PET RESCUE CHARITY DONATIONS DATA MANAGEMENT PROJECT REPORT

PREPARED BY: GROUP 1

Alex Yu

Wendell Destang

Nwume Ekenedilichukwu Cynthia

FOR: SUNDUS SHANEEF

SUBMISSION DATE:

April 3rd, 2025

TABLE OF CONTENTS

Alex Yu, Wendell Destang, Nwume Ekenedilichukwu Cynthia

Introduction

The Pet Rescue Charity organizes an annual city-wide donation drive to support its mission of rescuing and caring for animals. Volunteers, led by designated volunteer leaders, collect donations across different neighborhoods and submit weekly reports to the charity's main office.

To efficiently manage and analyze these donations, the organization requires a Central Donation Repository housed in an Oracle database. This project focuses on designing and implementing a data processing system to streamline donation tracking, validation, and reporting.

The project was divided into six key tasks:

Data Integration- Synchronizing the Oracle Address table with an updated SQL Server address table, ensuring the database has the most current donor addresses.
Data Validation & Loading- Developing a process to import donation records from CSV files, validate them, and reject incorrect or incomplete entries.
Data Mart Creation- Designing and implementing a star schema data mart for analyzing donations based on date, location, and volunteer contributions.
Data Transformation & Loading- Creating processes to transfer data from the Central Donation Repository to the Data Mart, ensuring it is optimized for analysis.
Reporting Views- Building database views to generate summary reports on donation trends, helping the charity track fundraising progress.
Database Security- Implementing user roles and permissions to control data access, ensuring secure and authorized operations.

By successfully executing these tasks, this project will enhance the charity's ability to manage, validate, and analyze donation data, ultimately improving their efficiency in fundraising and resource allocation.

Task 1: Updating the Address Table

The first task involved creating a data integration process to update the Address table in the Oracle database with the latest records from the master address table in SQL Server. This was accomplished using Talend, which facilitated the extraction, transformation, and loading (ETL) of address data.

To establish a connection to the SQL Server database, **Microsoft Azure** was used along with the provided credentials. The dbo.Address table from the Integration database was then retrieved and visualized.

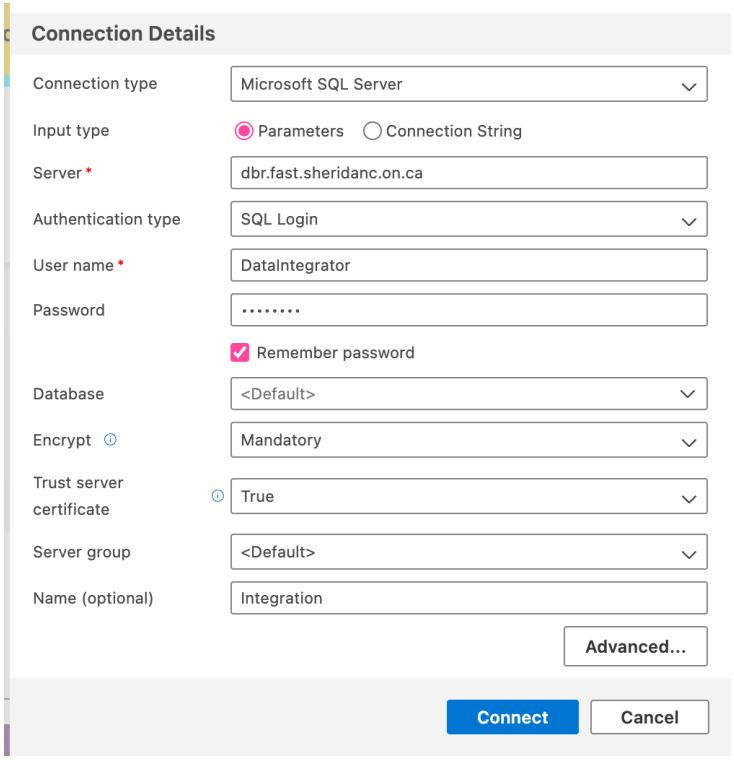
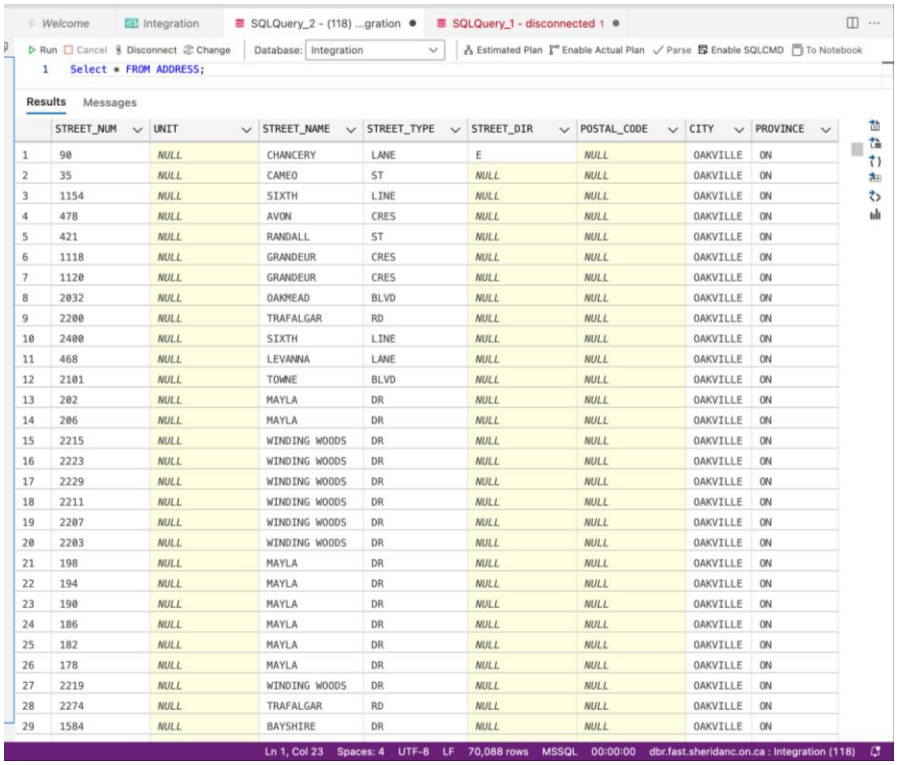


Fig 1: Data Integrator Connection Details on Microsoft Azure



Welcome | Integration | SQLQuery_2 - (118) ...gration | SQLQuery_1 - disconnected 1

Run | Cancel | Disconnect | Change | Database: Integration | Estimated Plan | Enable Actual Plan | Parse | Enable SQLCMD | To Notebook

```
1  Select * FROM ADDRESS;
```

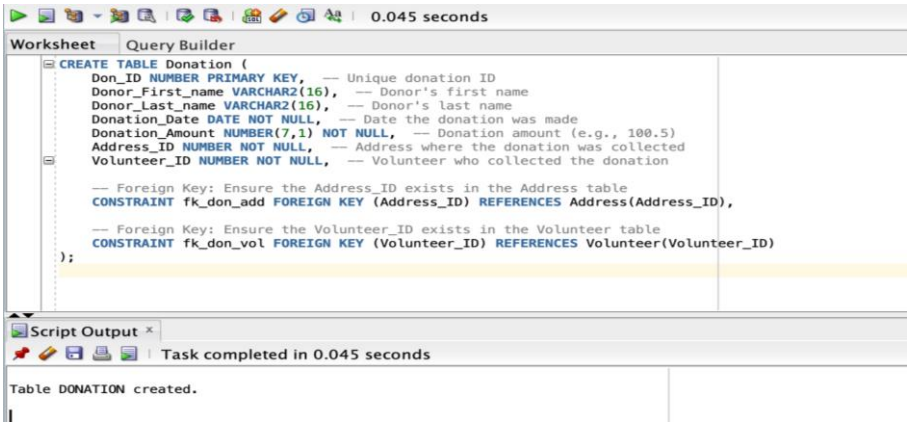
Results | Messages

	STREET_NUM	UNIT	STREET_NAME	STREET_TYPE	STREET_DIR	POSTAL_CODE	CITY	PROVINCE
1	90	NULL	CHANCERY	LANE	E	NULL	OAKVILLE	ON
2	35	NULL	CAMEO	ST	NULL	NULL	OAKVILLE	ON
3	1154	NULL	SIXTH	LINE	NULL	NULL	OAKVILLE	ON
4	478	NULL	AVON	CRES	NULL	NULL	OAKVILLE	ON
5	421	NULL	RANDALL	ST	NULL	NULL	OAKVILLE	ON
6	1118	NULL	GRANDEUR	CRES	NULL	NULL	OAKVILLE	ON
7	1120	NULL	GRANDEUR	CRES	NULL	NULL	OAKVILLE	ON
8	2032	NULL	OAKMEAD	BLVD	NULL	NULL	OAKVILLE	ON
9	2200	NULL	TRAFALGAR	RD	NULL	NULL	OAKVILLE	ON
10	2400	NULL	SIXTH	LINE	NULL	NULL	OAKVILLE	ON
11	468	NULL	LEVANNA	LANE	NULL	NULL	OAKVILLE	ON
12	2101	NULL	TOWNE	BLVD	NULL	NULL	OAKVILLE	ON
13	202	NULL	MAYLA	DR	NULL	NULL	OAKVILLE	ON
14	206	NULL	MAYLA	DR	NULL	NULL	OAKVILLE	ON
15	2215	NULL	WINDING WOODS	DR	NULL	NULL	OAKVILLE	ON
16	2223	NULL	WINDING WOODS	DR	NULL	NULL	OAKVILLE	ON
17	2229	NULL	WINDING WOODS	DR	NULL	NULL	OAKVILLE	ON
18	2211	NULL	WINDING WOODS	DR	NULL	NULL	OAKVILLE	ON
19	2207	NULL	WINDING WOODS	DR	NULL	NULL	OAKVILLE	ON
20	2203	NULL	WINDING WOODS	DR	NULL	NULL	OAKVILLE	ON
21	198	NULL	MAYLA	DR	NULL	NULL	OAKVILLE	ON
22	194	NULL	MAYLA	DR	NULL	NULL	OAKVILLE	ON
23	190	NULL	MAYLA	DR	NULL	NULL	OAKVILLE	ON
24	186	NULL	MAYLA	DR	NULL	NULL	OAKVILLE	ON
25	182	NULL	MAYLA	DR	NULL	NULL	OAKVILLE	ON
26	178	NULL	MAYLA	DR	NULL	NULL	OAKVILLE	ON
27	2219	NULL	WINDING WOODS	DR	NULL	NULL	OAKVILLE	ON
28	2274	NULL	TRAFALGAR	RD	NULL	NULL	OAKVILLE	ON
29	1584	NULL	BAYSHIRE	DR	NULL	NULL	OAKVILLE	ON

Ln 1, Col 23 | Spaces: 4 | UTF-8 | LF | 70,088 rows | MSSQL | 00:00:00 | dbr.fast.sheridanc.on.ca : Integration (118)

Fig 2: Master Address Table in Data Integration Database

Simultaneously, a connection was established to the Oracle database using the PRC user, where the necessary tables (Address, Donation, and Volunteer) were created. These tables were created using **Oracle SQL Developer.**



0.045 seconds

Worksheet | Query Builder

```
CREATE TABLE Donation (
    Don_ID NUMBER PRIMARY KEY,  -- Unique donation ID
    Donor_First_name VARCHAR2(16),  -- Donor's first name
    Donor_Last_name VARCHAR2(16),  -- Donor's last name
    Donation_Date DATE NOT NULL,  -- Date the donation was made
    Donation_Amount NUMBER(7,1) NOT NULL,  -- Donation amount (e.g., 100.5)
    Address_ID NUMBER NOT NULL,  -- Address where the donation was collected
    Volunteer_ID NUMBER NOT NULL,  -- Volunteer who collected the donation

    -- Foreign Key: Ensure the Address_ID exists in the Address table
    CONSTRAINT fk_don_add FOREIGN KEY (Address_ID) REFERENCES Address(Address_ID),

    -- Foreign Key: Ensure the Volunteer_ID exists in the Volunteer table
    CONSTRAINT fk_don_vol FOREIGN KEY (Volunteer_ID) REFERENCES Volunteer(Volunteer_ID)
);
```

Script Output

Task completed in 0.045 seconds

```
Table DONATION created.
```

Fig 3: Donation Table Creation

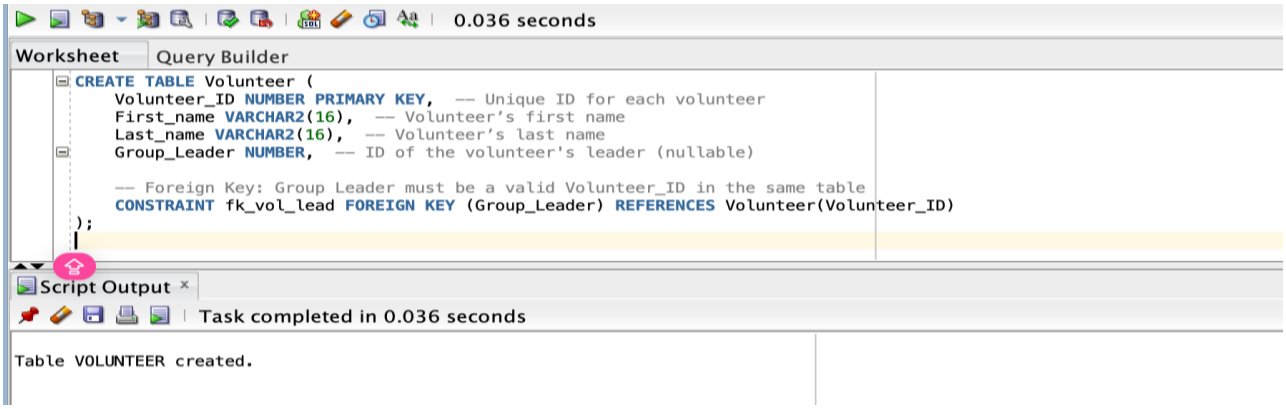


Fig 4: Volunteer Table Creation

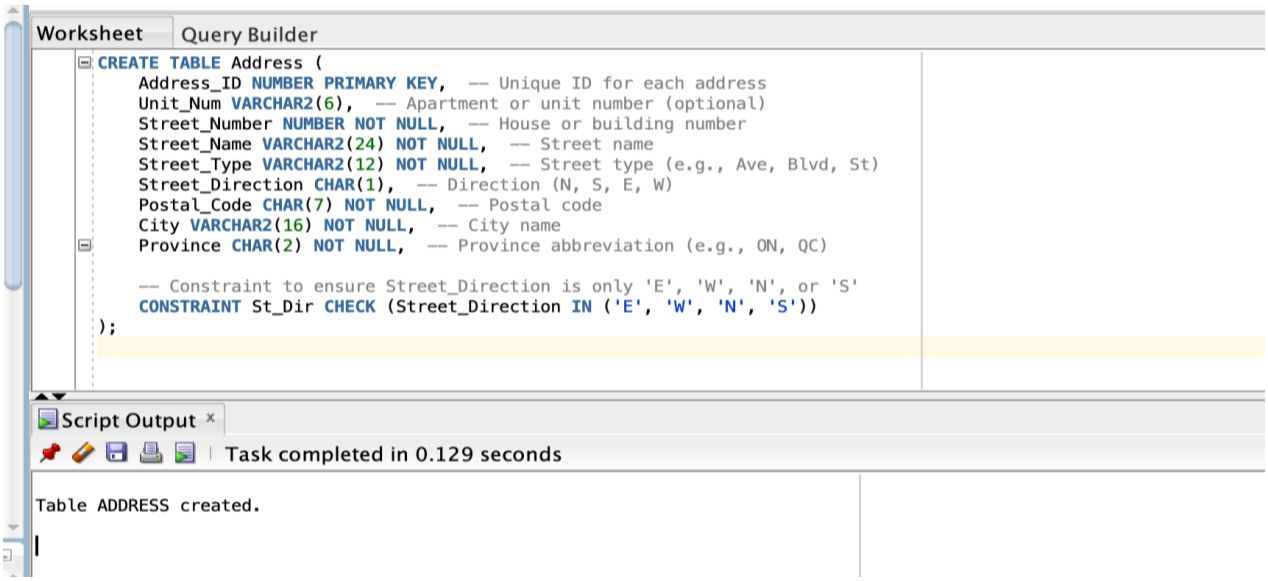


Fig 5: Address Table Creation

1. **Setting Up Connections in Talend** – Two separate database connections were created:

- One for SQL Server (source) using tDBInput
- One for Oracle Server (output) using tDBOutput

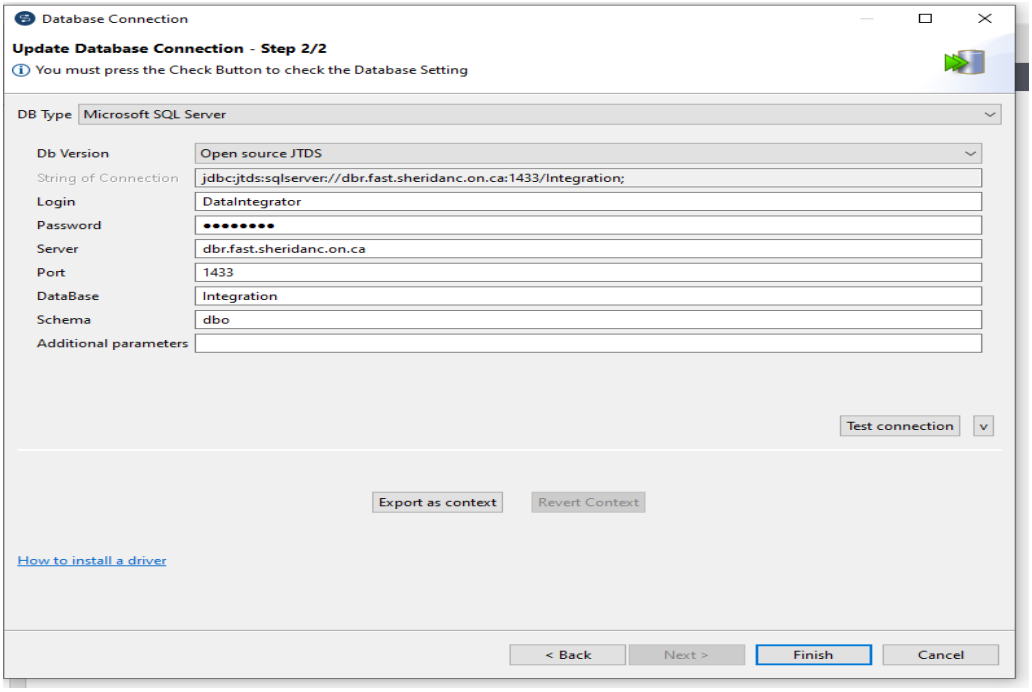


Fig 6: Data Integrator Connection On Talend Microsoft SQL Server

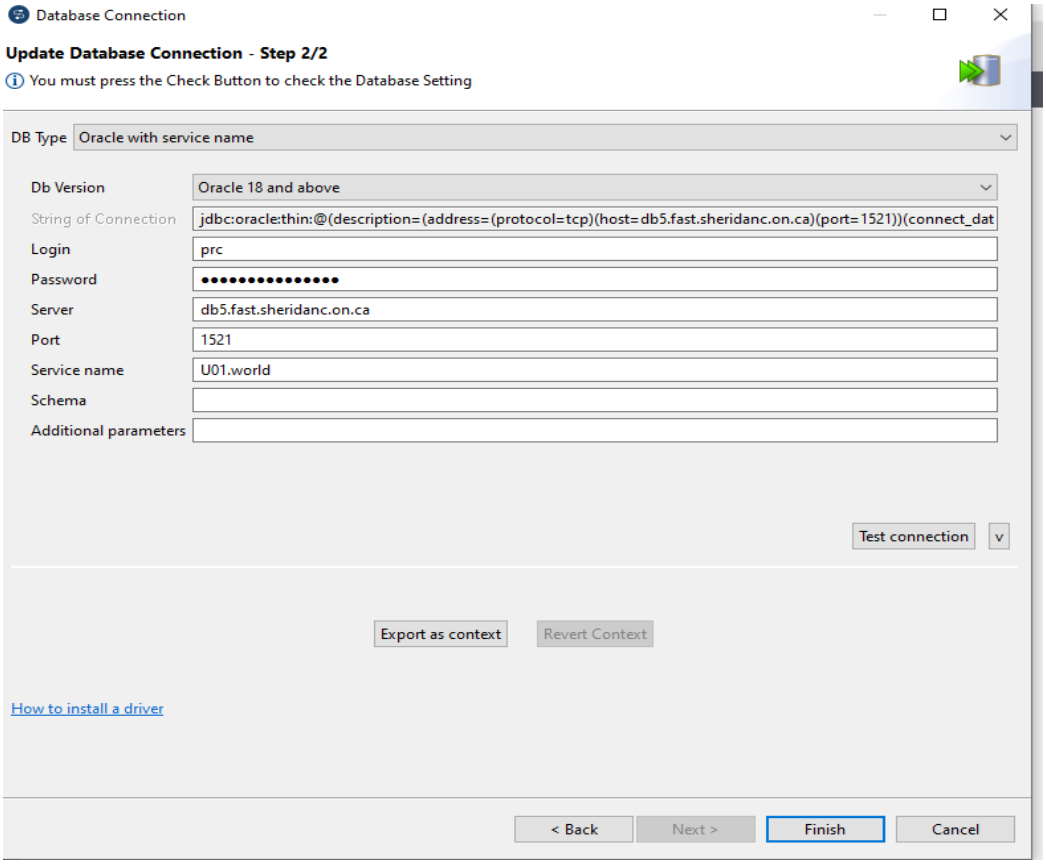


Fig 7: Oracle PRC Connection On Talend

These connections were added via **Metadata > Db Connections** in Talend.

2. Mapping the Address Table

- The SQL Server Address table (left side in Talend) was connected to a tMap component.
- The tMap component ensured correct mapping of input table columns to the Oracle Address table (right side in Talend).

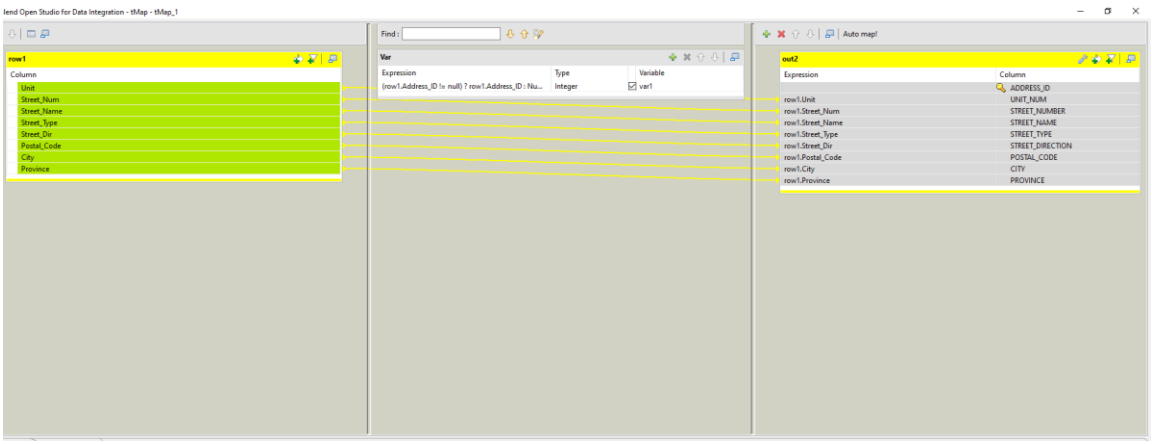


Fig 8: TMAP: Mapping the columns in the DataIntegration Database to the PRC Database

3. Handling Address_ID and Data Transformation

- The SQL Server table lacked an Address_ID, so a sequence generator was created in Talend to assign unique IDs incrementally.

Fig 9: DataIntegration Job Successfully Completed on Talend

By implementing this multi-step process, accurate and efficient data synchronization between SQL Server and Oracle databases was achieved, ensuring that the Address table remained up to date.

Task 2: Importing the csv data to the donations table

For this task, the donations data is first loaded into an intermediate table in the Oracle database using Talend, then filtered and loaded into the donations table using a PL/SQL script, with the leftover rejected data being manually extracted into a csv file.

The first step was to make another table identical to the donations table, called temp_donation. This will contain all of the records imported from the csv files, and there are no constraints added to any of the columns, such as primary or foreign keys. The table is created with the following SQL statement:

```
CREATE TABLE TEMP_DONATION(
    Don_ID NUMBER,
    Donor_First_name VARCHAR2(16),
    Donor_Last_name VARCHAR2(16),
    Donation_Date DATE,
    Donation_Amount NUMBER(7,1),
    Address_ID NUMBER,
    Volunteer_ID NUMBER
    );
```

Fig 10: Creating the TEMP donation table

This table is used for the next step, where the data is used to extract and insert the data into the temp_donation table, using Talend. The process is very similar to task 1, except the data will be loaded from csv files and a file delimited is created for the input and set to the directory of the file containing the donations data. The output is set to a connection to the temp_donation table in the Oracle server.

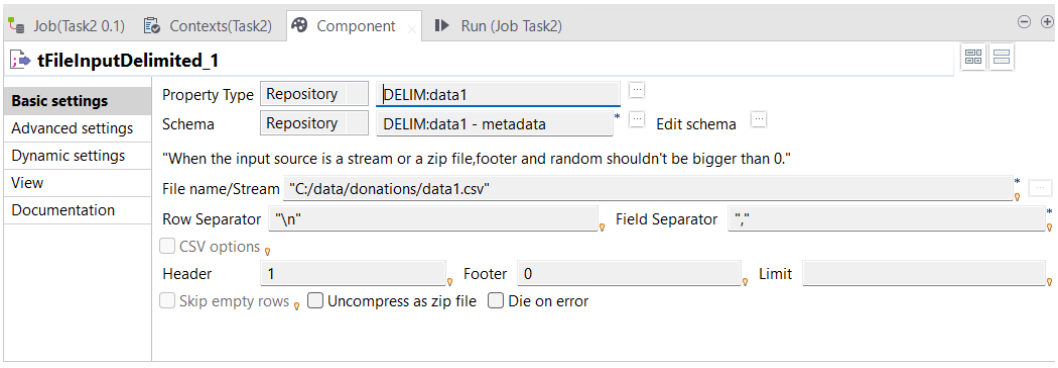


Fig 11: File input delimited

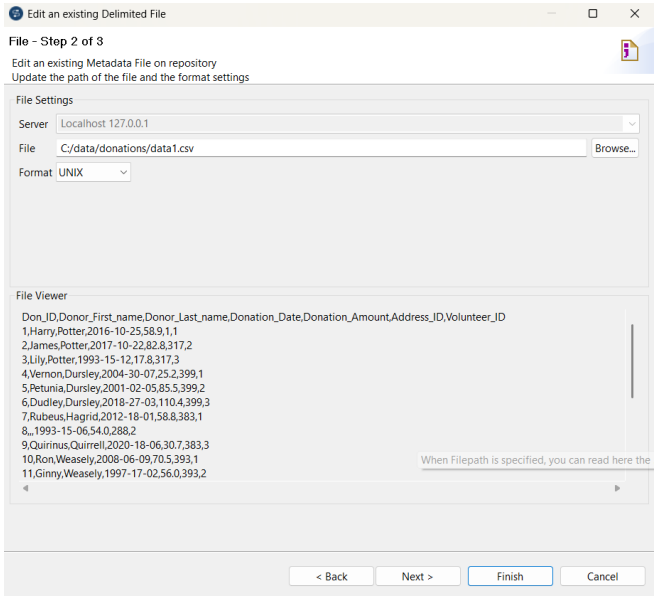


Fig 12: Location of the file and its contents

Because the temp_donations schema uses the BigDecimal variable type for numeric values while the file delimited input schema automatically uses types such as integer or float, the numeric data types in the input schema had to be changed to BigDecimal, so it matches the temp_donation schema. The tMap component could instead be used to do this using BigDecimal.valueOf(), but it will cause errors if any null values are passed into it, which essentially disallows invalid records with null numeric values to be loaded from the input.

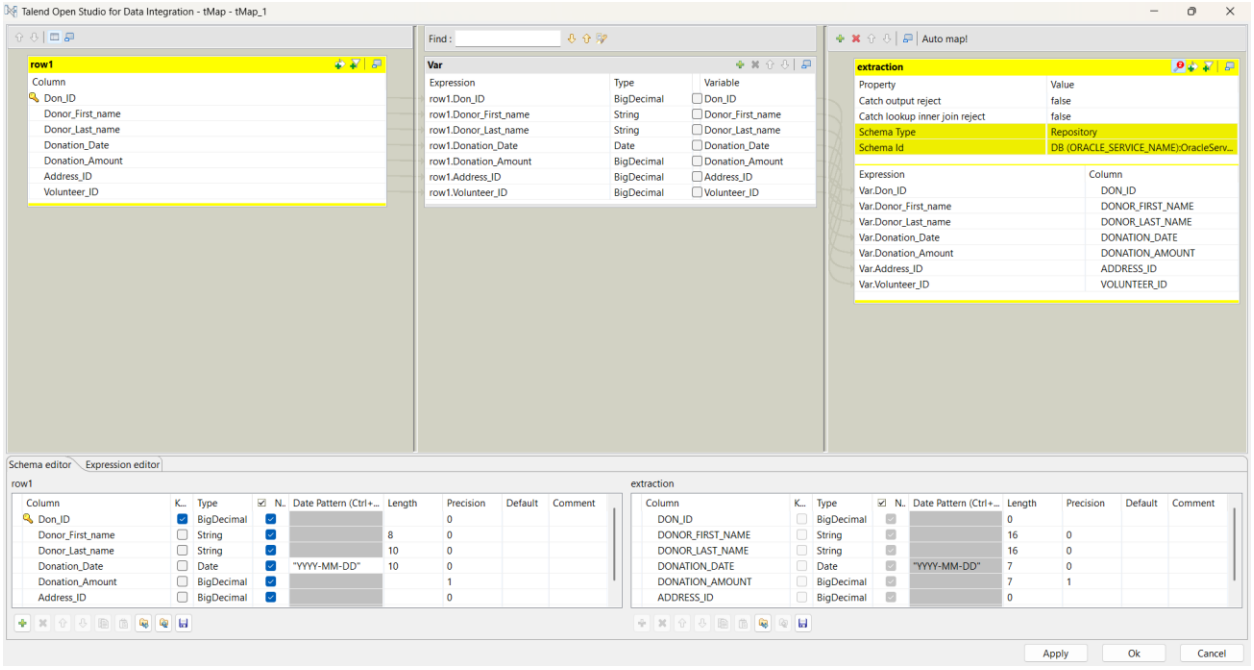


Fig 13: Mapping component; Maps columns from the file to the Oracle server

Each of the csv files are loaded in parallel, with the first Oracle server output dropping and recreating the temp_donations table to remove any old records that might linger around. Of course, the tMap might be unnecessary, but at least it makes it easy to see what is inside each of the schemas.

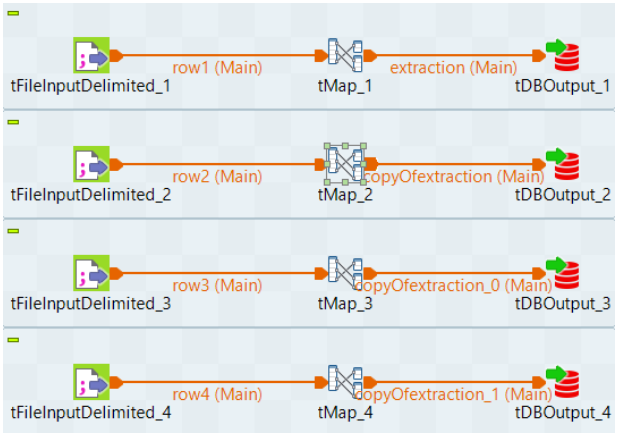


Fig 14: The Talend job used to load data from the csv files to the Oracle server

After the data is transported to the oracle server, the following PL/SQL script can be run to retrieve all of the valid entries in temp_donations and bulk insert them into the donations table.

```
DECLARE
    TYPE rec_donation IS TABLE OF DONATION%ROWTYPE;
    table_donation rec_donation;
    CURSOR cur IS SELECT * FROM TEMP_DONATION WHERE
        ADDRESS_ID IN (SELECT ADDRESS_ID FROM ADDRESS) AND
        VOLUNTEER_ID IN (SELECT VOLUNTEER_ID FROM VOLUNTEER) AND
        DON_ID NOT IN (SELECT DON_ID FROM DONATION) AND
        DON_ID IS NOT NULL AND
        DONATION_DATE IS NOT NULL AND
        DONATION_AMOUNT IS NOT NULL AND
        DONATION_AMOUNT > 0;
BEGIN
    OPEN cur;
        FETCH cur BULK COLLECT INTO table_donation;
    CLOSE cur;
    FOR ind IN 1 .. table_donation.LAST LOOP
        INSERT INTO DONATION VALUES table_donation(ind);
    END LOOP;
    DELETE FROM TEMP_DONATION WHERE DON_ID IN (SELECT DON_ID FROM DONATION);
END;
/
```

Fig 15: Script used to filter and transport the data from TEMP donations to donations

The script also deletes all of the entries in temp_donation that are present in the donation table, based on the primary key DON_ID. This effectively removes all of the valid entries from temp_donation, making it easy to manually extract the rejected data into a csv file.

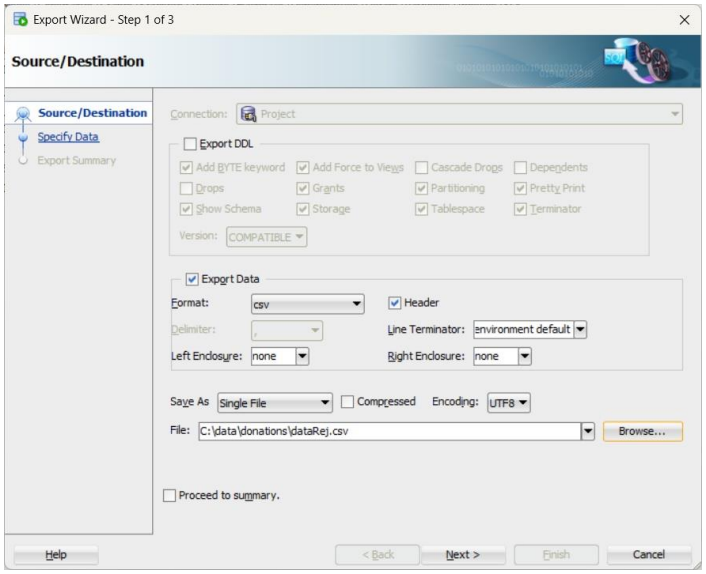


Fig 16: Exporting the rejected entries from temp_donation

This process has been tested multiple times using 4 different csv files containing a total of 81 records, with a few null or invalid entries. And the process has been completed on all of the tests without any issues.

Task 3: Creating a Star Schema and ER Diagram for the Donations Data Mart

Objective

The goal of this task was to design a star schema for the Donations Data Mart, ensuring efficient storage and retrieval of donation-related data. The schema focuses on capturing donation transactions at the grain level, which is defined as a unique combination of date (day level), address, and volunteer. The schema enables analytical queries on donation trends based on time, location, and volunteer contributions.

Schema Design

The star schema consists of one fact table (DONATION_FACT) and three dimension tables (DATE_DIMENSION, ADDRESS_DIMENSION, and VOLUNTEER_DIMENSION).

1. Fact Table:

DONATION_FACT

The fact table stores the core transactional data, including the number of donations and the total value of donations for each combination of date, address, and volunteer.

- Primary Key: FACT_ID
- Foreign Keys:
    - DATE_ID → Links to DATE_DIMENSION
    - ADDRESS_ID → Links to ADDRESS_DIMENSION
    - VOLUNTEER_ID → Links to VOLUNTEER_DIMENSION
- Measures:
    - DONATION_COUNT – The total number of donations
    - TOTAL_DONATIONS – The sum of all donations' values

This fact table allows aggregations such as total donations per day, per address, and per volunteer.

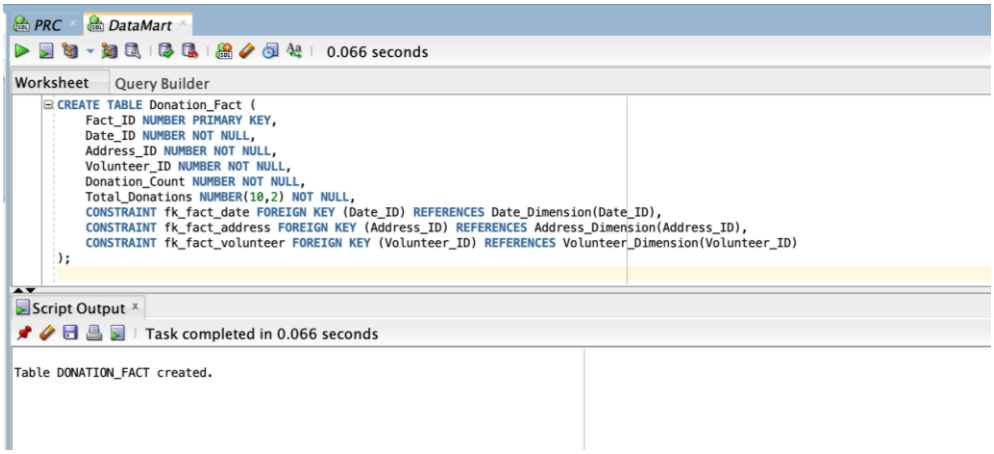


Fig 17: Creation Of Donation_Fact Table

2. Dimension Tables

Each dimension table provides detailed attributes for analyzing the donations from different perspectives.

- DATE_DIMENSION (Time-based analysis)
  - DATE_ID – Unique identifier for each date
  - FULL_DATE – Specific donation date (e.g., '2021-07-01')
  - YEAR – Year of the donation (e.g., 2021)
  - MONTH_NUMBER – Numeric month representation (1-12)
  - DAY_NUMBER – Day of the month (1-31)
  - MONTH_NAME_SHORT – Abbreviated month name (e.g., "Jan")
  - MONTH_NAME_LONG – Full month name (e.g., "January")

This table enables time-series analysis, allowing reports on donations by year, month, and specific dates.

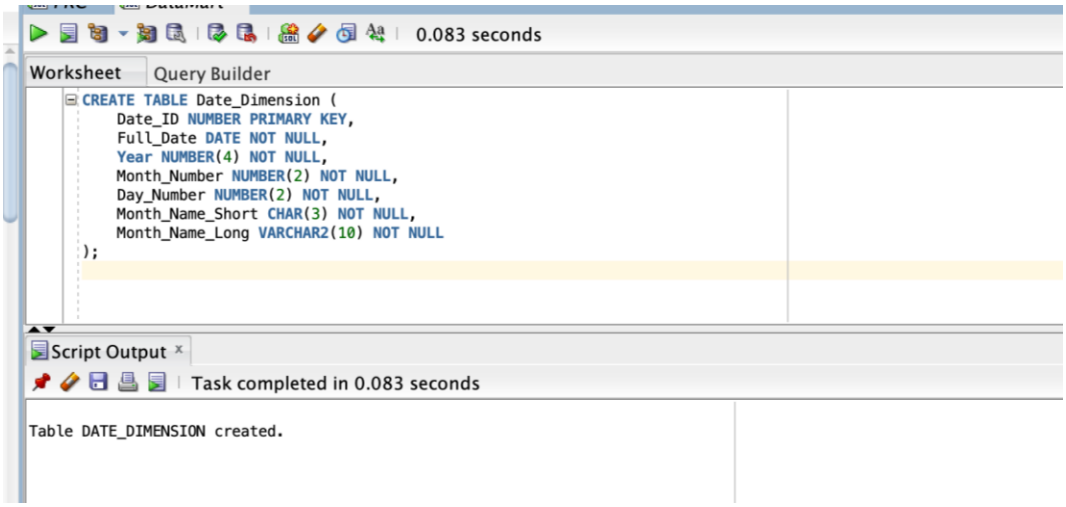


Fig 18: Creation Of Date_Dimension Table

- ADDRESS_DIMENSION (Location-based analysis
  - ADDRESS_ID – Unique identifier for each address
  - POSTAL_CODE – The postal code of the donation location
  - ADDRESS – Full address description

This table helps analyze donations based on geographical distribution.

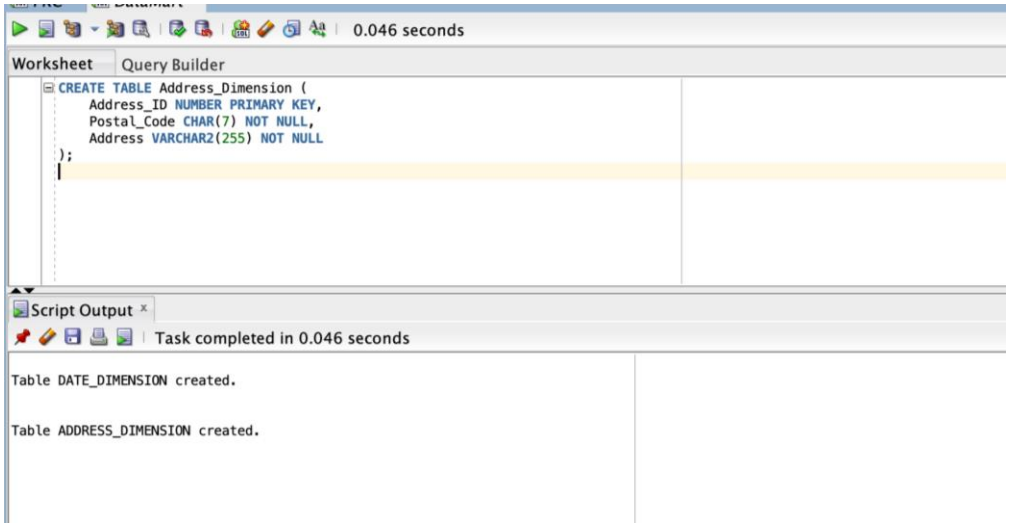


Fig 19: Creation Of Address_Dimension Table

- VOLUNTEER_DIMENSION (Volunteer-based analysis)
  - VOLUNTEER_ID – Unique identifier for each volunteer

- VOLUNTEER_NAME – The name of the volunteer

This table allows tracking of donations based on volunteer contributions.

ER Diagram Representation

The Entity-Relationship (ER) Diagram visually represents the fact and dimension tables and their relationships. The DONATION_FACT table serves as the central hub, with foreign keys linking it to the dimension tables. This design ensures that each donation record is associated with a specific date, address, and volunteer.

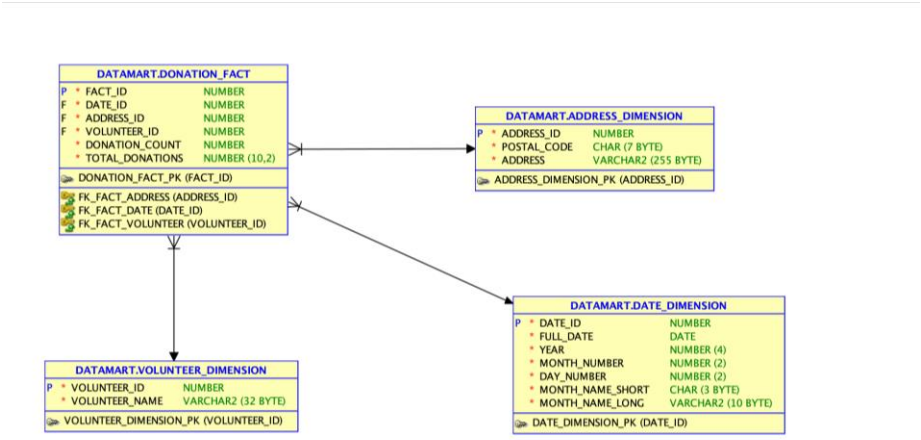


Fig 20: Entity Relationship Diagram showing the fact and dimension tables and their relationships

Database Implementation

To implement this schema in Oracle SQL Developer:

- The connection was established using the datamart user.
- The schema was structured following the star schema principles, ensuring optimized query performance for analytical processing.
- Primary keys were defined for unique identification, and foreign keys were established to maintain referential integrity between tables.

Task 4: Loading Data to Datamart

This task required creating a process which loads the data from the central repository to the star schema. For this process, a sequence was created to ensure that the surrogate keys for each dimension table were unique.

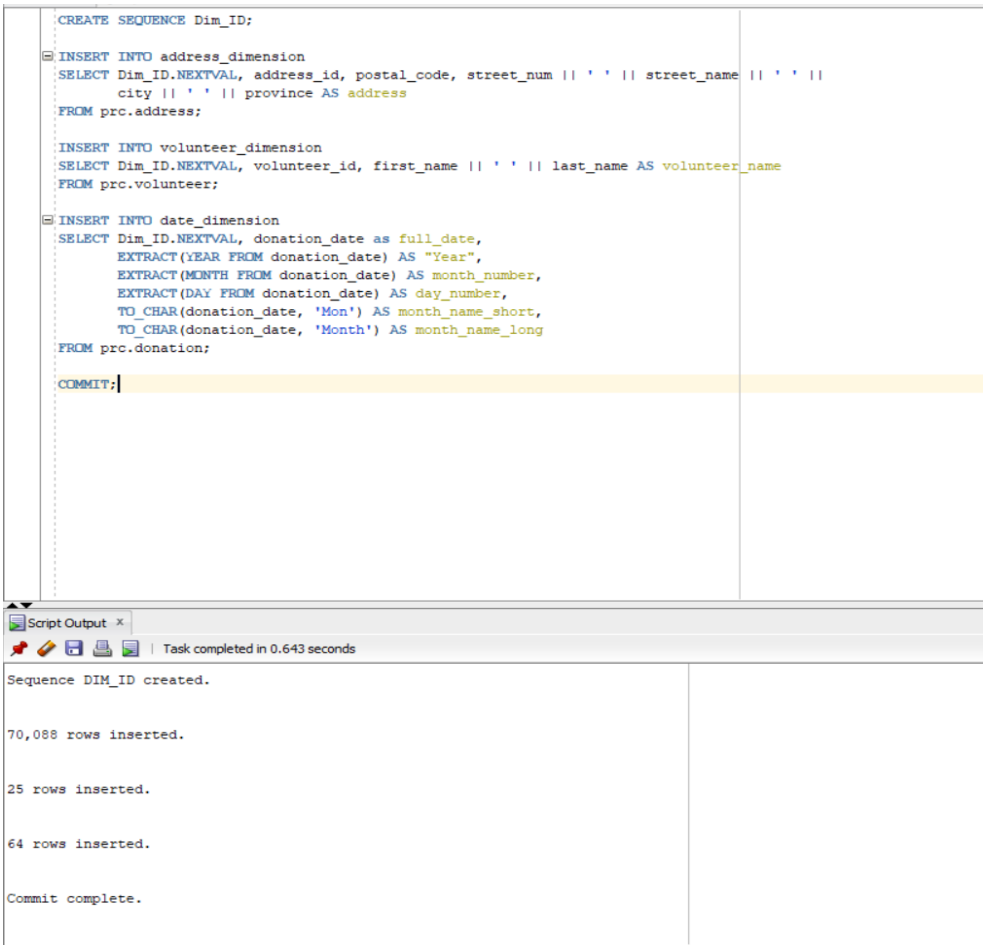


Fig 21: Insert statements to populate the dimension tables

Below is the insert statement for the fact table. The grain of the fact table is the intersection of date, address and volunteer.

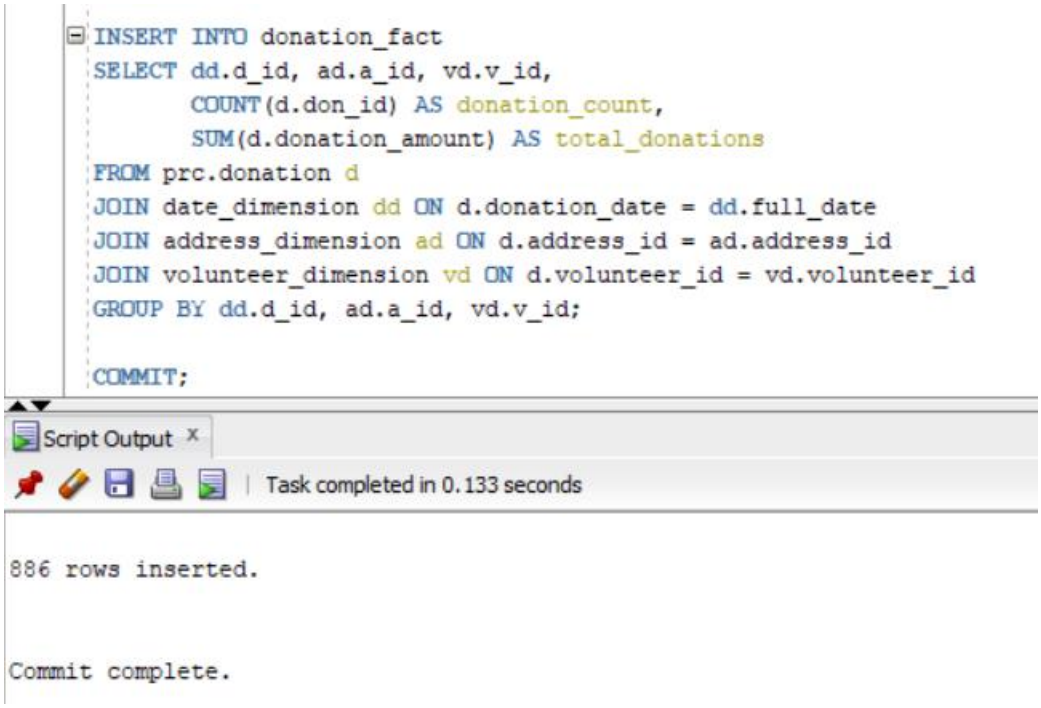


Fig 22: Insert statement to populate the fact table

Task 5: Creating Views

This task's objective was to create views based on certain result sets. The first view needed to address the following: The number of donations and total donations for each date showing the date hierarchy (year, long month name, day).

```sql
CREATE OR REPLACE VIEW donations_by_date AS
SELECT donation_date AS "Date", TO_CHAR(donation_date, 'YYYY') AS "Year",    -- TO_CHAR used to retrieve the year from donation date
       TO_CHAR(donation_date, 'Month') AS "Month",
       TO_CHAR(donation_date, 'DD') AS "Day",
       COUNT(don_id) AS Number_of_Donations,     -- Using don_id to determine to number of donations
       ROUND(SUM(donation_amount), 2) AS Total_Donation_Amount
FROM donation
GROUP BY donation_date, TO_CHAR(donation_date, 'YYYY'),
         TO_CHAR(donation_date, 'Month'),
         TO_CHAR(donation_date, 'DD')
ORDER BY donation_date, "Year", "Month", "Day";
```

Fig 23: SQL statement to create the view donations_by_date

The second view needed to address the following: The number of donations, sums and average donations by location hierarchy (postal code and address).

```sql
CREATE OR REPLACE VIEW donations_by_location AS
SELECT a.postal_code,
       a.street_num || ' ' || a.street_name || ' ' || a.city || ' ' || a.province AS Address,    -- Concatinating columns to create a single address line
       COUNT(d.don_id) AS Number_of_Donations, ROUND(SUM(d.donation_amount), 2) AS Total_Donation_Amount,
       ROUND(AVG(d.donation_amount), 2) AS Average_Donation
FROM donation d JOIN address a ON d.address_id = a.address_id    -- Join is needed as I am grabbing information from more than one table
GROUP BY a.postal_code, a.street_num || ' ' || a.street_name || ' ' || a.city || ' ' || a.province
ORDER BY a.postal_code, Address;
```

Fig 24: SQL statement to create the view donations_by_location

The third view needed to address the following: The number of donations, sums and average by volunteer leader and volunteer.

```sql
CREATE OR REPLACE VIEW donations_by_volunteer AS
SELECT v.group_leader AS Volunteer_Leader, v.first_name || ' ' || v.last_name AS Volunteer,    --Combining multiple columns to create the volunteer column
       COUNT(d.don_id) AS Number_of_Donations, ROUND(SUM(d.donation_amount), 2) AS Total_Donation_Amount,
       ROUND(AVG(d.donation_amount), 2) AS Average_Donation
FROM donation d JOIN volunteer v ON d.volunteer_id = v.volunteer    -- Join is needed to grab information from multiple tables
GROUP BY v.group_leader, v.first_name || ' ' || v.last_name
ORDER BY Volunteer_Leader, Volunteer;
```

Fig 25: SQL statement to create the view donations_by_volunteer

Task 6: Implementing Database Security

To ensure that data in the Pet Rescue Charity's Central Donation Repository remains secure, controlled, and protected against unauthorized access, a role-based security model was implemented. This involved creating two distinct user roles:

DMLUser – A user with permissions to modify data in the Donation, Address, and Volunteer tables.

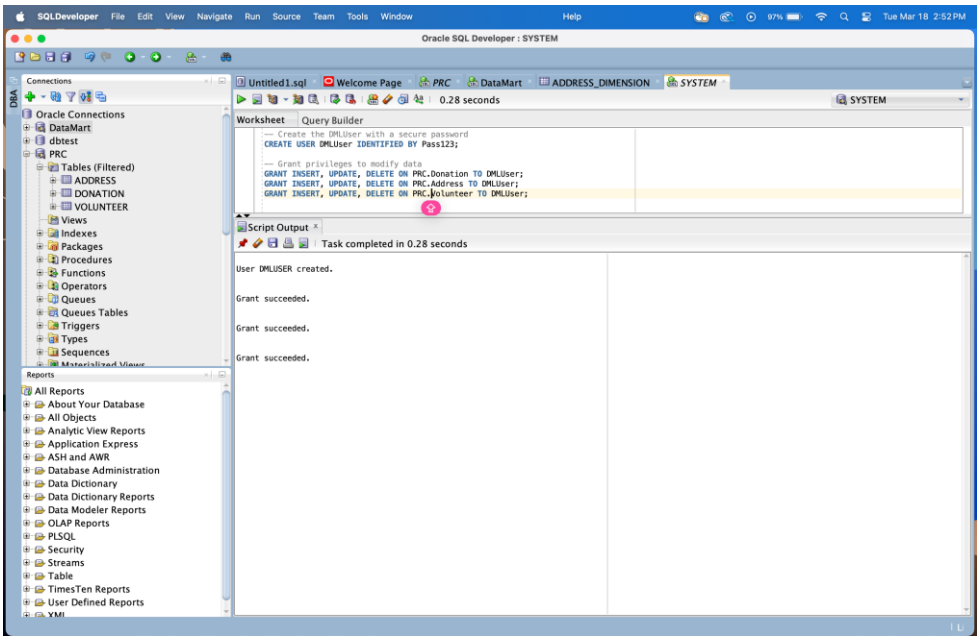


Fig 26: Creation of DMLUser with appropriate privileges

Dashboard – A read-only user with access to view reports but restricted from modifying any data.

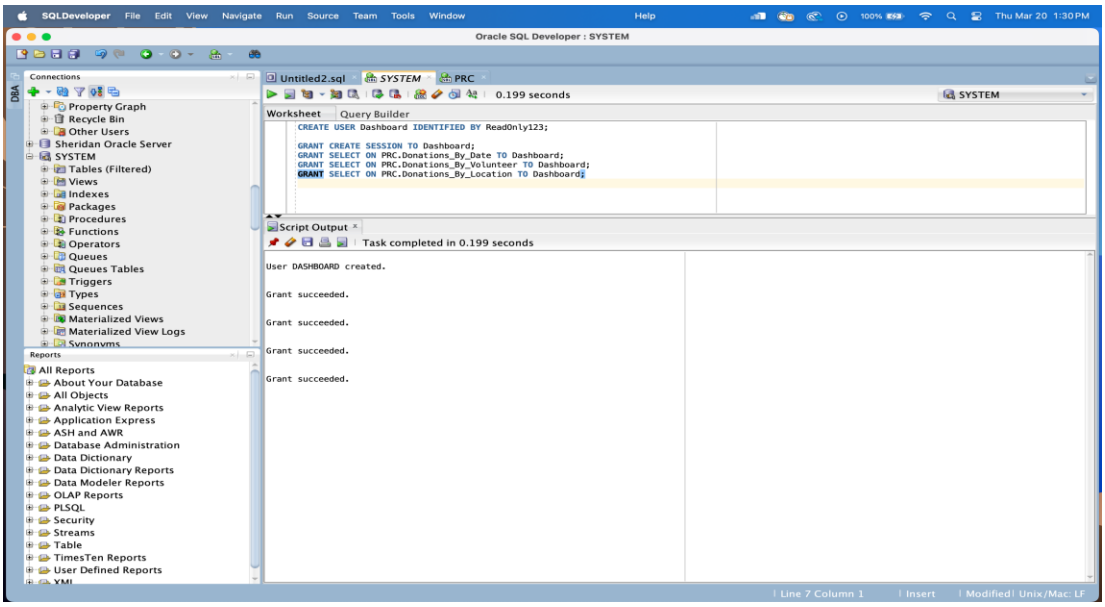


Fig 27: Creation of Dashbord User with appropriate privileges

By defining these roles, the system ensures that only authorized users can modify records, while report viewers cannot alter critical donation data.

Security Implementation Process

Logging in as SYS (Admin)

Before creating new users, a database administrator (DBA) needs to log in as SYS or SYSTEM, as these are the only users with permission to create new users and assign privileges.

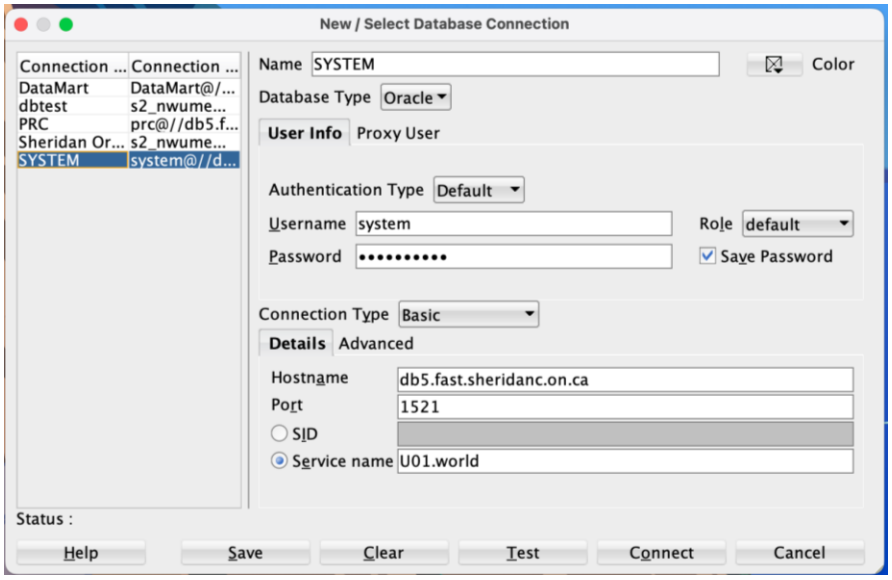


Fig 28: Connection Details of SYSTEM Oracle SQL Developer

Creating Users

The DMLUser and Dashboard accounts were created using SQL commands executed by the SYS user, who has administrative privileges.

Assigning Privileges

- DMLUser was granted INSERT, UPDATE, and DELETE privileges on core tables (Donation, Address, Volunteer).
- Dashboard was granted SELECT-only access to reporting views (Donations_By_Location, Donations_By_Volunteer, Donations_By_Date).
- DMLUser will handle all data modifications (insert, update, delete).
- Dashboard will have only read permissions for viewing reports.

This prevents unauthorized modifications and improves database security.

Testing and Validation

After creating the users and assigning privileges, tests were conducted to confirm access control:

- DMLUser successfully inserted, updated, and deleted records.

- Dashboard was blocked from modifying data but could successfully view reports.

Step 4: Verifying User Permissions

To confirm that privileges were assigned correctly, we execute the following SQL queries:

Check DMLUser Privileges:

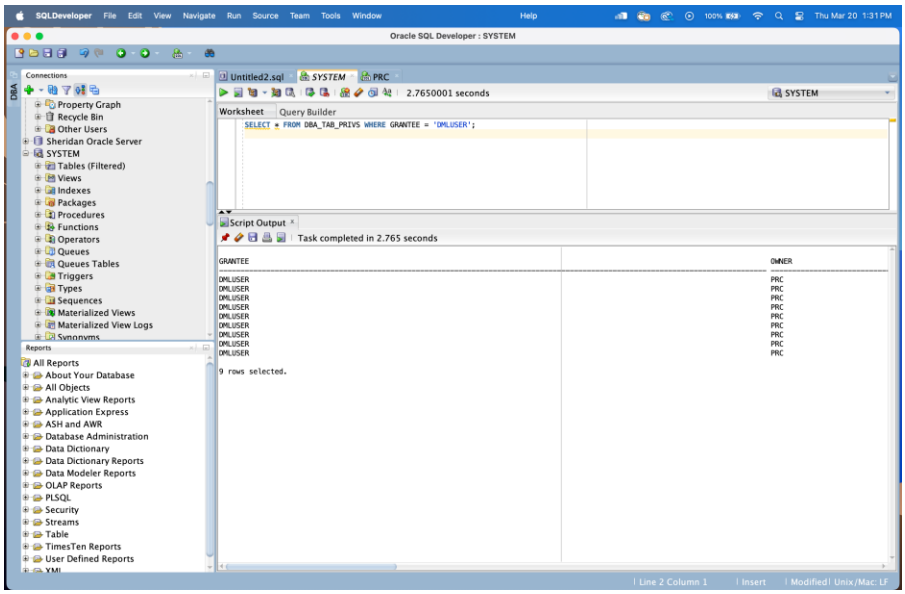


Fig 29: Confirming User Permissions for DMLUser

Check Dashboard Privileges

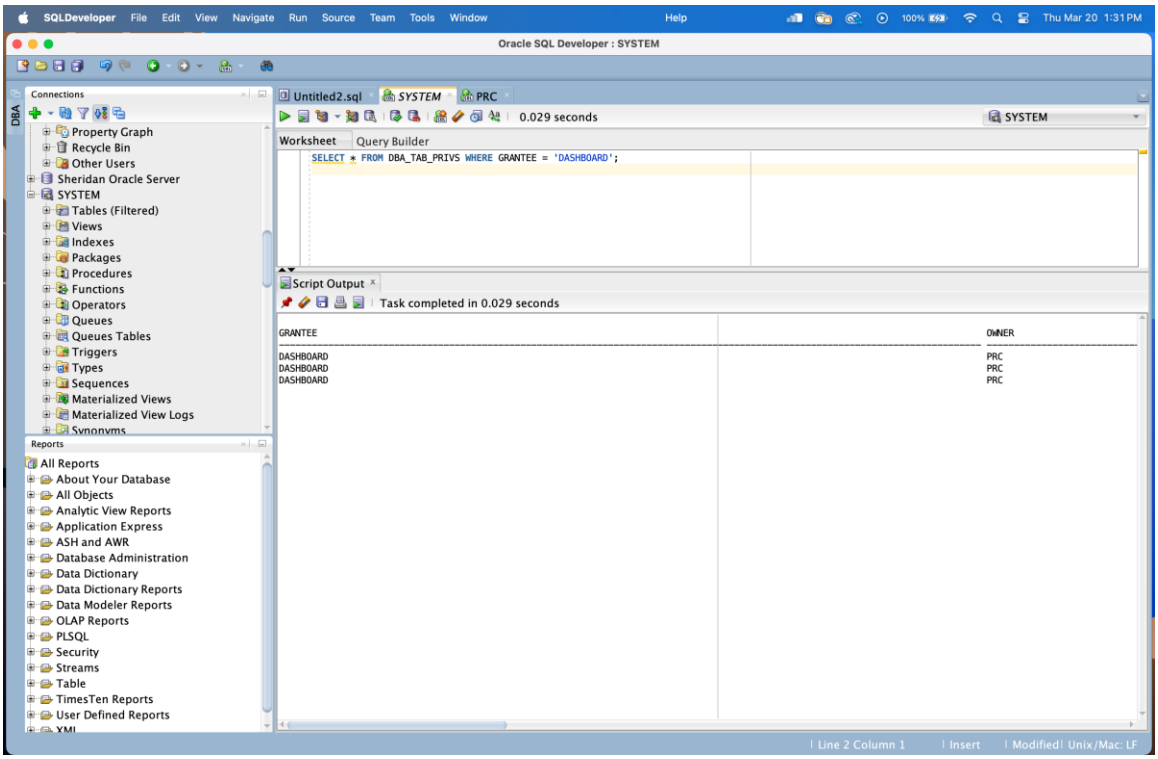


Fig 30: Confirming User Permissions for Dashboard

Expected Result:

- DMLUser has INSERT, UPDATE, DELETE privileges on core tables.
- Dashboard has **only SELECT access** on reporting views.

Conclusion

Conclusion

Alex Yu, Wendell Destang, Nwume Ekenedilichukwu Cynthia

The Pet Rescue Charity Donations Management System project successfully established a structured and efficient database solution for managing donations. Through the integration of Oracle SQL Developer, Talend, and command-line tools, we streamlined the process of storing, transforming, and analyzing donation data.

We began by designing and implementing a central donation repository in Oracle, ensuring that key entities such as donations, volunteers, and addresses were well-structured. The address synchronization process from SQL Server to Oracle improved data accuracy, while the CSV data import process enabled the seamless integration of donation records into the system. By validating records and handling errors efficiently, we ensured data integrity.

The development of a star schema-based data mart enabled efficient reporting and analysis of donation trends, helping stakeholders make informed decisions. The implementation of SQL views provided clear insights into donations based on time, location, and volunteers. Additionally, the security measures put in place ensured controlled access, safeguarding sensitive donation records.

Overall, this project demonstrated the effective use of database management techniques, ETL processes, and data warehousing concepts in a real-world scenario. The system we built enhances data organization, integrity, and accessibility, ultimately benefiting the Pet Rescue Charity by improving transparency and operational efficiency.