Friday, December 17, 2021

SEIS630 FINAL PROJECT WRITEUP

Xueqing Zhao

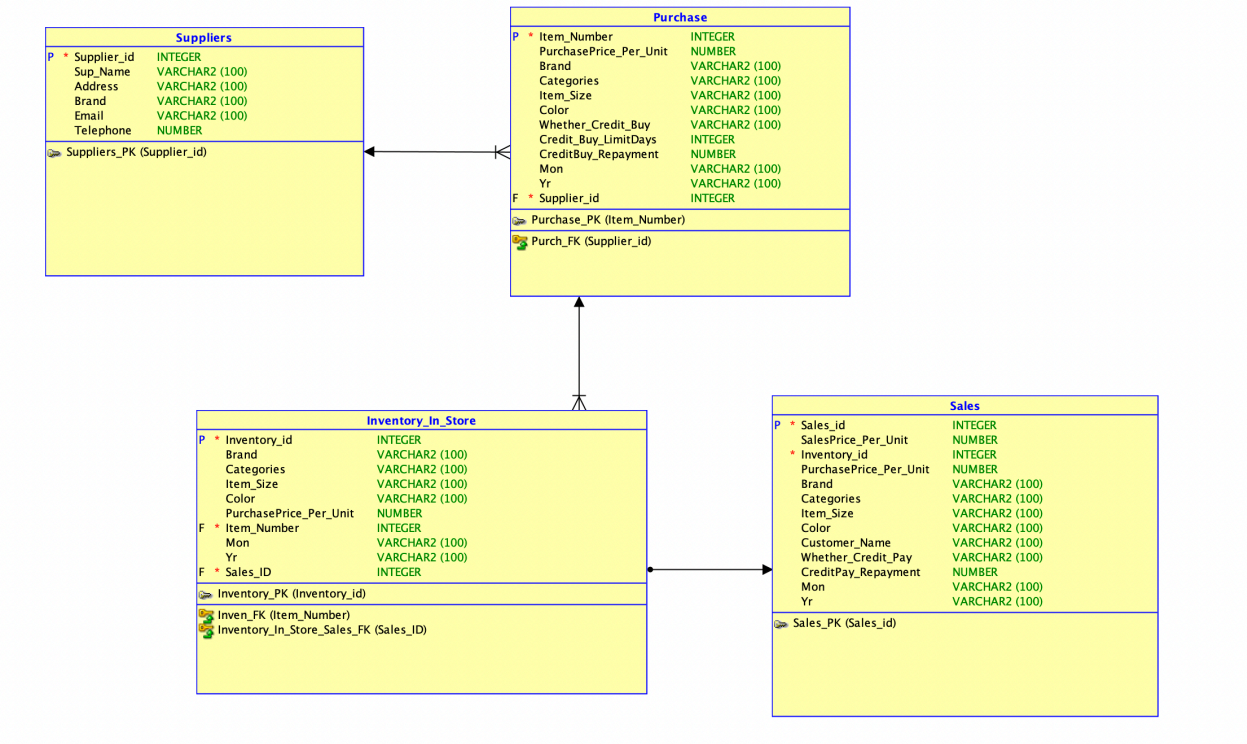
Overview

This project is to build a Clothing store Purchase-Inventory-Sales db application by using Oracle Data Modeler, Oracle APEX and Oracle Cloud.

Oracle Application Express (APEX) is a low-code development platform that enables to build scalable, secure enterprise apps.

Database Design / Redesign (Relational Model)

Clearly, using the Oracle Data Modeler, it is easy to identify required tables and draw the relationships between them.



As shown in above model, there are four tables displaying the contents and relationships to build the further Purchase-Inventory-Sales db. I delete many tables of previous drafts. Some are not required, and some could be presented in a better way, such as view.

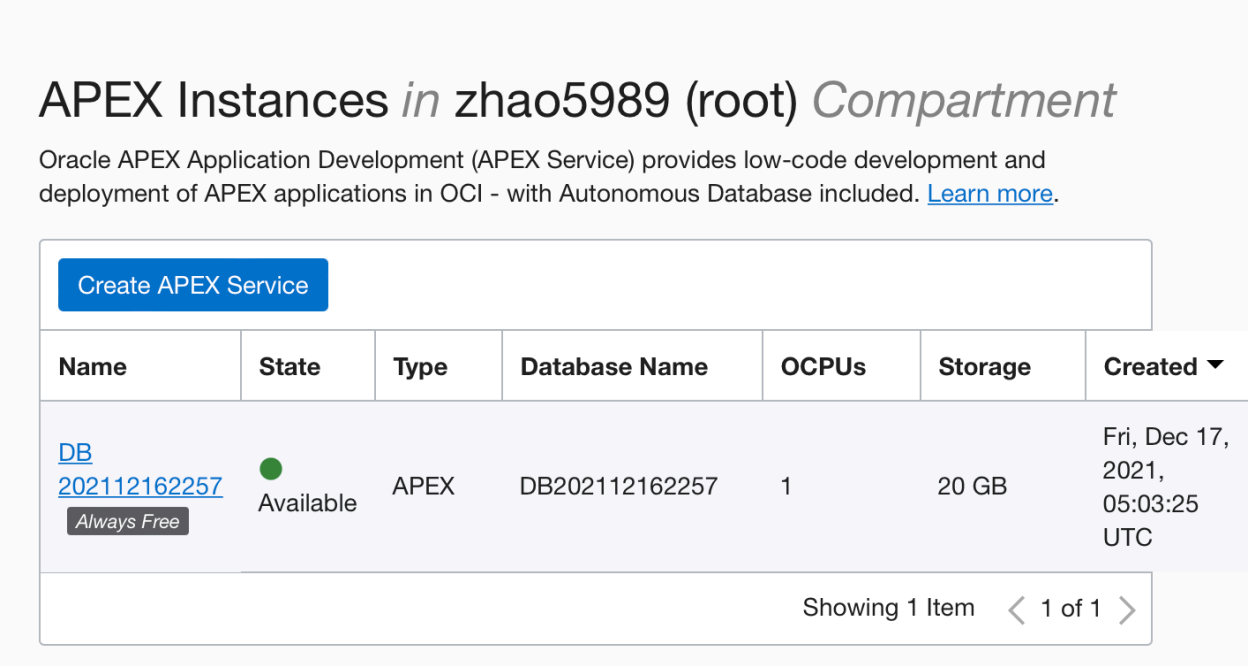
Next, generating the ddl file and uploading to Oracle APEX, four tables, columns, types, constructs, and constraints are produced.

Oracle APEX Database

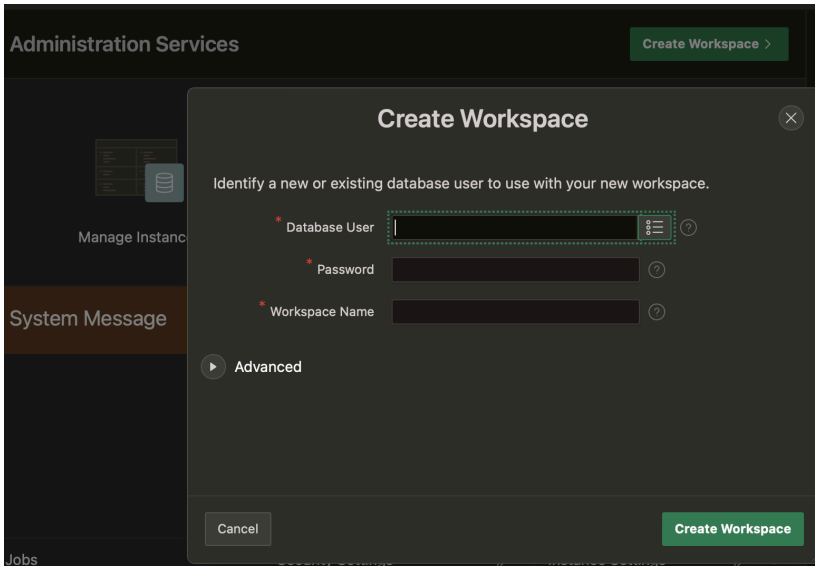
Oracle APEX is a low-code development platform to enable us to build DB-backed Web APP easily.

How to reach this goal? The first step is to build the Database. Below are steps for DB build.

1. we need to Create an APEX Service instance on the APEX Instances page of ORACLE Cloud. When it is done, it will connect to an autonomous database storage where our data will be stored and backup automatically.



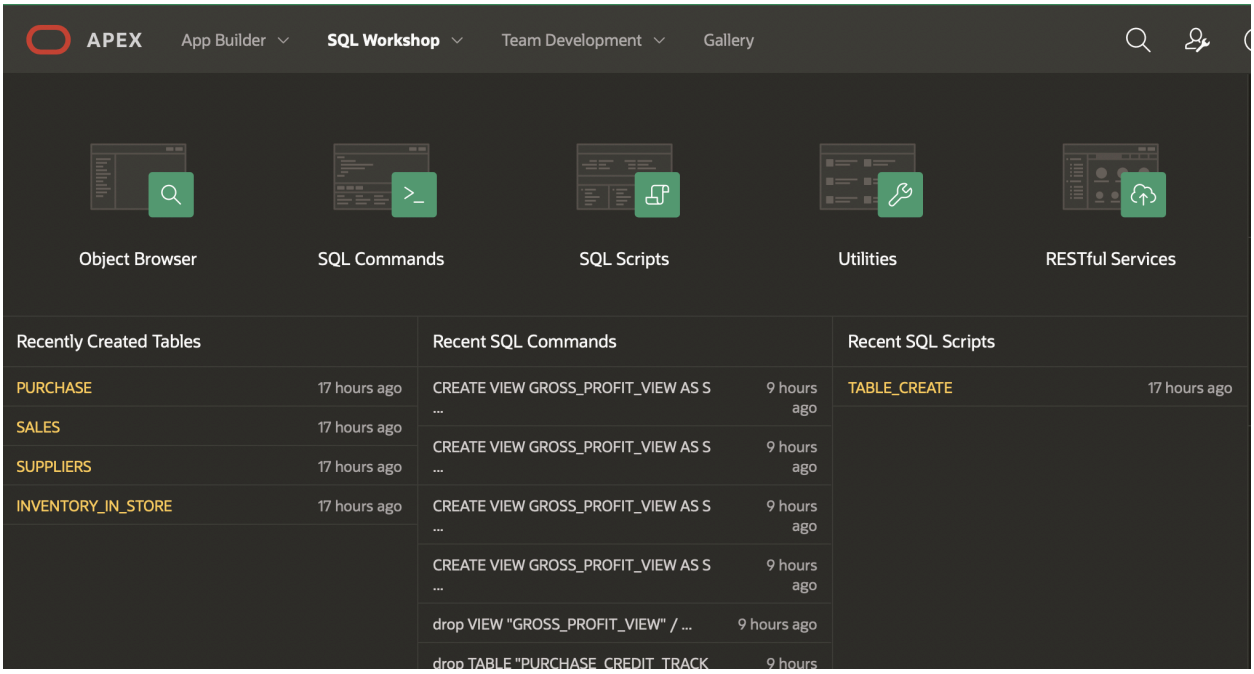
2. Laugh Apex and Choose Administration Services. We will create a Workspace here with User, Password and workspace Name. This helps database admin security and privacy.

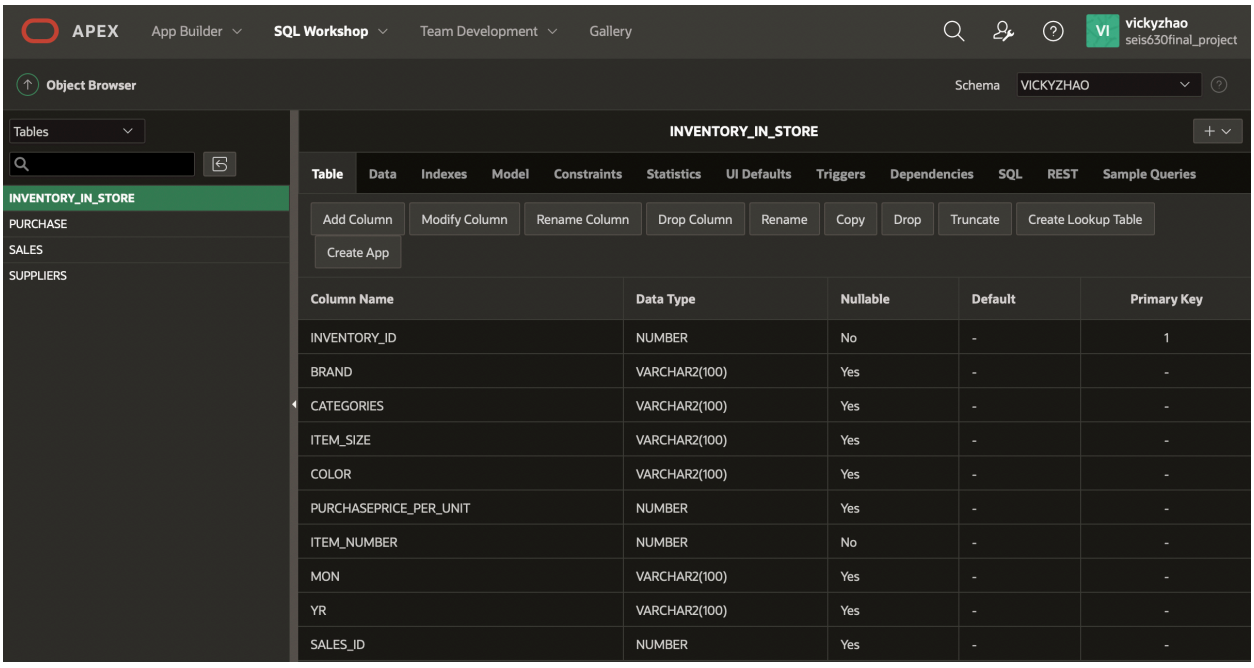


3. Login to Oracle Application Express with register User, Password, and Workspace. Then, it is time to create tables.

TABLE CREATE

Oracle APEX has built-in SQL Platform like LIVESQL to develop database. Obviously, We could upload the ddl file to SQL Scripts. Thereby, four tables are created.





Besides, as seen as above picture, we could click add column/add data by an easy no-sql way. For each table, it is convenient to further develop its construct and find its related information, such as constrains, triggers, data, etc.

PL/SOL TRIGGER

- **1. Automatically incremental index for pk column.**

```
CREATE SEQUENCE INVEN_ID
MINVALUE 1
MAXVALUE 999999999
START WITH 1
INCREMENT BY 1
NOCACHE;

CREATE OR REPLACE TRIGGER INVENTORY_INSERT_T1
BEFORE INSERT ON INVENTORY_IN_STORE FOR EACH ROW
DECLARE
BEGIN
    SELECT INVEN_ID.nextval INTO :NEW.INVENTORY_ID
    FROM DUAL;
END;
```

- **2. When new purchases get in Table Purchase, related data automatically insert into Table INVENTORY_IN_STORE at the same time.**

```
CREATE OR REPLACE TRIGGER PUR_INVEN_UPDATE_T2
AFTER INSERT ON PURCHASE FOR EACH ROW
BEGIN
    INSERT INTO INVENTORY_IN_STORE(BRAND,
    CATEGORIES,ITEM_SIZE,COLOR,PURCHASEPRICE_PER_UNIT,
    ITEM_NUMBER,MON,YR)
    VALUES(:NEW.BRAND,:NEW.CATEGORIES,
    :NEW.ITEM_SIZE,:NEW.COLOR, :NEW.PURCHASEPRICE_PER_UNIT,
    :NEW.ITEM_NUMBER,:NEW.MON,:NEW.YR);
END;
```

INSERT/UPDATE STATEMENT

In SQL, we usually use insert/update to add value to table.

“ INSERT INTO TABLE VALUES……
UPDATE TABLE SET…… ”

```
INSERT INTO PURCHASE(
ITEM_NUMBER,
PURCHASEPRICE_PER_UNIT,
CATEGORIES,
ITEM_SIZE,
COLOR,
WHETHER_CREDIT_BUY,
CREDIT_BUY_LIMITDAYS,
CREDITBUY_REPAYMENT,
MON,
YR,
SUPPLIER_ID)
VALUES(556,31,'SCARF','NORMAL','PINK','YES',60,0,'DEC','2021',1);

UPDATE PURCHASE
SET PURCHASE.BRAND =(
SELECT BRAND
FROM SUPPLIERS
WHERE SUPPLIERS.SUPPLIER_ID = PURCHASE.SUPPLIER_ID);
```

For data, we can also use no-code way to insert rows directly based on tabs structure.

VIEW STATEMENT

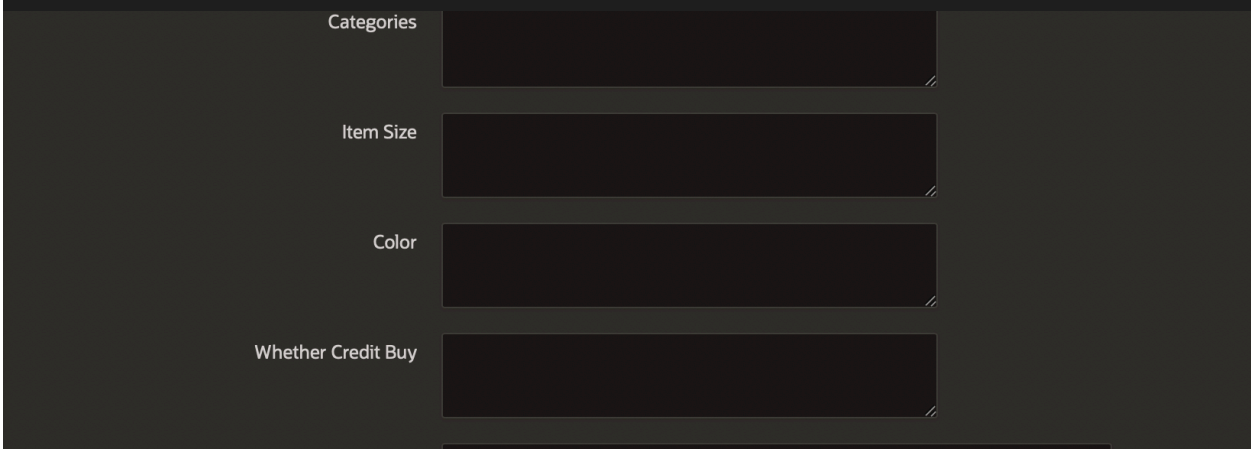
One of profound experience I have in this project is View Statement.

In SQL, a view is a virtual table based on the result-set of an SQL statement. A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.

How does VIEW STATEMENT help me in this project?

For advantages,

```sql
CREATE VIEW CUSTOMER_CREDIT_CONTROL AS
SELECT SALES_ID,CUSTOMER_NAME,MON,YR,
SALESPRICE_PER_UNIT-CREDITPAY_REPAYMENT AS DUE_AMOUT
FROM SALES
WHERE SALESPRICE_PER_UNIT-CREDITPAY_REPAYMENT IS NOT NULL
AND SALESPRICE_PER_UNIT-CREDITPAY_REPAYMENT >0
ORDER BY MON DESC
WITH READ ONLY;
```



- Views can represent a subset of the data contained in a table.
- Views can join and simplify multiple tables into a single virtual table.
- Views can act as aggregated tables, where the database engine aggregates data (sum, average, etc.) and presents the calculated results as part of the data.

Hence, I reduce redundant tables in previous draft and create views for the same purpose. In order to monitor corresponding purchase, inventory and sales activities, four views are created: PURCHASE_CREDIT_CONTROL VIEW, CUSTOMER_CREDIT_CONTROL VIEW, CUSTOMERS VIEW, GROSS_PROFIT VIEW.

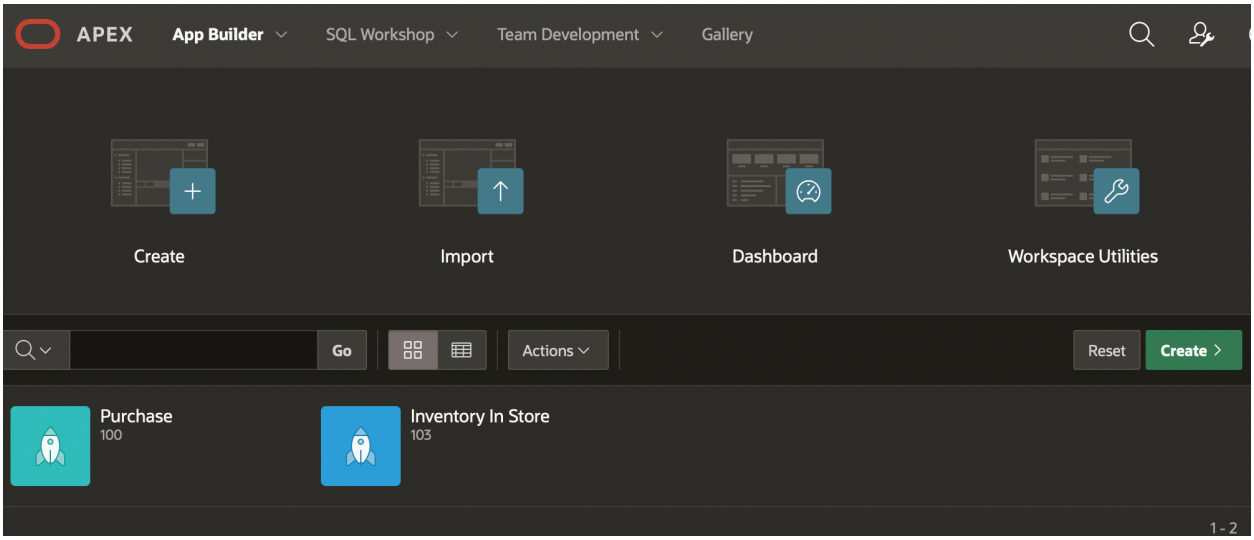
DB-backed Web Applications

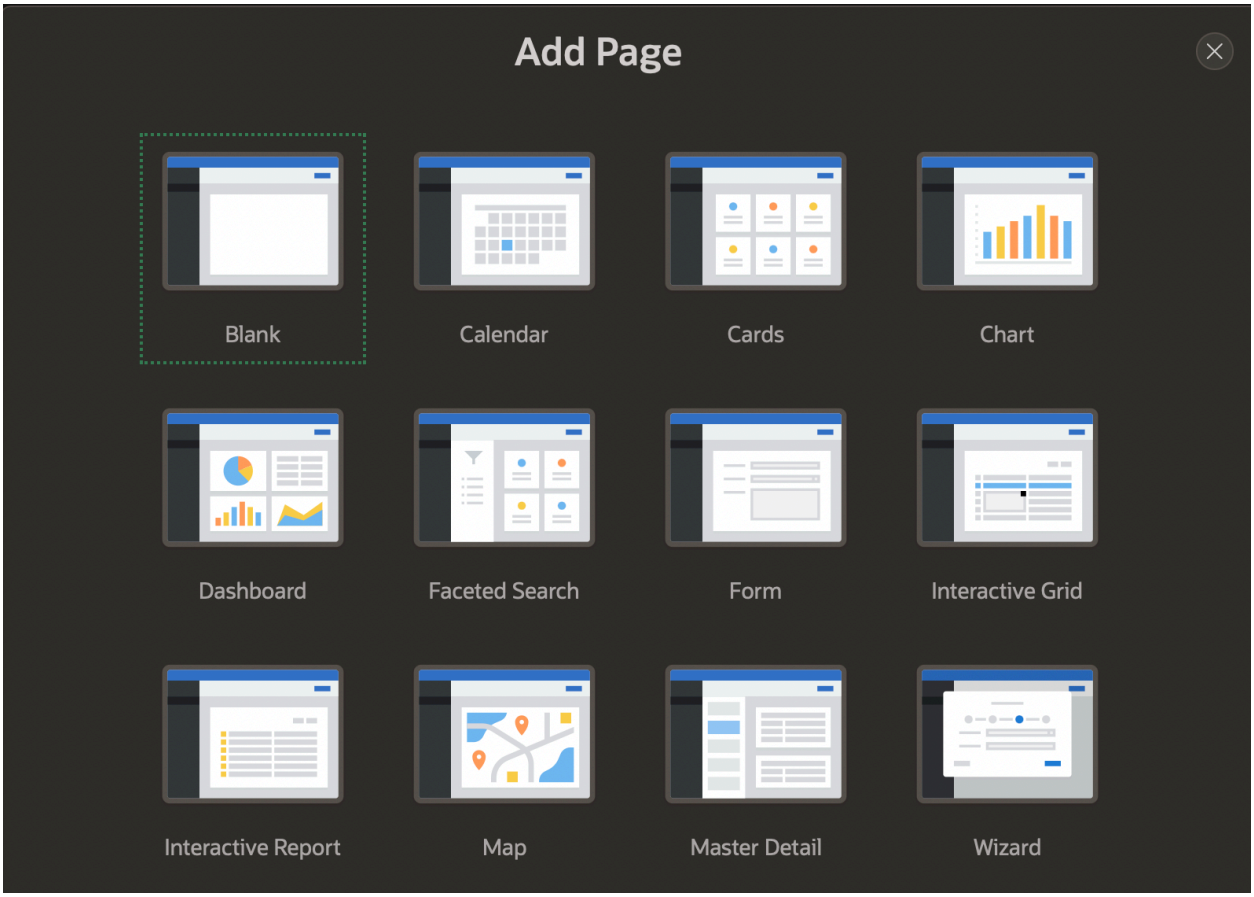
The final step is to build Web App that is built on App Builder Function.

Clicking on the Create, choose new application and add pages on existing data and tables.

One good thing is APEX has rich built-in data analysis themes with simple data analysis and visualization tools. For example, Dashboard, Chart, etc. It is very benefit for users to produce desired data analysis reports.

Also, we can connect two tables to produce combined data reports.





I have build an App Purchase_Inventory_Sales based on produced tables and views. By the App, data can be easily displayed, added, filtered, grouped, visualized, and so on.

Conclusion

By doing this class and this project, I have a lot of fun. I get to know about DB knowledge. I was interested in how the DB app works before. Now, this question has been gotten in touch a little bit by exploring in this project.

Meanwhile, I realize a great many drawbacks in my knowledge. I caused many errors when I was writing the trigger. In order to solve those problems, I have referred to others' sample and Oracle Official Documents. Although it is still very hard for me, and I can only write easy trigger code, I have better understand how trigger works. Besides, There are some problems that I haven't solved yet. For instance, set a UPDATE and DELETE RESTRICT Trigger for Parent Table. I tried to set this trigger between Sales Table and Inventory_in_store Table but without success. These bugs will be solved in the future. Besides, some interested trigger area appeal to me where I will further explore over the time, e.g., Using Compound DML Triggers to Avoid Mutating-Table Error; Triggers for Building Complex Updatable Views, etc.

And finally, I sincerely appreciate Professor's help and Classmates' advice.

Hope you enjoy my Project Processes.

Thank You!