
-------
 57966
(1 row)
```

Remove Rows with NULL in Critical Fields

```
DROP TABLE IF EXISTS cohort_non_null;


CREATE TABLE cohort_non_null AS
SELECT *
FROM cohort_no_duplicates
WHERE enrollment_id IS NOT NULL
AND assigned_cohort IS NOT NULL
AND apply_date IS NOT NULL;
```

Purpose: Ensure essential fields are present.

Why: Missing enrollment_id, assigned_cohort, or apply_date makes records unusable for analysis, as they break links or timelines.

Final clean table: -

```
DROP TABLE IF EXISTS final_cohort_master;

CREATE TABLE final_cohort_master AS

SELECT *

FROM cohort_date_standardized;
```

Purpose: Consolidate all cleaning steps into the final dataset.
Why: Provides a reliable, standardized, and clean table for reporting, visualization, and further validation.

Final Validation: -

- Raw: **57,966** rows
- After Deduplication: **57,966** rows
- After NULL removal: **51,341** rows
- Final: **51,341** rows
- Difference from raw: **6,625 rows removed**

Observations

- No duplicates were found (deduplication did not reduce row count).
- A total of 6,625 rows were dropped due to missing critical fields (e.g., enrollment_id, apply_date).
- Final dataset is clean, unique, and complete for analysis.

Conclusion: -

The project integrates Cohort and Learner Opportunity data into a single Master Table to facilitate appropriate learner participation and program performance analysis.

Cohort Table: Sets up training cohorts by IDs, codes, dates, and size. It had undergone data quality checks no duplicates, no missing values.

Learner Opportunity Table: Tracks learner enrollments with associated cohorts. Validation found missing values in key fields and possible duplicates.

Master Table Build: Both datasets were joined on assigned_cohort → cohort_id. Stepped cleaning process was implemented.

Cleaning Steps:

- Created working copy for traceability.
- Removed duplicates (none).
- Dropped NULL rows in critical columns (enrollment_id, assigned_cohort, apply_date).
- Normalized dates and rolled up into final_cohort_master.

Validation Results:

Raw rows: 57,966

After cleaning: 51,341

6,625 rows dropped for missing critical fields.

Observations:

- Data set is now clean, unique, and ready for analysis.
- Deduplication had no effect (no duplicates).
- Main problem was missing data in critical fields.

Learner_Raw Data

Learner education schema(column name, datatype) details

Column Name	Data Type	Description
learner_id	TEXT	Primary key – Learners unique id
country	TEXT	Country they resides
degree	TEXT	Degree they are pursuing
institution	TEXT	Institution they are attending
major	TEXT	Major they are enrolled

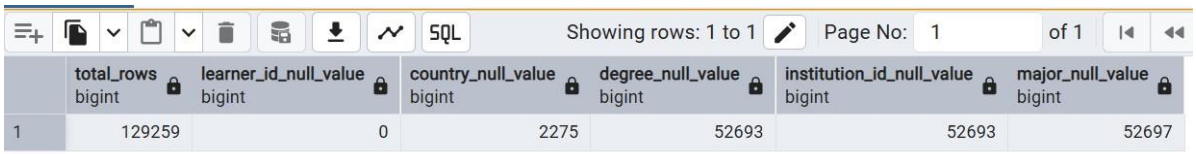
Data Quality Checks

Before deep diving into ETL we perform data quality checks by finding missing values, duplicate records, and inconsistent format.

Missing Values

Missing values of dataset are verified here for handling.

```
-- Checking Missing Values
SELECT
COUNT(*) AS total_rows,
SUM(CASE WHEN learner_id ILIKE 'null' THEN 1 ELSE 0 END) AS learner_id_null_value,
SUM(CASE WHEN country ILIKE 'null' THEN 1 ELSE 0 END) AS country_null_value,
SUM(CASE WHEN degree ILIKE 'null' THEN 1 ELSE 0 END) AS degree_null_value,
SUM(CASE WHEN institution ILIKE 'null' THEN 1 ELSE 0 END) AS institution_id_null_value,
SUM(CASE WHEN major ILIKE 'null' THEN 1 ELSE 0 END) AS major_null_value
FROM learner_raw;
```

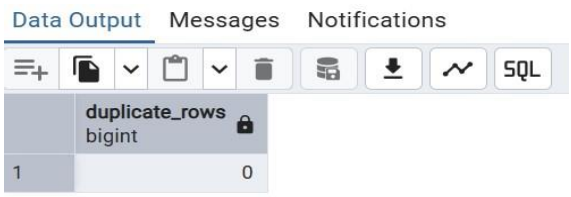


	total_rows bigint	learner_id_null_value bigint	country_null_value bigint	degree_null_value bigint	institution_id_null_value bigint	major_null_value bigint
1	129259	0	2275	52693	52693	52697

Duplicate Records

We check if there exist any duplicate records in the database to avoid redundancy.

```sql
-- checking duplicate values by learner_id
select learner_id,count(*)
from learner_raw
group by learner_id
having count(*)>1

--Checking duplicate records
select count(*) duplicate_rows
from (select
row_number() over(partition by learner_id,country,degree,institution,major) as rn
from learner_raw) t
where rn>1
```



Checking Data Validation

We check the data validations and format in each column and refine it for better performance.

```sql
-- Checking Inconsistencies
select
distinct learner_id
from learner_raw order by learner_id


select
distinct country
from learner_raw order by country

select
distinct degree
from learner_raw order by degree

select
distinct institution
from learner_raw order by institution

select
distinct major
from learner_raw order by major
```

	degree text
1	Graduate Student
2	High School Student
3	Not in Education
4	NULL
5	Other Professional
6	Parent of Student
7	Teacher/Educator
8	Undergraduate Student

Data Output Messages Notifications

Showing rows: 1 to 1000 Page No: 1

	institution text
61	9th
62	a
63	A
64	A-E FUNAI
65	A a u
66	A and A co
67	A B M College
68	A B M COLLEGE JSR
69	A C Patil Colleg Of Engineering
70	A D Patel Institute of Technology
71	A e kalsekar college of commerce and management
72	A levels
73	A Mother%27s Touch Preschool and Daycare
74	A p j Abdul kalam paramedical College
75	A P J Abdul Kalam Technologcal University, Kerala
76	A P J Abdul Kalam Technological University
77	A P Shah Institute of technology
78	A U
79	A U D Primary School

Data Output Messages Notifications

Showing rows: 1 to 1000 Page No:

	major text
723	BBA in Business Analytics
724	BBA LLB
725	BBA LLB HONSCORPORATE LAW
726	Bbaib
727	Bbd
728	BBIS
729	Bbit
730	BBIT
731	Bbs
732	BBS
733	Bca
734	BCA
735	BCA with Analytics
736	BCM
737	Bcom
738	BCOM
739	Bcom Computers
740	Bcom Hon

ETL & Cleaning Steps

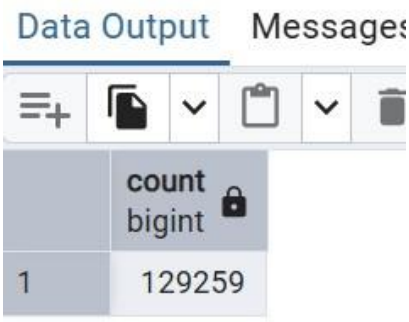
Creation of Master Table

To perform ETL operations we create a master table and load data in it to handle missing values, inconsistencies, and data formats.

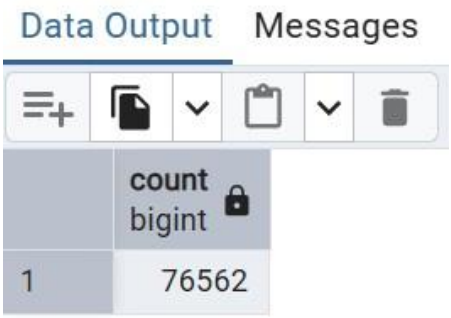
Count of rows

Before performing ETL we check the total number of rows of master_raw table and master_clean table.

```
--count of rows from raw table
select count(*) from master
```




```
--count of rows from clean table
select count(*) from master_clean
```



Handling Missing Values

To avoid data inaccuracy and improve performance during analysis and to provide accurate results while data visualization.

```
--select non missing values
create or replace procedure non_missing_values()
Language plpgsql
as $$
Begin
Execute 'DROP TABLE IF EXISTS master_clean';
Execute 'create table master_clean as
select *
from master_raw where learner_id not ILIKE ''null'' and
country not ILIKE ''null'' and degree not ILIKE ''null'' and
institution not ILIKE ''null'' and major not ILIKE ''null''
';
End;
$$;

call non_missing_values();
```

Duplicate Values

There are no duplicate records in dataset.

Data Validation

Data validation is important during analysis and ETL process to maintain only necessary data avoiding irregularities.

```
-- data validation
create or replace procedure data_validation()
Language plpgsql
as $$
Begin
Execute 'update master_clean
set major=''Accountancy and Bussiness Management'' where lower(major) like ''accountancy%''
';
Execute 'update master_clean
set major=''Accounting'' where lower(major) = ''accounting'' or lower(major) = ''accountant''
';
Execute 'update master_clean
set major=''Finance'' where lower(major) like ''finance''
';
End;
$$;

call data_validation()
```

Data Output | Messages | Notifications

Showing rows: 1 to 1000 | Page

	major text
20	Accountancy and Bussiness Management
21	Accounting
22	Accounting and Economics
23	Accounting and finance
24	Accounting and Finance
25	Accounting and Information Systems
26	Accounting and Information Technology

Validation & Refinement

Observation:

We have observed that there is a clear difference between number of records in raw_table and master_clean table:

Raw Table: 129,259 Rows

Master Clean Table: 76562 Rows

Difference: 52697 Rows

Marketing Campaign Data All Accounts (2023-2024):

Dataset Overview:

Column Name	Overview
Ad Account Name	College account names managing the ads (e.g., SLU, Brand Awa, RIT)
Campaign name	Name of the marketing campaign (e.g., Digital Marketing, Data Analytics, Competitions)

Delivery status	Status of the ad campaign (e.g., completed, inactive)
Delivery level	Type of ad delivery unit (campaign level)
Reach	Number of unique users who saw the ad
Outbound clicks	Number of clicks leading users away from the platform
Outbound type	Classification of outbound actions (numeric representation in dataset)
Result type	Type of primary results achieved (e.g., Website applications, Reach, ThruPlay)
Results	Count of the achieved result type
Cost per result	Average cost incurred per achieved result
Amount spent (AED)	Total money spent on the campaign in AED currency
CPC (cost per link click)	Average cost per individual link clicks
dates	Campaign reporting date in MM/DD/YYYY format

Dataset Statistics:

Column	Data Type	Non-Null count	Unique Values
Ad Account Name	TEXT	0	0
Campaign name	TEXT	2	0
Delivery status	TEXT	0	0
Delivery level	TEXT	0	0
Reach	TEXT	0	0
Outbound clicks	NUMERIC	2	0
Outbound type	NUMERIC	2	0
Result type	TEXT	0	0
Results	NUMERIC	0	0
Cost per result	NUMERIC	0	0
Amount spent (AED)	NUMERIC	0	0
CPC (cost per link click)	NUMERIC	2	0
dates	DATE	0	0

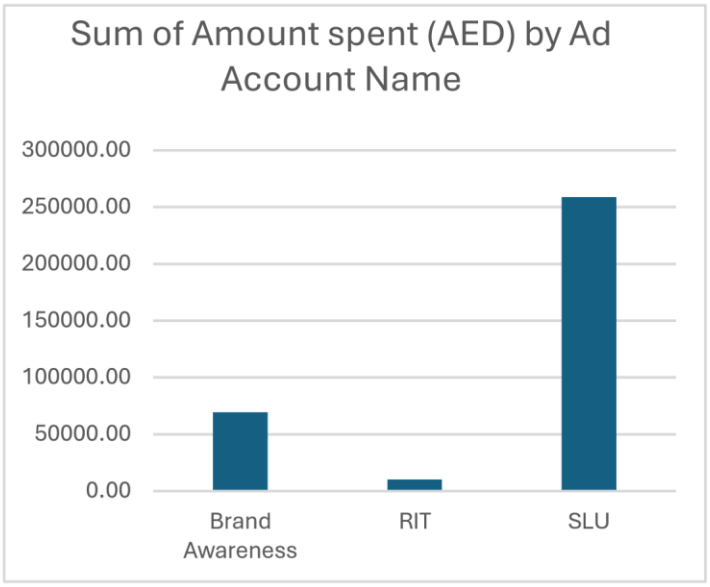
Identification of missing values, duplicates, and inconsistencies:

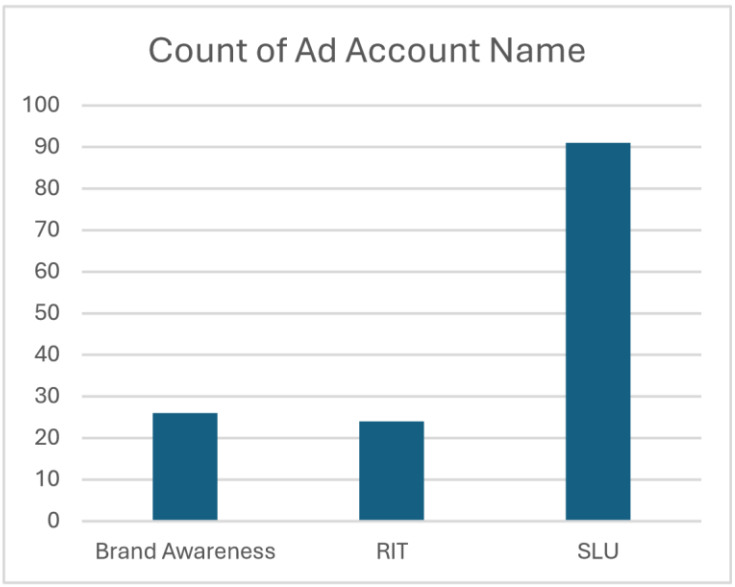
Missing Values: 08 (there are over all 8 missing values in the entire table(blanks))

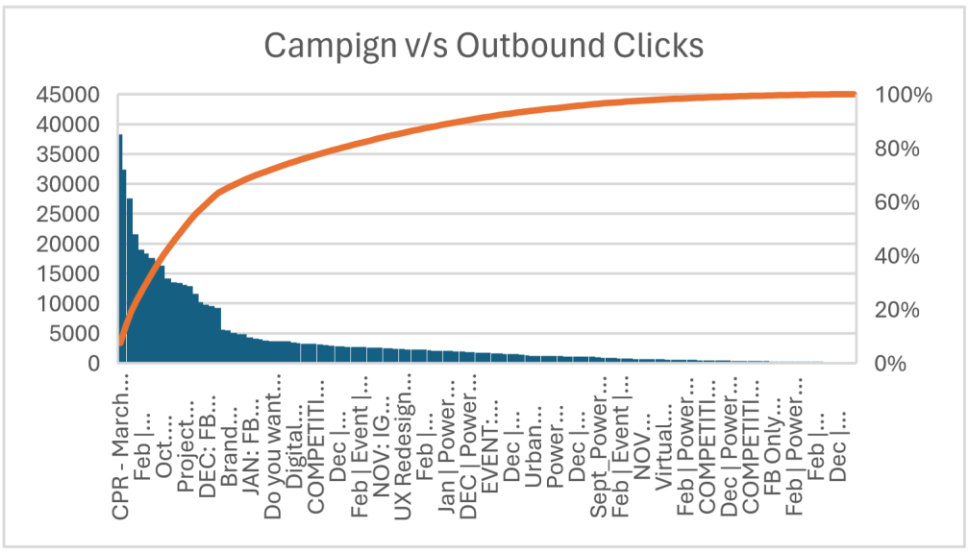
Duplicates: no duplicates

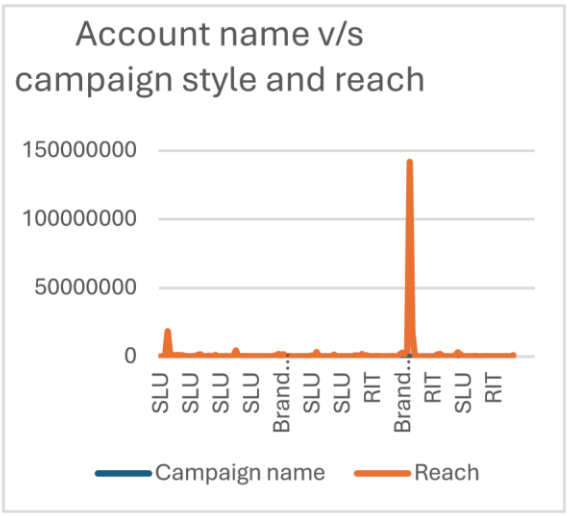
Inconsistencies:

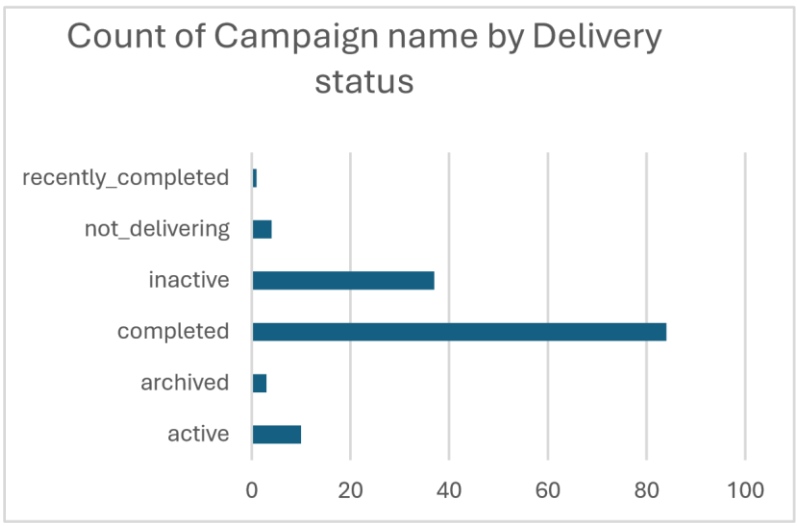
a) There are consistent in the date, like different format than expected, have anomalies, outliers in cost per result, amount, results.


Sum of Amount spent (AED) by Ad Account Name
300000.00
250000.00
200000.00
150000.00
100000.00
50000.00
0.00
Brand Awareness
RIT
SLU


Count of Ad Account Name
100
90
80
70
60
50
40
30
20
10
0
Brand Awareness
RIT
SLU


Campign v/s Outbound Clicks
45000
40000
35000
30000
25000
20000
15000
10000
5000
0
100%
80%
60%
40%
20%
0%
CPR - March...
Feb |...
Oct....
Project...
DEC: FB...
Brand...
JAN: FB...
Do you want...
Digital...
COMPETITI...
Dec |...
Feb | Event |...
NOV: IG...
UX Redesign...
Feb |...
Jan | Power...
DEC | Power...
EVENT:...
Dec |...
Urban...
Power...
Dec |...
Sept_Power...
Feb | Event |...
NOV...
Virtual...
Feb | Power...
COMPETITI...
Dec | Power...
COMPETITI...
FB Only...
Feb | Power...
Feb |...
Dec |...


Account name v/s campaign style and reach
150000000
100000000
50000000
0
SLU
SLU
SLU
SLU
Brand...
SLU
SLU
RIT
Brand...
RIT
SLU
RIT
Campaign name
Reach


Count of Campaign name by Delivery status
recently_completed
not_delivering
inactive
completed
archived
active
0
20
40
60
80
100

Key Findings Distribution:

- Most metrics (Reach, Results, Amount Spent, etc.) are stored as **TEXT** in the raw file, preventing direct numeric analysis until casting/cleaning.
- Initial inspection suggests multiple repeated Ad Account Name values; Campaign name varies but needs normalization (case, extra spaces).
- Outbound clicks and CPC have very limited distinct values in the raw stats output, indicating possible data parsing/formatting issues.

By Campaign:

- No declared primary key — multiple campaigns can share the same reporting date and ad account.
- Campaign naming inconsistencies (e.g., spacing, casing) could cause false duplicates when grouping.

Temporal:

- dates column present but needs validation — ensure it is a true DATE type and not a text string.
- Cannot yet compute start–end ranges or trends until data cleaning.

Quality:

- Many columns show **0 non-null counts** in your posted stats output — likely due to import or quoting issues when using \copy from CSV.
- Potential duplicates across Ad Account Name + Campaign name + dates need checking once cleaned.
- Numeric fields (Reach, Results, Amount Spent, etc.) are text in raw import — requires conversion before meaningful analysis.

Data Cleaning performed

1. **Data Type Fixes**
    - Cast Reach, Outbound clicks, Results, Cost per result, Amount spent (AED), CPC (cost per link click) to NUMERIC.
    - Cast dates to DATE type (MM/DD/YYYY parsing).
    - Ensure Ad Account Name, Campaign name, Delivery status, etc. remain as TEXT.
2. **Cleaning & Standardization** ○ Trim whitespace, normalize case in Campaign name and Ad Account Name.
    - Replace blank strings ('') with NULL where applicable.
    - Remove any non-numeric characters from numeric columns before casting.
3. **Key & Constraints**
    - Create a **composite primary key** on (ad_account_name, campaign_name, dates) if business rules support it. ○ Consider unique index on (campaign_name, dates) if each campaign has one record per date.
4. **Feature Engineering** ○ Add spend_per_click = amount_spent_aed / outbound_clicks. ○ Add result_rate = results / reach.
    - Derive campaign duration once grouped by campaign.
5. **Validation Rules** ○ Flag negative or zero metrics where not expected.
    - Flag future dates or unrealistic spend/reach values.
6. **Analytics Prep** ○ Create an indexed cleaned table (marketing_campaign_clean). ○ Build a view with essential metrics, standardized names, and derived features for direct use in Power BI/Tableau.

Data Quality Report

Marketing_Campaign:

Issues Detected

- **Blank Campaign Names**
  Two records are missing values in the *Campaign Name* field while still containing data in all other columns. Since a campaign name is a critical identifier, these rows cannot be reliably used in analysis and were removed.
- **Results Column – Potential Outliers**
  At first glance, the *Results* column contains unusually high values. However, when cross-checked against the *Reach* metric, these values align with expected performance. This occurs when the *Result Type* is **"Reach"** or **"ThruPlay"**, confirming the values are valid.

Cleaning Logic Applied

- Removed two rows with blank *Campaign Name* values.

Testing Methodology

- Initial row count: 141
- Final row count after cleaning: 139
- Manual cross-check of *Results* vs *Reach* confirmed consistency in cases flagged as potential outliers.

Why Create a Calculated Date Table

A calculated date table serves as a common time dimension, enabling connections between datasets that lack a direct primary key or shared column — for example, linking the Marketing Campaign table with other datasets.

By standardizing and centralizing date-related information, it allows for more comprehensive analysis, such as:

- Comparing cohort joinings during periods with active marketing campaigns versus periods without.
- Measuring the impact of specific campaigns on learner enrollment.
- Identifying which marketing campaigns have been most effective in driving registrations.

This unified time-based structure ensures consistent filtering, trend analysis, and cross-dataset insights that would otherwise be difficult to achieve.

Cognito Data Quality & ETL Report

Comprehensive Data Assessment and ETL Plan

Dataset: Cognito Raw Data
Records: 129178 **Fields:** 9

Executive Summary

This report assesses the Cognito raw dataset and outlines an ETL plan to achieve a clean, standardized, and analytics-ready master table. We identified missing values across key attributes, 0 duplicate keys, and format inconsistencies in date fields. The proposed ETL pipeline extracts the raw data, removes duplicates, standardizes text and date formats, and enforces integrity checks, ensuring reliable inputs for downstream reporting.

Table of Contents

1. Cognito Data Overview

The dataset contains user-level attributes. The table below lists columns, inferred data types, and missing value rates.

Column	Data Type	Missing %
user_id	object	0.0%
email	object	0.0%
gender	object	33.18%
UserCreateDate	object	0.0%
UserLastModifiedDate	object	0.0%
birthdate	datetime64[ns]	33.18%
city	object	33.18%
zip	object	33.19%
state	object	33.24%

2. Data Quality Checks

We focus on missing values, duplicates, and format inconsistencies prior to ETL.

2.1 Check Missing Values

Purpose	**Identify completeness gaps in key attributes to inform imputation or exclusion rules.**
Why	Missing data can bias analysis and break downstream joins; handling them consistently improves reliabi

SQL	SELECT COUNT(*) AS total_rows, SUM(CASE WHEN user_id IS NULL THEN 1 ELSE 0 END) AS missing_user_id, SUM(CASE WHEN email IS NULL THEN 1 ELSE 0 END) AS missing_email, SUM(CASE WHEN gender IS NULL THEN 1 ELSE 0 END) AS missing_gender, SUM(CASE WHEN birthdate IS NULL THEN 1 ELSE 0 END) AS missing_birthdate, SUM(CASE WHEN city IS NULL THEN 1 ELSE 0 END) AS missing_city, SUM(CASE WHEN zip IS NULL THEN 1 ELSE 0 END) AS missing_zip, SUM(CASE WHEN state IS NULL THEN 1 ELSE 0 END) AS missing_state, SUM(CASE WHEN UserCreateDate IS NULL THEN 1 ELSE 0 END) AS missing_usercreatedate, SUM(CASE WHEN UserLastModifiedDate IS NULL THEN 1 ELSE 0 END) AS missing_userlastmodifi FROM cognito_raw;

2.2 Check Duplicate Records

Purpose	**Ensure uniqueness of user records and prevent double-counting.**
Why	Duplicate rows inflate metrics and create inconsistencies across systems.
SQL	-- Duplicate full rows SELECT COUNT(*) AS duplicate_rows FROM (SELECT *, ROW_NUMBER() OVER (PARTITION BY user_id, email, gender, birthdate, city, zip, state, (FROM cognito_raw) t WHERE rn > 1; -- Duplicate user_id keys SELECT user_id, COUNT(*) AS cnt FROM cognito_raw GROUP BY user_id HAVING COUNT(*) > 1;

3. Cognito Sample Data

A 10-row sample provides a quick preview of raw records and helps validate fields.

	email	gender	UserCreateDate	UserLastModifiedDate	birthdate	city	zip	st
36-433c-a941-a612b3d2fbb8	gikonyosalome19@gmail.com	Female	2024-11-17T21:25:56.381Z	2024-11-17T21:32:50.783Z	1996-05-04	NAIVASHA	20117	NA
83-4e21-816b-101cf05f9a79	evelyn.natasha.guo@gmail.com	nan	2025-01-19T02:07:06.113Z	2025-01-19T02:07:20.726Z	NaT	nan	nan	na
82-4889-ae96-3b77ff60f1e4	lauren.singh@rocketmail.com	Female	2024-03-26T23:23:54.329Z	2024-09-27T13:47:51.806Z	1990-04-05	Queens Village	11428	NY
24-4de8-9f10-59ebf8fd019b	anihmercy2019@gmail.com	Female	2024-03-31T19:04:21.735Z	2024-09-27T16:12:28.564Z	1998-12-28	Ibadan	200221	Oy
17-4b6a-9462-fc2bc7fdfb91	lagrimasamie@gmail.com	Female	2024-03-25T20:36:26.352Z	2024-04-08T16:10:24.503Z	1999-05-05	Malolos City	3000	Bu
ef-4dab-b7ef-d953aee7e746	kolayinka777@gmail.com	nan	2023-06-16T15:19:21.404Z	2023-06-16T15:20:06.170Z	NaT	nan	nan	na
04-4b66-a1e2-72172db26b33	ujjwal.pandey2103@gmail.com	Male	2024-05-21T17:58:57.614Z	2024-09-27T16:04:21.994Z	2000-03-21	New Delhi	110045	De
e-4594-9248-4a5d53ff5a7e	amaliataabazuing@gmail.com	Female	2023-07-05T22:25:14.656Z	2024-09-08T08:55:42.155Z	2002-04-22	Kumasi	233	As
6a-42e8-ab0d-ed9c0adc6139	230888@d230.org	nan	2023-01-05T16:33:12.709Z	2023-01-05T19:32:14.802Z	NaT	nan	nan	na
8a-40fd-a1c5-796aed4440f7	vonyedika35@gmail.com	nan	2023-06-22T12:56:37.433Z	2023-06-22T12:57:57.221Z	NaT	nan	nan	na

4. Validation & Refinement

We validate record counts, null handling, and duplicate removal after loading to staging and master tables. Key checks include: (i) record count parity vs. raw; (ii) absence of unintended duplicates; (iii) date and text types correctly cast; and (iv) referential integrity where applicable.

5. ETL & Cleaning Steps

Purpose	**Create a de-duplicated, typed staging table from raw Cognito data.**
Why	Provides a controlled surface for transformations without changing the raw source.

SQL	CREATE TABLE IF NOT EXISTS stg_cognito AS
	SELECT DISTINCT
	  user_id,
	  LOWER(email) AS email,
	  NULLIF(gender,'') AS gender,
	  CAST(NULLIF(birthdate,'') AS DATE) AS birthdate,
	city, zip, state,
	  CAST(UserCreateDate AS TIMESTAMP) AS user_created_at,
	  CAST(UserLastModifiedDate AS TIMESTAMP) AS user_modified_at
	FROM cognito_raw;

Purpose	**Ensure each user_id is unique; drop exact duplicates.**
Why	Prevents metric inflation and inconsistent joins.

SQL	DELETE FROM stg_cognito t
	USING (
	  SELECT MIN(ctid) AS keep_ctid, user_id
	  FROM stg_cognito
	  GROUP BY user_id
	  HAVING COUNT(*) > 1
	) d
	WHERE t.user_id = d.user_id
	  AND t.ctid <> d.keep_ctid;

Purpose	**Enforce not-null constraints for critical identifiers.**
Why	Null keys break referential integrity and analytics grouping.
SQL	DELETE FROM stg_cognito WHERE user_id IS NULL OR email IS NULL;
Purpose	**Normalize all date/time fields to canonical types.**
Why	Consistent date types avoid parsing errors and enable time-based analytics.
SQL	UPDATE stg_cognito SET user_created_at = CAST(user_created_at AS TIMESTAMP), user_modified_at = CAST(user_modified_at AS TIMESTAMP);

6. Final Data Quality Summary

Column	Missing %	Duplicates (user_id)	Date Format Issues
user_id	0.0%	0	No
email	0.0%	-	No
gender	33.18%	-	No
UserCreateDate	0.0%	-	No
UserLastModifiedDate	0.0%	-	No
birthdate	33.18%	-	No
city	33.18%	-	No
zip	33.19%	-	No
state	33.24%	-	No

The ETL flow delivers a consistent, duplicate-free, and typed dataset suitable for analytics and reporting.