Task 1

Dataset

(<https://www.kaggle.com/aungpyaeap/supermarket-sales>)

Hi, we’ve chosen dataset which contains invoice information of a supermarket. It contains more than 900 rows. Unfortunately, the database doesn’t have image column itself. So, we decided to create one!



(FK\_INVOICE is a foreign key from main table “Supermarket”)

Task 2

Import Data…

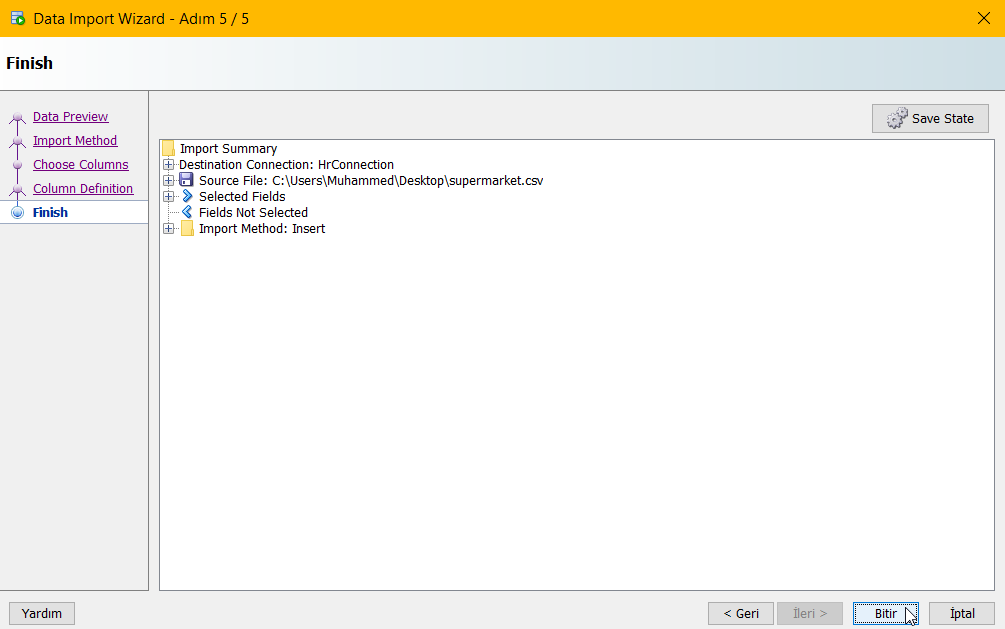
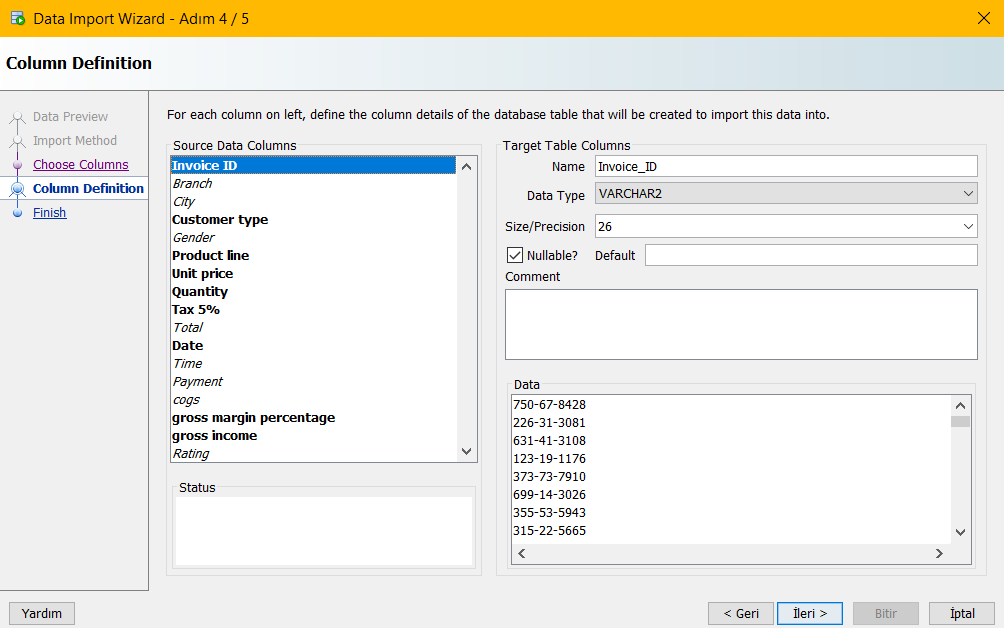
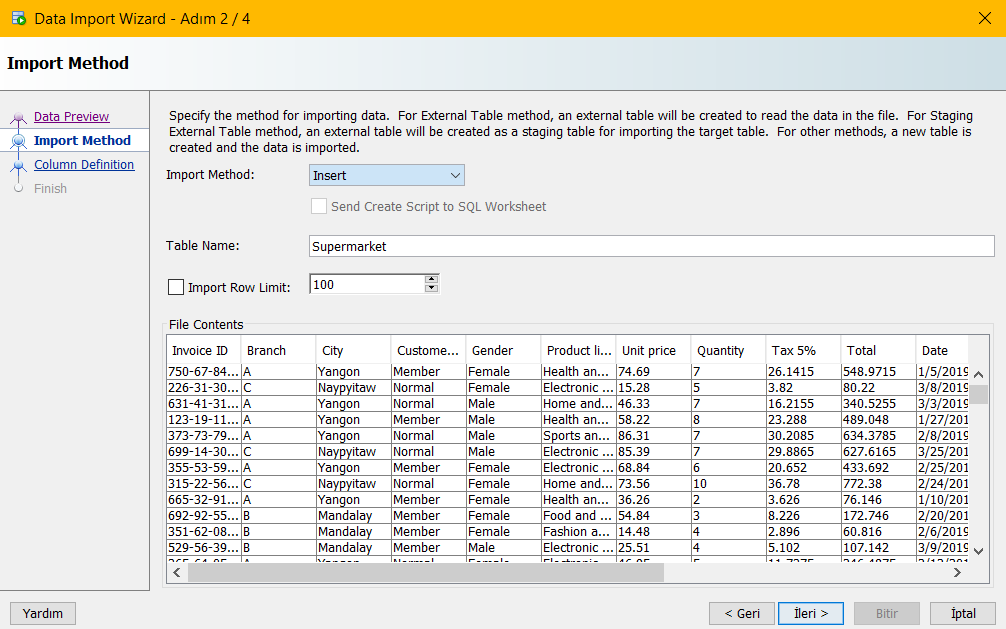
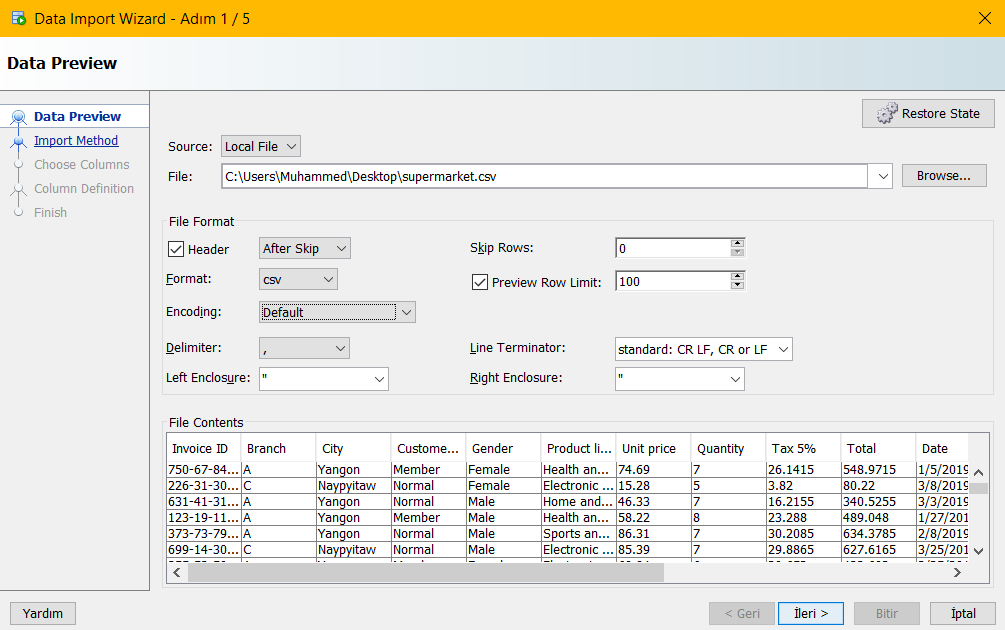
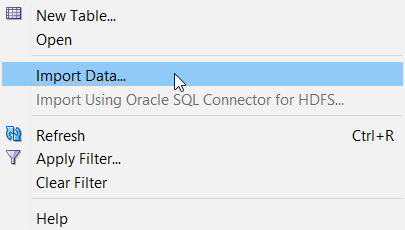
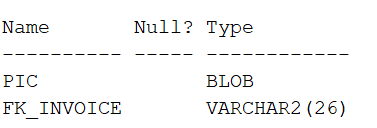


Image Table

The column holding Images has BLOB data type.



Derived Columns

ALTER TABLE supermarket

ADD (

price\_with\_tax AS (cogs+tax\_5\_percent)

); -- Column that is sum of total price and tax.

ALTER TABLE supermarket

ADD (

intallment\_12mo AS (((cogs+tax\_5\_percent)/12) \* 1.10)

); -- Installment price for 12 month with 10% interest

Tables

We have 2 tables in our database. “Supermarket” is the main one and “Pictures” table that hold image. And they’re connected by the foreign key constraint. (FK\_INVOICE)

Task 3

[Unfortunately, we could not make connection between the frontend and backend part.](https://www.youtube.com/watch?v=Y_oD111dK7c&ab_channel=AuroraVibes)

Task 4

Functions & Procedures

create or replace FUNCTION get\_5\_percent

(p\_invoice IN supermarket.invoice\_id%TYPE)

RETURN NUMBER IS

v\_number NUMBER;

BEGIN

SELECT (unit\_price \* quantity) \* 5/10

INTO v\_number

FROM supermarket

WHERE invoice\_id = p\_invoice;

RETURN v\_number;

END get\_5\_percent; --Calculates the 5% tax for the price

create or replace FUNCTION get\_total

(payment\_type IN supermarket.payment%TYPE)

RETURN NUMBER IS

v\_total supermarket.cogs%TYPE;

BEGIN

SELECT sum(cogs)

INTO v\_total

FROM supermarket

WHERE payment = payment\_type;

RETURN v\_total;

END get\_total; -- Calculates total number of money earned with given payment type (Cash, Ewallet & etc.)

CREATE OR REPLACE PROCEDURE create\_invoice (

invoice IN VARCHAR(26),

branch IN VARCHAR(26),

city IN VARCHAR(26),

customer\_type IN VARCHAR(26),

gender IN VARCHAR(26),

prod\_line IN VARCHAR(26),

unt\_price IN NUMBER(38),

quantity IN NUMBER(38),

tax\_5 IN VARCHAR(26),

total NUMBER(38),

datee IN DATE,

time IN VARCHAR(26),

payment IN VARCHAR(26),

cogs IN NUMBER(38),

gross\_margin IN NUMBER(38),

gross\_inco IN VARCHAR(26),

rating IN NUMBER(38)

) AS

BEGIN

INSERT INTO SUPERMARKET(Invoice\_id, Branch, city, customer\_type, gender, product\_line, unit\_price,

quantity, tax\_5\_percent, total, datee, time, payment, cogs, gross\_margin\_percentage, gross\_income, rating)

VALUES (invoice, branch, city, customer\_type, gender, prod\_line, unt\_price, quantity, tax\_5, total,

datee, time, payment, cogs, gross\_margin, gross\_inco, rating);

END;-- This procedure is defined to add new invoice to the table

create or replace PROCEDURE print\_bill(

invoiceID supermarket.invoice\_id%TYPE

)

IS

r\_sp supermarket%ROWTYPE;

BEGIN

SELECT \*

INTO r\_sp

FROM supermarket

WHERE invoice\_id = invoiceID;

dbms\_output.put\_line(r\_sp.branch || ' ' || r\_sp.city || ' ' || r\_sp.customer\_type || ' ' || r\_sp.gender

|| ' ' || r\_sp.product\_line || ' ' || r\_sp.unit\_price || ' ' || r\_sp.quantity || ' ' || r\_sp.tax\_5\_percent

|| ' ' || r\_sp.total || ' ' || r\_sp.datee || ' ' || r\_sp.time || ' ' || r\_sp.payment || ' ' || r\_sp.cogs

|| ' ' || r\_sp.gross\_margin\_percentage || ' ' || r\_sp.gross\_income || ' ' || r\_sp.rating);

END; --With this procedure, you can print any invoice information you want.

Collections

DECLARE

TYPE t\_invoice\_date IS TABLE OF supermarket.datee%TYPE

INDEX BY VARCHAR(11);

v\_invoice\_date\_tab t\_invoice\_date;

BEGIN

FOR sm\_rec IN

(SELECT invoice\_id, datee FROM supermarket)

LOOP

v\_invoice\_date\_tab(sm\_rec.invoice\_id) := sm\_rec.datee;

END LOOP;

DECLARE

TYPE t\_invoice\_date IS TABLE OF supermarket.datee%TYPE

INDEX BY VARCHAR(11);

v\_invoice\_date\_tab t\_invoice\_date;

v\_invoice\_date\_count NUMBER(4);

BEGIN

FOR sm\_rec IN

(SELECT invoice\_id, datee FROM supermarket)

LOOP

v\_invoice\_date\_tab(sm\_rec.invoice\_id) := sm\_rec.datee;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE(v\_invoice\_date\_tab.COUNT);

END;

END;

Cursors

DECLARE

CURSOR curs IS

select invoice\_id from supermarket

where city = 'Yangon' AND customer\_type = 'Member';

invoiceID supermarket.invoice\_id%TYPE;

BEGIN

OPEN curs;

LOOP

FETCH curs INTO invoiceID;

EXIT WHEN curs%NOTFOUND;

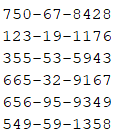
DBMS\_OUTPUT.PUT\_LINE(invoiceID);

END LOOP;

CLOSE curs;

END;

Output:



DECLARE

CURSOR curs IS

select product\_line, unit\_price from supermarket

where unit\_price > (select avg(unit\_price) from supermarket);

productLine supermarket.product\_lıne%TYPE;

unitPrice supermarket.unit\_price%TYPE;

BEGIN

OPEN curs;

LOOP

FETCH curs INTO productLine, unitPrice;

EXIT WHEN curs%NOTFOUND;

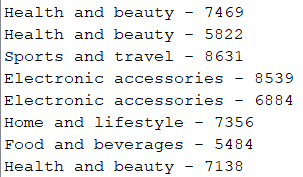
DBMS\_OUTPUT.PUT\_LINE(productLine || ' - ' || unitPrice);

END LOOP;

CLOSE curs;

END;

Output:



DECLARE

CURSOR curs IS

select customer\_type, gender, rating from supermarket;

BEGIN

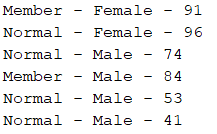
FOR v\_dep\_records IN curs LOOP

DBMS\_OUTPUT.PUT\_LINE(v\_dep\_records.customer\_type || ' - ' || v\_dep\_records.gender || ' - ' || v\_dep\_records.rating);

END LOOP;

END;

Output:



DECLARE

CURSOR curs IS

select customer\_type, gender, product\_line from supermarket

where city = 'Yangon';

v\_record curs%ROWTYPE;

BEGIN

OPEN curs;

LOOP

FETCH curs INTO v\_record;

EXIT WHEN curs%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE(v\_record.customer\_type || ' - ' || v\_record.gender || ' - ' || v\_record.product\_line);

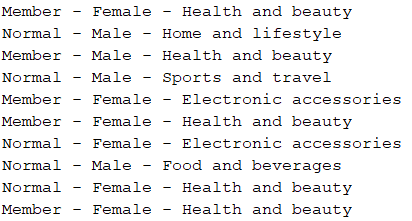
END LOOP;

DBMS\_OUTPUT.PUT\_LINE(SQL%ROWCOUNT);

CLOSE curs;

END;

Output:



Packages

create or replace PACKAGE BODY sm\_package AS

PROCEDURE print\_bill(invoiceID supermarket.invoice\_id%TYPE) IS

r\_sp supermarket%ROWTYPE;

BEGIN

SELECT \* INTO r\_sp FROM supermarket

WHERE invoice\_id = invoiceID;

dbms\_output.put\_line(r\_sp.branch || ' ' || r\_sp.city || ' ' || r\_sp.customer\_type || ' ' || r\_sp.gender

|| ' ' || r\_sp.product\_line || ' ' || r\_sp.unit\_price || ' ' || r\_sp.quantity || ' ' || r\_sp.tax\_5\_percent

|| ' ' || r\_sp.total || ' ' || r\_sp.datee || ' ' || r\_sp.time || ' ' || r\_sp.payment || ' ' || r\_sp.cogs

|| ' ' || r\_sp.gross\_margin\_percentage || ' ' || r\_sp.gross\_income || ' ' || r\_sp.rating);

END;

END sm\_package;

create or replace PACKAGE BODY sm\_functions AS

FUNCTION get\_5\_percent

(p\_invoice IN supermarket.invoice\_id%TYPE)

RETURN NUMBER IS

v\_number NUMBER;

BEGIN

SELECT (unit\_price \* quantity) \* 5/10

INTO v\_number

FROM supermarket

WHERE invoice\_id = p\_invoice;

RETURN v\_number;

END get\_5\_percent;

FUNCTION get\_total

(payment\_type IN supermarket.payment%TYPE)

RETURN NUMBER IS

v\_total supermarket.cogs%TYPE;

BEGIN

SELECT sum(cogs)

INTO v\_total

FROM supermarket

WHERE payment = payment\_type;

RETURN v\_total;

END get\_total;

END sm\_package; –-This one contains the 5% percent calculator and get\_total function which can calculate total number of money earned from specific payment type.

Triggers

CREATE TABLE Invoice\_Log(

ID NUMBER,

OPERATION\_DATE DATE,

ACTION VARCHAR(15),

AUTHOR VARCHAR(15)); --Loggings table

create or replace TRIGGER Invoice\_delete

AFTER DELETE ON "SUPERMARKET" FOR EACH ROW

BEGIN

INSERT INTO Invoice\_LOG(ID, NEW\_INV\_ID, OLD\_INV\_ID, AUTHOR, OPERATION\_DATE, ACTION)

VALUES(inv\_seq.NEXTVAL, :NEW.INVOICE\_ID ,:OLD.INVOICE\_ID ,USER, SYSDATE, 'DELETE');

END; --Saves deleted invoice information

CREATE OR REPLACE TRIGGER drop\_user

BEFORE drop on SCHEMA

BEGIN

IF ora\_dict\_obj\_name ='SUPERMARKET'

THEN

RAISE\_APPLICATION\_ERROR( -20001, 'This table can not be deleted!' );

END IF;

END; --This trigger prevents drop of main table

create or replace TRIGGER Invoice\_update

AFTER UPDATE ON "SUPERMARKET" FOR EACH ROW

BEGIN

INSERT INTO Invoice\_LOG(ID, NEW\_INV\_ID, OLD\_INV\_ID, AUTHOR, OPERATION\_DATE, ACTION)

VALUES(inv\_seq.NEXTVAL, :NEW.INVOICE\_ID ,:OLD.INVOICE\_ID ,USER, SYSDATE, 'UPDATE');

END; -- Saves information of updated invoice.

Dynamic SQL

DECLARE

my\_query VARCHAR(100);

intCl NUMBER(4);

BEGIN

my\_query := 'Select count(invoice\_id) from supermarket';

EXECUTE IMMEDIATE my\_query INTO intCl;

DBMS\_OUTPUT.PUT\_LINE('Number of bills: ' || intCl);

END; -- Outputs number of bills

DECLARE

plsql\_block VARCHAR(300) := 'BEGIN create\_invoice(:a, :b, :c, :d, :e, :f, :g, :h, :i, :j, :k, :l, :m, :n, :o, :p, :q); END;';

new\_invoice IN VARCHAR(26) := '120-06-4234';

new\_branch IN VARCHAR(26) := 'A';

new\_city IN VARCHAR(26) := 'Yangon';

new\_customer\_type IN VARCHAR(26) := 'Member';

new\_gender IN VARCHAR(26) := 'Male';

new\_prod\_line IN VARCHAR(26) := 'Electronic accessories';

new\_unt\_price IN NUMBER(38) := 3061;

new\_quantity IN NUMBER(38) := 6;

new\_tax\_5 IN VARCHAR(26) := '9.183';

new\_total NUMBER(38) := 192843;

new\_datee IN DATE := '12/03/2019';

new\_time IN VARCHAR(26) := '20:36';

new\_payment IN VARCHAR(26) := 'Cash';

new\_cogs IN NUMBER(38) := 18366;

new\_gross\_margin IN NUMBER(38) := 4761904762;

new\_gross\_inco IN VARCHAR(26) := 9.183;

new\_rating IN NUMBER(38) := 89;

BEGIN

EXECUTE IMMEDIATE plsql\_block

USING IN new\_invoice, new\_branch, new\_city, new\_customer\_type, new\_gender, new\_prod\_line, new\_unt\_price,

new\_quantity, new\_tax\_5, new\_total, new\_datee, new\_time, new\_payment, new\_cogs, new\_gross\_margin, new\_gross\_inco,

new\_rating;

END;