TAKE-HOME ASSIGNMENT: ANALYTICS ENGINEER

Mayah Bosworth

Overview

- Clean raw data
  - Correct data types and format
  - Remove irrelevant columns
  - Remove NULL values
  - Use logic to substitute missing information
- Create dimension and fact tables
  - Implement PKs and FKs to link tables
- Output sales_performance table optimized for tableau reporting

Project Structure

```
project/
├── python-scripts/
│   ├── main.py
│   ├── find_path_function.py
│   ├── db_connection_function.py
│   ├── loading-function.py
│   ├── run_sql_script.py
│   ├── detect_encoding_function.py
│   ├── run_etl_function.py
│   ├── sales_performance_analysis_function.py
│   └── run_analysis.py
├── sql-scripts/
│   ├── create_dim_customer_table.sql
│   ├── create_dim_product_table.sql
│   ├── create_fact_sales_table.sql
│   ├── create_raw_customers_table.sql
│   ├── create_raw_products_table.sql
│   ├── create_raw_orders_table.sql
│   ├── create_sales_performance_table.sql
│   ├── populate_sales_performance_table.sql
│   └── sales-metrics-analysis/
│       ├── customer_segment_sales_metrics.sql
│       ├── monthly_revenue_sales_metrics.sql
│       ├── product_category_sales_metrics.sql
│       ├── region_sales_metrics.sql
│       ├── top_customer_sales_metrics.sql
│       └── top_month_metrics.sql
└── csv/
    ├── orders/
    ├── customers/
    └── products/
```

Initial Issues in .csv Files

customers.csv	• NULL values for customer_id • missing customer_name for some rows • inconsistent region strings • irrelevant columns	• filter out rows with a NULL customer_id using CAST and REGEXP • standardize region strings using LOWER • removed irrelevant columns during table creation
orders.csv	• utf-16le file encoding • NULL values for order_id • invalid dates • irrelevant columns. • some customer_id values did not exist in the customers table	• use pandas to handle the utf-16le encoding • filter out rows where order_id was NULL • correct invalid dates (leap year) • removed irrelevant columns during table creation • added "Unknown" entries to dim_customer for missing customer information
products.csv	• utf-16le file encoding • NULL values for product_id • irrelevant columns	• use pandas to handle the utf-16le encoding • filter out rows where product_id was NULL • removed irrelevant columns during table creation

SQL Data Cleaning Preview

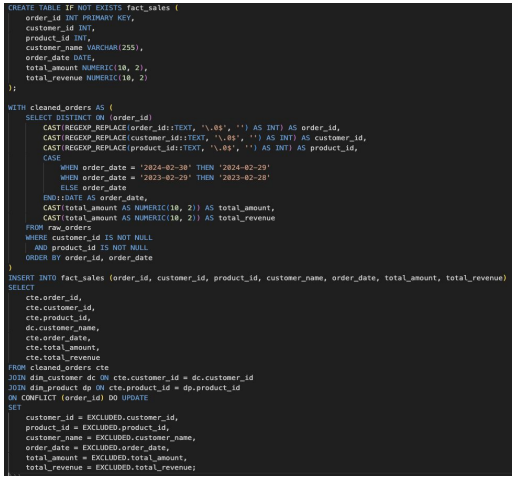
dim_customers_table.sql

```sql
CREATE TABLE IF NOT EXISTS dim_customer (
    customer_id INT PRIMARY KEY,
    customer_name VARCHAR(255),
    region VARCHAR(100),
    segment VARCHAR(100)
);

INSERT INTO dim_customer (customer_id, customer_name, region, segment)
SELECT
    CAST(REGEXP_REPLACE(customer_id::TEXT, '\.0$', '') AS INT) AS customer_id,
    customer_name,
    LOWER(region) AS region,
    segment
FROM raw_customers
WHERE customer_id IS NOT NULL
  AND CAST(customer_id AS TEXT) ~ '^[0-9]+(\.0)?$'
ON CONFLICT (customer_id) DO UPDATE
SET
    customer_name = EXCLUDED.customer_name,
    region = EXCLUDED.region,
    segment = EXCLUDED.segment;
```

dim_products_table.sql

```sql
CREATE TABLE IF NOT EXISTS dim_product (
    product_id INT PRIMARY KEY,
    category VARCHAR(100)
);

INSERT INTO dim_product (product_id, category)
SELECT
    CAST(REGEXP_REPLACE(product_id::TEXT, '\.0$', '') AS INT) AS product_id,
    category
FROM raw_products
WHERE product_id IS NOT NULL
  AND CAST(product_id AS TEXT) ~ '^[0-9]+(\.0)?$'
ON CONFLICT (product_id) DO UPDATE
SET
    category = EXCLUDED.category;
```

fact_sales_table.sql



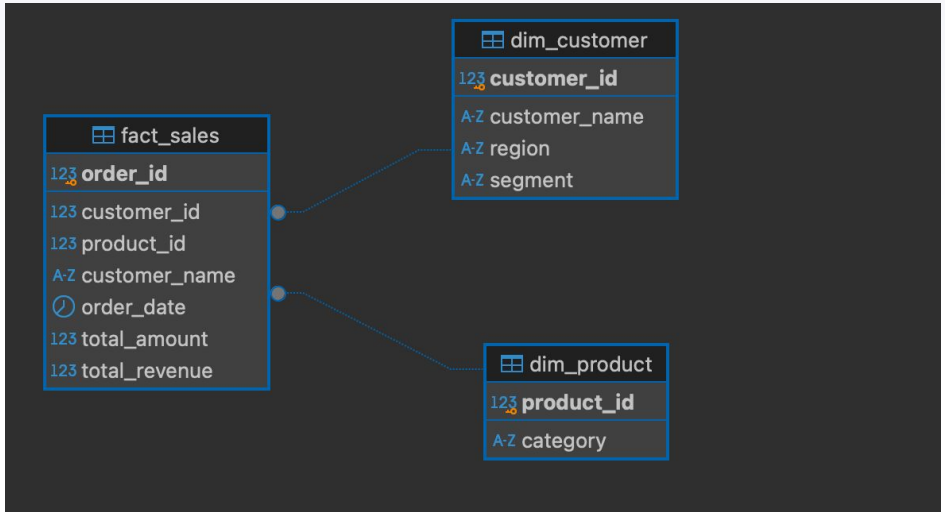
```sql
CREATE TABLE IF NOT EXISTS fact_sales (
    order_id INT PRIMARY KEY,
    customer_id INT,
    product_id INT,
    customer_name VARCHAR(255),
    order_date DATE,
    total_amount NUMERIC(10, 2),
    total_revenue NUMERIC(10, 2)
);

WITH cleaned_orders AS (
    SELECT DISTINCT ON (order_id)
        CAST(REGEXP_REPLACE(order_id::TEXT, '\.0$', '') AS INT) AS order_id,
        CAST(REGEXP_REPLACE(customer_id::TEXT, '\.0$', '') AS INT) AS customer_id,
        CAST(REGEXP_REPLACE(product_id::TEXT, '\.0$', '') AS INT) AS product_id,
        CASE
            WHEN order_date = '2024-02-30' THEN '2024-02-29'
            WHEN order_date = '2023-02-29' THEN '2023-02-28'
            ELSE order_date
        END::DATE AS order_date,
        CAST(total_amount AS NUMERIC(10, 2)) AS total_amount,
        CAST(total_amount AS NUMERIC(10, 2)) AS total_revenue
    FROM raw_orders
    WHERE customer_id IS NOT NULL
      AND product_id IS NOT NULL
    ORDER BY order_id, order_date
)
INSERT INTO fact_sales (order_id, customer_id, product_id, customer_name, order_date, total_amount, total_revenue)
SELECT
    cte.order_id,
    cte.customer_id,
    cte.product_id,
    dc.customer_name,
    cte.order_date,
    cte.total_amount,
    cte.total_revenue
FROM cleaned_orders cte
JOIN dim_customer dc ON cte.customer_id = dc.customer_id
JOIN dim_product dp ON cte.product_id = dp.product_id
ON CONFLICT (order_id) DO UPDATE
SET
    customer_id = EXCLUDED.customer_id,
    product_id = EXCLUDED.product_id,
    customer_name = EXCLUDED.customer_name,
    order_date = EXCLUDED.order_date,
    total_amount = EXCLUDED.total_amount,
    total_revenue = EXCLUDED.total_revenue;
```

ETL Structure

- Step 1: load raw data
- Step 2: create raw tables
- Step 3: create dim tables
- Step 4: create fact table
- Step 5: create sales performance table
- Step 6: populate sales performance table

```python
def run_etl():
    try:
        print("starting ETL pipeline...", flush=True)

        # step 1: load raw data
        print("\nstep 1: loading raw data...", flush=True)
        load_csv_to_db('raw_customers', 'customers.csv', unique_column='customer_id')
        load_csv_to_db('raw_orders', 'orders.csv', unique_column='order_id')
        load_csv_to_db('raw_products', 'products.csv', unique_column='product_id')
        print("raw data loaded successfully\n", flush=True)

        # step 2: create raw tables
        print("step 2: creating raw tables...", flush=True)
        run_sql_script('create_raw_customers_table.sql')
        run_sql_script('create_raw_orders_table.sql')
        run_sql_script('create_raw_products_table.sql')
        print("raw tables created successfully\n", flush=True)

        # step 3: create dimensional tables
        print("step 3: creating dimensional tables...", flush=True)
        run_sql_script('create_dim_customer_table.sql')
        run_sql_script('create_dim_product_table.sql')
        print("dimensional tables created successfully\n", flush=True)

        # step 4: create fact table
        print("step 4: creating fact table...", flush=True)
        run_sql_script('create_fact_sales_table.sql')
        print("fact table created successfully\n", flush=True)

        print("ETL pipeline completed successfully", flush=True)

        # step 5: create sales performance table
        print("step 5: creating sales performance table...", flush=True)
        run_sql_script('create_sales_performance_table.sql')
        print("sales performance table created successfully\n", flush=True)

        # setp 6: populate sales performance table
        print("step 6: populating sales performance table...", flush=True)
        run_sql_script('populate_sales_performance_table.sql')
        print("sales performance table populated successfully\n", flush=True)

    except Exception as e:
        print("\nETL pipeline failed", flush=True)
        print("error:", e, flush=True)

    finally:
        print("ETL pipeline finished", flush=True)
```

Schema Diagram



This schema represents a star schema design, where the `fact_sales` table serves as the central fact table. It connects to two dimension tables, `dim_customer` and `dim_product` via foreign key relationships.

Snapshot of sales_performance table:

	order_id	customer_name	customer_id	region	segment	product_id	category	order_month	total_revenue
1	1	Amanda Vang	13	west	Corporate	8	Electronics	1	486.3
2	2	Unknown	999	Unknown	Unknown	12	Furniture	4	26.41
3	3	Heather Pace	20	east	Retail	999	Clothing	4	382.1
4	4	Mary Austin	27	east	Small Business	6	Electronics	2	279.09
5	5	Sherry Jones	15	south	Small Business	8	Electronics	7	338.14
6	6	Mary Austin	27	east	Small Business	12	Furniture	3	405.81
7	7	Cynthia Lee	25	north	Retail	15	Electronics	3	285.68
8	8	Steven Price	5	east	Retail	9	Furniture	5	498.79
9	9	Emily Warren	10	north	Retail	4	Clothing	1	362.35
10	10	Kyle Gonzalez	24	west	Small Business	15	Electronics	12	470.01

Tableau use case examples:

category	total_revenue_generated
Clothing	994,035.37
Electronics	759,462.74
Furniture	610,085.05
Accessories	144,707.84

Determine top performing product categories

order_month	monthly_revenue
1	339,378.35
2	307,354.45
3	182,656.8
4	156,074.4
5	173,217.6
6	204,444
7	221,965.2
8	190,518.3
9	124,531.2
10	156,184.2
11	148,483.8
12	303,482.7

Track monthly revenue trends

segment	total_revenue_generated
Small Business	1,000,028.47
Retail	865,962.36
Corporate	510,932.57
Unknown	131,367.6

Understand which customer segments generate the most revenue