NYU

ADVANCED PROJECT I
ECE-GY 9953
BK33

PRESENTED BY Brian Catraguna & Palak Keni

Submission Date: 20 December 2023

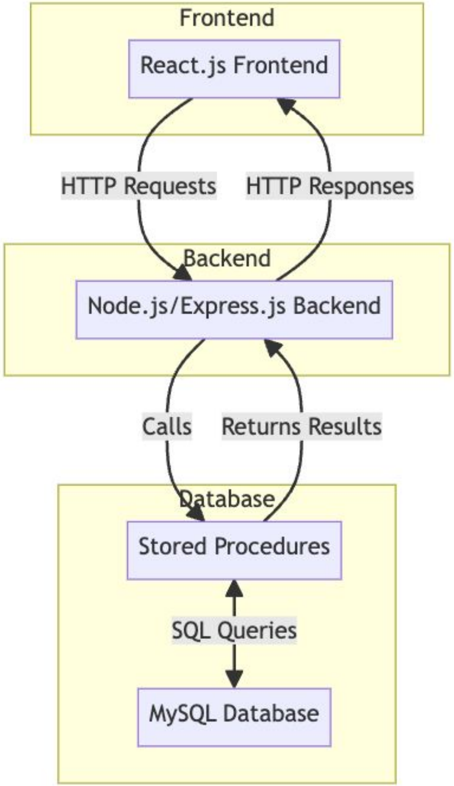
Business Case

- Awesome Superstore Inc. is a global e-commerce platform that connects sellers from around the world to customers in different markets, specializing in the categories of technology, furniture, and office supplies. The company aims to leverage technology to create a seamless online shopping experience, tapping into markets in Africa, Asia Pacific, Europe, Latin America, and the US & Canada. To achieve this, Awesome Superstore Inc. plans to develop an Online Transaction Processing (OLTP) database and a web application for day-to-day operations, along with a data warehousing system and dashboard for in-depth data analysis and business monitoring.

AWESOME INC.

NYU

Web Application Architecture

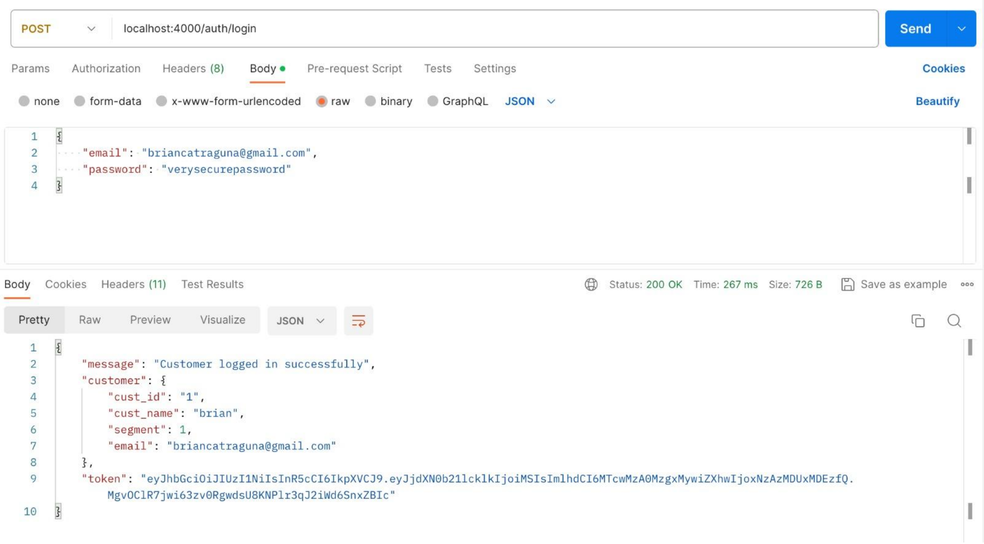


Frontend Tech Overview

- Javascript as the primary programming language
- ReactJS as the framework used
- Material UI as the UI component library
- Axios as the networking tool to hit backend endpoints
- React router as the routing tool
- Redux and local storage for state management
- Toast Notifications for success and failure messages.
- We created a total of 12 different pages for the web application

REST API

- REST (Representational State Transfer) API is used for communication between the client (our frontend) and our backend server, allowing them to exchange data and perform actions over the internet using HTTP methods such as PUT, POST, GET, DELETE



Backend Tech Overview

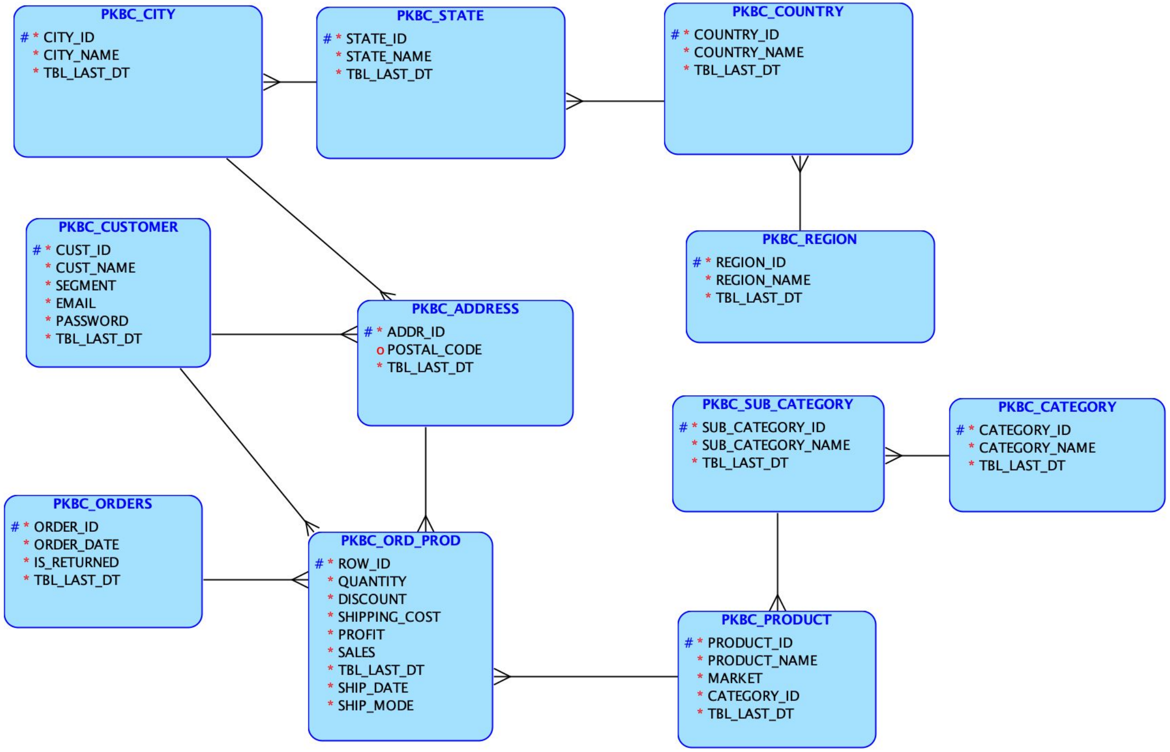
- NodeJS for server side runtime environment
- ExpressJS for the framework for routing API endpoints
- BcryptJS for password encryption tool
- Express validator for HTTP requests validation
- Nodemailer and Google's SMTP server for generating and sending emails
- JSONwebtoken for user authentication
- We created a total of 24 API endpoints for the frontend to interact with our backend. In which 6 are public endpoints and the rest required authentication and a signed JSONwebtoken to access resources

OLTP Database Tech Overview

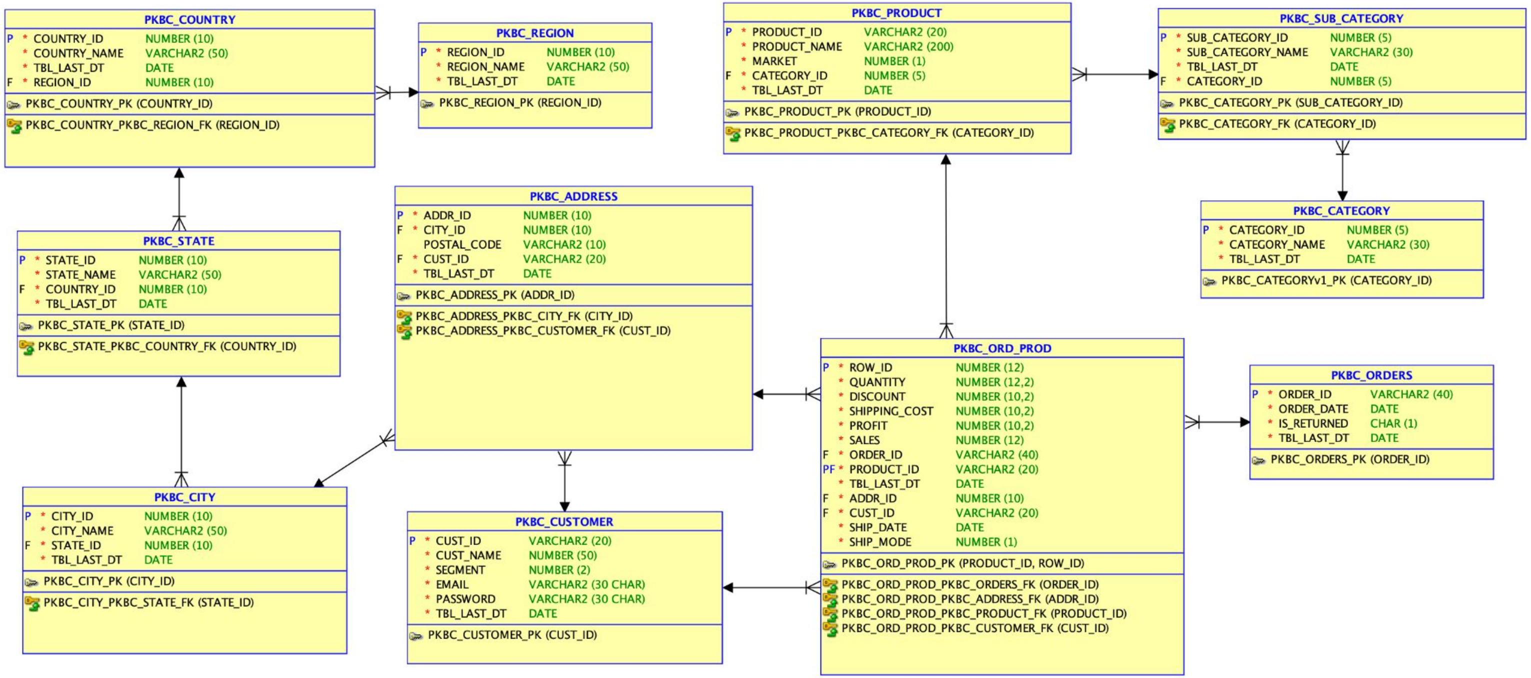
- MySQL database as the database used.
- Use of stored procedures and parametrized queries to prevent SQL injection. We have created a total of 22 stored procedures for the backend server to interact with DB
- Database indexing on frequently used columns to increase performance in lookups. We added 5 indexes on the database for columns that we frequently query on
- Triggers to insert and update ‘tbl_last_dt’ on every table to record the last time we updated the row for ETL purposes
- Using transaction (commit & rollback) in Stored Procedures to maintain data integrity

NYU

Logical Model (OLTP)



Relational Model (OLTP)

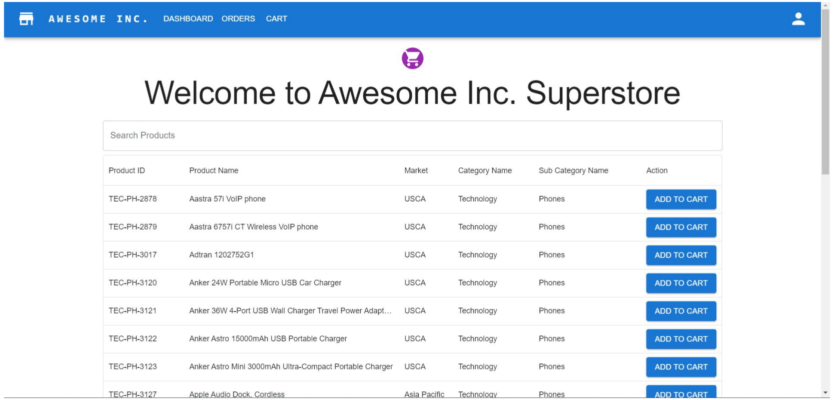


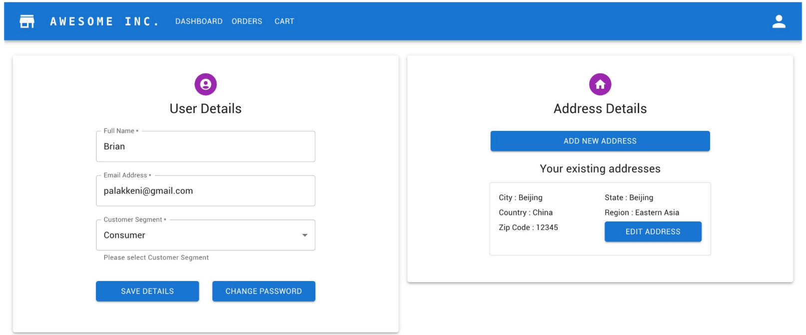
Application Functionality

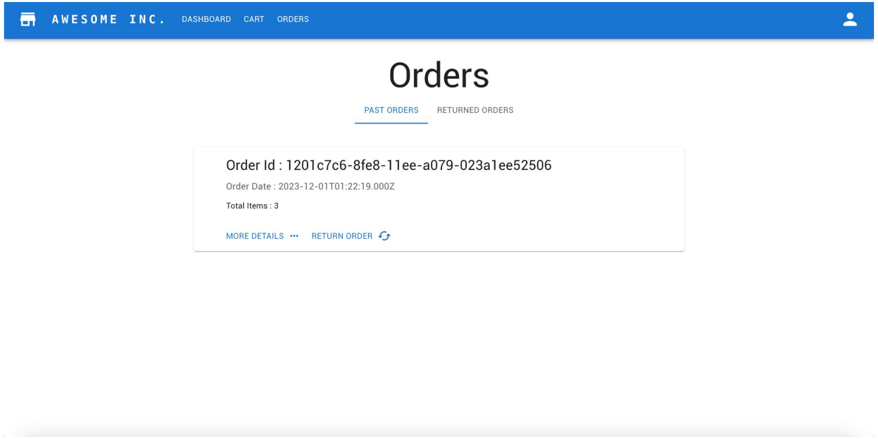
Customers users are able to :

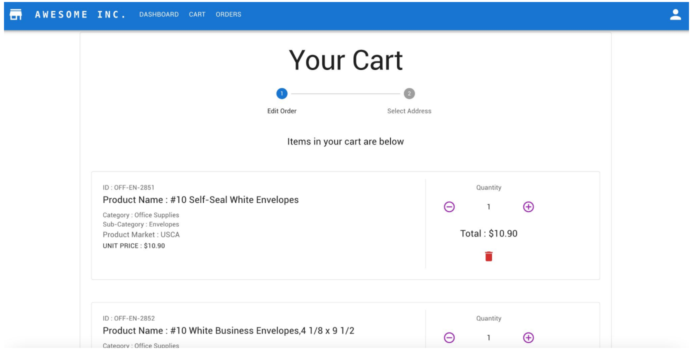
- Create and log into their accounts.
- Forgot and reset password through OTP email
- View and search for products that are available in the market
- Edit their profile such as changing their full name, email, or customer segment
- Changing password once logged in
- Adding address related to the customer
- Editing existing address
- Adding products to cart, adjust the quantity
- Checkout the cart by selecting address, selecting ship mode and placing orders
- Viewing past orders and filter by non-returned and returned orders
- Requesting for return orders
- Viewing details of orders by viewing products that are related to the order

Screenshots









NYU

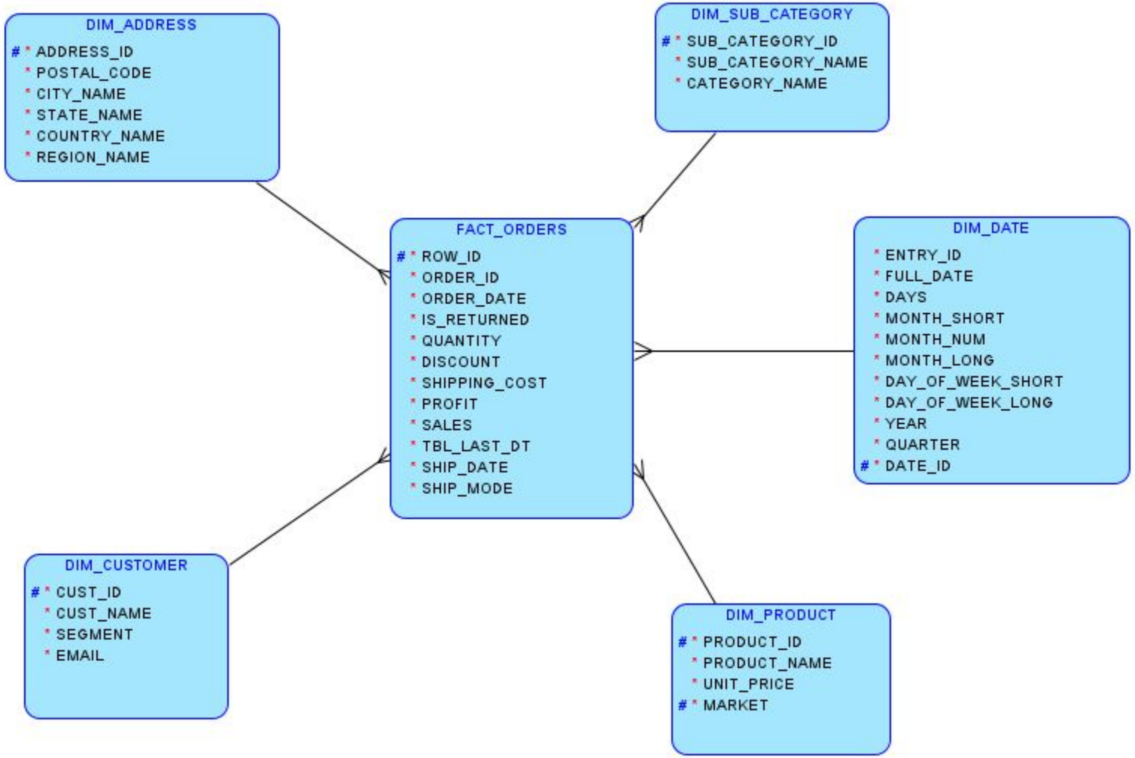
History Table (OLTP)

- Added history table to preserve deleted data and track history
- We added history tables for table
  - Pkbc_address
  - Pkbc_customer
  - Pkbc_ord_prod
  - Pkbc_order
  - pkbc_product

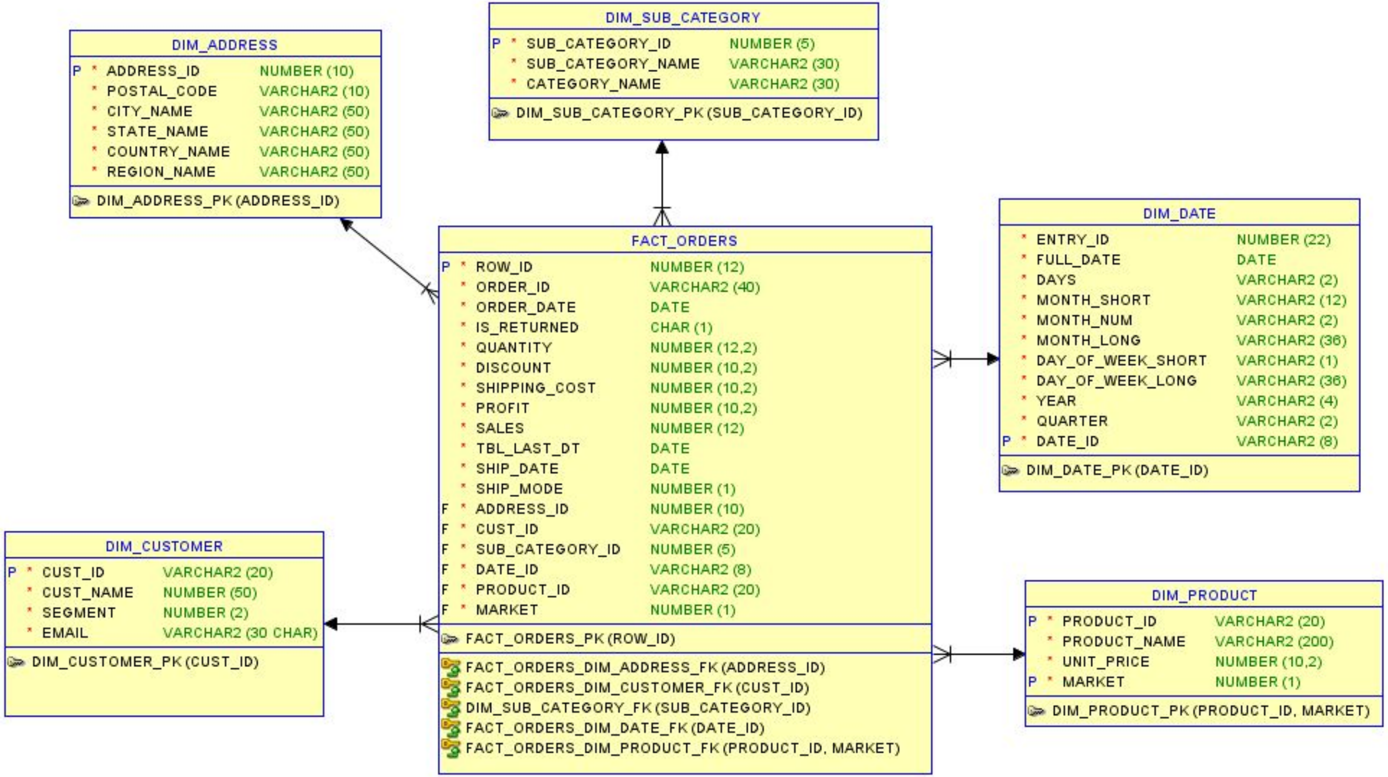
```
drop table if exists pkbc_address_history;
create table pkbc_address_history as select * from
pkbc_address where 1 = 2;
alter table pkbc_address_history add constraint
pk_address_history primary key (addr_id);

delimiter $$
drop trigger if exists td_pkbc_address;
create trigger td_pkbc_address
before delete on pkbc_address for each row
begin
insert into pkbc_address_history
select * from pkbc_address
where addr_id = old.addr_id;
update pkbc_address_history
set tbl_last_dt=current_timestamp()
where addr_id=old.addr_id;
end$$
delimiter ;
```

NYU

Logical Model (DW)



Relational Model (DW)



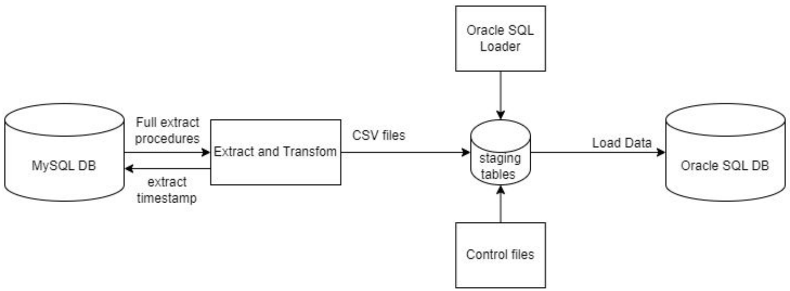
ETL Approach

- Used ETL approach to integrate MySQL OLTP data into Oracle Data Warehouse
- Leveraged Change Data Capture (CDC) for incremental updates on scheduled basis
- Minimizes data transfers by extracting only changes after initial load
- Extracts taken to CSV files, loaded into Oracle staging tables via SQL*Loader
- Implemented star schema data model for simplicity and fast query performance
- Staging tables used to land, transform, and stage data before data warehouse tables

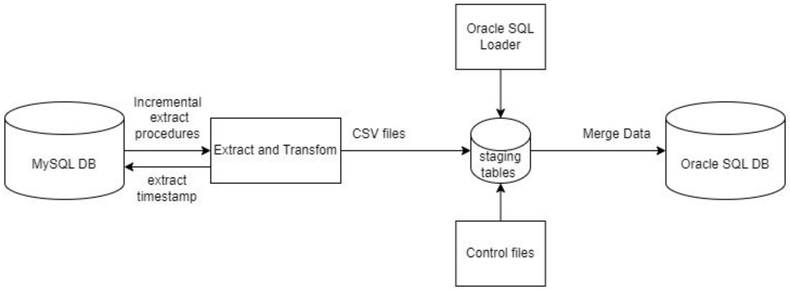
ETL Approach(.. contd)

- Initial full extract export from MySQL, apply transformations, load into warehouse (baseline)
- etl_extract_date table tracks full extract timestamps
- Then periodic incremental extracts from MySQL - only changed data since last extract
- Apply similar ETL process on extracts and merge into data warehouse tables
- Achieves ongoing synchronization of data warehouse with only incremental deltas

FULL ETL PROCESS



INCREMENTAL ETL PROCESS



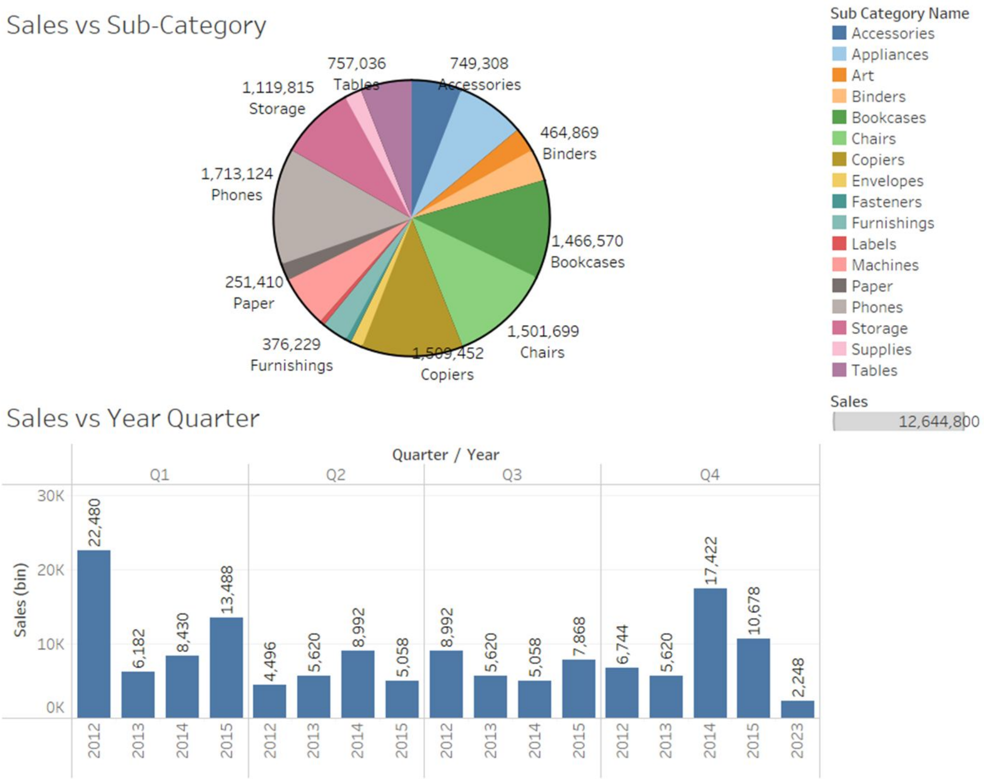
NYU

Partition Table

- Partition on orders table in data warehouse based on order date

```
CREATE TABLE fact_pkbc_orders (
    row_id          NUMBER(12) NOT NULL,
    order_id        VARCHAR2(40) NOT NULL,
    order_date      DATE NOT NULL,
    is_returned     CHAR(1) NOT NULL,
    quantity        NUMBER(12, 2) NOT NULL,
    discount        NUMBER(10, 2) NOT NULL,
    shipping_cost   NUMBER(10, 2) NOT NULL,
    profit          NUMBER(10, 2) NOT NULL,
    sales           NUMBER(12) NOT NULL,
    ship_date       DATE NOT NULL,
    ship_mode       NUMBER(1) NOT NULL,
    address_id      NUMBER(10) NOT NULL,
    cust_id         VARCHAR2(20) NOT NULL,
    sub_category_id NUMBER(5) NOT NULL,
    product_id      VARCHAR2(20) NOT NULL,
                market      NUMBER(1) NOT NULL,
    date_id         VARCHAR2(8) NOT NULL
)PARTITION BY RANGE (order_date)
     ( PARTITION P1 VALUES less than(to_date('01-JAN-2012', 'DD-MON-YYYY')),
       PARTITION P2 VALUES less than(to_date('01-JAN-2013', 'DD-MON-YYYY')),
       PARTITION P3 VALUES less than(to_date('01-JAN-2014', 'DD-MON-YYYY')),
       PARTITION P4 VALUES less than(to_date('01-JAN-2015', 'DD-MON-YYYY')),
       PARTITION P5 VALUES less than(to_date('01-JAN-2016', 'DD-MON-YYYY')),
       PARTITION P6 VALUES less than(to_date('01-JAN-2017', 'DD-MON-YYYY')),
       PARTITION P7 VALUES less than(to_date('01-JAN-2018', 'DD-MON-YYYY')),
       PARTITION P8 VALUES less than(to_date('01-JAN-2019', 'DD-MON-YYYY')),
       PARTITION P9 VALUES less than(to_date('01-JAN-2020', 'DD-MON-YYYY')),
       PARTITION P10 VALUES less than(to_date('01-JAN-2021', 'DD-MON-YYYY')),
       PARTITION P11 VALUES less than(to_date('01-JAN-2022', 'DD-MON-YYYY')),
       PARTITION P12 VALUES less than(to_date('01-JAN-2023', 'DD-MON-YYYY')),
       PARTITION P13 VALUES less than(to_date('01-JAN-2024', 'DD-MON-YYYY'))
     );
```

Data Analytics



Lessons Learnt

- Combined complementary skills for well-rounded project
- UI/frontend and backend development
- Covered each other's weaknesses
- Communication was key for collaboration
- Alignment on designs and decisions
- Resolve challenges together
- Enable productive teamwork
- Tools like Git improved code management
- Track changes, fixes, features
- Support concurrent working
- Streamline productivity
- Clean, simple code was crucial
- Simplify troubleshooting
- Avoid new issues
- Facilitate contributions