Web Traffic on E-commerce Platform

Milestone1: Problem definition

Group 11

Sanay Shah

Venkata Sai Prasad Aka

Abdul Mateen

857-318-6077 (Tel of Student 1)
857-260-8799 (Tel of Student 2)
857-381-1139 (Tel of Student 3)

shah.sana@northeastern.ed
u aka.v@northeastern.edu
mateen.a@northeastern.edu

Percentage of Effort Contributed by Student 1: 33%

Percentage of Effort Contributed by Student 2: 33%

Percentage of Effort Contributed by Student 3: 33%

Signature of Student 1: Sanay Shah

Signature of Student 2: Venkata Sai Prasad Aka

Signature of Student 3: Abdul Mateen

Submission Date: 03/26/2023

Problem Definition

An e-commerce fashion company focusing on young people with an age range of 15-35 years was established since early 2016. Since the pandemic hit in 2020, the company has seen the potential for the development of digital shopping because people spend more time accessing the internet. Few local companies like Fashion Campus have also started online thrifting and selling used products. Can this new segment of market can boom in coming years and is the web traffic is helping overtime to increase the sales

Objective:

To analyze website traffic and sales. Additionally, to figure out lifestyle and fashion industry product trends on e-commerce platform over last few years and e-commerce growth to understand the broader shift towards online shopping.

Dataset:

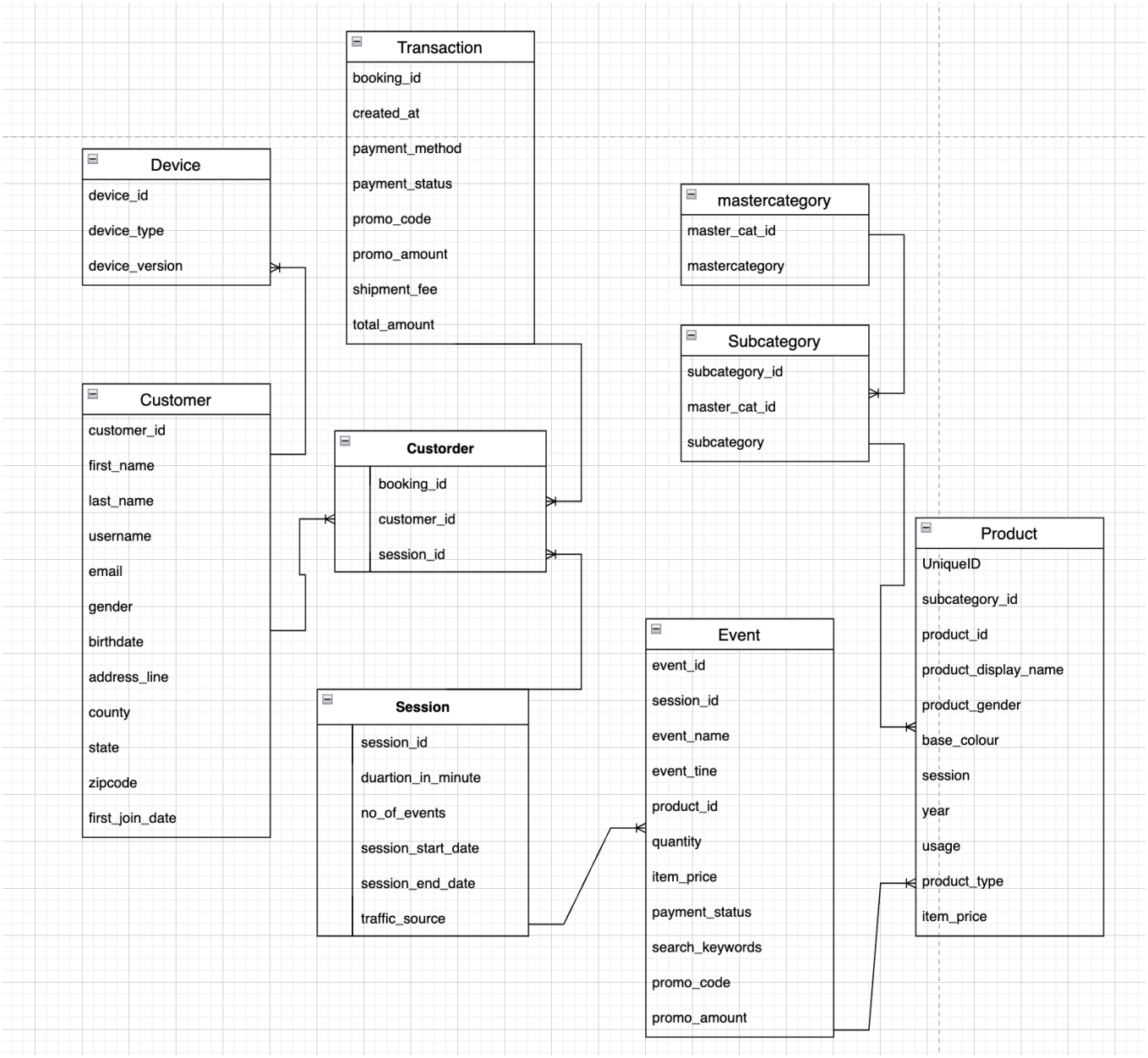
https://www.kaggle.com/datasets/latifahhukma/fashion- campus?select=transactions.csv

File Name:
click_stream_new.csv ,customer.csv,Product.csv ,Transaction_new.csv (Instead of latitude and longitude in dataset planning to create New columns Country, State, Zipcode)

Objective:

1. Figure out the trend in traffic on the application /website month by month.
2. Number of repeat customers
3. Percentage of Customers visiting the website/application and buying the products in one go
4. Which category of goods different age groups are preferring to buy and from which brand?
5. In what price bracket people are willing to spend?
6. Thrifting or buying and selling used clothes on digital platforms has a good potential to grow

ER Schema (diagram):



Customer
customer_ID, device_ID, first_name, last_name, username, email, gender, birthdate, age, address_line, county, state, zip_code, first_join_date.

Device
device_ID, device_type, device_version.

Transaction
booking_ID, created_at, payment_method, payment_status, promo_code, promo_amount, shipment_fee, total_amount.

Custorder
Booking_id, customer_id, session_id.

Session
session_ID, traffic_source, duration_in_minute, no_of_events, session_start_date, session_end_date.

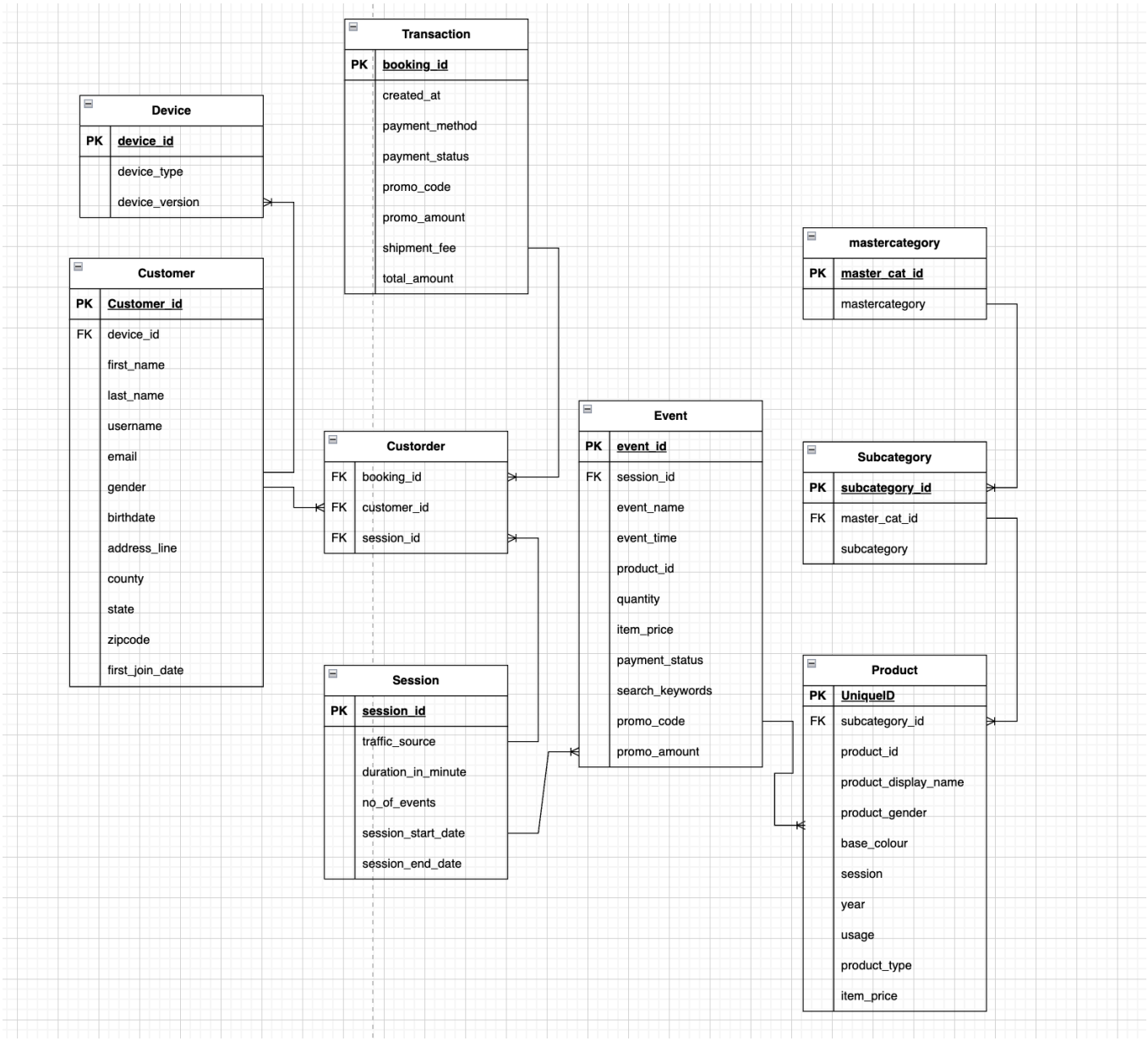
Subcategory
Subcategory_id, master_cat_id, subcategory.

MasterCategory
master_cat_ID, mastercategory

Product
UniqueID, subcategory_id, product_id, product_display_name, product_gender, base_colour, session, year, usage, product_type, item_price

Event
Event_id, session_id, event_name, event_name, event_time, product_id, quantity, item_price, payment_status, search_keywords, promo_code, promo_amount

Relational Model:



Customer: This Customer table stores the information regarding all the customers in the platform. The attributes of the table includes:

Attributes	**Data Type**	**Key**	**Constraint & Comments**
customer_ID	INTEGER	PRIMARY	NOT NULL
device_ID	INTEGER	FOREIGN	FOREIGN Key which references device_ID from Device. This is used to fetch the details of the type of device customer is using.
first_name	VARCHAR(100)		NOT NULL
last_name	VARCHAR(100)		
username	VARCHAR(100)		NOT NULL
email	VARCHAR(150)		NOT NULL
birthdate	DATE		
age	INTEGER		This age is calculated field from the birthdate attribute.
address_line	VARCHAR(100)		
county	VARCHAR(100)		
state	VARCHAR(50)		
zipcode	INTEGER		
first_join_date	DATE		This attribute contains first signed up date in the application.
gender	VARCHAR(5)		

Device: This Device table has the data containing the type of device, customer is using. The attributes of the table includes:

Attributes	**Data Type**	**Key**	**Constraint & Comments**
device_ID	INTEGER	PRIMARY	NOT NULL
device_type	VARCHAR(100)		
Device_version	VARCHAR(100)		

Transaction: This is a driving table of the database which stores the information about the transactions, customer has made.

Attributes	**Data Type**	**Key**	**Constraint & Comments**
booking_ID	VARCHAR(100)	PRIMARY	NOT NULL
created_at	TIMESTAMP		
Payment_method	VARCHAR(100)		
Payment_status	VARCHAR(100)		
promo_code	VARCHAR(100)		
promo_amount	INTEGER		
shipment_fee	DOUBLE		

Total_amount	VARCHAR(100)		

Custorder: The data contained in this custorder table has details about order details. The attributes of the table include:

Attributes	**Data Type**	**Key**	**Constraint & Comments**
booking_id	VARCHAR(100)	FOREIGN	NOT NULL
customer_id	INTEGER	FOREIGN	NOT NULL
session_id	VARCHAR(100)	FOREIGN	NOT NULL

Product: This Product table stores the details about the description of the product along with year.

Attributes	**Data Type**	**Key**	**Constraint & Comments**
Product_ID	INTEGER	PRIMARY	NOT NULL
subcategory_id	INTEGER	FOREIGN	FOREIGN Key which references master_cat_ID from MasterCategory.
product_display_name	VARCHAR(100)		
product_gender	VARCHAR(10)		
base_colour	VARCHAR(20)		
season	VARCHAR(100)		
year	VARCHAR(100)		
usage	VARCHAR(100)		
product_type	VARCHAR(200)		
Item_price	NUMERIC		

MasterCategory: This table has the details of category of the product like Apparel, Accessories, etc.

Attributes	Data Type	Key	Constraint & Comments
master_cat_ID	INTEGER	PRIMARY	NOT NULL
mastercategory	VARCHAR(100)		

SubCategory: In this table, it contains the details of product subcategory like Topwear, Bottomwear, etc., where the parent entity is MasterCategory.

Attributes	Data Type	Key	Constraint & Comments
subcategory_ID	INTEGER	PRIMARY	NOT NULL
Master_cat_id	INTEGER	FOREIGN	
subcategory	VARCHAR(100)		

Article: In this table, it contains the details of product subcategory like Topwear, Bottomwear, etc., where the parent entity is MasterCategory.

Attributes	Data Type	Key	Constraint & Comments
article_ID	INTEGER	PRIMARY	NOT NULL
season_ID	INTEGER	FOREIGN	
article_type	VARCHAR(100)		
base_color	VARCHAR(100)		

Session:

Attributes	Data Type	Key	Constraint & Comments
season_ID	INTEGER	PRIMARY	NOT NULL
traffic_source	VARCHAR(100)		
duration_in_mins	VARCHAR(200)		
no_of_events	INTEGER		
session_start_date	TIMESTAMP		
session_end_date	TIMESTAMP		

Event:

Attributes	Data Type	Key	Constraint & Comments
event_id	VARCHAR(200)	PRIMARY	NOT NULL
Session_id	VARCHAR(200)	FOREIGN	
Event_name	VARCHAR(200)		
Event_time	INTEGER		
Product_id	TIMESTAMP		
quantity	TIMESTAMP		
Item_price	DOUBLE		
Payment_status	VARCHAR(200)		
Search_keywords	VARCHAR(200)		
Promo_code	VARCHAR(200)		
Promo_amount	DOUBLE		

Implementation:

```
CREATE TABLE Customer (
    customer_ID INTEGER PRIMARY KEY NOT NULL,
    device_ID INTEGER NOT NULL,
    first_name VARCHAR(100) NOT NULL,
    last_name VARCHAR(100),
    username VARCHAR(100) NOT NULL,
    email VARCHAR(150) NOT NULL,
    birthdate DATE,
    age INTEGER,
    address_line VARCHAR(100),
    county VARCHAR(100),
    state VARCHAR(50),
    zipcode INTEGER,
    first_join_date DATE,
    gender VARCHAR(5),
    FOREIGN KEY (device_ID) REFERENCES Device(device_ID)
);

CREATE TABLE Device (
    device_ID INTEGER PRIMARY KEY NOT NULL,
    device_type VARCHAR(100),
    Device_version VARCHAR(100)
);

CREATE TABLE Transaction (
    booking_ID VARCHAR(100) PRIMARY KEY NOT NULL,
    created_at TIMESTAMP,
    Payment_method VARCHAR(100),
    Payment_status VARCHAR(100),
    promo_code VARCHAR(100),
    promo_amount INTEGER,
    shipment_fee DOUBLE,
    Total_amount VARCHAR(100)
);

CREATE TABLE Custorder (
    booking_id VARCHAR(100) NOT NULL,
    customer_id INTEGER NOT NULL,
    session_id VARCHAR(100) NOT NULL,
    PRIMARY KEY (booking_id, customer_id, session_id),
    FOREIGN KEY (booking_id) REFERENCES Transaction(booking_ID),
     FOREIGN KEY (customer_id) REFERENCES Customer(customer_ID),
    FOREIGN KEY (session_id) REFERENCES Session(session_ID)
);

CREATE TABLE Product (
    Product_ID INTEGER PRIMARY KEY NOT NULL,
    subcategory_id INTEGER,
```

```
    product_display_name VARCHAR(100),
    product_gender VARCHAR(10),
    base_colour VARCHAR(20),
    season VARCHAR(100),
    year VARCHAR(100),
    usage VARCHAR(100),
    product_type VARCHAR(200),
    Item_price NUMERIC,
    FOREIGN KEY (subcategory_id) REFERENCES
SubCategory(subcategory_ID)
);

CREATE TABLE MasterCategory (
    master_cat_ID INTEGER PRIMARY KEY NOT NULL,
    mastercategory VARCHAR(100)
);

CREATE TABLE SubCategory (
    subcategory_ID INTEGER PRIMARY KEY NOT NULL,
    Master_cat_id INTEGER,
    subcategory VARCHAR(100),
    FOREIGN KEY (Master_cat_id) REFERENCES
MasterCategory(master_cat_ID)
);


CREATE TABLE Session (
    season_ID INTEGER PRIMARY KEY NOT NULL,
    traffic_source VARCHAR(100),
    duration_in_mins VARCHAR(200),
    no_of_events INTEGER,
    session_start_date TIMESTAMP,
    session_end_date TIMESTAMP
);

CREATE TABLE Event (
    event_id VARCHAR(200) PRIMARY KEY NOT NULL,
    Session_id VARCHAR(200),
    Event_name VARCHAR(200),
    Event_time INTEGER,
    Product_id TIMESTAMP,
    quantity TIMESTAMP,
    Item_price DOUBLE,
    Payment_status VARCHAR(200),
    Search_keywords VARCHAR(200),
    Promo_code VARCHAR(200),
    Promo_amount DOUBLE,
    FOREIGN KEY (Session_id) REFERENCES Session(season_ID),
    FOREIGN KEY (Product_id) REFERENCES Product(Product_ID)
);
```

Data Population Methodology:

We are going to use the data which we have fetched from our data source which is CSV format which includes Clickstream, Product Mastercategory, Subcategory, Article, Season, Customer and Transactional Data. Apart from that we have added multiple columns (Address, State, County and Zip code)for that we are going to create our own data using python.

Warehouse Design Proposal:

The MultiDim schema could have the following structure:

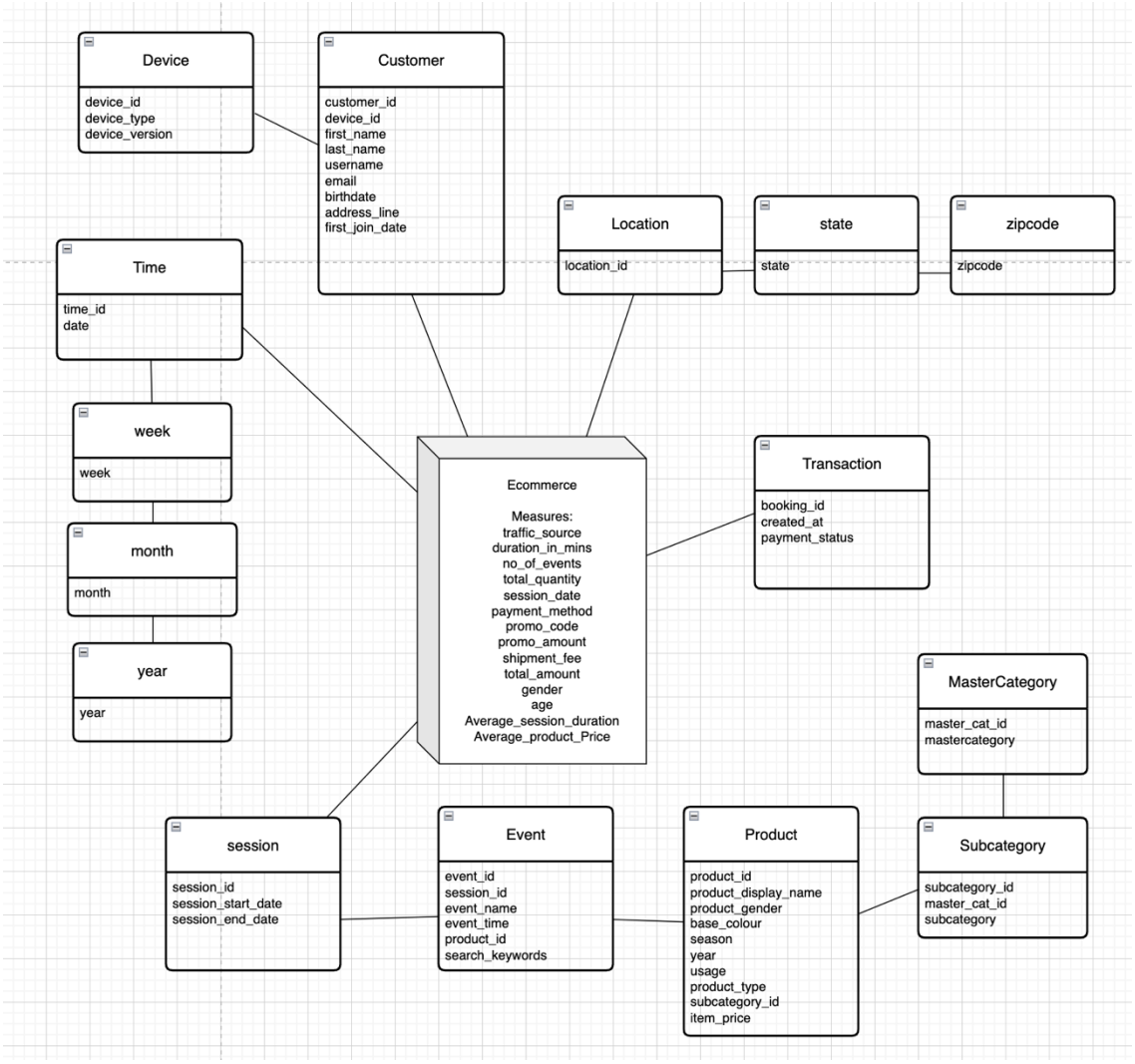
Dimensions:

- customer (customer_id, device_id, first_name, last_name, username, email, gender, birthdate, age, address_line, first_join_date, prev_username)
- Device (device_id, device_type, device_version)
- Time (time_id, date, week, month, quarter, year)
- Transaction (booking_id, created_at, payment_method, payment_status, promo_code)
- Location (location_id, state, city, zipcode)
- MasterCategory (master_cat_id, mastercategory)
- Subcategory (subcategory_id, master_cat_id, subcategory)
- Product (product_id, product_display_name, product_gender, base_colour, season, year, usage, product_type, subcategory_id, item_price)
- Session (session_id, traffic_source,session_start_date, session_end_date)
- Event (event_id, session_id, event_name, event_time, product_id, search_keywords)

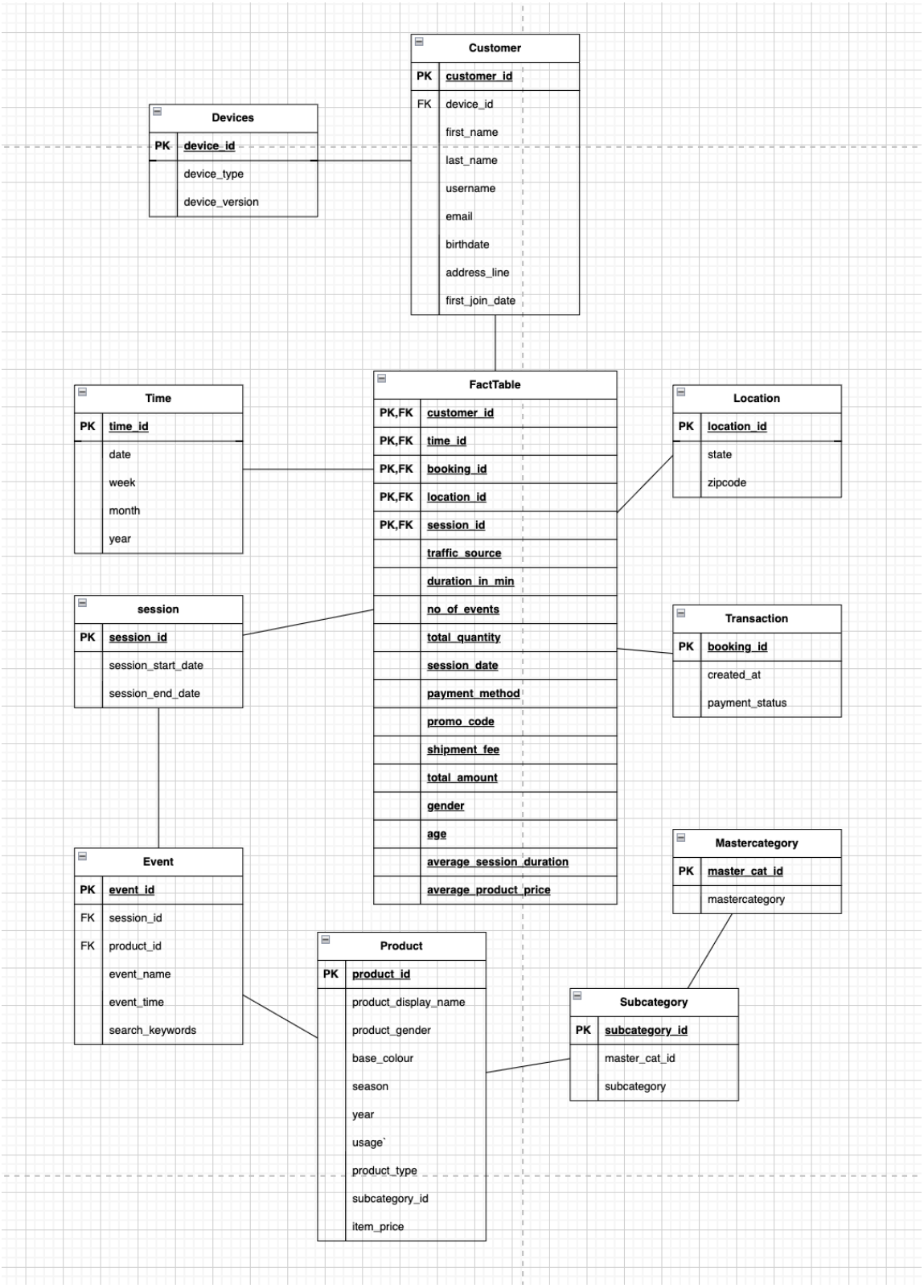
Measures:

- traffic_source
- duration_in_mins
- no_of_events
- total_quantity
- session_date
- payment_method
- promo_code
- promo_amount
- shipment_fee
- total_amount
- gender
- age
- Average_session_duration
- Average_product_Price

Conceptual Model:



Logical Model:



SCHEMA IMPLEMENTATION:

```
CREATE TABLE FactTable (
    customer_id INT,
    time_id INT,
    booking_id INT,
    location_id INT,
    session_id INT,
    traffic_source VARCHAR(50),
    duration_in_min INT,
    no_of_events_total_quantity INT,
    session_date DATE,
    payment_method VARCHAR(20),
    promo_code VARCHAR(20),
    shipment_fee DECIMAL(10, 2),
    total_amount DECIMAL(10, 2),
    gender CHAR(1),
    age INT,
    average_session_duration DECIMAL(10, 2),
    average_product_price DECIMAL(10, 2),
    PRIMARY KEY (customer_id, time_id, booking_id, location_id, session_id
),
    FOREIGN KEY (customer_id) REFERENCES Customer(customer_id),
    FOREIGN KEY (time_id) REFERENCES Time(time_id),
    FOREIGN KEY (booking_id) REFERENCES Transaction(booking_id),
    FOREIGN KEY (location_id) REFERENCES Location(location_id),
    FOREIGN KEY (session_id) REFERENCES Session(session_id)
);



CREATE TABLE IF NOT EXISTS public.customer_dim
 (
        cust_key integer NOT NULL DEFAULT
nextval('customer_dim_cust_key_seq'::regclass),
        customer_id integer NOT NULL,
        device_id character varying(100) COLLATE pg_catalog."default",
        first_name character varying(100) COLLATE pg_catalog."default",
        last_name character varying(100) COLLATE pg_catalog."default",
        username character varying(100) COLLATE pg_catalog."default",
        email character varying(150) COLLATE pg_catalog."default",
        gender character varying(5) COLLATE pg_catalog."default",
        birthdate date,
        age integer,
```

```
    address_line character varying(100) COLLATE pg_catalog."default",
    first_join_date date,
    prev_username character varying(200) COLLATE pg_catalog."default",
    CONSTRAINT customer_dim_pkey PRIMARY KEY (cust_key)
);


CREATE TABLE IF NOT EXISTS public.customers_dim
(
    customer_id integer NOT NULL,
    device_id character varying(100) COLLATE pg_catalog."default",
    first_name character varying(100) COLLATE pg_catalog."default",
    last_name character varying(100) COLLATE pg_catalog."default",
    username character varying(100) COLLATE pg_catalog."default",
    email character varying(150) COLLATE pg_catalog."default",
    gender character varying(5) COLLATE pg_catalog."default",
    birthdate date,
    age integer,
    first_join_date date,
    CONSTRAINT customers_dim_pkey PRIMARY KEY (customer_id)
);


CREATE TABLE IF NOT EXISTS public.device_dim
(
    device_id character varying(100) COLLATE pg_catalog."default" NOT NULL,
    device_type character varying(100) COLLATE pg_catalog."default",
    device_version character varying(100) COLLATE pg_catalog."default",
    CONSTRAINT pk_device PRIMARY KEY (device_id)
);



CREATE TABLE IF NOT EXISTS public.event_dim
(
    event_id character varying(200) COLLATE pg_catalog."default" NOT NULL,
    session_id character varying(200) COLLATE pg_catalog."default",
    event_name character varying(100) COLLATE pg_catalog."default",
    event_time timestamp without time zone,
    product_id integer,
    quantity double precision,
    search_keywords character varying(200) COLLATE pg_catalog."default",
```

```
        CONSTRAINT pk_event PRIMARY KEY (event_id)
);



CREATE TABLE IF NOT EXISTS public.location_dim
(
        location_id integer NOT NULL,
        state character varying(200) COLLATE pg_catalog."default",
        city character varying(200) COLLATE pg_catalog."default",
        zipcode integer,
        CONSTRAINT location_dim_pkey PRIMARY KEY (location_id)
);



CREATE TABLE IF NOT EXISTS public.mastercategory_dim
(
        master_cat_id integer NOT NULL,
        mastercategory character varying(100) COLLATE pg_catalog."default",
        CONSTRAINT mastercategory_pkey PRIMARY KEY (master_cat_id)
);



CREATE TABLE IF NOT EXISTS public.product_dim
(
        product_id integer NOT NULL,
        product_display_name character varying(100) COLLATE pg_catalog."default",
        product_gender character varying(10) COLLATE pg_catalog."default",
        base_colour character varying(20) COLLATE pg_catalog."default",
        season character varying(100) COLLATE pg_catalog."default",
        year character varying(100) COLLATE pg_catalog."default",
        usage character varying(100) COLLATE pg_catalog."default",
        product_type character varying(200) COLLATE pg_catalog."default",
        subcategory_id integer,
        item_price numeric(10, 2),
        CONSTRAINT pk_product PRIMARY KEY (product_id)
);



CREATE TABLE IF NOT EXISTS public.products_dim
(
        product_id integer NOT NULL,
        product_display_name character varying(100) COLLATE pg_catalog."default",
        product_gender character varying(10) COLLATE pg_catalog."default",
```

```
    base_colour character varying(20) COLLATE pg_catalog."default",
    season character varying(100) COLLATE pg_catalog."default",
    year character varying(100) COLLATE pg_catalog."default",
    usage character varying(100) COLLATE pg_catalog."default",
    "Product_type" character varying(200) COLLATE pg_catalog."default",
    subcategory_id integer,
    item_price numeric(10, 2),
    created_on timestamp without time zone,
    modified_on timestamp without time zone,
    CONSTRAINT pk_products PRIMARY KEY (product_id)
);


CREATE TABLE IF NOT EXISTS public.session_dim
(
    session_id character varying(100) COLLATE pg_catalog."default" NOT NULL,
    traffic_source character varying(100) COLLATE pg_catalog."default",
    duration_in_mins bigint,
    no_of_events integer,
    session_start_date timestamp without time zone,
    session_end_date timestamp without time zone,
    total_quantity double precision,
    CONSTRAINT pk_session PRIMARY KEY (session_id)
);




CREATE TABLE IF NOT EXISTS public.subcategory_dim
(
    subcategory_id integer NOT NULL,
    master_cat_id integer,
    subcategory character varying(100) COLLATE pg_catalog."default",
    CONSTRAINT pk_subcategory PRIMARY KEY (subcategory_id)
);


CREATE TABLE IF NOT EXISTS public.time_dim
(
    time_id integer NOT NULL DEFAULT nextval('time_dim_time_id_seq'::regclass),
    date date,
    week character(7) COLLATE pg_catalog."default",
```

```
    month character(7) COLLATE pg_catalog."default",
    quarter integer,
    year character(4) COLLATE pg_catalog."default",
    CONSTRAINT time_dim_pkey PRIMARY KEY (time_id),
    CONSTRAINT time_dim_date_key UNIQUE (date)
);
```

```
CREATE TABLE IF NOT EXISTS public.transaction_dim
(
    booking_id character varying(100) COLLATE pg_catalog."default" NOT NULL,
    created_at timestamp without time zone,
    payment_method character varying(100) COLLATE pg_catalog."default",
    payment_status character varying(100) COLLATE pg_catalog."default",
    promo_code character varying(100) COLLATE pg_catalog."default",
    promo_amount integer,
    shipment_fee integer,
    total_amount double precision,
    total_quantity integer,
    CONSTRAINT pk_transactiondim PRIMARY KEY (booking_id)
);
```

OLAP Operations:

1. Slicing - Filtering data based on one or more dimensions:

```
-- Total sales by gender in 2022

SELECT gender, SUM(total_amount) as total_sales

FROM FactTable ft

JOIN Time t ON ft.time_id = t.time_id

WHERE t.year = 2022

GROUP BY gender;
```

2. Dicing - Filtering data based on two or more dimensions:

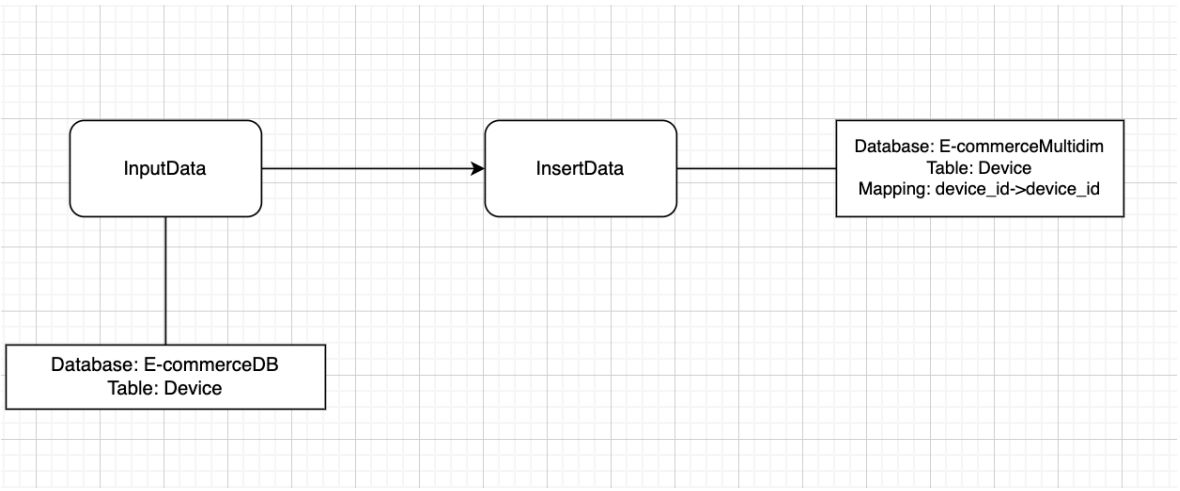
```
-- Total sales in 2022 for each state, grouped by product gender
SELECT l.state, p.product_gender, SUM(ft.total_amount) as total_sales
FROM FactTable ft
JOIN Time t ON ft.time_id = t.time_id
JOIN Location l ON ft.location_id = l.location_id
JOIN Product p ON ft.product_id = p.product_id
WHERE t.year = 2022
```

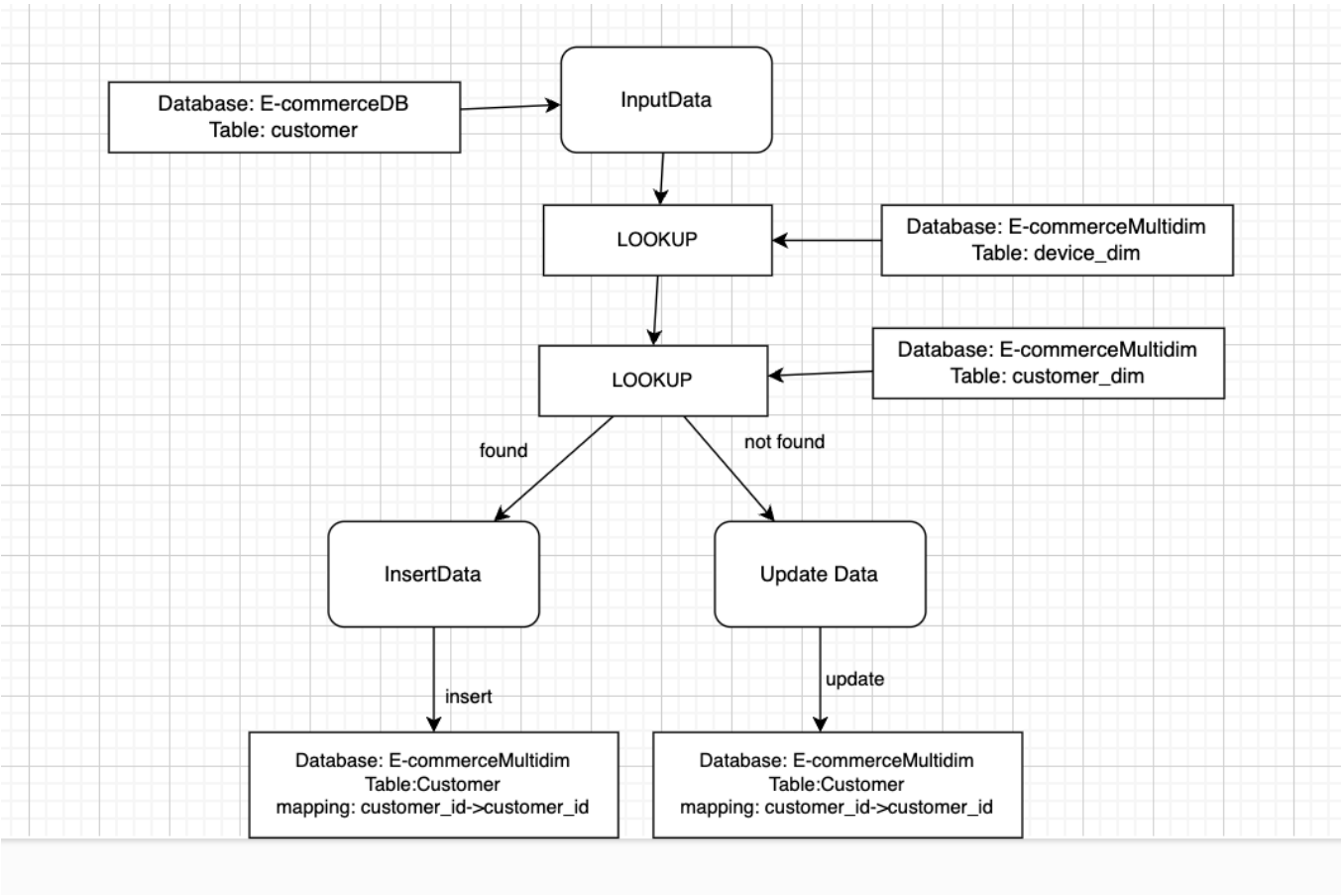
3. Roll-up - Aggregating data to a higher level of the hierarchy:

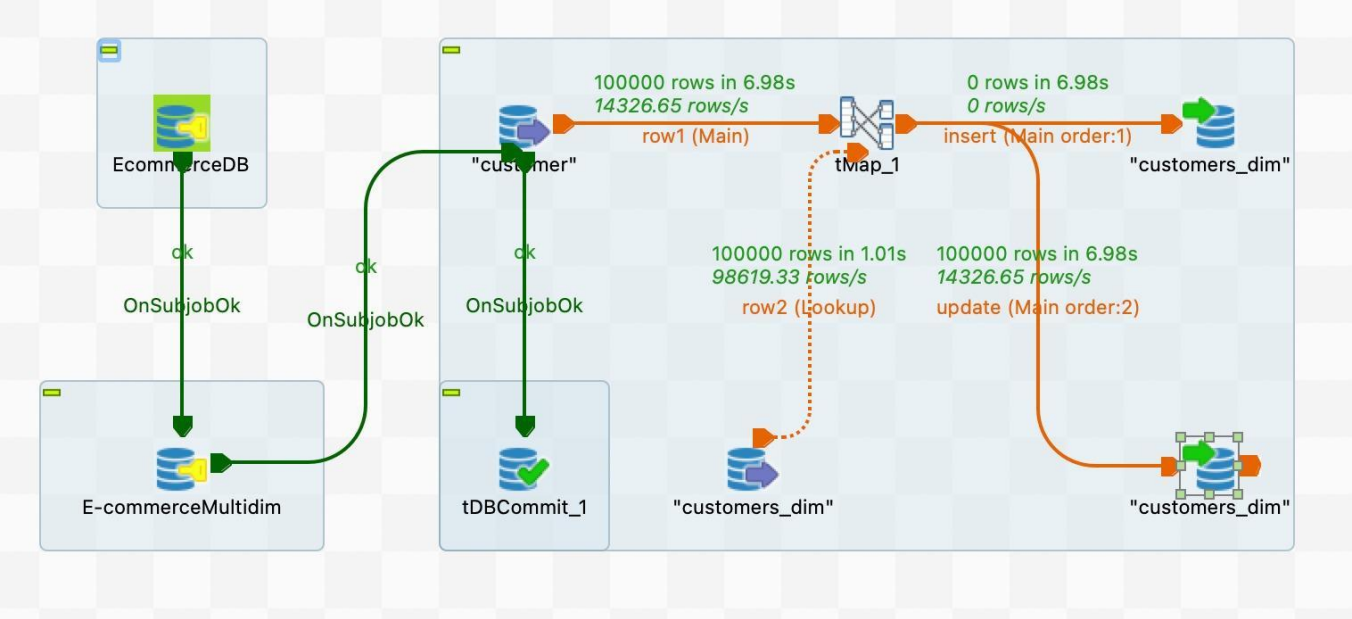
```
-- Total sales by year and quarter
SELECT t.year, CEIL(t.month / 3) as quarter, SUM(ft.total_amount) as total_sales
FROM FactTable ft
JOIN Time t ON ft.time_id = t.time_id
GROUP BY t.year, CEIL(t.month / 3)
ORDER BY t.year, quarter;
```

Control and Data Flows:

Loading device and customer dimensions:

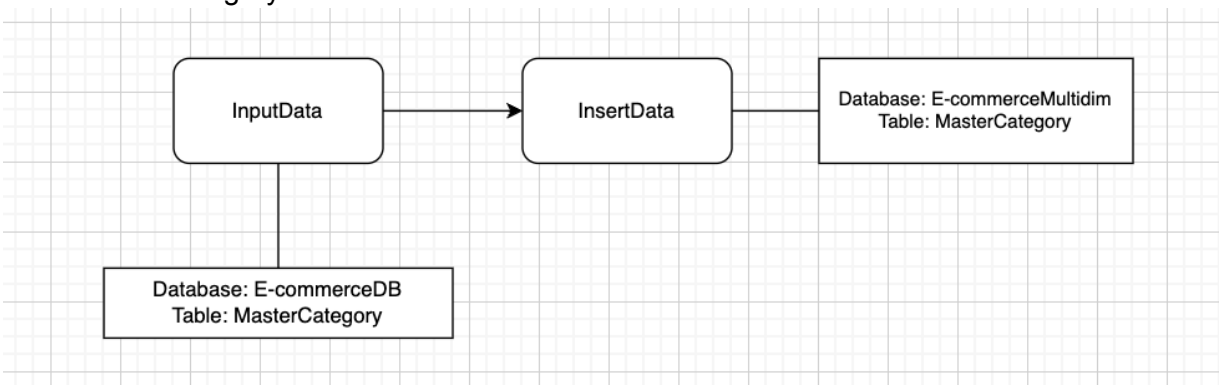

InputData
InsertData
Database: E-commerceMultidim
Table: Device
Mapping: device_id->device_id
Database: E-commerceDB
Table: Device

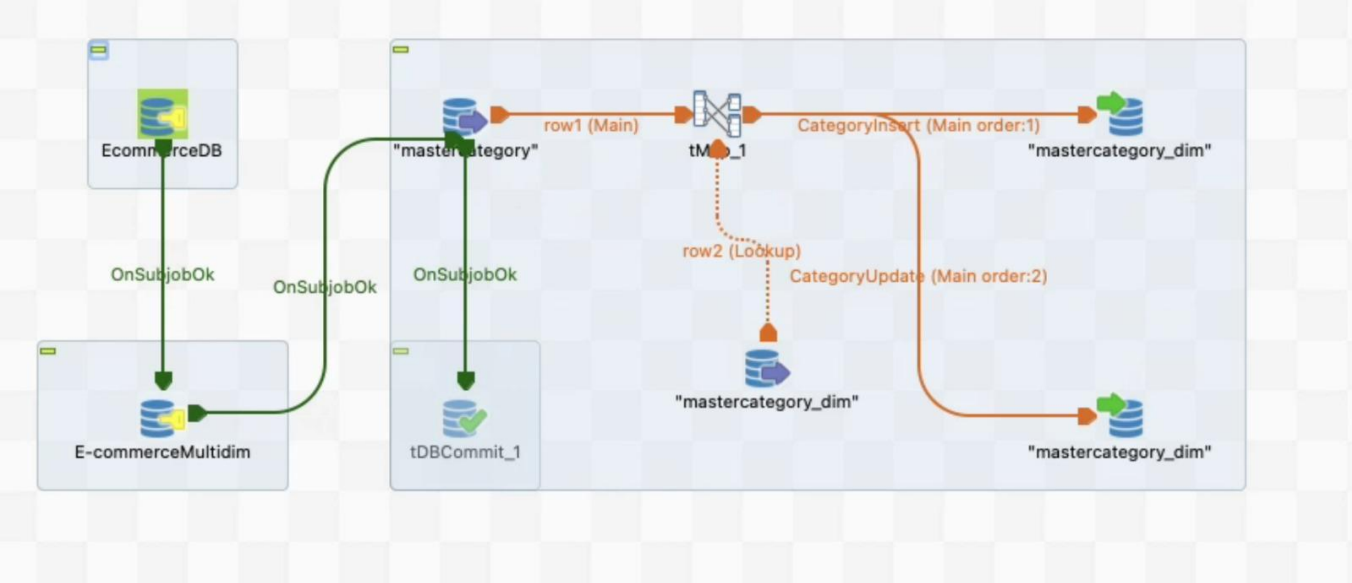

Database: E-commerceDB
Table: customer
InputData
LOOKUP
Database: E-commerceMultidim
Table: device_dim
LOOKUP
Database: E-commerceMultidim
Table: customer_dim
found
not found
InsertData
Update Data
insert
update
Database: E-commerceMultidim
Table:Customer
mapping: customer_id->customer_id
Database: E-commerceMultidim
Table:Customer
mapping: customer_id->customer_id



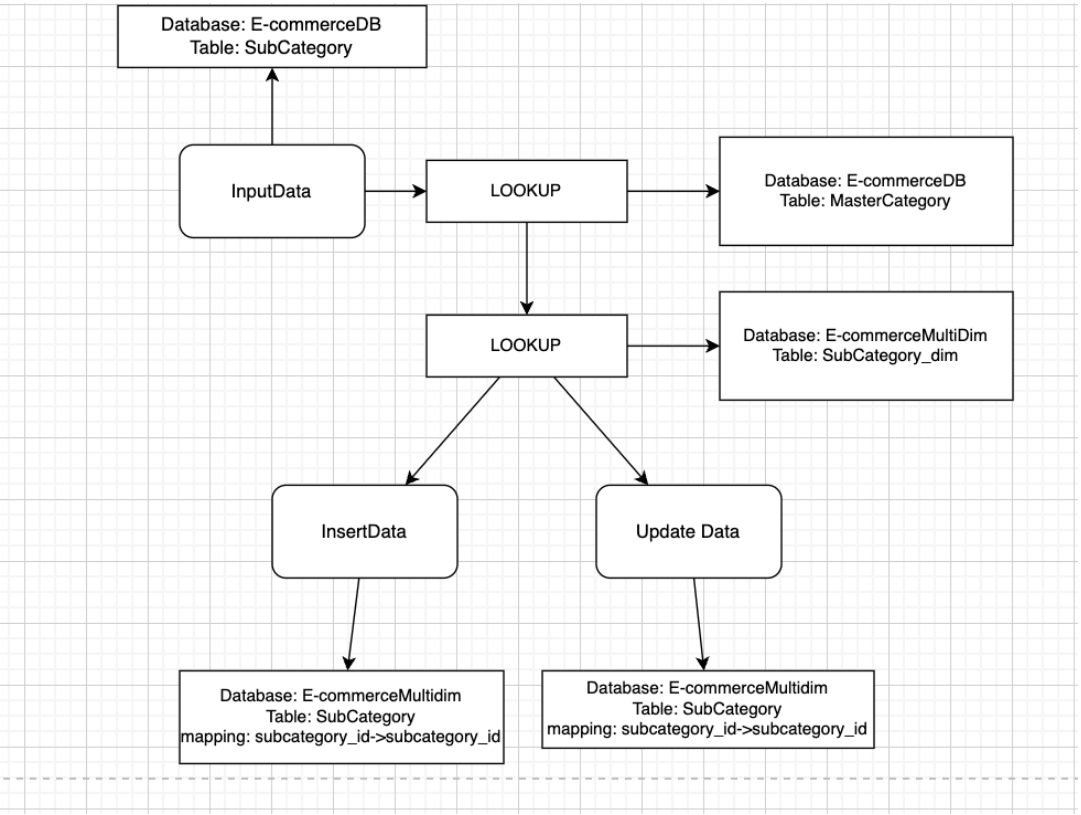
In the customer dimension we used Type 1 Slowly changing dimensions to enable simply overwrite data in dimensions. When the customer record is initiated, you may not get all attributes. Therefore, when the customer record is initiated at the operational database, there will be empty or null records in the customer records. Once the ETL is executed, those empty records will be created in the data warehouse. Once these attributes are filled in the operational databases, that has to be updated in the data warehouse.

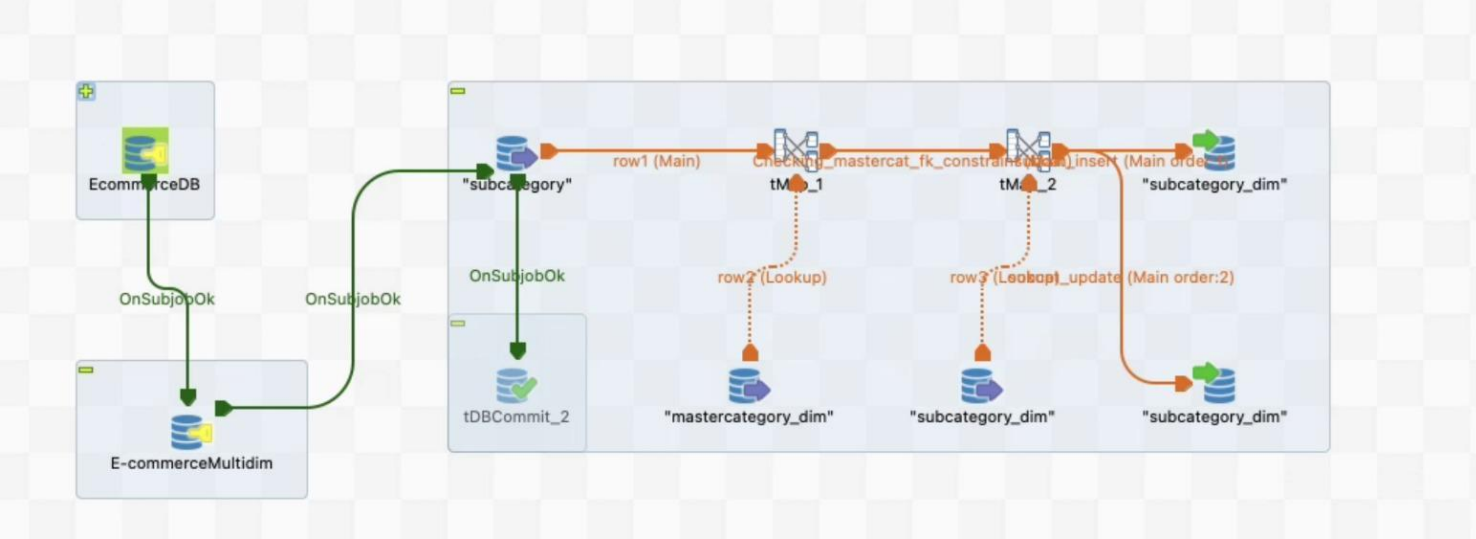
Load MasterCategory:





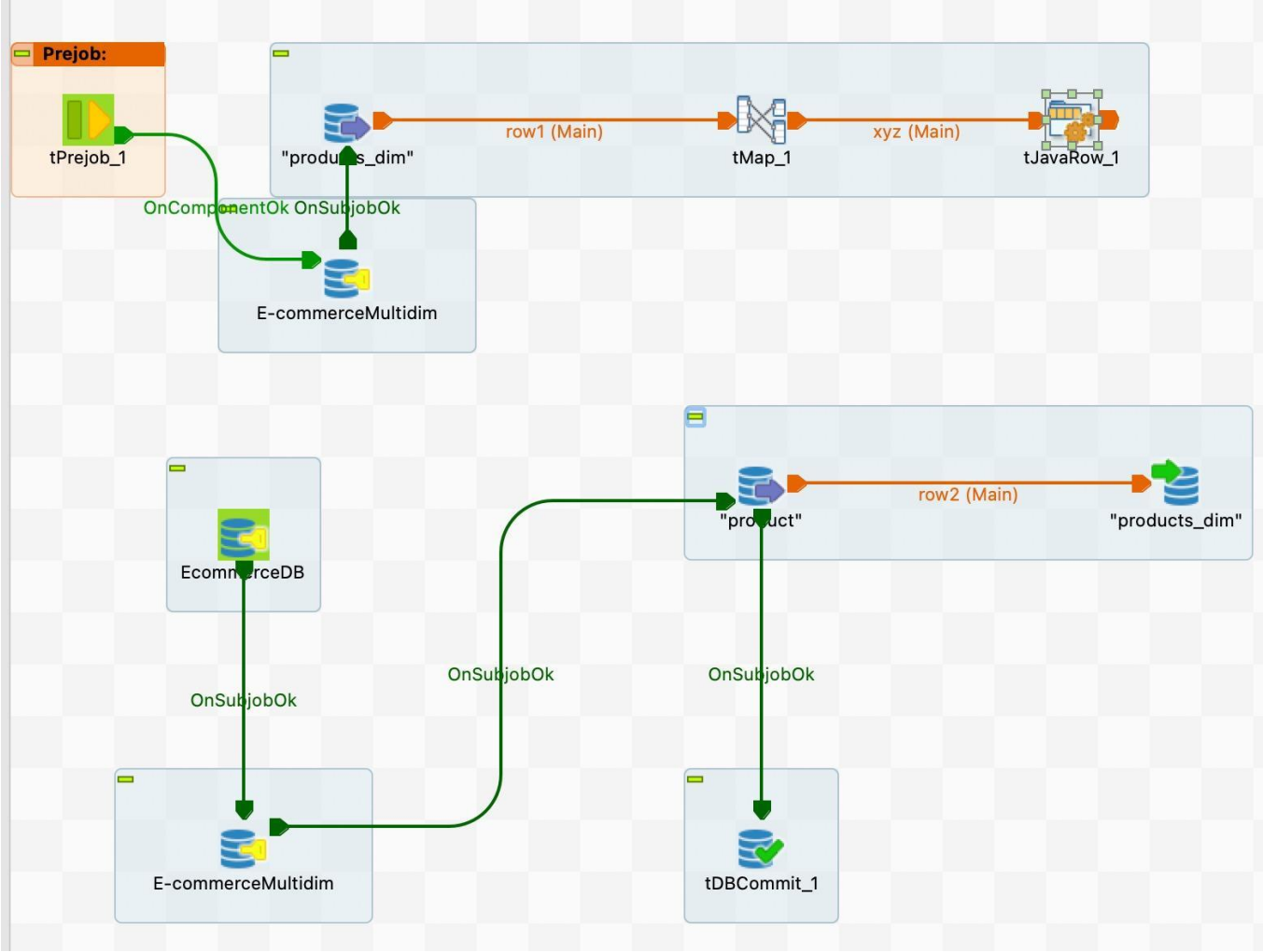
Load SubCategory:





In the subcategory dimension it looks for the mastercategory_id and proceeds to insert in the table.

Load Product:



Implemented Incremental Load:

ETL Incremental Loading is often advantageous when dealing with data sources of relatively larger sizes. Compared to Full loading, ETL Incremental Loading only uploads the data that is either newly added or changed instead of fully dumping the entire dataset.

Overview

Incremental load is a process of loading only the changes (new or modified records) from the source to the target system, instead of reloading all data. This approach saves time and resources, especially when dealing with large datasets.

In our project, we are using Talend to perform an incremental load from the E-commerce database to the E-commerce multidimensional database.

Approach

We are using a context variable to store the last_modified date of the target table `product_dim` in the E-commerce multidimensional database. We then use this variable to verify if there are new or modified records in the source table `product` in the E-commerce database. If there are, we perform an incremental load to update the target table with the changes.

The incremental load process consists of the following steps:

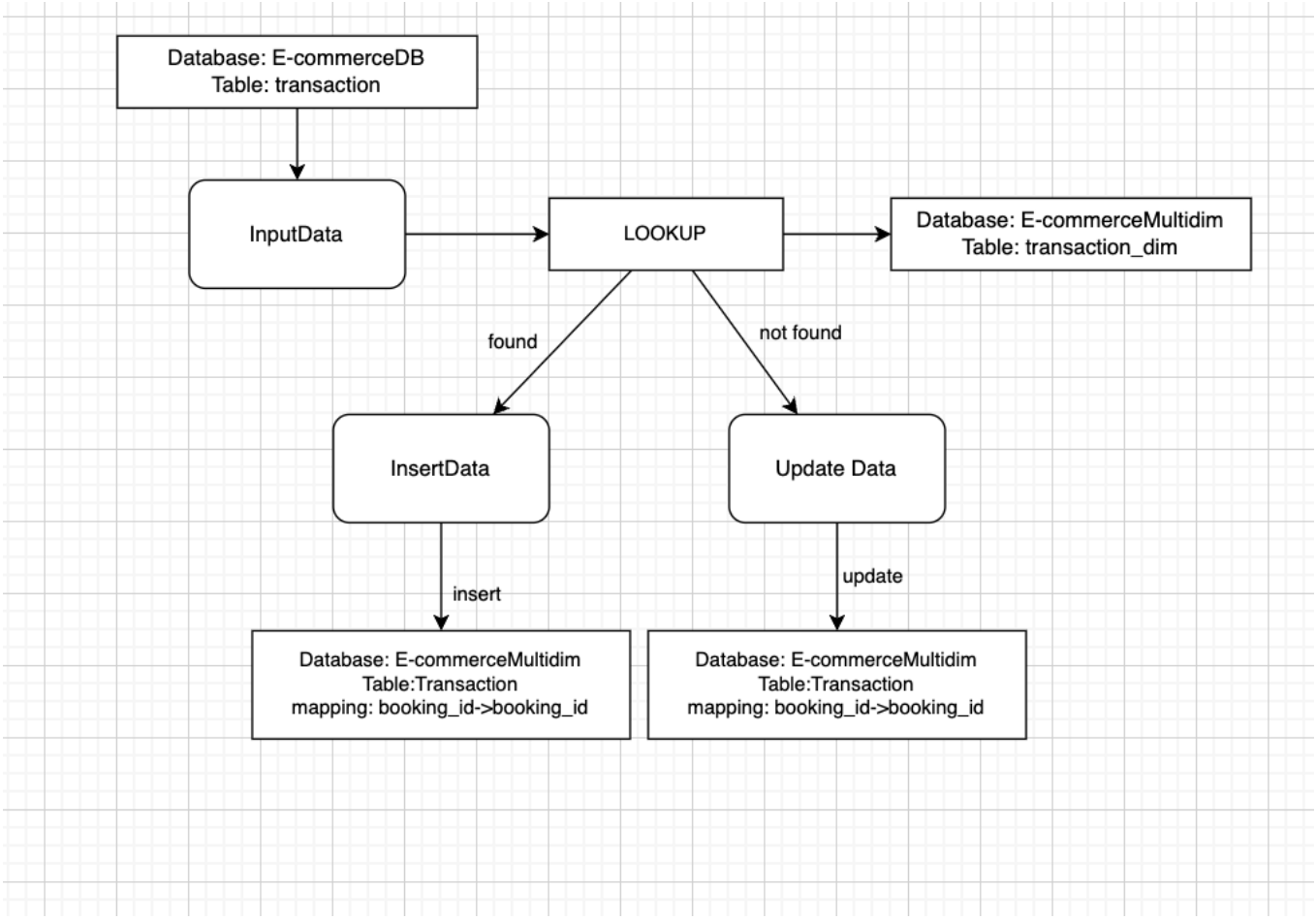
A pre-job `tJavaRow` component reads the last_modified date of the target table `product_dim` from the E-commerce multidimensional database and stores it in a context variable named `last_modified_date`. This component is executed at the beginning of the job.

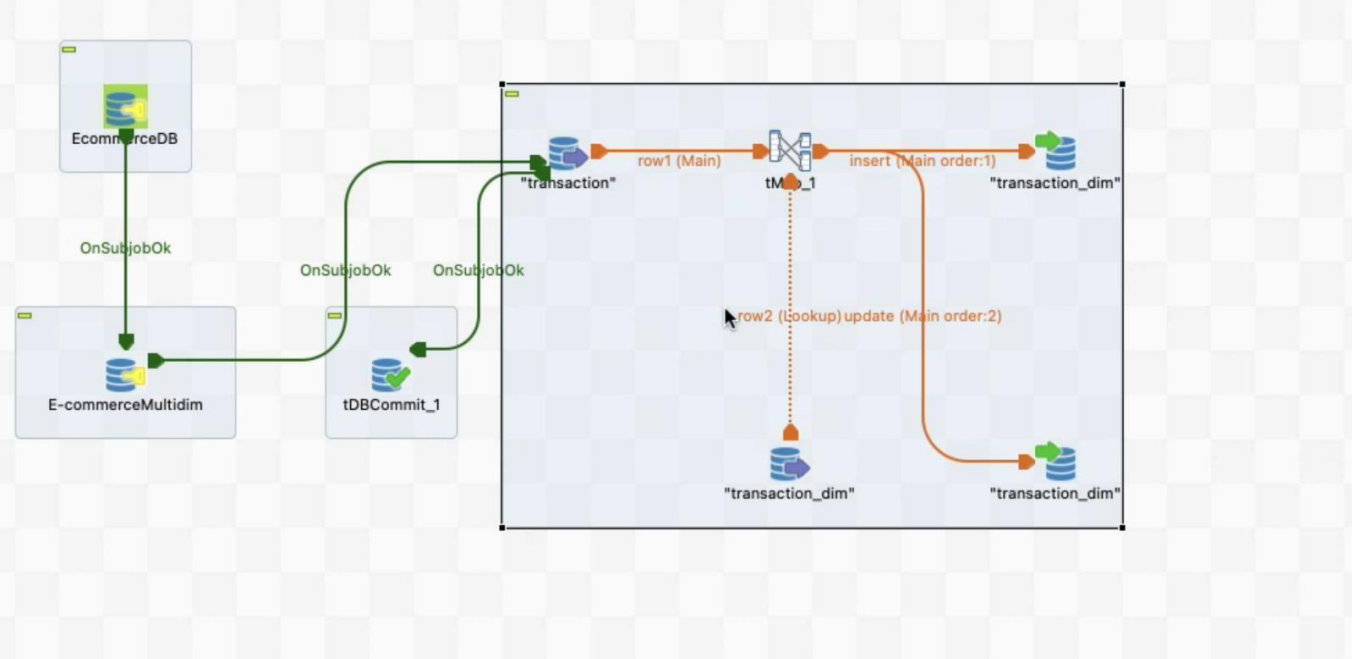
A `tJava` component reads the `last_modified_date` context variable and uses it to query the source table `product` in the E-commerce database. If there are new or modified records since the `last_modified_date`, the component sets a context variable `incremental_load` to true.

Conclusion

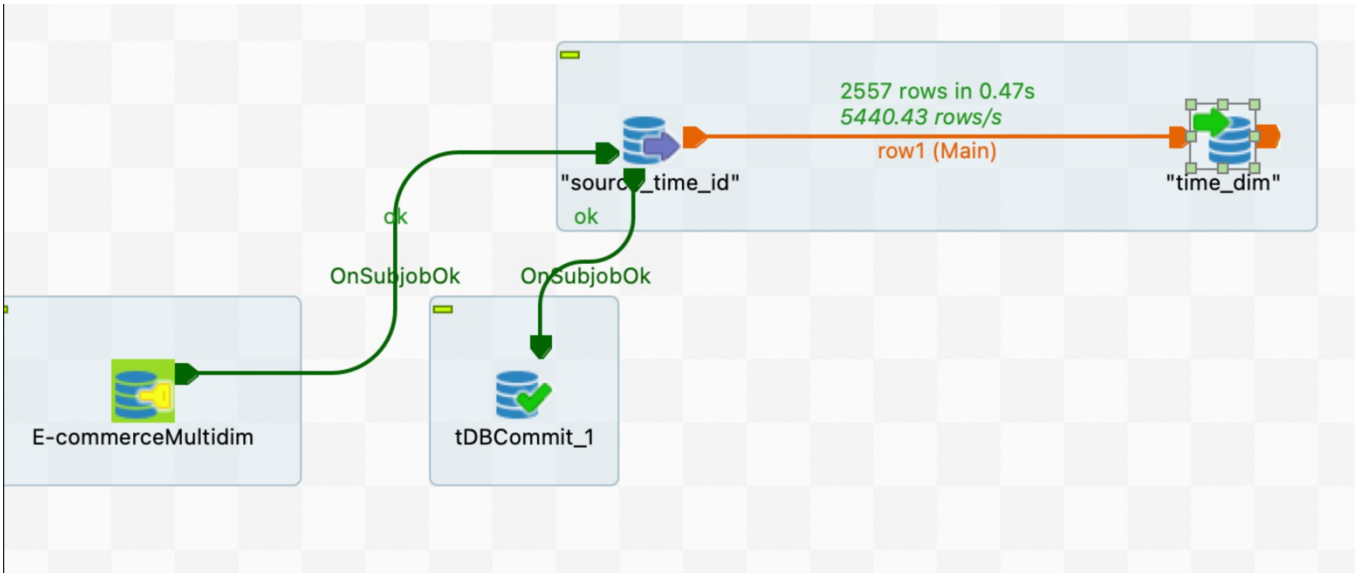
The incremental load approach is an efficient way to update the target database with only the changes from the source database. By using Talend's context variables and components such as `tJavaRow` and `tJava`, we were able to implement an incremental load process that saves time and resources. The pre-job component ensures that the last_success context variable is up-to-date before running the incremental load, while the incremental load component filters and loads only the changed records.

Load Transaction:



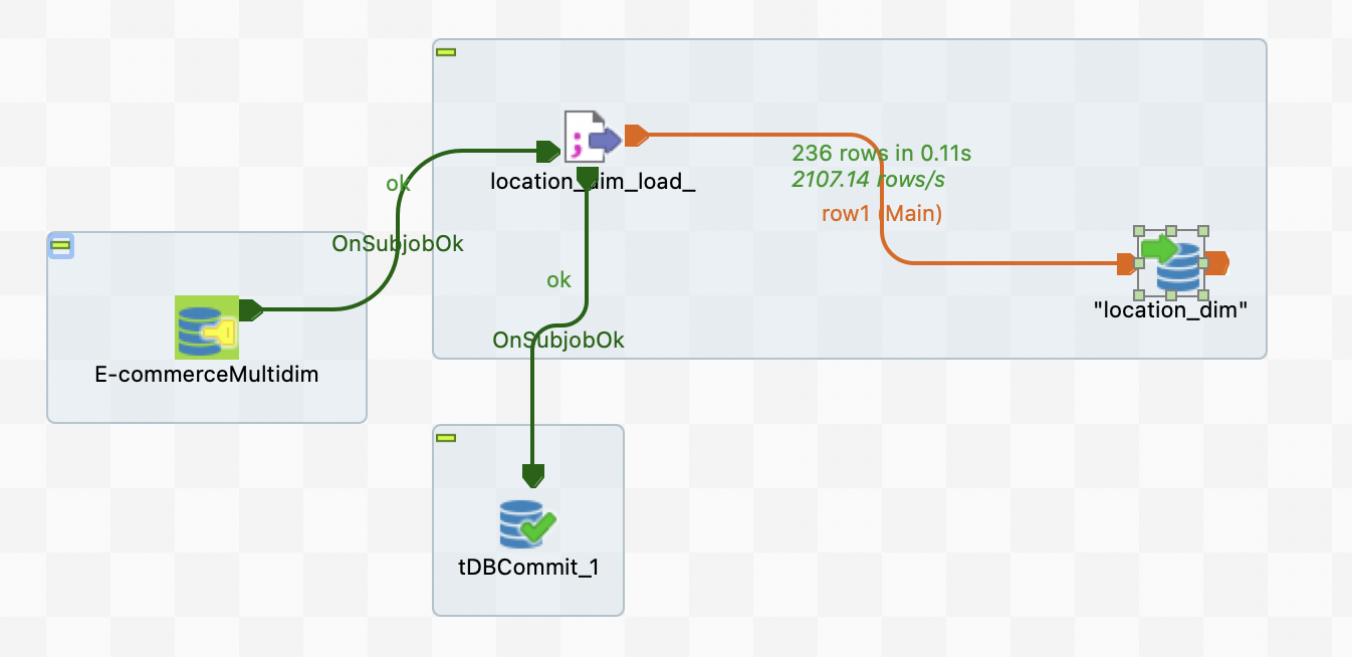


Load Time:



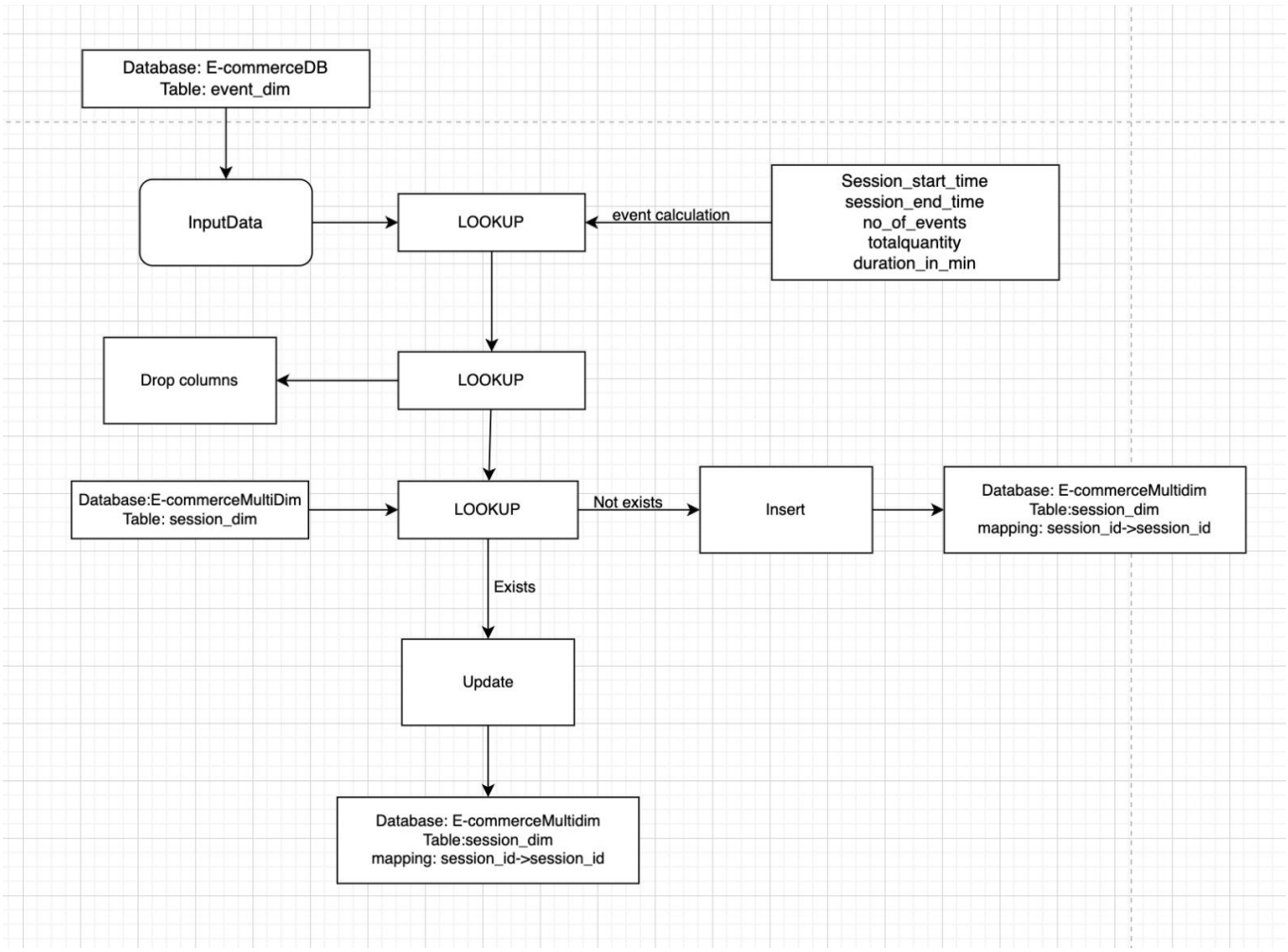
In the above ETL we extracted and loaded the data from the external source and transferred to the time dimension in data warehouse.

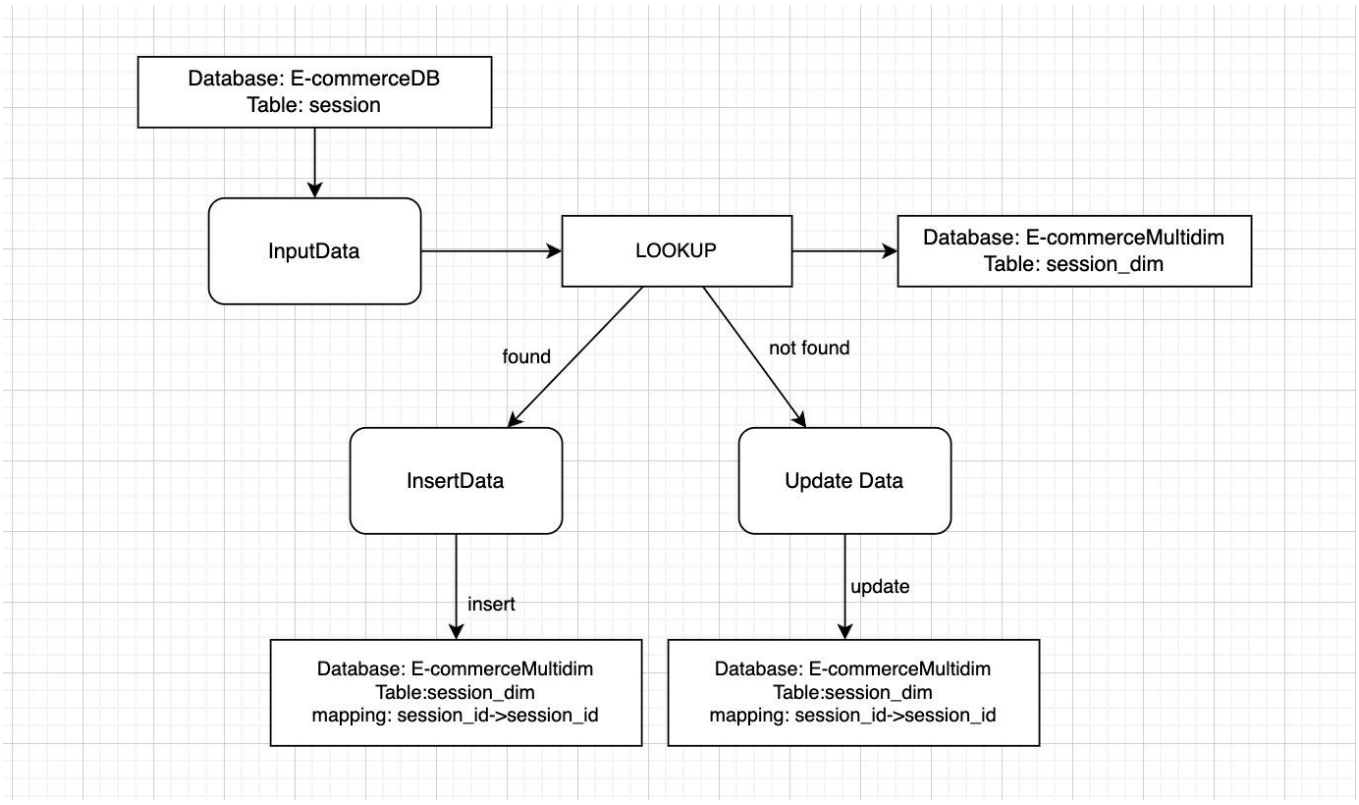
Load Location Dimension:

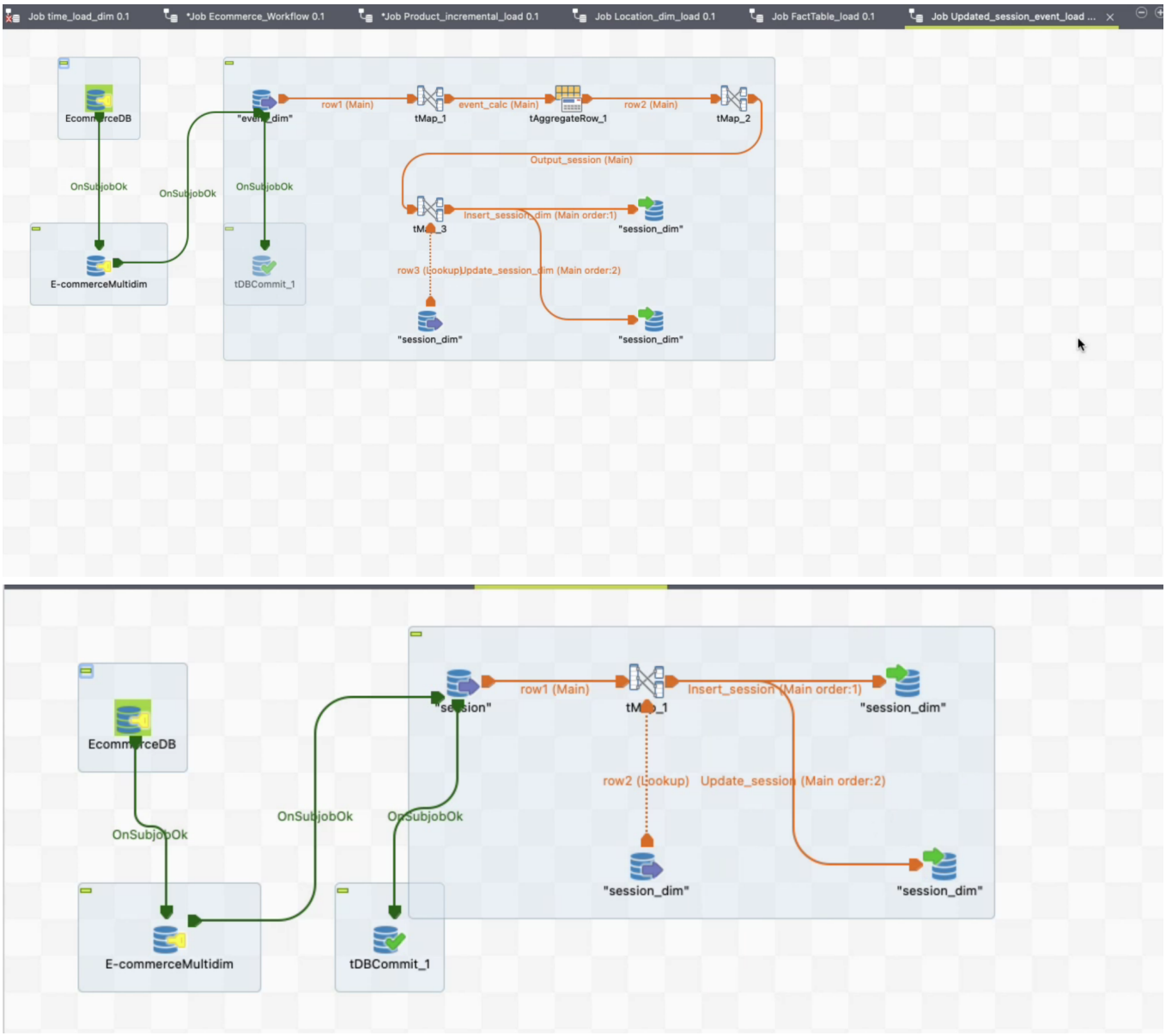


In the ETL process described, data was taken from an external source, transformed and loaded into the location dimension of a data warehouse

Load Session Dimension:

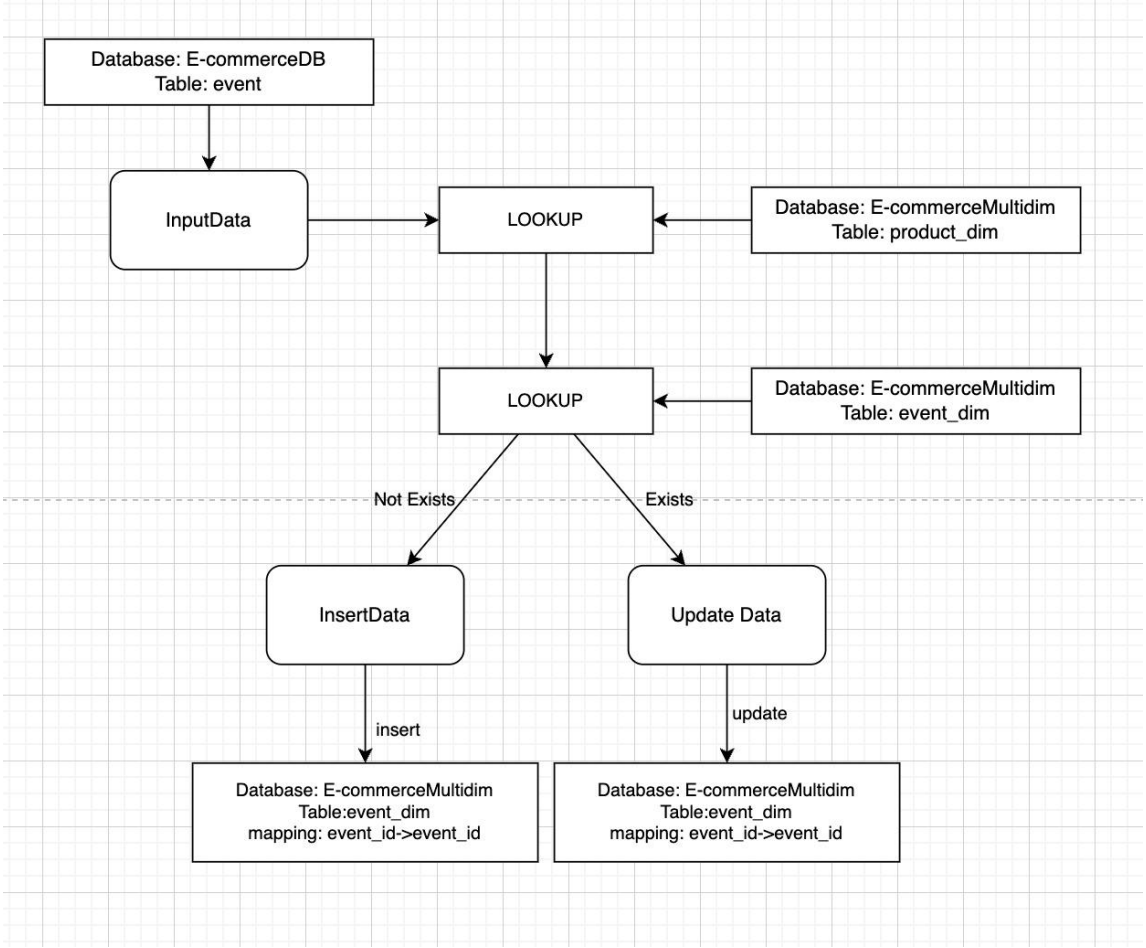

Database: E-commerceDB
Table: event_dim
InputData
LOOKUP
event calculation
Session_start_time
session_end_time
no_of_events
totalquantity
duration_in_min
Drop columns
LOOKUP
Database:E-commerceMultiDim
Table: session_dim
LOOKUP
Not exists
Insert
Database: E-commerceMultidim
Table:session_dim
mapping: session_id->session_id
Exists
Update
Database: E-commerceMultidim
Table:session_dim
mapping: session_id->session_id

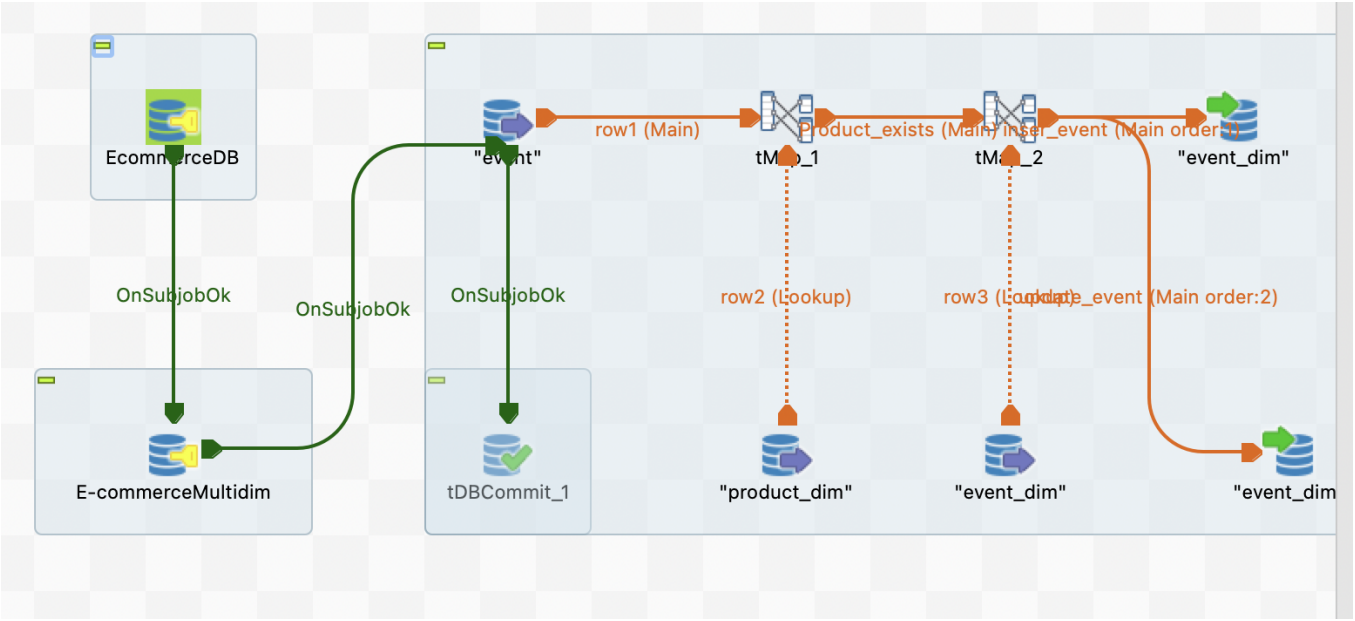

Database: E-commerceDB
Table: session
InputData
LOOKUP
Database: E-commerceMultidim
Table: session_dim
found
not found
InsertData
Update Data
insert
update
Database: E-commerceMultidim
Table:session_dim
mapping: session_id->session_id
Database: E-commerceMultidim
Table:session_dim
mapping: session_id->session_id



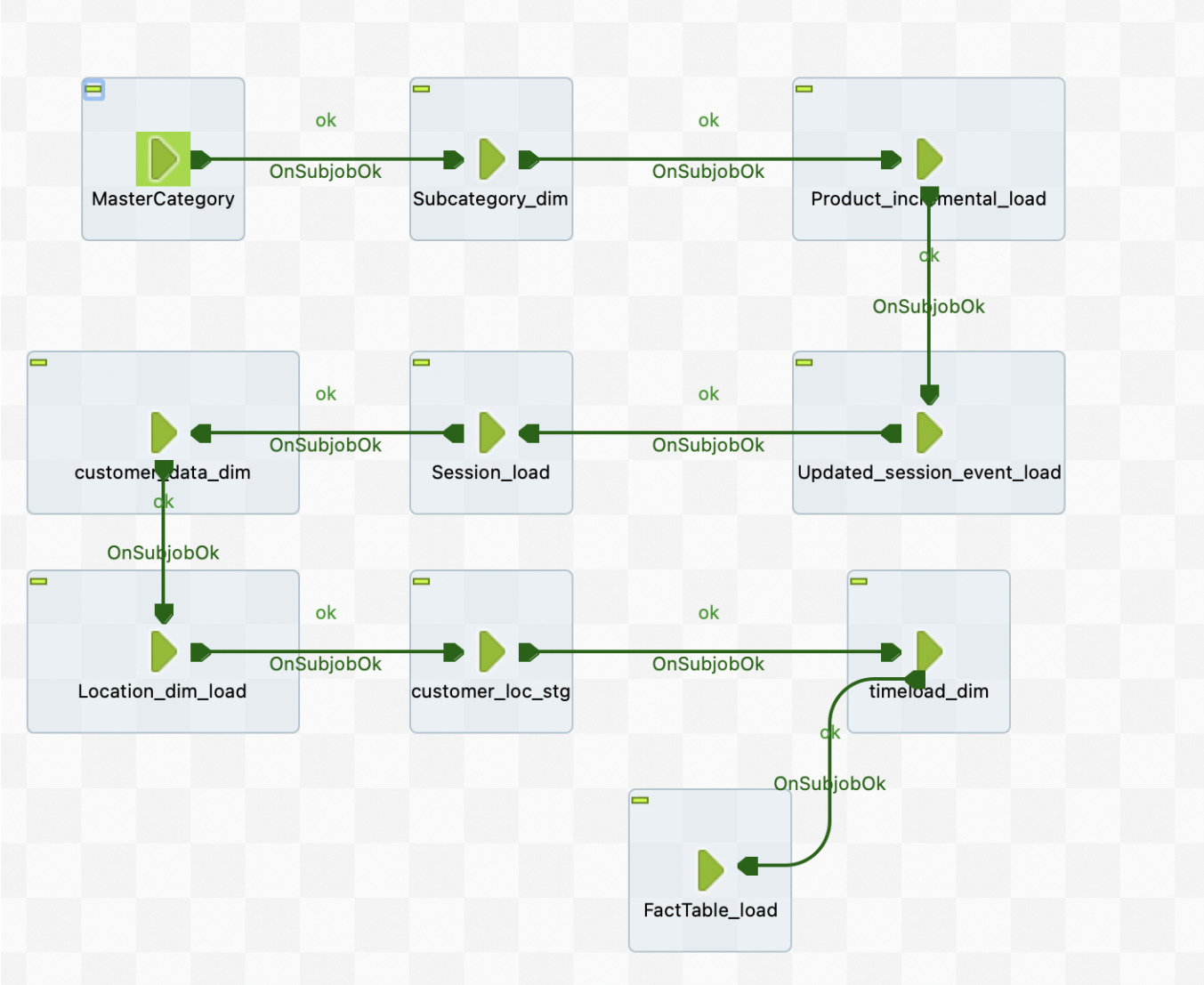
For the session dimension while extracting and loading the data, It was too big and couldn't handle the ETL process. So we created two ETL schemas to load the session dimension.

Loading the event Dimension:



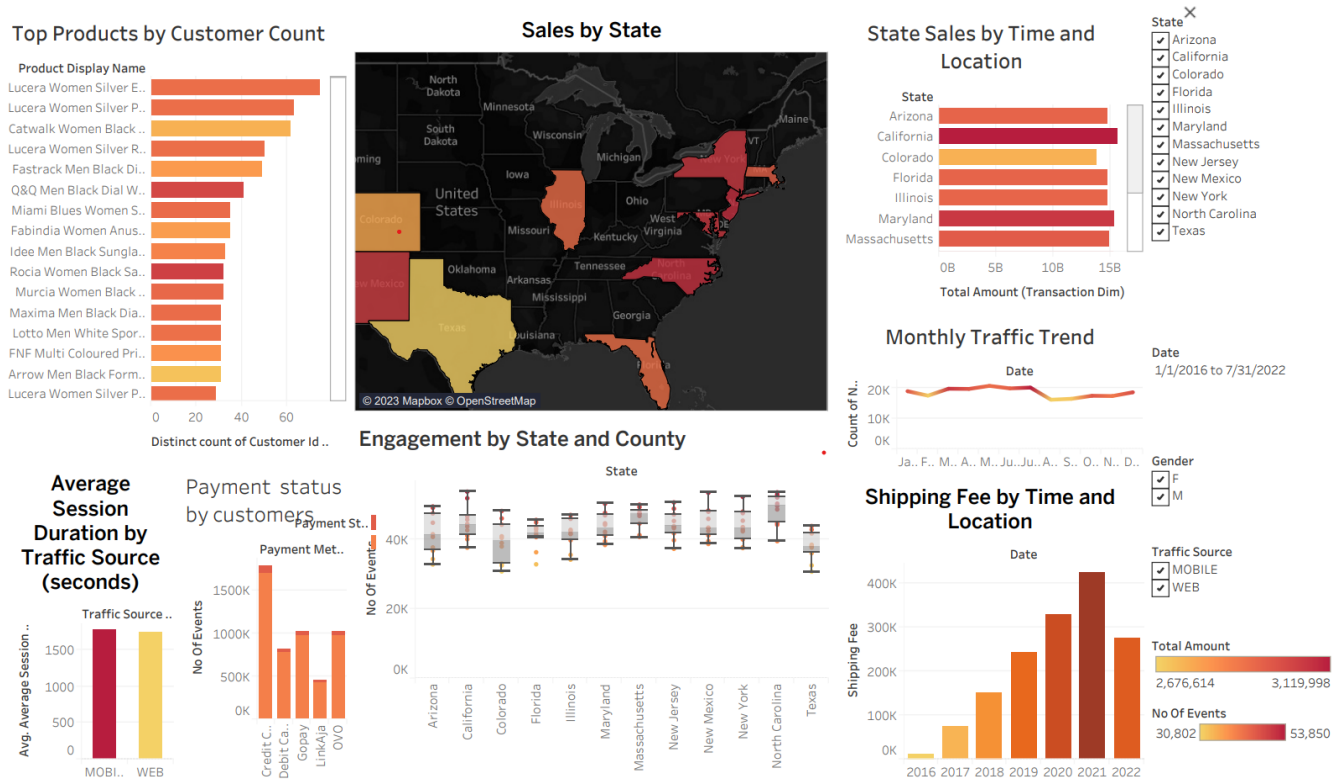


ETL process for the web traffic on ecommerce platform:



In the above ETL process we used the dummy event dimension instead of the original because here, it has more than 12 crore data which can't be handled in talend. So we implemented the event dimension with some sample data.

Analysis:



After completing Talend jobs, Once the data is pushed successfully to the Data warehouse in Postgres, It is connected to the Tableau to perform further analysis.

1. Figure out the trend in traffic on the application website month by month. From this, most traffic was in the month of May to July and less traffic is from September to October.

2. Engagement of Customers by each state and county. The session duration is shown using Box-and-whisker plot categorized by state and county.

3. Top Products bought by the customers were analyzed where Lucera Women silver earrings was the most bought item by customers.

4. Type of device, customers use to purchase products. Where most of the customers prefer to shop using mobile devices rather than web browsers.

5. Payment type and status by customers. Customers used different payment types but most of them used Credit card to purchase products.

6. Sales by each state, where most of the sales are done in the Northeast coast like in Massachusetts and Maine.

7. State sales and shipping fee analysis by time and location.