

**KOLEJ UNIVERSITI TUNKU ABDUL RAHMAN**

**FACULTY OF COMPUTING AND INFORMATION TECHNOLOGY**

**Assignment**

**BAIT3003 Data Warehouse Technology**

**January 2021 Semester**

|  |  |
| --- | --- |
| Name | Student ID |
| Tan Teoh Xin Ee | 20WMR08887 |
| Tan Yi Hong | 20WMR08890 |
| Tan Wei Siong | 20WMR08888 |
| Nigel Lee Jian Hsee | 20WMR08882 |

Programme : RDS2

Tutorial Group : Group 2

Date of Submission to Tutor : 16 April 2021

**BAIT3003 Data Warehouse Technology**

**Assignment Assessment Form**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task**  **No.** | **Task Descriptions** | **Weightage** | **Criteria** | **Ratings** | **Marks** | **CLO** |
| 1 | Design of Data warehouse  (logical design) | 5% | * Include the relevant dimensions. * Include the correct measures in the fact table. | * Excellent (5) * Good (4) * Moderate (2-3) * Poor (0-1) |  | 1 |
|  | Design of Data warehouse  (physical design) | 15% | * Create TABLE statements * Appropriate data types and size of attributes * Proper Integrity constraints | * Excellent (13-15) * Good (10-12) * Moderate (6-9) * Poor (0-5) |  | 1 |
| 2 | ETL  (initial loading) | 20% | * VIEWS, SELECT,INSERT,PROCEDURES for each of the dimensions and fact table. * Variety of techniques necessary to achieve the correct data loading | * Excellent (18-20) * Good (14-17) * Moderate (9-13) * Poor (0-8) |  | 1 |
|  | ETL  (subsequent loading) | 20% | * VIEWS, SELECT,INSERT,PROCEDURES for each of the dimensions and fact table. * Logic to scrub dirty data | * Excellent (18-20) * Good (15-17) * Moderate (9-14) * Poor (0-8) |  | 1 |
| 3 | \*Business Analytic queries design  (Individual marks awarded)) | 30% | * Clear and proper identification of information needs * Flexible query to cater for variety of inputs, use of multiple tables * Meaningful report handlings * Data values formatted accordingly | * Excellent (25-30) * Good (16-24) * Moderate (9-15) * Poor (0-8) |  | 3 |
| 4 | Assignment Report | 10% | * Comprehensive coverage * Quality of report presented * All tasks numbered, header / footer used, proper formatting | * Excellent (9-10) * Good (7-8) * Moderate (4-6) * Poor (0-3) |  | 1 |

**Group Member: Task 3 marks Total marks**

**1. Tan Teoh Xin Ee**  **( ) ( )**

**2. Tan Yi Hong ( ) ( )**

**3. Tan Wei Siong ( ) ( )**

**4. Nigel Lee Jian Hsee ( ) ( )**

Chapter 1 Design of Data Warehouse

* 1. Logical Design
     1. Original Database
     2. Star Schema Dimension and Fact Tables
  2. Physical Design
     1. Dimension Tables
     2. Fact Table

Chapter 2 Extract, Transform, Load Process

2.1 Script for initial loading

2.2 Script for subsequent loading

Chapter 3 Business Analytics Reports

3.1 Tan Teoh Xin Ee

3.1.1 Top 10 Sales on the Menu List

3.1.2 Top 10 Promotion Code used in Restaurant and Menu List in Selected Year

3.1.3 Top 10 Discount Amount and Percentage on Promotion Code

3.2 Tan Yi Hong

3.2.1 Top 10 Most Popular Restaurant For Year 2020 Second Half Compared To First Half For Selected City

3.2.2 Total Sales Of Each State For Year 2020 Compared To 2019

3.2.3 Top 10 Most Order Delivered Rider For 2020 Second Half Compared to First Half

3.3 Tan Wei Siong

3.3.1 Top 3 total order restaurants in each time section within the selected city in 2020

3.3.2 Menu order quantity less than 100 of the Top 3 least order restaurants in the selected city

3.3.3 Top 10 spending customers in 2019 and their spend in 2020

3.4 Nigel Lee Jian Hsee

3.4.1 Rank Total Sales of Each State based on each Meal Section

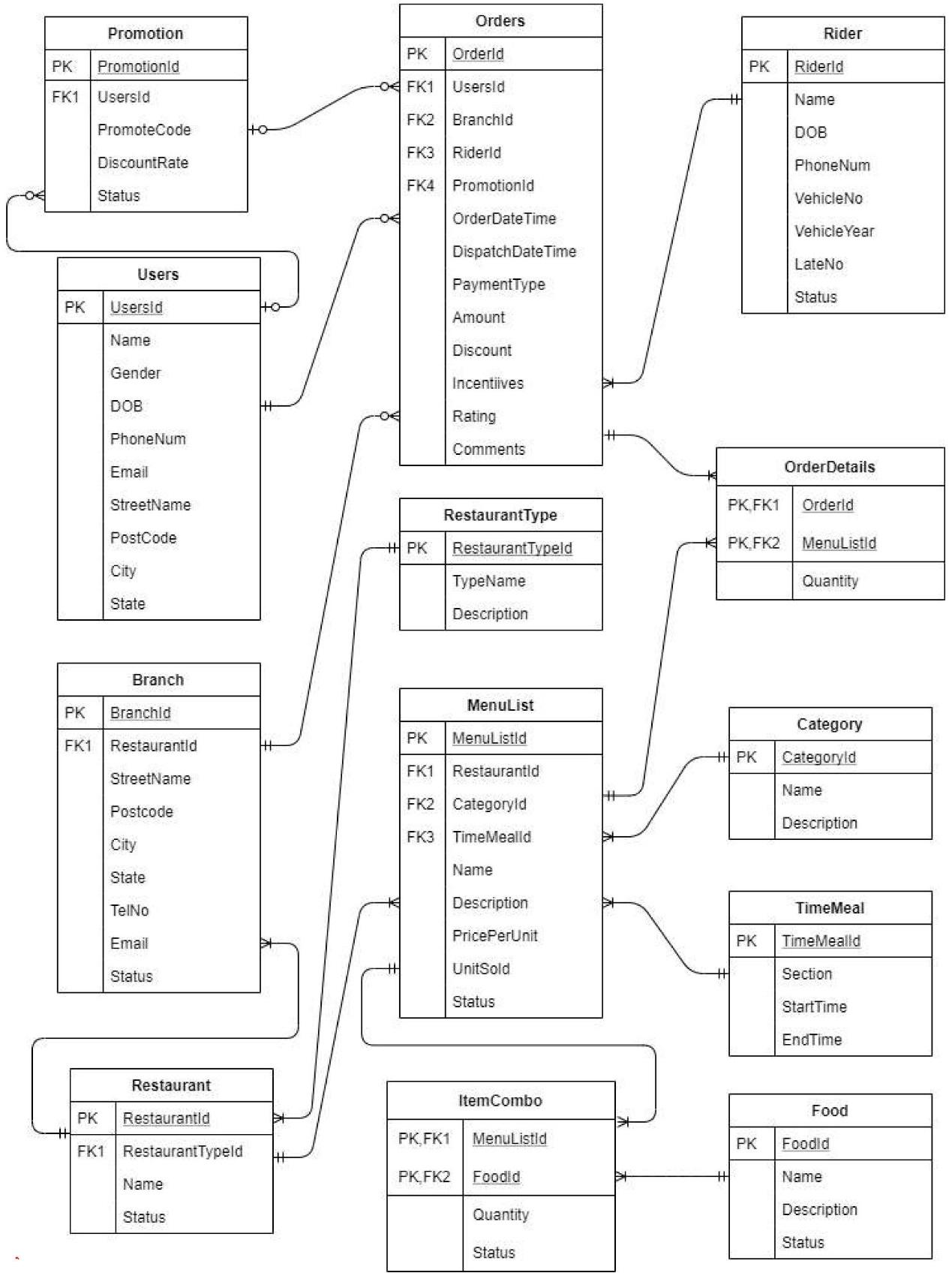
3.4.2 Top 3 Branch for each restaurant type based on the input of state and year

3.4.3 Compared Top 10 City sales percentage with Top City

# Chapter 1 : Design of Data Warehouse

## 1.1 Logical Design

### 1.1.1 Original Database



**Category Table**

CREATE TABLE Category(

CategoryId NUMBER(5) NOT NULL,

Name VARCHAR(20) NOT NULL,

Description VARCHAR(50),

PRIMARY KEY (CategoryId)

);

CREATE TABLE Food(

FoodId NUMBER(5) NOT NULL,

Name VARCHAR(100) NOT NULL,

Description VARCHAR(350),

Status VARCHAR(10) NOT NULL,

PRIMARY KEY (FoodId),

CONSTRAINT chk\_status\_f CHECK (Status in ('Active', 'Inactive'))

);

**Restaurant Type Table**

CREATE TABLE RestaurantType(

RestaurantTypeId NUMBER(5) NOT NULL,

TypeName VARCHAR(20) NOT NULL,

Description VARCHAR(50),

PRIMARY KEY (RestaurantTypeId)

);

**Rider Table**

CREATE TABLE Rider(

RiderId NUMBER(5) NOT NULL,

Name VARCHAR(50) NOT NULL,

DOB DATE NOT NULL,

PhoneNum NUMBER(15) NOT NULL,

VehicleNo VARCHAR(10) NOT NULL,

VehicleYear NUMBER(4) NOT NULL,

LateNo NUMBER(5) NOT NULL,

Status VARCHAR(10) NOT NULL,

PRIMARY KEY (RiderId),

CONSTRAINT chk\_LateNo CHECK (LateNo >= 0),

CONSTRAINT chk\_status CHECK (Status IN ('Active', 'Inactive'))

);

**Time Meal Table**

CREATE TABLE TimeMeal(

TimeMealId NUMBER(5) NOT NULL,

TimeSection VARCHAR(25) NOT NULL,

StartTime TIMESTAMP,

EndTime TIMESTAMP,

PRIMARY KEY (TimeMealId)

);

**Users Table**

CREATE TABLE Users(

UsersId NUMBER(5) NOT NULL,

Name VARCHAR(50) NOT NULL,

Gender VARCHAR(1) NOT NULL,

DOB DATE NOT NULL,

PhoneNum NUMBER(15) NOT NULL,

Email VARCHAR(50) NOT NULL,

StreetName VARCHAR(200) NOT NULL,

PostCode NUMBER(7) NOT NULL,

City VARCHAR(30) NOT NULL,

State VARCHAR(30) NOT NULL,

PRIMARY KEY (UsersId),

CONSTRAINT chk\_gender CHECK (Gender in ('M', 'F')),

CONSTRAINT chk\_email CHECK (REGEXP\_LIKE(email,'^[a-zA-Z]\w+@(\S+)$'))

);

**ItemCombo Table**

CREATE TABLE ItemCombo(

MenuListId NUMBER(5) NOT NULL,

FoodId NUMBER(5) NOT NULL,

Quantity NUMBER(4) NOT NULL,

Status VARCHAR(10) NOT NULL,

PRIMARY KEY (MenuListId, FoodId),

FOREIGN KEY (FoodId) REFERENCES Food (FoodId),

CONSTRAINT chk\_quantity CHECK (Quantity > 0),

CONSTRAINT chk\_status\_ic CHECK (Status in ('Active', 'Inactive'))

);

**Promotion Table**

CREATE TABLE Promotion(

PromotionId NUMBER(5) NOT NULL,

PromoteCode VARCHAR(15) NOT NULL,

DiscountRate NUMBER(2) NOT NULL,

Status VARCHAR(10) NOT NULL,

UsersId NUMBER(5),

PRIMARY KEY (PromotionId),

FOREIGN KEY (UsersId) REFERENCES Users (UsersId),

CONSTRAINT chk\_dis\_rate CHECK (DiscountRate BETWEEN 1 AND 100),

CONSTRAINT chk\_status\_promo CHECK (Status in ('Active', 'Inactive'))

);

**Restaurant Table**

CREATE TABLE Restaurant(

RestaurantId NUMBER(5) NOT NULL,

Name VARCHAR(60) NOT NULL,

RestaurantTypeId NUMBER(5) NOT NULL,

Status VARCHAR(10) NOT NULL,

PRIMARY KEY (RestaurantId),

FOREIGN KEY (RestaurantTypeId) REFERENCES RestaurantType (RestaurantTypeId),

CONSTRAINT chk\_status\_res CHECK (Status in ('Active', 'Inactive'))

);

**Branch Table**

CREATE TABLE Branch(

BranchId NUMBER(5) NOT NULL,

RestaurantId NUMBER(5) NOT NULL,

StreetName VARCHAR(200) NOT NULL,

Postcode NUMBER(7) NOT NULL,

City VARCHAR(30) NOT NULL,

State VARCHAR(30) NOT NULL,

TelNo NUMBER(15) NOT NULL,

Email VARCHAR(50) NOT NULL,

Status VARCHAR(10) NOT NULL,

PRIMARY KEY (BranchId),

FOREIGN KEY (RestaurantId) REFERENCES Restaurant (RestaurantId),

CONSTRAINT chk\_status\_br CHECK (Status IN ('Active', 'Inactive'))

);

**MenuList Table**

CREATE TABLE MenuList(

MenuListId NUMBER(5) NOT NULL,

Name VARCHAR(100) NOT NULL,

Description VARCHAR(350),

PricePerUnit NUMBER(6,2) NOT NULL,

UnitSold NUMBER(6) NOT NULL,

RestaurantId NUMBER(5) NOT NULL,

CategoryId NUMBER(5) NOT NULL,

TimeMealId NUMBER(5) NOT NULL,

Status VARCHAR(10) NOT NULL,

PRIMARY KEY (MenuListId),

FOREIGN KEY (RestaurantId) REFERENCES Restaurant (RestaurantId),

FOREIGN KEY (CategoryId) REFERENCES Category (CategoryId),

FOREIGN KEY (TimeMealId) REFERENCES TimeMeal (TimeMealId),

CONSTRAINT chk\_price CHECK (PricePerUnit > 0),

CONSTRAINT chk\_unit\_sold CHECK (UnitSold >= 0),

CONSTRAINT chk\_status\_ml CHECK (Status IN ('Active', 'Inactive'))

);

**Orders Table**

CREATE TABLE Orders(

OrderId NUMBER NOT NULL,

OrderDateTime TIMESTAMP NOT NULL,

DispatchDateTime TIMESTAMP,

PaymentType VARCHAR(15) NOT NULL,

Amount NUMBER(7,2) DEFAULT 0.0,

Discount NUMBER(7,2) DEFAULT 0.0,

Incentives NUMBER(7,2) DEFAULT 0.0,

Rating NUMBER(1),

Comments VARCHAR(100),

UsersId NUMBER(5) NOT NULL,

BranchId NUMBER(5) NOT NULL,

RiderId NUMBER(5),

PromotionId NUMBER(5),

PRIMARY KEY (OrderId),

FOREIGN KEY (UsersId) REFERENCES Users (UsersId),

FOREIGN KEY (BranchId) REFERENCES Branch (BranchId),

FOREIGN KEY (RiderId) REFERENCES Rider (RiderId),

FOREIGN KEY (PromotionId) REFERENCES Promotion (PromotionId),

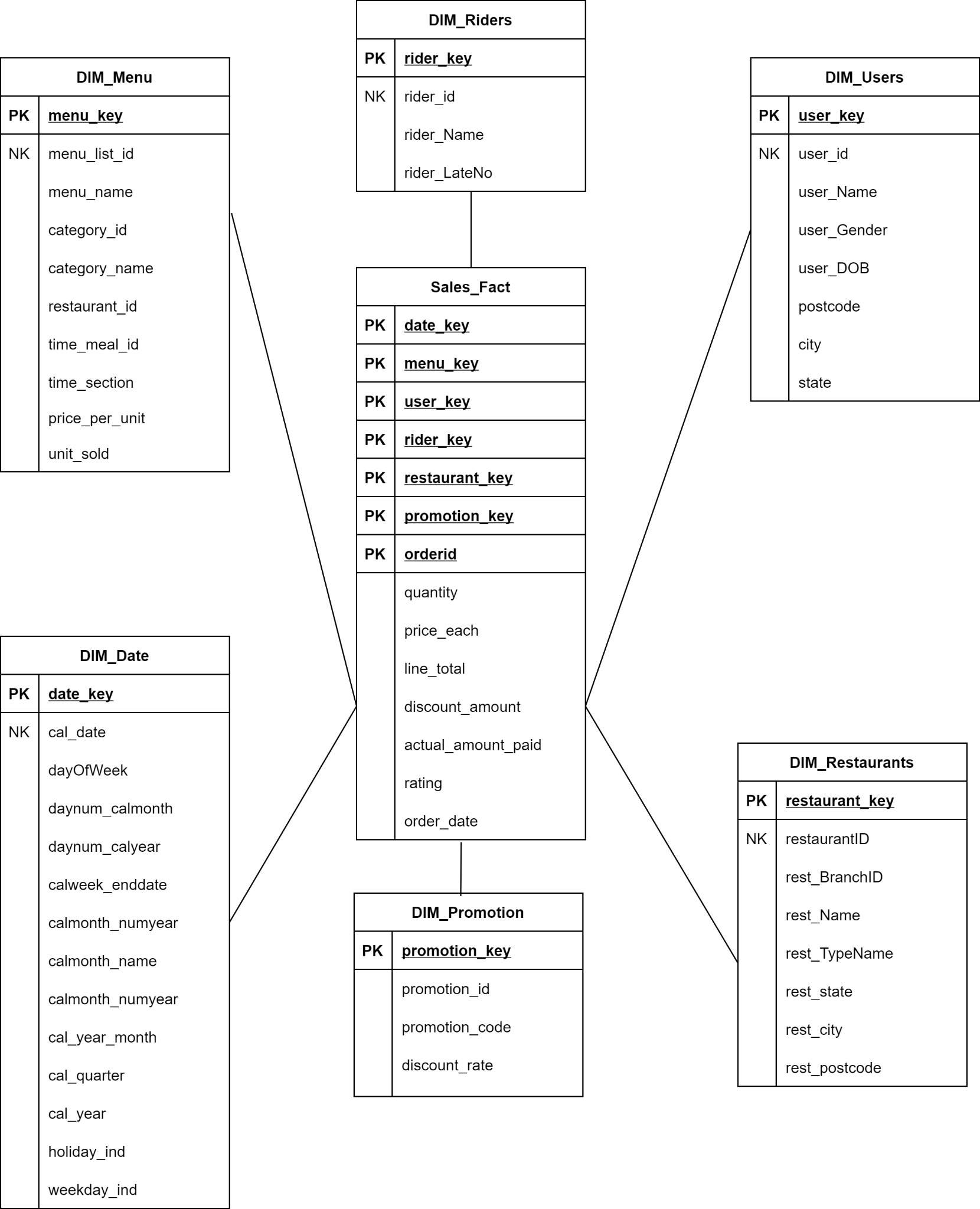
CONSTRAINT chk\_order\_dispatch\_dt CHECK (DispatchDateTime > OrderDateTime),

CONSTRAINT chk\_pay\_type CHECK (PaymentType IN ('Cash', 'Card', 'FPX', 'E-wallet')),

CONSTRAINT chk\_rating CHECK (Rating BETWEEN 1 AND 5)

);

### 1.1.2 Star Schema Dimension and Fact Tables



## 1.2 Physical Design

### 1.2.1 Dimension Table

**Menu Dimension**

CREATE SEQUENCE DIM\_menu\_seq

START with 10001

INCREMENT by 1;

CREATE TABLE DIM\_menu

(menu\_key number NOT NULL,

menu\_list\_id number(5) NOT NULL,

menu\_name varChar(100) NOT NULL,

category\_id number(5) NOT NULL,

category\_name varChar(50) NOT NULL,

time\_meal\_id number(5) NOT NULL,

time\_section varchar(50) NOT NULL,

price\_per\_unit number(6,2) NOT NULL,

unit\_sold number(6) NOT NULL,

restaurant\_id number(5) NOT NULL,

PRIMARY KEY(menu\_key)

);

**Restaurant Dimension**

CREATE SEQUENCE DIM\_Restaurant\_seq

START with 10001

INCREMENT by 1;

CREATE TABLE Dim\_Restaurants

(restaurant\_key number NOT NULL,

restaurantID number(5) NOT NULL,

rest\_BranchID number(5) NOT NULL,

rest\_name varchar(60) NOT NULL,

rest\_TypeName varchar(20) NOT NULL,

rest\_city varchar(30) NOT NULL,

rest\_state varchar(20) NOT NULL,

rest\_postcode NUMBER(7) NOT NULL,

PRIMARY KEY(restaurant\_key)

);

**Users Dimension**

CREATE SEQUENCE dim\_users\_seq

START with 10001

INCREMENT by 1;

CREATE TABLE Dim\_Users

(user\_key number NOT NULL,

user\_ID number(5) NOT NULL,

user\_Name varchar(50) NOT NULL,

user\_Gender char(1) NOT NULL,

user\_dob date NOT NULL,

postcode number(7) NOT NULL,

city varchar(30) NOT NULL,

state varchar(30) NOT NULL,

PRIMARY KEY(user\_key)

);

**Rider Dimension**

CREATE SEQUENCE dim\_rider\_seq

START with 10001

INCREMENT by 1;

CREATE TABLE DIM\_riders

(rider\_key number NOT NULL,

rider\_id number(5) NOT NULL,

rider\_Name varchar(50) NOT NULL,

rider\_LateNo number(5) NOT NULL,

PRIMARY KEY(rider\_key)

);

**Promotion Dimension**

CREATE SEQUENCE dim\_promotion\_seq

START with 10001

INCREMENT by 1;

CREATE TABLE DIM\_Promotion

(promotion\_key number NOT NULL,

promotion\_id number(5) NOT NULL,

promotion\_code varchar(15) NOT NULL,

discount\_rate number(2) NOT NULL,

PRIMARY KEY(promotion\_key)

);

**Date Dimension**

CREATE SEQUENCE date\_seq

START with 100001

INCREMENT by 1;

CREATE TABLE DIM\_Date

(Date\_key number NOT NULL, -- surrogate key

cal\_date date NOT NULL, -- every date

dayOfWeek number(1) NOT NULL, -- 1 to 7

dayNum\_calMonth number(2) NOT NULL, -- 1 to 31

dayNum\_calYear number(3) NOT NULL, -- 1 to 366

calWeek\_endDate date NOT NULL,

calWeek\_numYear number(2) NOT NULL, -- 1 to 53 weeks

calMonth\_name varchar(9) NOT NULL, -- JAN TO DEC

calMonth\_numYear number(2) NOT NULL, -- 01 to 12

cal\_year\_month char(7) NOT NULL, -- 'YYYY-MM'

cal\_quarter char(2) NOT NULL, -- 'Q1' to 'Q4'

cal\_Year number(4) NOT NULL,

holiday\_ind char(1) NOT NULL, -- 'Y' or 'N'

weekday\_ind char(1) NOT NULL, -- 'Y' or 'N'

PRIMARY KEY(Date\_key)

);

### 1.2.2 Fact Table

CREATE TABLE temp\_order AS SELECT \* FROM orders;

UPDATE temp\_order

SET promotionid = 10000

WHERE promotionid IS NULL;

CREATE TABLE SALES\_FACT

(date\_key number NOT NULL,

menu\_key number NOT NULL,

user\_key number NOT NULL,

rider\_key number NOT NULL,

restaurant\_key number NOT NULL,

promotion\_key number NOT NULL,

orderid number NOT NULL,

quantity number(3) NOT NULL,

price\_each number(7,2) NOT NULL,

line\_total number(9,2) NOT NULL,

discount\_amount number(7,2) NOT NULL,

actual\_amount\_paid number(9,2) NOT NULL,

rating number(1) NOT NULL,

order\_date date NOT NULL,

PRIMARY KEY(date\_key,menu\_key, user\_key,

rider\_key,restaurant\_key,promotion\_key, orderID),

FOREIGN KEY (date\_Key) REFERENCES DIM\_DATE (date\_Key),

FOREIGN KEY (menu\_key) REFERENCES DIM\_menu (menu\_key),

FOREIGN KEY (user\_key) REFERENCES DIM\_Users (user\_key),

FOREIGN KEY (rider\_key) REFERENCES DIM\_Riders (rider\_key),

FOREIGN KEY (restaurant\_key) REFERENCES DIM\_restaurants (restaurant\_key),

FOREIGN KEY (promotion\_key) REFERENCES DIM\_promotion (promotion\_key)

);

# Chapter 2 : Design of Data Warehouse

## 2.1 Script for initial loading

**Menu Dimension**

INSERT into DIM\_menu

SELECT DIM\_menu\_seq.nextval,m.menuListId,

UPPER(m.name), m.categoryID,UPPER(c.name),

m.timemealID, UPPER(t.timeSection),

m.pricePerUnit, m.unitSold,

m.restaurantid

FROM menulist m, timemeal t, category c

WHERE m.categoryid = c.categoryid AND

m.timemealid = t.timemealid;

**Restaurant Dimension**

INSERT into DIM\_Restaurants

SELECT DIM\_Restaurant\_seq.nextval,

A.restaurantID,B.BranchID, UPPER(A.name),

UPPER(C.typeName), UPPER(B.city),

UPPER(B.state), B.postcode

FROM Restaurant A, Branch B, RestaurantType C

WHERE (A.restaurantID = B.restaurantID) AND

1. restaurantTypeID = C.restaurantTypeID);

**Users Dimension**

INSERT into DIM\_USERS

SELECT dim\_users\_seq.nextval,

usersID, UPPER(name),

UPPER(gender),

dob, postcode,

UPPER(city), UPPER(state)

FROM users;

**Riders Dimension**

INSERT into DIM\_riders

SELECT dim\_rider\_seq.nextval,

riderID, UPPER(name),

lateNO

FROM rider;

**Promotion Dimension**

INSERT into DIM\_Promotion values (10000, 10000,'NONE', 0);

INSERT into DIM\_Promotion

SELECT dim\_promotion\_seq.nextval,

promotionId, UPPER(promotecode),

discountrate

FROM promotion;

**Date Dimension**

DECLARE

start\_date date; -- start of analysis date

end\_date date; -- end of analysis date

v\_dayOfWeek number(1);

v\_dayNumCalMth number(2);

v\_dayNumCalYr number(3);

v\_weekEndDate date;

v\_weekYear number(2);

v\_calMonthName varchar(9);

v\_calMonthNo number(2);

v\_calYear\_month char(7);

v\_quarter char(2);

v\_calYear number(4);

v\_weekDay\_ind char(1);

v\_holiday\_ind char(1);

BEGIN

-- set the start and end date e.g. date from 1/01/2015 to 31/03/2021

start\_date := to\_date('01/01/2015','dd/mm/yyyy');

end\_date := to\_date('31/03/2021','dd/mm/yyyy');

v\_holiday\_ind := 'N';

WHILE (start\_date <= end\_date) LOOP

v\_dayOfWeek := to\_char(start\_date,'D');

v\_dayNumCalMth := extract (day from start\_date);

v\_dayNumCalYr := to\_char(start\_date,'ddd');

v\_weekEndDate := start\_date+(7-to\_char(start\_date,'d'));

v\_weekYear := to\_char(start\_date,'ww');

v\_calMonthName := to\_char(start\_date,'MONTH');

v\_calMonthNo := extract (month from start\_date);

v\_calYear\_month:= to\_char(start\_date,'YYYY-MM');

v\_calYear := extract (year from start\_date);

if (v\_calMonthNo <=3) then

v\_quarter :='Q1';

elsif (v\_calMonthNo <=6) then

v\_quarter :='Q2';

elsif (v\_calMonthNo <=9) then

v\_quarter :='Q3';

else

v\_quarter :='Q4';

end if;

if (v\_dayOfWeek BETWEEN 2 and 6) then

v\_weekDay\_ind := 'Y';

else

v\_weekDay\_ind := 'N';

end if;

insert into DIM\_Date values(

date\_seq.nextval, start\_date, v\_dayOfWeek,

v\_dayNumCalMth,v\_dayNumCalYr, v\_weekEndDate, v\_weekYear,

v\_calMonthName, v\_calMonthNo, v\_calYear\_month, v\_quarter,

v\_calYear, v\_holiday\_ind, v\_weekDay\_ind);

/\*

dbms\_output.put\_line(date\_seq.nextval||' date is : '||to\_char(start\_date,'dd-mm-yyyy')||

' '||v\_dayOfWeek||' '||v\_dayNumCalMth||' '||v\_dayNumCalYr||' '||v\_weekDay\_ind);

\*/

start\_date := start\_date+1;

end LOOP;

END;

/

CREATE OR REPLACE procedure prc\_Updt\_Holiday(in\_Date IN date) IS

BEGIN

update DIM\_Date

set holiday\_ind = 'Y'

where cal\_date = in\_Date;

if (SQL%NOTFOUND) then

dbms\_output.put\_line('No rows updated');

else

dbms\_output.put\_line('Data updated');

end if;

exception

when OTHERS then

dbms\_output.put\_line('Database errors...');

END;

/

Execute the prc\_Updt\_Holiday procedure and key in a date to update the date which is holiday.

**Sales Fact**

CREATE TABLE temp\_order AS SELECT \* FROM orders;

UPDATE temp\_order

SET promotionid = 10000

WHERE promotionid IS NULL;

INSERT into Sales\_Fact

SELECT d.date\_key,m.menu\_key, u.user\_key,

r.rider\_key,e.restaurant\_key,p.promotion\_key,

b.orderid,b.quantity,m.price\_per\_unit, (b.quantity\*m.price\_per\_unit)

AS line\_total, (b.quantity\*m.price\_per\_unit\*(p.discount\_rate/100)) AS discount\_amount,(b.quantity\*m.price\_per\_unit)-((b.quantity\*m.price\_per\_unit)\*(p.discount\_rate/100)) AS actual\_amount\_paid,o.rating, o.orderdatetime

FROM DIM\_Date D

JOIN temp\_order o

on TRUNC(d.cal\_date) = TRUNC(o.orderdatetime)

JOIN orderdetails b

on b.orderid = o.orderid

JOIN DIM\_menu M

on M.menu\_list\_id = b.menulistid

JOIN DIM\_Users U

on U.user\_id = o.usersid

JOIN DIM\_Riders R

on R.rider\_id = o.riderid

JOIN DIM\_promotion P

on P.promotion\_id = o.promotionid

JOIN DIM\_restaurants E

on E.rest\_branchID = o.branchID;

In order to solve the null value in the orders table and not changing the original table in the database. A temp\_order was created and will update rows where the promotion id is null into 10000. Our team decided not to change the original table to keep the integrity of the database. Promotion id start with 10001 and 10000 was unused. Temp\_order table value will be inserted into Sales\_fact with NOT NULL for the promotion id.

## 2.2 Script for subsequent loading

**Menu Dimension**

INSERT into DIM\_menu

SELECT DIM\_menu\_seq.nextval,m.menuListId, UPPER(m.name),

m.categoryID,UPPER(c.name),

m.timemealID,UPPER(t.timeSection),m.pricePerUnit,

m.unitSold,m.restaurantid

FROM menulist m, timemeal t, category c

WHERE m.categoryid = c.categoryid AND m.timemealid =

t.timemealid AND m.menuListId > 14078;

The last menu list ID was 14078. All id is integer data type. If the menulist table consist any new menu list id after 14078 will be loaded into DIM\_menu,

**Restaurant Dimension**

INSERT into DIM\_Restaurants

SELECT DIM\_Restaurant\_seq.nextval,

A.restaurantID,B.BranchID, UPPER(A.name),

UPPER(C.typeName), UPPER(B.city),

UPPER(B.state), B.postcode

FROM Restaurant A, Branch B, RestaurantType C

WHERE (A.restaurantID = B.restaurantID) AND

(A.restaurantTypeID = C.restaurantTypeID) AND

B.BranchID>10540 ;

The last branch ID was 14078. All id is integer data type. If the branch table consists of any new branch id after 10540 will be loaded into DIM\_Restaurants.

**Users Dimension**

INSERT into DIM\_USERS

SELECT dim\_users\_seq.nextval,

usersID, UPPER(name), UPPER(gender),

dob, postcode, UPPER(city),

UPPER(state)

FROM users

WHERE usersID > 10900;

The last user ID was 10900. All id is integer data type. If the user table consists of any new users id after 10540 will be loaded into DIM\_Users.

**Riders Dimension**

INSERT into DIM\_riders

SELECT dim\_rider\_seq.nextval,

riderID, UPPER(name),lateNO

FROM rider

WHERE rider\_id>10100;

The last rider ID was 10100. All id is integer data type. If the rider table consists of any new rider id after 10100 will be loaded into DIM\_riders.

**Promotion Dimension**

INSERT into DIM\_Promotion

SELECT dim\_promotion\_seq.nextval,

promotionId, UPPER(promotecode),

discountrate

FROM promotion

WHERE promotionId>10065;

**Date Dimension**

DECLARE

start\_date date; -- start of analysis date

end\_date date; -- end of analysis date

v\_dayOfWeek number(1);

v\_dayNumCalMth number(2);

v\_dayNumCalYr number(3);

v\_weekEndDate date;

v\_weekYear number(2);

v\_calMonthName varchar(9);

v\_calMonthNo number(2);

v\_calYear\_month char(7);

v\_quarter char(2);

v\_calYear number(4);

v\_weekDay\_ind char(1);

v\_holiday\_ind char(1);

BEGIN

-- set the start and end date e.g. date from 01/04/2021 to 30/04/2021

start\_date := to\_date('01/04/2021,'dd/mm/yyyy');

end\_date := to\_date('30/04/2021,'dd/mm/yyyy');

v\_holiday\_ind := 'N';

WHILE(start\_date <= end\_date) LOOP

v\_dayOfWeek := to\_char(start\_date,'D');

v\_dayNumCalMth := extract (day from start\_date);

v\_dayNumCalYr := to\_char(start\_date,'ddd');

v\_weekEndDate := start\_date+(7-to\_char(start\_date,'d'));

v\_weekYear := to\_char(start\_date,'ww');

v\_calMonthName := to\_char(start\_date,'MONTH');

v\_calMonthNo := extract (month from start\_date);

v\_calYear\_month:= to\_char(start\_date,'YYYY-MM');

v\_calYear := extract (year from start\_date);

if (v\_calMonthNo <=3) then

v\_quarter :='Q1';

elsif (v\_calMonthNo <=6) then

v\_quarter :='Q2';

elsif (v\_calMonthNo <=9) then

v\_quarter :='Q3';

else

v\_quarter :='Q4';

end if;

if (v\_dayOfWeek BETWEEN 2 and 6) then

v\_weekDay\_ind := 'Y';

else

v\_weekDay\_ind := 'N';

end if;

INSERT into DIM\_Date values(

date\_seq.nextval, start\_date, v\_dayOfWeek,

v\_dayNumCalMth,v\_dayNumCalYr, v\_weekEndDate,

v\_weekYear,v\_calMonthName, v\_calMonthNo,

v\_calYear\_month, v\_quarter,

v\_calYear, v\_holiday\_ind, v\_weekDay\_ind);

/\*

dbms\_output.put\_line(date\_seq.nextval||' date is : '||to\_char(start\_date,'dd-mm-yyyy')||

' '||v\_dayOfWeek||' '||v\_dayNumCalMth||' '||v\_dayNumCalYr||' '||v\_weekDay\_ind);

\*/

start\_date := start\_date+1;

end LOOP;

END;

/

EXEC prc\_Updt\_Holiday('31-DEC-21')

The last date was 31/03/2021. New data can be generated by just changing the start date and the end date. After the new date was generated and will be inserted into DIM\_Date. Then execute the prc\_Updt\_Holiday procedure and key in to update the date which is holiday.

**Sales Fact**

DROP TABLE temp\_table

CREATE TABLE temp\_order AS SELECT \* FROM orders;

UPDATE temp\_order

SET promotionid = 10000

WHERE promotionid IS NULL;

INSERT into Sales\_Fact

SELECT d.date\_key, m.menu\_key, u.user\_key,

r.rider\_key, e.restaurant\_key, p.promotion\_key, b.orderid,

b.quantity, m.price\_per\_unit, (b.quantity\*m.price\_per\_unit)

AS line\_total, (b.quantity\*m.price\_per\_unit\*(p.discount\_rate/100)) AS discount\_amount,(b.quantity\*m.price\_per\_unit)-((b.quantity\*m.price\_per\_unit)\*(p.discount\_rate/100)) AS actual\_amount\_paid,o.rating, o.orderdatetime

FROM DIM\_Date D

join temp\_order o

on TRUNC(d.cal\_date) = TRUNC(o.orderdatetime)

join orderdetails b

on b.orderid = o.orderid

join DIM\_menu M

on M.menu\_list\_id = b.menulistid

join DIM\_Users U

on U.user\_id = o.usersid

join DIM\_Riders R

on R.rider\_id = o.riderid

join DIM\_promotion P

on P.promotion\_id = o.promotionid

join DIM\_restaurants E

on E.rest\_branchID = o.branchID

WHERE TRUNC(o.orderdatetime) > ‘31-MAR-2021’;

Temp\_order will be dropped first and create a new one by copying the original orders table. Any orders after 31-MAR-2021 will be updated into the Sales\_Fact table.

# Chapter 3 : Business Analytics Reports

## 3.1 Tan Teoh Xin Ee

### 3.1.1 Top 10 Sales on the Menu List

### This report is used to analysis on the top 10 sales for the menu list in different restaurant. We are able to know that what is the most sales menu in what amount of quantity sold in the certain restaurant. At the end of the report, we can see that the rank for the quantity sold and also rank of sales. Therefore, we can use this report to focus more on the certain restaurant on which menu list.

### Script/Coding

### set linesize 90

### set pagesize 150

### column menu\_name format a12

### column restaurant format a18

### column ranks format 99

### column rankq format 9999

### column sales format 999999999.99

### column ranks heading 'Sales|Rank'

### column rankq heading 'Quantity|Rank'

### create or replace view menuList\_view as

### select \*

### from(

### select (A.menu\_list\_id)as Menu\_List\_ID,

### (a.menu\_name) as Menu\_Name,

### (sum(b.quantity)) as Quantity,

### (sum(b.line\_total)) as Sales,

### (c.rest\_name) as Restaurant,

### RANK() OVER (ORDER BY sum(b.line\_total) desc) ranks,

### RANK() OVER (ORDER BY sum(b.quantity) desc) rankq

### from dim\_menu a, sales\_fact b, dim\_restaurants c

### where a.menu\_key = b.menu\_key and

### c.restaurant\_key = b.restaurant\_key and

### c.restaurantid = a.restaurant\_id

### group by a.menu\_list\_id,a.menu\_name,c.rest\_name

### order by sum(b.line\_total) desc)

### where rownum <=10;

### TTITLE CENTER 'Top 10 Sales on the Menu List'SKIP 1 -

### CENTER ============================= SKIP 1

### select \* from menuList\_view;

### Sample output

### SQL> set linesize 90

### SQL> set pagesize 150

### SQL> column menu\_name format a12

### SQL> column restaurant format a18

### SQL> column ranks format 99

### SQL> column rankq format 9999

### SQL> column sales format 999999999.99

### SQL> column ranks heading 'Sales|Rank'

### SQL> column rankq heading 'Quantity|Rank'

### SQL> create or replace view menuList\_view as

### 2 select \*

### 3 from(

### 4 select (A.menu\_list\_id)as Menu\_List\_ID,

### 5 (a.menu\_name) as Menu\_Name,

### 6 (sum(b.quantity)) as Quantity,

### 7 (sum(b.line\_total)) as Sales,

### 8 (c.rest\_name) as Restaurant,

### 9 RANK() OVER (ORDER BY sum(b.line\_total) desc) ranks,

### 10 RANK() OVER (ORDER BY sum(b.quantity) desc) rankq

### 11 from dim\_menu a, sales\_fact b, dim\_restaurants c

### 12 where a.menu\_key = b.menu\_key and

### 13 c.restaurant\_key = b.restaurant\_key and

### 14 c.restaurantid = a.restaurant\_id

### 15 group by a.menu\_list\_id,a.menu\_name,c.rest\_name

### 16 order by sum(b.line\_total) desc)

### 17 where rownum <=10;

### View created.

### SQL>

### SQL> TTITLE CENTER 'Top 10 Sales on the Menu List'SKIP 1 -

### > CENTER ============================= SKIP 1

### SQL> select \* from menuList\_view;

### Top 10 Sales on the Menu List

### =============================

### Sales Quantity

### MENU\_LIST\_ID MENU\_NAME QUANTITY SALES RESTAURANT Rank Rank

### ------------ ------------ ---------- ------------- ------------------ ----- --------

### 13991 500G WAGYU B 1207 1086300.00 NAUGHTY NURI'S 1 1157

### EEF

### 13990 300G WAGYU B 1056 633600.00 NAUGHTY NURI'S 2 1407

### EEF

### 13827 BEEF GOULASH 5154 402012.00 FARQUHAR MANSION F 3 23

### SOUP

### 12380 PAD KRA PAW 4339 375800.79 FEI FAN STEAMBOAT 4 55

### 12378 THAI BBQ CHI 4386 364695.90 FEI FAN STEAMBOAT 5 54

### CKEN WINGS (

### 5PCS)

### 13989 200G WAGYU B 868 347200.00 NAUGHTY NURI'S 6 1691

### EEF

### 13571 HAIDILAO 4 P 1151 322280.00 HAIDILAO HOTPOT 7 1259

### AX SET

### 11906 POMFRET FISH 1928 289200.00 MING GARDEN 8 478

### IN TWO PREP

### ARATIONS

### 12379 STIR FRIED P 4150 287055.50 FEI FAN STEAMBOAT 9 66

### RAWN, CHICKE

### N & CHILI

### 12399 FRIED CHICKE 3535 262049.55 IMPIAN MAJU MAMAK 10 93

### N (DRUMSTICK

### )

### 10 rows selected.

### 3.1.2 Top 10 Promotion Code used in Restaurant and Menu List in Selected Year

### This report is used to analysis on Top 10 Promotion code applied in which restaurant and menu list in the selected year. We are able to see that in certain restaurant has applied amount of promotion code on the certain menu list. Therefore, by using this report we able to tackle on the customers who from the certain restaurant and ordered certain menu list.

### Script/Coding

### set linesize 90

### set pagesize 150

### column menu\_name format a12

### column restaurant format a18

### create or replace view promo\_view as

### select \*

### from(

### select (a.promotion\_id) as Promo\_ID,

### (a.promotion\_code) as Code,

### (sum(b.quantity)) as Quantity,

### (c.menu\_name) as Menu\_Name,

### (d.rest\_name) as Restaurant,

### RANK() OVER (ORDER BY sum(b.quantity) desc) Rank

### from dim\_promotion a, sales\_fact b, dim\_menu c, dim\_restaurant d

### where a.promotion\_key = b.promotion\_key and

### c.menu\_key = b.menu\_key and

### d.restaurant\_key = b.restaurant\_key and

### extract(year from b.order\_date) = '&year' and

### a.promotion\_id != 10000

### group by a.promotion\_id,a.promotion\_code,c.menu\_name,d.rest\_name

### order by sum(b.quantity) desc)

### where rownum <=10;

### TTITLE CENTER 'Total 10 Promotion Code used by Restaurant and Menu List'SKIP 1 -

### CENTER ========================================================== SKIP 1

### select \* from promo\_view;

### Sample output

### SQL> set linesize 90

### SQL> set pagesize 150

### SQL> column menu\_name format a12

### SQL> column restaurant format a18

### SQL> create or replace view promo\_view as

### 2 select \*

### 3 from(

### 4 select (a.promotion\_id) as Promo\_ID,

### 5 (a.promotion\_code) as Code,

### 6 (sum(b.quantity)) as Quantity,

### 7 (c.menu\_name) as Menu\_Name,

### 8 (d.rest\_name) as Restaurant,

### 9 RANK() OVER (ORDER BY sum(b.quantity) desc) Rank

### 10 from dim\_promotion a, sales\_fact b, dim\_menu c, dim\_restaurant d

### 11 where a.promotion\_key = b.promotion\_key and

### 12 c.menu\_key = b.menu\_key and

### 13 d.restaurant\_key = b.restaurant\_key and

### 14 extract(year from b.order\_date) = '&year' and

### 15 a.promotion\_id != 10000

### 16 group by a.promotion\_id,a.promotion\_code,c.menu\_name,d.rest\_name

### 17 order by sum(b.quantity) desc)

### 18 where rownum <=10;

### Enter value for year: 2020

### old 14: extract(year from b.order\_date) = '&year' and

### new 14: extract(year from b.order\_date) = '2020' and

### View created.

### SQL> TTITLE CENTER 'Total 10 Promotion Code used by Restaurant and Menu List'SKIP 1 -

### > CENTER ========================================================== SKIP 1

### SQL> select \* from promo\_view;

### Total 10 Promotion Code used by Restaurant and Menu List

### ==========================================================

### PROMO\_ID CODE QUANTITY MENU\_NAME RESTAURANT RANK

### ---------- --------------- ---------- ------------ ------------------ ----------

### 10029 SUPPERTIME 41 BEEF GOULASH Farquhar Mansion F 1

### SOUP

### 10041 KAKIONLINE2020 40 PASSION DRAG The Black Whale Bu 2

### ONFRUIT WHAL

### CANO

### 10048 HELLO2020 38 MINI CURRY F Fei Fan Steamboat 3

### ISH POT

### 10035 PANDADISC 37 SPICY TENDER Burger King 4

### CRISP

### 10046 MACALLAN20 36 SINGLE BBQ B Burger King 5

### EEFACON

### 10040 PANDABUY 36 DARK KNIGHT Spade Burger 5

### 10012 EATMORE 34 HOT CHOCOLAT Coffee Smith 7

### E

### 10056 PANDA50 34 BEEF GOULASH Farquhar Mansion F 7

### SOUP

### 10010 SHOPMOOD 34 SPRIT Mr.Dakgalbi 7

### 10032 LUNCH 33 ORGASHROOM Spade Burger 10

### 10 rows selected.

### 3.1.3 Top 10 Discount Amount and Percentage on Promotion Code

### This report is used to analysis on the Top 10 discount amount from certain promotion code and the percentage compare to the first discount amount on promotion code. We are able to observe that the from what restaurant had applied what promotion code in the most and what is the discount amount had gave to the customers. Therefore, by using this report we can know that the amount of company lost from the discount and from what promotion code.

### Script/Coding

### set linesize 90

### set pagesize 150

### column percentage format 999.99

### column dis\_amount format 9999999.99

### create or replace view dis\_view1 as

### select \*

### from(

### select (a.promotion\_id) as Promo\_ID,

### (a.promotion\_code) as Code,

### (sum(b.discount\_amount)) as Dis\_amount,

### (c.rest\_name) as Restaurant,

### RANK() OVER (ORDER BY sum(b.discount\_amount) desc) Rank

### from dim\_promotion a, sales\_fact b, dim\_restaurants c

### where a.promotion\_key = b.promotion\_key and

### c.restaurant\_key = b.restaurant\_key

### group by a.promotion\_id,a.promotion\_code,c.rest\_name

### order by sum(b.discount\_amount) desc)

### where rownum <=1;

### create or replace view dis\_view2 as

### select \*

### from(

### select (a.promotion\_id) as Promo\_ID,

### (a.promotion\_code) as Code,

### (sum(b.discount\_amount)) as Dis\_amount,

### (c.rest\_name) as Restaurant,

### RANK() OVER (ORDER BY sum(b.discount\_amount) desc) Rank

### from dim\_promotion a, sales\_fact b, dim\_restaurants c

### where a.promotion\_key = b.promotion\_key and

### c.restaurant\_key = b.restaurant\_key

### group by a.promotion\_id,a.promotion\_code,c.rest\_name

### order by sum(b.discount\_amount) desc)

### where rownum <=10;

### create or replace view disView as

### select b.Promo\_ID,

### b.Code,

### b.Dis\_amount,

### b.Restaurant,

### b.Rank,

### round(b.dis\_amount/a.dis\_amount\*100,2) Percentage

### from dis\_view1 a, dis\_view2 b;

### TTITLE CENTER 'Total 10 Discount Amount and Percentage'SKIP 1 -

### CENTER ========================================== SKIP 1

### select \* from disView;

Sample output

SQL> set linesize 90

SQL> set pagesize 150

SQL> column percentage format 999.99

SQL> column dis\_amount format 9999999.99

SQL> create or replace view dis\_view1 as

2 select \*

3 from(

4 select (a.promotion\_id) as Promo\_ID,

5 (a.promotion\_code) as Code,

6 (sum(b.discount\_amount)) as Dis\_amount,

7 (c.rest\_name) as Restaurant,

8 RANK() OVER (ORDER BY sum(b.discount\_amount) desc) Rank

9 from dim\_promotion a, sales\_fact b, dim\_restaurants c

10 where a.promotion\_key = b.promotion\_key and

11 c.restaurant\_key = b.restaurant\_key

12 group by a.promotion\_id,a.promotion\_code,c.rest\_name

13 order by sum(b.discount\_amount) desc)

14 where rownum <=1;

View created.

SQL>

SQL> create or replace view dis\_view2 as

2 select \*

3 from(

4 select (a.promotion\_id) as Promo\_ID,

5 (a.promotion\_code) as Code,

6 (sum(b.discount\_amount)) as Dis\_amount,

7 (c.rest\_name) as Restaurant,

8 RANK() OVER (ORDER BY sum(b.discount\_amount) desc) Rank

9 from dim\_promotion a, sales\_fact b, dim\_restaurants c

10 where a.promotion\_key = b.promotion\_key and

11 c.restaurant\_key = b.restaurant\_key

12 group by a.promotion\_id,a.promotion\_code,c.rest\_name

13 order by sum(b.discount\_amount) desc)

14 where rownum <=10;

View created.

SQL>

SQL> create or replace view disView as

2 select b.Promo\_ID,

3 b.Code,

4 b.Dis\_amount,

5 b.Restaurant,

6 b.Rank,

7 round(b.dis\_amount/a.dis\_amount\*100,2) Percentage

8 from dis\_view1 a, dis\_view2 b;

View created.

SQL> TTITLE CENTER 'Total 10 Discount Amount and Percentage'SKIP 1 -

> CENTER ========================================== SKIP 1

SQL> select \* from disView;

Total 10 Discount Amount and Percentage

==========================================

PROMO\_ID CODE DIS\_AMOUNT RESTAURANT RANK PERCENTAGE

---------- --------------- ----------- ------------------ ---------- ----------

10058 PANDAMANIA 23638.80 NAUGHTY NURI'S 1 100.00

10003 GRABPAY 15547.50 NAUGHTY NURI'S 2 65.77

10057 BOOZEAT 15107.10 NAUGHTY NURI'S 3 63.91

10040 PANDABUY 14331.00 NAUGHTY NURI'S 4 60.62

10025 PANDAMORE 13735.33 FEI FAN STEAMBOAT 5 58.11

10017 CHRISTMAS 12222.60 NAUGHTY NURI'S 6 51.71

10056 PANDA50 11599.58 FEI FAN STEAMBOAT 7 49.07

10052 NEWPANDA 11462.33 FEI FAN STEAMBOAT 8 48.49

10059 ONLYONPANDA 10898.30 NAUGHTY NURI'S 9 46.10

10016 MERDEKA 10786.01 POCHA KOREAN BBQ 10 45.63

10 rows selected.

## 3.3 Tan Yi Hong

### 3.3.1 Top 10 Most Popular Restaurant For Year 2020 Second Half Compared To First Half For Selected City

This report is to show the most popular restaurants in the second half of the year 2020 and compared to the first half of the year for the selected city. With this report, ACME Food Delivery (AFD) can know what are the most popular restaurants that received the most orders from customers in the selected city throughout the first half and second half of the year 2020. The company can use this report to analyze what is the area that received most orders and also the trend for the popularity of restaurants. This will help to predict the future popularity of restaurants and to boost their sales by making events or sales with the delivery options.

### Script/Coding

### set pagesize 200

### column rest\_name format a20 TRUNC

### column Rest\_branchid format 99999 TRUNC

### column Total\_Orders\_1stHalf format 999 TRUNC

### column Total\_Orders\_2ndHalf format 999 TRUNC

### column Average\_Rating format 9.00 TRUNC

### column Total\_Sales\_1stHalf format 999,999.00 TRUNC

### column Total\_Sales\_2ndHalf format 999,999.00 TRUNC

### column Popular\_Ranking\_1stHalf format 99 TRUNC

### column Popular\_Ranking\_2ndHalf format 99 TRUNC

### COLUMN Total\_Orders\_1stHalf HEADING 'Total|Orders|1stHalf'

### COLUMN Total\_Sales\_1stHalf HEADING 'Total|Sales|1stHalf'

### COLUMN Average\_Rating\_1stHalf HEADING 'Avg|Rating|1stHalf'

### COLUMN Popular\_Ranking\_1stHalf HEADING 'Rank|1st|Half'

### COLUMN Total\_Orders\_2ndHalf HEADING 'Total|Orders|2ndHalf'

### COLUMN Total\_Sales\_2ndHalf HEADING 'Total|Sales|2ndHalf'

### COLUMN Average\_Rating\_2ndHalf HEADING 'Avg|Rating|2ndHalf'

### COLUMN Popular\_Ranking\_2ndHalf HEADING 'Rank|2nd|Half'

### COLUMN Rest\_branchid HEADING 'Rest|Branch|ID'

### COLUMN Rest\_Name HEADING 'Rest|Name'

### CREATE OR REPLACE VIEW top\_rest2020\_firstHalf AS

### SELECT \*

### FROM(SELECT r.rest\_branchid, r.rest\_name, r.rest\_city, COUNT(distinct s.orderid) AS Total\_Orders\_1stHalf,

### SUM(s.line\_total) AS Total\_Sales\_1stHalf, ROUND(AVG(s.rating),2) AS Average\_Rating\_1stHalf,

### RANK() OVER (ORDER BY count(distinct s.orderid) DESC) Popular\_Ranking\_1stHalf

### FROM sales\_fact s, DIM\_restaurants r

### WHERE s.restaurant\_key = r.restaurant\_key

### AND rest\_city = UPPER('&city\_input')

### AND EXTRACT(year FROM s.order\_date) = 2020

### AND (EXTRACT(month FROM s.order\_date) BETWEEN 1 AND 6)

### GROUP BY rest\_branchid, rest\_name, rest\_city

### ORDER BY Total\_Orders\_1stHalf DESC)

### WHERE Popular\_Ranking\_1stHalf <= 10;

### CREATE OR REPLACE VIEW top\_rest2020\_secondHalf AS

### SELECT \*

### FROM(SELECT r.rest\_branchid, r.rest\_name,

### COUNT(distinct s.orderid) AS Total\_Orders\_2ndHalf,

### SUM(s.line\_total) AS Total\_Sales\_2ndHalf, ROUND(AVG(s.rating),2) AS Average\_Rating\_2ndHalf

### FROM sales\_fact s, DIM\_restaurants r

### WHERE s.restaurant\_key = r.restaurant\_key

### AND EXTRACT(year FROM s.order\_date) = 2020

### AND (EXTRACT(month FROM s.order\_date) BETWEEN 7 AND 12)

### GROUP BY r.rest\_branchid, r.rest\_name);

### set linesize 99

### TTITLE CENTER 'Top 10 Most Popular Restaurant For Year 2020 Second Half Compared To First Half For Selected City' SKIP 1 -

### CENTER ================================================================================================= SKIP 1

### SELECT a.rest\_branchid, a.rest\_name, a.Total\_Orders\_1stHalf, a.Total\_Sales\_1stHalf,

### a.Average\_Rating\_1stHalf, a.Popular\_Ranking\_1stHalf, b.Total\_Orders\_2ndHalf,

### b.Total\_Sales\_2ndHalf, b.Average\_Rating\_2ndHalf,

### RANK() OVER (ORDER BY b.Total\_Orders\_2ndHalf DESC) Popular\_Ranking\_2ndHalf

### FROM top\_rest2020\_firstHalf a, top\_rest2020\_secondHalf b

### WHERE a.rest\_branchid = b.rest\_branchid;

### Sample output

### SQL> set linesize 99

### SQL> TTITLE CENTER 'Top 10 Most Popular Restaurant For Year 2020 Second Half Compared To First Half For Selected City' SKIP 1 -

### > CENTER ================================================================================================= SKIP 1

### SQL>

### SQL> SELECT a.rest\_branchid, a.rest\_name, a.Total\_Orders\_1stHalf, a.Total\_Sales\_1stHalf,

### 2 a.Average\_Rating\_1stHalf, a.Popular\_Ranking\_1stHalf, b.Total\_Orders\_2ndHalf,

### 3 b.Total\_Sales\_2ndHalf, b.Average\_Rating\_2ndHalf,

### 4 RANK() OVER (ORDER BY b.Total\_Orders\_2ndHalf DESC) Popular\_Ranking\_2ndHalf

### 5 FROM top\_rest2020\_firstHalf a, top\_rest2020\_secondHalf b

### 6 WHERE a.rest\_branchid = b.rest\_branchid;

### Top 10 Most Popular Restaurant For Year 2020 Second Half Compared To First Half For Selected City

### =================================================================================================

### Rest Total Total Avg Rank Total Total Avg Rank

### Branch Rest Orders Sales Rating 1st Orders Sales Rating 2nd

### ID Name 1stHalf 1stHalf 1stHalf Half 2ndHalf 2ndHalf 2ndHalf Half

### ------ -------------------- ------- ----------- ---------- ---- ------- ----------- ---------- ----

### 10106 TEXAS CHICKEN 51 11,791.82 3.02 6 57 15,532.47 2.68 1

### 10078 KOBQ 59 30,976.00 2.84 1 54 26,104.00 3.08 2

### 10422 RAJA LAUT CURRY HOUS 53 3,012.00 2.87 3 54 3,278.90 2.98 2

### 10043 LK WESTERN 51 21,282.00 3.05 6 50 21,681.20 3.02 4

### 10386 ARABELLA RESTAURANT 54 22,193.00 3.47 2 45 18,396.00 2.57 5

### 10031 THE TOKYO RESTAURANT 51 24,816.00 3.23 6 42 15,567.00 2.51 6

### 10452 HAIDILAO HOTPOT 51 31,139.00 3.24 6 39 24,027.00 3.14 7

### 10507 NASI KANDAR SUBAIDAH 53 3,501.50 3.19 3 39 2,728.50 3.38 7

### 10317 IMPIAN MAJU MAMAK 52 41,771.42 3.22 5 38 28,511.91 3.14 9

### 10013 BLACK MARKET 51 20,267.00 2.63 6 32 12,540.00 2.95 10

### 10 rows selected.

### 3.2.2 Total Sales Of Each State For Year 2020 Compared To 2019

This report shows the total sales from each state for the year 2020 compared to year 2019. The purpose of this report is to analyze the growth or the drop of the sales in each state. ACME Food Delivery (AFD) can use this report to see which state has the highest sales and also lowest sales, as well as the increase or decrease of sales. The company can help to increase the sales for the lower rank states to make more profit from them, and also find out the reason for the trend of the sales.

### Script/Coding

### set pagesize 200

### column state format a20

### column Total\_Sales2019 format 999,999,999.00

### column Total\_Sales2020 format 999,999,999.00

### column Sales\_DIff format 9,999,999.00

### column Percent\_Sales\_Diff format 999.00

### COLUMN Total\_Order2019 HEADING 'Total|Order|2019'

### COLUMN Total\_Order2020 HEADING 'Total|Order|2020'

### COLUMN Total\_Sales2019 HEADING 'Total|Sales|2019'

### COLUMN Total\_Sales2020 HEADING 'Total|Sales|2020'

### COLUMN Sales\_Diff HEADING 'Sales|Diff'

### COLUMN Percent\_Sales\_Diff HEADING 'Percent|Sales|Diff'

### CREATE OR REPLACE VIEW top\_state2020 AS

### SELECT A.\*, 2020 AS Year\_NUM

### FROM (SELECT u.state, COUNT(\*) AS Total\_Order2020,

### SUM(line\_total) AS Total\_Sales2020

### FROM dim\_users u, sales\_fact s

### WHERE EXTRACT(year from s.order\_date) = 2020 AND u.user\_key = s.user\_key

### GROUP BY u.state

### ORDER BY Total\_Sales2020 DESC) A;

### CREATE OR REPLACE VIEW top\_state2019 AS

### SELECT B.\*,2019 AS Year\_NUM

### FROM (SELECT u.state, count(\*) AS Total\_Order2019,

### SUM(line\_total) AS Total\_Sales2019

### FROM dim\_users u, sales\_fact s

### WHERE EXTRACT(year FROM s.order\_date) = 2019 AND u.user\_key = s.user\_key

### GROUP BY u.state

### ORDER BY Total\_Sales2019 DESC) B;

### set linesize 100

### TTITLE CENTER 'Total Sales Of Each State For Year 2020 Compared To 2019' SKIP 1 -

### CENTER ========================================================== SKIP 1

### SELECT a.state, a.total\_order2019 AS Total\_Order2019, a.total\_sales2019 AS Total\_Sales2019,

### b.total\_order2020 AS Total\_Order2020, b.total\_sales2020 AS Total\_Sales2020,

### (b.total\_sales2020-a.total\_sales2019) AS Sales\_Diff,

### ROUND((((b.total\_sales2020-a.total\_sales2019)/a.total\_sales2019) \* 100),2) AS Percent\_Sales\_Diff

### FROM top\_state2019 a, top\_state2020 b

### WHERE a.state = b.state

### ORDER BY Total\_Sales2019 DESC;

### Sample output

### SQL> set linesize 100

### SQL> TTITLE CENTER 'Total Sales Of Each State For Year 2020 Compared To 2019' SKIP 1 -

### > CENTER ========================================================== SKIP 1

### SQL>

### SQL> SELECT a.state, a.total\_order2019 AS Total\_Order2019, a.total\_sales2019 AS Total\_Sales2019,

### 2 b.total\_order2020 AS Total\_Order2020, b.total\_sales2020 AS Total\_Sales2020,

### 3 (b.total\_sales2020-a.total\_sales2019) AS Sales\_Diff,

### 4 ROUND((((b.total\_sales2020-a.total\_sales2019)/a.total\_sales2019) \* 100),2) AS Percent\_Sales\_Diff

### 5 FROM top\_state2019 a, top\_state2020 b

### 6 WHERE a.state = b.state

### 7 ORDER BY Total\_Sales2019 DESC;

### Total Sales Of Each State For Year 2020 Compared To 2019

### ==========================================================

### Total Total Total Total Percent

### Order Sales Order Sales Sales Sales

### STATE 2019 2019 2020 2020 Diff Diff

### -------------------- ---------- --------------- ---------- --------------- ------------- -------

### SELANGOR 32661 3,558,775.59 33068 3,707,359.89 148,584.30 4.18

### WILAYAH PERSEKUTUAN 20080 2,202,502.18 19971 2,212,835.00 10,332.82 .47

### PULAU PINANG 12089 1,282,468.94 11653 1,276,590.91 -5,878.03 -.46

### SARAWAK 11139 1,194,311.46 10625 1,192,964.17 -1,347.29 -.11

### PAHANG 9519 1,088,194.06 9406 1,034,951.55 -53,242.51 -4.89

### JOHOR 9077 995,997.95 8717 978,499.14 -17,498.81 -1.76

### PERAK 7862 880,220.09 7361 796,928.39 -83,291.70 -9.46

### NEGERI SEMBILAN 5554 584,223.10 5308 600,257.67 16,034.57 2.74

### KEDAH 4005 440,428.66 3844 460,019.75 19,591.09 4.45

### SABAH 3169 347,194.28 3161 331,770.15 -15,424.13 -4.44

### MELAKA 2027 230,062.61 2070 256,700.91 26,638.30 11.58

### PERLIS 2009 224,859.25 1992 230,967.90 6,108.65 2.72

### KELANTAN 1582 169,872.35 1705 185,542.83 15,670.48 9.22

### 13 rows selected.

### 3.2.3 Top 10 Most Order Delivered Rider For 2020 Second Half Compared to First Half

This report will show the top 10 riders for 2020 second half compared to the first half of the year 2020. This is to know which rider has the best performance among all of them and analyze their performance throughout the year. ACME Food Delivery (AFD) can give bonus rewards for the best performance rider to encourage all other riders to keep improving their performance such as maintaining 0 late number. On the other hand, by giving rewards for the good performance riders, this will also attract more people to come to work as a rider for them because of the benefits provided.

Script/Coding

set pagesize 200

column rider\_name format a20 TRUNC

column rider\_id format 99999 TRUNC

column Total\_Sales\_1stHalf format 999,999.00

column Total\_Sales\_2ndHalf format 999,999.00

column Total\_Orders\_1stHalf format 999

column Total\_Orders\_2ndHalf format 999

column Rank\_1stHalf format 99

column Rank\_2ndHalf format 99

column Late\_No\_1stHalf format 99

column Late\_No\_2ndHalf format 99

COLUMN Total\_Orders\_1stHalf HEADING 'Total|Orders|1stHalf'

COLUMN Total\_Orders\_2ndHalf HEADING 'Total|Orders|2ndHalf'

COLUMN Late\_No\_1stHalf HEADING 'Late|No|1stHalf'

COLUMN Late\_No\_2ndHalf HEADING 'Late|No|2ndHalf'

COLUMN Rank\_1stHalf HEADING 'Rank|1st|Half'

COLUMN Rank\_2ndHalf HEADING 'Rank|2nd|Half'

COLUMN Rider\_id HEADING 'Rider|ID'

COLUMN Rider\_name HEADING 'Rider|Name'

CREATE OR REPLACE VIEW top\_rider\_firstHalf AS

SELECT \*

FROM(

SELECT r.rider\_id, r.rider\_name, COUNT(distinct s.orderid) AS Total\_Orders\_1stHalf,

r.rider\_lateno AS Late\_No\_1stHalf, SUM(s.line\_total) AS Total\_Sales\_1stHalf,

RANK() OVER (ORDER BY count(distinct s.orderid) DESC) Rank\_1stHalf

FROM sales\_fact s, DIM\_riders r

WHERE s.rider\_key = r.rider\_key

AND EXTRACT(year FROM s.order\_date) = 2020

AND (EXTRACT(month FROM s.order\_date) BETWEEN 1 AND 6)

GROUP BY rider\_id, rider\_name, rider\_lateno

ORDER BY Total\_Orders\_1stHalf DESC

)

WHERE Rank\_1stHalf <=10;

CREATE OR REPLACE VIEW top\_rider\_secondHalf AS

SELECT r.rider\_id, r.rider\_name, COUNT(distinct s.orderid) AS Total\_Orders\_2ndHalf,

r.rider\_lateno AS Late\_No\_2ndHalf, SUM(s.line\_total) AS Total\_Sales\_2ndHalf

FROM sales\_fact s, DIM\_riders r

WHERE s.rider\_key = r.rider\_key

AND EXTRACT(year FROM s.order\_date) = 2020

AND (EXTRACT(month FROM s.order\_date) BETWEEN 7 AND 12)

GROUP BY rider\_id, rider\_name, rider\_lateno;

set linesize 95

TTITLE CENTER 'Top 10 Most Order Delivered Rider For 2020 Second Half Compared to First Half' SKIP 1 -

CENTER ============================================================================= SKIP 1

SELECT a.\*, b.Total\_Orders\_2ndHalf, b.Late\_No\_2ndHalf, b.Total\_Sales\_2ndHalf,

RANK() OVER (ORDER BY Total\_Orders\_2ndHalf DESC) Rank\_2ndHalf

FROM top\_rider\_firstHalf a, top\_rider\_secondHalf b

WHERE a.rider\_id = b.rider\_id;

## Sample output

## SQL> set linesize 95

## SQL> TTITLE CENTER 'Top 10 Most Order Delivered Rider For 2020 Second Half Compared to First Half' SKIP 1 -

## > CENTER ============================================================================= SKIP 1

## SQL>

## SQL> SELECT a.\*, b.Total\_Orders\_2ndHalf, b.Late\_No\_2ndHalf, b.Total\_Sales\_2ndHalf,

## 2 RANK() OVER (ORDER BY Total\_Orders\_2ndHalf DESC) Rank\_2ndHalf

## 3 FROM top\_rider\_firstHalf a, top\_rider\_secondHalf b

## 4 WHERE a.rider\_id = b.rider\_id;

## Top 10 Most Order Delivered Rider For 2020 Second Half Compared to First Half

## =============================================================================

## Total Late Total Rank Total Late Total Rank

## Rider Rider Orders No Sales 1st Orders No Sales 2nd

## ID Name 1stHalf 1stHalf 1stHalf Half 2ndHalf 2ndHalf 2ndHalf Half

## ------ -------------------- ------- ------- ----------- ---- ------- ------- ----------- ----

## 10001 FLOSSIE MEGANY 238 0 67,040.48 7 268 0 94,719.52 1

## 10072 LIBBY SCHOFFEL 238 0 77,045.80 7 262 0 73,209.82 2

## 10075 CAREY ZEBEDEE 242 0 71,609.30 3 254 0 57,629.14 3

## 10043 VERNOR ZORER 238 0 81,333.46 7 247 0 67,358.60 4

## 10010 SISSIE DAKHNO 238 0 66,232.09 7 244 0 77,889.88 5

## 10029 CARTER CORNEY 241 0 64,084.06 4 234 0 66,560.06 6

## 10098 ZORA STREIGHT 240 0 75,101.78 6 232 0 58,357.82 7

## 10082 DALLAS CADOGAN 241 0 73,111.53 4 222 0 63,715.02 8

## 10092 SEE BLACKHURST 243 0 79,153.47 2 219 0 67,346.06 9

## 10003 TIMOTHEA DAEN 248 0 61,823.02 1 210 0 74,302.07 10

## 10 rows selected.

## 3.3 Tan Wei Siong

### 3.3.1 Top 3 total order restaurants in each time section within the selected city in 2020

This report will show the most popular restaurant in every time section within the selected city in 2020. The aim of this report is to analyze the most popular restaurant within the same city with the customer. Time section can help to inform the restaurant it is famous for breakfast, diner or other else. For instance, if a restaurant has the most ordered quantity for breakfast, the restaurant can be defined as the most popular restaurant for the breakfast section. ACME Food Delivery (AFD) can use the result of this report to suggest the restaurant as most people pick or most popular to their customers when it is in the same location with the customers. This could help the ADF to improve their income.

Script/coding

Break on time\_section skip 1 duplicates

TTITLE ON

TTITLE CENTER 'Top 3 total order restaurants in each time section within the selected city in 2020' SKIP 1-

CENTER ========================================================================== SKIP 2

COLUMN time\_section format a30

COLUMN rest\_name format a30

Set linesize 90

Set pagesize 120

SELECT time\_section,

rest\_name,

Total\_Order,

time\_section\_Order AS Rank

FROM(

SELECT C.time\_section,

RANK() OVER (Partition by

C.time\_section

Order by COUNT

(DISTINCT A.orderID) DESC) AS

time\_section\_Order,

COUNT(DISTINCT A.orderID) AS

Total\_Order,

B.rest\_branchID,

B.rest\_name

FROM Sales\_fact A, DIM\_restaurants B, DIM\_menu C

Where A.restaurant\_key = B.restaurant\_key and

B.rest\_city = UPPER('&city\_input') and

C.restaurant\_id = B.restaurantID and

Extract(year from A.order\_Date) = 2020

Group by B.rest\_branchID, B.rest\_name, C.time\_section

Order by time\_section\_Order, Total\_Order DESC

)

Where time\_section\_Order<=3

Order by time\_section, Total\_Order DESC;

Sample Output

SQL> SELECT time\_section,

2 rest\_name,

3 Total\_Order,

4 time\_section\_Order AS Rank

5 FROM(

6 SELECT C.time\_section,

7 RANK() OVER (Partition by

8 C.time\_section

9 Order by COUNT

10 (DISTINCT A.orderID) DESC) AS

11 time\_section\_Order,

12 COUNT(DISTINCT A.orderID) AS

13 Total\_Order,

14 B.rest\_branchID,

15 B.rest\_name

16 FROM Sales\_fact A, DIM\_restaurants B, DIM\_menu C

17 Where A.restaurant\_key = B.restaurant\_key and

18 B.rest\_city = UPPER('&city\_input') and

19 C.restaurant\_id = B.restaurantID and

20 Extract(year from A.order\_Date) = 2020

21 Group by B.rest\_branchID, B.rest\_name, C.time\_section

22 Order by time\_section\_Order, Total\_Order DESC

23 )

24 Where time\_section\_Order<=3

25 Order by time\_section, Total\_Order DESC;

Enter value for city\_input: kuala lumpur

old 18: B.rest\_city = UPPER('&city\_input') and

new 18: B.rest\_city = UPPER('kuala lumpur') and

Top 3 total order restaurants in each time section within the selected city in 2020

==========================================================================

TIME\_SECTION REST\_NAME TOTAL\_ORDER RANK

------------------------------ ------------------------------ ----------- ----------

ALL DAY R CAFE 103 1

ALL DAY BURGER KING 97 2

ALL DAY THE ALLEY 95 3

BREAKFAST R CAFE 103 1

BREAKFAST BURGER KING 97 2

BREAKFAST SUBWAY 94 3

DINNER NADU MALAXIANGGUO 88 1

DINNER FEI FAN STEAMBOAT 86 2

DINNER SANTOUKA RAMEN 81 3

LUNCH R CAFE 103 1

LUNCH GANGNAM 88 RESTAURANT 91 2

LUNCH MY ELEPHANT 90 3

LUNCH AND DINNER R CAFE 103 1

LUNCH AND DINNER THE ALLEY 95 2

LUNCH AND DINNER GANGNAM 88 RESTAURANT 91 3

LUNCH AND DINNER LITTLE HANOI 91 3

LUNCH AND DINNER MARRYBROWN 91 3

17 rows selected.

\

### 3.3.2 Menu order quantity less than 100 of the Top 3 least order restaurants in the selected city

The purpose of this report is to analyze the top 3 least ordered restaurants and  their menu order quantity which is less than 100. With this report, the ACME Food Delivery (AFD) can know which menu items of the restaurant are not famous for the customer with the same location. The AFD can issue the promotion code or discuss with the restaurant for an affordable price for these menu items to help to improve their sales.

### Script/coding

### set linesize 120

### set pagesize 100

### Column rest\_name format a35

### Column menu\_name format a40

### Break on restaurant\_ID skip 1 duplicates

### CREATE OR REPLACE view Top\_3\_Least\_Order\_Set\_view AS

### SELECT B.restaurant\_ID, SUM(C.quantity) AS Set\_Meals

### FROM DIM\_Restaurant A, DIM\_Menu B, Sales\_Fact C

### Where B.restaurant\_ID = A.restaurantID and

### B.category\_name ='SET MEAL' and

### C.menu\_key = B.menu\_key

### Group by B.restaurant\_ID, A.rest\_Name

### Order by B.restaurant\_ID;

### CREATE OR REPLACE view Top\_3\_Least\_Order\_ALA\_view AS

### SELECT B.restaurant\_ID, SUM(C.quantity) AS Ala\_Carte

### FROM DIM\_Restaurant A, DIM\_Menu B, Sales\_Fact C

### Where B.restaurant\_ID = A.restaurantID and

### B.category\_name ='ALA CARTE' and

### C.menu\_key = B.menu\_key

### Group by B.restaurant\_ID, A.rest\_Name

### Order by B.restaurant\_ID;

### CREATE OR REPLACE view Top\_3\_Least\_Order\_Rank\_view AS

### SELECT \*

### FROM(

### Select A.restaurant\_ID, A.rest\_Name,A.rest\_BranchID,

### RANK() OVER (ORDER BY A.Total\_Ordered) Ranking,

### A.Total\_Ordered, B.Set\_Meals, C.Ala\_Carte

### FROM Top\_3\_Least\_Order\_view A

### JOIN Top\_3\_Least\_Order\_Set\_view B

### ON B.restaurant\_ID = A.restaurant\_ID

### JOIN Top\_3\_Least\_Order\_ALA\_view C

### ON C.restaurant\_ID = A.restaurant\_ID

### Order by 4)

### Where ROWNUM <=3;

### TTITLE ON

### TTITLE CENTER 'Top 3 least order restaurant in the selected city' SKIP 1-

### CENTER ================================================= SKIP 2

### Select \* FROM Top\_3\_Least\_Order\_Rank\_view;

### TTITLE CENTER 'Menu order quantity less than 100 of the Top 3 least order restaurant in the selected city' SKIP 1-

### CENTER ========================================================================================== SKIP 2

### SELECT restaurant\_ID,rest\_Name,menu\_name,

### Meal\_Ordered, Rank

### FROM(

### SELECT A.restaurant\_ID, A.rest\_Name,

### A.rest\_BranchID,

### B.menu\_name,

### SUM(C.quantity) As Meal\_Ordered,

### RANK() OVER (Partition by A.restaurant\_ID

### Order by SUM(C.quantity) DESC)

### AS Rank

### FROM Top\_3\_Least\_Order\_Rank\_view A

### JOIN DIM\_menu B

### ON B.restaurant\_ID = A.restaurant\_ID

### JOIN Sales\_Fact C

### ON C.menu\_key = B.menu\_key

### Group by A.restaurant\_ID, A.rest\_Name,

### A.rest\_BranchID, B.menu\_name

### HAVING SUM(C.quantity)<=100

### Order by A.restaurant\_ID, Meal\_Ordered

### )

### Order by restaurant\_ID, Rank ASC;

### Sample output

### SQL> SELECT restaurant\_ID,rest\_Name,menu\_name,

### 2 Meal\_Ordered, Rank

### 3 FROM(

### 4 SELECT A.restaurant\_ID, A.rest\_Name,

### 5 A.rest\_BranchID,

### 6 B.menu\_name,

### 7 SUM(C.quantity) As Meal\_Ordered,

### 8 RANK() OVER (Partition by A.restaurant\_ID

### 9 Order by SUM(C.quantity) DESC)

### 10 AS Rank

### 11 FROM Top\_3\_Least\_Order\_Rank\_view A

### 12 JOIN DIM\_menu B

### 13 ON B.restaurant\_ID = A.restaurant\_ID

### 14 JOIN Sales\_Fact C

### 15 ON C.menu\_key = B.menu\_key

### 16 Group by A.restaurant\_ID, A.rest\_Name,

### 17 A.rest\_BranchID, B.menu\_name

### 18 HAVING SUM(C.quantity)<=100

### 19 Order by A.restaurant\_ID, Meal\_Ordered

### 20 )

### 21 Order by restaurant\_ID, Rank ASC;

### Menu order quantity less than 100 of the Top 3 least order restaurant in the selected city

### ==========================================================================================

### RESTAURANT\_ID REST\_NAME MENU\_NAME MEAL\_ORDERED RANK

### ------------- ----------------------------------- ---------------------------------------- ------------ ----------

### 10047 GOO NOODLE HOUSE AYAM TANDOORI NASI PUTIH 94 1

### 10047 GOO NOODLE HOUSE MEE GORENG 68 2

### 10047 GOO NOODLE HOUSE TANDOORI DOUBLE CHEESE NAAN SET 54 3

### 10047 GOO NOODLE HOUSE CHEEZY CHEESE NAAN SET 46 4

### 10047 GOO NOODLE HOUSE ONION NAAN SET 24 5

### 10047 GOO NOODLE HOUSE TANDOORI AYAM 9 6

### 10074 PIZZA HUT PAN-SEARED SALMON 86 1

### 10074 PIZZA HUT PAN-SEARED TUNA FISH 73 2

### 10101 TEXAS CHICKEN CHICKEN BITES RICE 71 1

### 10101 TEXAS CHICKEN FRENCH FRIES 3 2

### 10101 TEXAS CHICKEN CHICKEN BITES MEEHOON 2 3

### 11 rows selected.

### 3.3.3 Top 10 spending customers in 2019 and their spend in 2020

The purpose of this report is to show the top 10 spending customers in 2019 and their spend in 2020. The report can the ACME Food Delivery (AFD) to know the increase or decrease percentage for the customers spending from 2019 to 2020.

### Script/Coding

### Column Total\_Spend\_2019 format 99999.00;

### Column Total\_Discount\_2019 format 9999.00;

### Column Line\_Total\_2019 format 99999.00;

### Column AVG\_Spend\_2019 foramt 999.99;

### Column Total\_Spend\_2020 format 99999.00;

### Column Total\_Discount\_2020 format 9999.00;

### Column Line\_Total\_2020 format 99999.00;

### Column AVG\_Spend\_2020 format 999.99;

### Column user\_Name format a12;

### Column AVG\_Spend format 9999.00;

### Column Total\_Spend\_Perc format 999.00;

### Column user\_ID format 9999999;

### Column AVG\_RANK format 99;

### COLUMN Total\_Spend\_2019 HEADING 'Total|Spend|2019'

### COLUMN Total\_Spend\_2020 HEADING 'Total|Spend|2020'

### COLUMN AVG\_Rank HEADING 'AVG|Rank'

### COLUMN LINE\_TOTAL\_2019 HEADING 'LINE|TOTAL|2019'

### COLUMN LINE\_TOTAL\_2020 HEADING 'LINE|TOTAL|2020'

### COLUMN Total\_dist\_2019 HEADING 'Total|dist|2019'

### COLUMN Total\_dist\_2020 HEADING 'Total|dist|2020'

### COLUMN Total\_Spend\_Perc HEADING 'Total|Spend|Perc'

### COLUMN AVG\_SPEND\_2019 HEADING 'AVG|SPEND|2019'

### COLUMN AVG\_SPEND\_2020 HEADING 'AVG|SPEND|2020'

### set pagesize 100

### set linesize 125

### CREATE OR REPLACE view Top\_10\_Spend\_Cust2019\_view AS

### SELECT \*

### FROM(

### SELECT A.user\_ID, A.user\_Name,

### SUM(line\_total) AS Line\_Total\_2019,

### SUM(discount\_amount) AS Total\_Dist\_2019,

### SUM(line\_total) - SUM(discount\_amount) AS Total\_Spend\_2019,

### ROUND((SUM(line\_total) - SUM(discount\_amount))/COUNT(DISTINCT (B.OrderID)),2) AS AVG\_Spend\_2019

### FROM DIM\_Users A, Sales\_Fact B

### Where A.user\_key = B.user\_key and

### Extract(year from order\_Date) = 2019

### Group by A.user\_ID, A.user\_Name

### Order by Total\_Spend\_2019 DESC)

### Where ROWNUM <=10;

### CREATE OR REPLACE view Top\_10\_Spend\_Cust2020\_view AS

### SELECT A.user\_ID, A.user\_Name,

### SUM(line\_total) AS Line\_Total\_2020,

### SUM(discount\_amount) AS Total\_Dist\_2020,

### SUM(line\_total) - SUM(discount\_amount) AS Total\_Spend\_2020,

### ROUND((SUM(line\_total) - SUM(discount\_amount))/COUNT(DISTINCT (B.OrderID)),2) AS AVG\_Spend\_2020

### FROM DIM\_Users A, Sales\_Fact B

### Where A.user\_key = B.user\_key and

### Extract(year from order\_Date) = 2020

### Group by A.user\_ID, A.user\_Name

### Order by Total\_Spend\_2020 DESC;

### TTITLE CENTER 'Top 10 spending customers in 2019 and their spend in 2020' SKIP 1-

### CENTER ========================================================= SKIP 2

### Select A.user\_ID, A.user\_Name, A.Line\_Total\_2019,

### A.Total\_Dist\_2019, A.Total\_Spend\_2019,

### Rank() OVER (Order by A.AVG\_Spend\_2019 DESC) AS AVG\_Rank,

### A.AVG\_Spend\_2019,

### B.Line\_Total\_2020,

### B.Total\_Dist\_2020,

### B.Total\_Spend\_2020,

### Rank() OVER (Order by B.AVG\_Spend\_2020 DESC) AS AVG\_Rank,

### B.AVG\_Spend\_2020,

### Round((((B.Total\_Spend\_2020 -

### A.Total\_Spend\_2019)/

### A.Total\_Spend\_2019)\*100),2)

### AS Total\_Spend\_Perc

### FROM Top\_10\_Spend\_Cust2019\_view A

### LEFT JOIN Top\_10\_Spend\_Cust2020\_view B ON

### A.user\_ID = B.user\_ID

### Order by A.Total\_Spend\_2019 DESC;

### Sample output

### SQL> Select A.user\_ID, A.user\_Name, A.Line\_Total\_2019,

### 2 A.Total\_Dist\_2019, A.Total\_Spend\_2019,

### 3 Rank() OVER (Order by A.AVG\_Spend\_2019 DESC) AS AVG\_Rank,

### 4 A.AVG\_Spend\_2019,

### 5 B.Line\_Total\_2020,

### 6 B.Total\_Dist\_2020,

### 7 B.Total\_Spend\_2020,

### 8 Rank() OVER (Order by B.AVG\_Spend\_2020 DESC) AS AVG\_Rank,

### 9 B.AVG\_Spend\_2020,

### 10 Round((((B.Total\_Spend\_2020 -

### 11 A.Total\_Spend\_2019)/

### 12 A.Total\_Spend\_2019)\*100),2)

### 13 AS Total\_Spend\_Perc

### 14 FROM Top\_10\_Spend\_Cust2019\_view A

### 15 LEFT JOIN Top\_10\_Spend\_Cust2020\_view B ON

### 16 A.user\_ID = B.user\_ID

### 17 Order by A.Total\_Spend\_2019 DESC;

### Top 10 spending customers in 2019 and their spend in 2020

### =========================================================

### LINE Total Total AVG LINE Total Total AVG Total

### TOTAL dist Spend AVG SPEND TOTAL dist Spend AVG SPEND Spend

### USER\_ID USER\_NAME 2019 2019 2019 Rank 2019 2020 2020 2020 Rank 2020 Perc

### -------- ------------ --------- ---------- --------- ---- ---------- --------- ---------- --------- ---- ------- -------

### 10343 COB SANDYFOR 29553.92 1941.47 27612.45 2 563.52 16504.05 2235.62 14268.43 3 303.58 -48.33

### D10343

### 10456 MARIQUILLA G 30143.30 2553.28 27590.02 5 459.83 6250.46 787.36 5463.10 10 151.75 -80.20

### EERITS10456

### 10208 COB SANDYFOR 28431.00 2858.5 25572.50 1 568.28 13802.58 2028.72 11773.86 7 255.95 -53.96

### D

### 10816 MAUREEN BIRK 27716.98 2493.3 25223.68 6 458.61 14501.52 2727.47 11774.05 5 294.35 -53.32

### BECK10816

### 10729 MAMIE BARFFO 24976.66 1210.74 23765.92 4 485.02 33475.99 864.88 32611.11 1 582.34 37.22

### RD10729

### 10268 ESTEBAN HEBB 25828.94 2657.98 23170.96 7 399.5 13180.39 1204.94 11975.45 8 234.81 -48.32

### LE

### 10335 DEBRA WINSIO 25617.09 2624.13 22992.96 9 353.74 11992.16 1574.75 10417.41 9 217.03 -54.69

### WIECKI10335

### 10570 LEXINE LONGS 31401.59 8780.12 22621.47 8 390.03 20238.57 2045.37 18193.20 2 319.18 -19.58

### TAFF10570

### 10185 GRANT FAIN 24246.13 1903.54 22342.59 10 349.1 12762.32 1201.54 11560.78 6 281.97 -48.26

### 10589 ZECHARIAH SR 26874.95 4916.49 21958.46 3 499.06 16849.53 2216.05 14633.48 4 298.64 -33.36

### EENAN10589

### 10 rows selected.

## 3.4 Nigel Lee Jian Hsee

### 3.4.1 Rank Total Sales of Each State based on each Meal Section

### The purpose of this report will rank each time section sales based on user state in the year of 2020. Total sales rank doesn’t mean the sales rank based on user state and time meal section is the same. For instance, the reports show that Sarawak was the third highest in total sales but only ranked fifth in Lunch Section Meal, and fourth in Dinner. This report allows ACME Food Delivery (AFD) to know on which meal time section has better sales and may focus to improve lower sales state on different time meal sections or find out on which time section meal in specific state is more popular. AFD is able to improve lower time section sales by having promo code that might only be able to be used during that time section.

### Script/coding

### set linesize 125

### set pagesize 120

### Create or Replace view T\_Breakfast2020 AS

### Select U.state as state,

### sum(S.line\_total) AS Total\_Breakfast,

### ROW\_NUMBER() OVER(Order By sum(S.line\_total) DESC)

### Ranks

### From sales\_fact S,dim\_menu M,dim\_users U

### Where S.menu\_key = M.menu\_key AND

### M.time\_section = 'BREAKFAST'

### AND U.user\_key=S.user\_key

### AND extract(year from S.order\_date) = 2020

### Group by u.state;

### Create or Replace view T\_LUNCH2020 AS

### Select U.state as state,

### sum(S.line\_total) AS Total\_LUNCH,

### ROW\_NUMBER() OVER(Order By sum(S.line\_total) DESC)

### Ranks

### From sales\_fact S,dim\_menu M,dim\_users U

### Where S.menu\_key = M.menu\_key

### AND M.time\_section = 'LUNCH'

### AND U.user\_key=S.user\_key

### AND extract(year from S.order\_date) = 2020

### Group by U.state;

### Create or Replace view T\_DINNER2020 AS

### Select U.state as state,

### sum(S.line\_total) AS Total\_DINNER,

### ROW\_NUMBER() OVER(Order By sum(S.line\_total) DESC)

### Ranks

### From sales\_fact S,dim\_menu M,dim\_users U

### Where S.menu\_key = M.menu\_key

### AND M.time\_section = 'DINNER'

### AND U.user\_key=S.user\_key

### AND extract(year from S.order\_date) = 2020

### Group by U.state;

### Create or Replace view T\_LD2020 AS

### Select U.state as state,

### sum(S.line\_total) AS Total\_LUNCH\_DINNER,

### ROW\_NUMBER() OVER(Order By sum(S.line\_total) DESC)

### Ranks

### From sales\_fact S,dim\_menu M,dim\_users U

### Where S.menu\_key = M.menu\_key

### AND M.time\_section = 'LUNCH AND DINNER'

### AND U.user\_key=S.user\_key

### AND extract(year from S.order\_date) = 2020

### Group by U.state;

### Create or Replace view T\_ALLDAY2020 AS

### Select U.state as state,

### sum(S.line\_total) AS Total\_ALL\_DAY,

### ROW\_NUMBER() OVER(Order By sum(S.line\_total) DESC)

### Ranks

### From sales\_fact S,dim\_menu M,dim\_users U

### Where S.menu\_key = M.menu\_key

### AND M.time\_section = 'ALL DAY'

### AND U.user\_key=s.user\_key

### AND extract(year from S.order\_date) = 2020

### Group by U.state;

### column ranks format 99

### column state format a20

### column Total\_Breakfast format 9999999.99

### column Total\_Lunch format 9999999.99

### column Total\_Dinner format 9999999.99

### column Total\_LUNCH\_DINNER format 9999999.99

### column Total\_All\_Day format 9999999.99

### column Total\_Breakfast format 9999999.99

### column Total format 9999999.99

### column Total\_Breakfast HEADING 'Total|Breakfast'

### column Total\_Lunch HEADING 'Total|Lunch'

### column Total\_Dinner HEADING 'Total|Dinner'

### column Total\_LUNCH\_DINNER HEADING 'Total|Lunch|Dinner'

### column Total\_ALL\_DAY HEADING 'Total|All|Day'

### TTITLE CENTER 'Rank of Each Time Meal Section Sales on each State'SKIP 1 -

### CENTER ======================================================================== SKIP

### select B.state,Total\_Breakfast,B.ranks,

### Total\_Lunch,L.ranks,

### Total\_Dinner,D.ranks,

### Total\_LUNCH\_DINNER,LD.ranks,

### Total\_ALL\_DAY,A.ranks,

### (Total\_Breakfast+Total\_Lunch+Total\_Dinner+Total\_LUNCH\_DINNER+Total\_ALL\_DAY) AS TOTAL

### FROM T\_Breakfast2020 B

### join T\_LUNCH2020 L

### on B.state = L.state

### join T\_DINNER2020 D

### on B.state = D.state

### join T\_LD2020 LD

### on B.state = LD.state

### join T\_ALLDAY2020 A

### on B.state = A.state

### order by total desc;

Sample output

set linesize 125

SQL> set pagesize 120

SQL>

SQL> Create or Replace view T\_Breakfast2020 AS

2 Select U.state as state,

3 sum(S.line\_total) AS Total\_Breakfast,

4 ROW\_NUMBER() OVER(Order By sum(S.line\_total) DESC)

5 Ranks

6 From sales\_fact S,dim\_menu M,dim\_users U

7 Where S.menu\_key = M.menu\_key AND

8 M.time\_section = 'BREAKFAST'

9 AND U.user\_key=S.user\_key

10 AND extract(year from S.order\_date) = 2020

11 Group by u.state;

View created.

SQL>

SQL> Create or Replace view T\_LUNCH2020 AS

2 Select U.state as state,

3 sum(S.line\_total) AS Total\_LUNCH,

4 ROW\_NUMBER() OVER(Order By sum(S.line\_total) DESC)

5 Ranks

6 From sales\_fact S,dim\_menu M,dim\_users U

7 Where S.menu\_key = M.menu\_key

8 AND M.time\_section = 'LUNCH'

9 AND U.user\_key=S.user\_key

10 AND extract(year from S.order\_date) = 2020

11 Group by U.state;

View created.

SQL>

SQL> Create or Replace view T\_DINNER2020 AS

2 Select U.state as state,

3 sum(S.line\_total) AS Total\_DINNER,

4 ROW\_NUMBER() OVER(Order By sum(S.line\_total) DESC)

5 Ranks

6 From sales\_fact S,dim\_menu M,dim\_users U

7 Where S.menu\_key = M.menu\_key

8 AND M.time\_section = 'DINNER'

9 AND U.user\_key=S.user\_key

10 AND extract(year from S.order\_date) = 2020

11 Group by U.state;

View created.

SQL>

SQL> Create or Replace view T\_LD2020 AS

2 Select U.state as state,

3 sum(S.line\_total) AS Total\_LUNCH\_DINNER,

4 ROW\_NUMBER() OVER(Order By sum(S.line\_total) DESC)

5 Ranks

6 From sales\_fact S,dim\_menu M,dim\_users U

7 Where S.menu\_key = M.menu\_key

8 AND M.time\_section = 'LUNCH AND DINNER'

9 AND U.user\_key=S.user\_key

10 AND extract(year from S.order\_date) = 2020

11 Group by U.state;

View created.

SQL>

SQL> Create or Replace view T\_ALLDAY2020 AS

2 Select U.state as state,

3 sum(S.line\_total) AS Total\_ALL\_DAY,

4 ROW\_NUMBER() OVER(Order By sum(S.line\_total) DESC)

5 Ranks

6 From sales\_fact S,dim\_menu M,dim\_users U

7 Where S.menu\_key = M.menu\_key

8 AND M.time\_section = 'ALL DAY'

9 AND U.user\_key=s.user\_key

10 AND extract(year from S.order\_date) = 2020

11 Group by U.state;

View created.

SQL>

SQL> column ranks format 99

SQL> column state format a20

SQL> column Total\_Breakfast format 9999999.99

SQL> column Total\_Lunch format 9999999.99

SQL> column Total\_Dinner format 9999999.99

SQL> column Total\_LUNCH\_DINNER format 9999999.99

SQL> column Total\_All\_Day format 9999999.99

SQL> column Total\_Breakfast format 9999999.99

SQL> column Total format 9999999.99

SQL> column Total\_Breakfast HEADING 'Total|Breakfast'

SQL> column Total\_Lunch HEADING 'Total|Lunch'

SQL> column Total\_Dinner HEADING 'Total|Dinner'

SQL> column Total\_LUNCH\_DINNER HEADING 'Total|Lunch|Dinner'

SQL> column Total\_ALL\_DAY HEADING 'Total|All|Day'

SQL>

SQL>

SQL>

SQL> TTITLE CENTER 'Rank of Each Time Meal Section Sales on each State'SKIP 1 -

> CENTER ======================================================================== SKIP

SQL> select B.state,Total\_Breakfast,B.ranks,

2 Total\_Lunch,L.ranks,

3 Total\_Dinner,D.ranks,

4 Total\_LUNCH\_DINNER,LD.ranks,

5 Total\_ALL\_DAY,A.ranks,

6 (Total\_Breakfast+Total\_Lunch+Total\_Dinner+Total\_LUNCH\_DINNER+Total\_ALL\_DAY) AS TOTAL

7 FROM T\_Breakfast2020 B

8 join T\_LUNCH2020 L

9 on B.state = L.state

10 join T\_DINNER2020 D

11 on B.state = D.state

12 join T\_LD2020 LD

13 on B.state = LD.state

14 join T\_ALLDAY2020 A

15 on B.state = A.state

16 order by total desc;

Rank of Each Time Meal Section Sales on each State

========================================================================

Total Total

Total Total Total Lunch All

STATE Breakfast RANKS Lunch RANKS Dinner RANKS Dinner RANKS Day RANKS TOTAL

-------------------- ----------- ----- ----------- ----- ----------- ----- ----------- ----- ----------- ----- -----------

SELANGOR 206131.47 1 93462.78 1 59323.71 1 863652.34 1 2339541.02 1 3562111.32

WILAYAH PERSEKUTUAN 120795.63 2 57117.90 2 36035.67 2 520394.81 2 1333422.68 2 2067766.69

SARAWAK 77206.60 3 23438.13 5 17803.91 4 298060.02 3 795700.52 3 1212209.18

PULAU PINANG 52270.28 4 21918.56 7 20248.43 3 261908.98 4 742425.11 4 1098771.36

PAHANG 51779.29 5 22543.34 6 13742.60 5 243502.60 5 654169.01 5 985736.84

JOHOR 48048.95 7 29909.99 3 13275.59 7 229314.73 6 634933.25 6 955482.51

PERAK 51429.51 6 25505.42 4 13563.44 6 194421.75 7 512434.03 7 797354.15

NEGERI SEMBILAN 35490.74 8 21265.77 8 8267.84 8 158594.52 8 389614.83 8 613233.70

SABAH 14840.92 11 13844.44 9 7067.58 9 97580.59 9 261357.50 9 394691.03

KEDAH 19645.74 9 10445.08 10 5329.44 10 87510.11 10 249726.23 10 372656.60

KELANTAN 15292.60 10 9365.17 11 5263.52 11 64211.53 11 171996.39 11 266129.21

PERLIS 9186.03 12 4142.45 13 1713.95 12 56974.50 12 137582.73 12 209599.66

MELAKA 7847.30 13 5277.83 12 563.07 13 47841.02 13 136362.54 13 197891.76

13 rows selected.

### 3.4.2 Top 3 Branch for each restaurant type based on the input of state and year

### The purpose of this report is to find out the top 3 branches for each type of restaurant based on the state and year of report user’s input. Example below show the result of the top three branches of each restaurant type in the state of Wilayah Persekutuan. Detail of the restaurant such as name, branch id total order received, quantity of item ordered, the sales and average spending per order. ACME Food Delivery (AFD) able to recommend the users top restaurant based on different restaurant type to the report user.

### Script/coding

### set linesize 90

### set pagesize 120

### break on rest\_typename skip 1 duplicates

### TTITLE CENTER 'TOP 3 For Each Restaurant Branch Based on Input Year and State'SKIP 1 -

### CENTER ======================================================================== SKIP 1

### column TotalSales format 9999999.99

### column AveragePerOrder format 9999999.99

### column ranks format 99

### column rest\_name format a25

### column rest\_branchid format 99999

### column TotalQty format 99999

### column Total\_Order format 9999

### column rest\_typename format a10

### column AveragePerOrder HEADING 'Average| Per |Order '

### column TotalQty HEADING 'Total|QTY'

### column Total\_Order HEADING 'Total|Order'

### column rest\_branchid HEADING 'Branch|ID'

### column rest\_typename HEADING 'Resturant|Type'

### SELECT rest\_typename,

### rest\_name,

### rest\_branchid,

### Total\_Order,

### TotalQty,

### Total\_Sales,ranks,

### AveragePerOrder

### FROM

### ( SELECT rest\_typename,

### rest\_name, rest\_branchid,

### sum(quantity) as TotalQTY,

### sum(line\_total) as Total\_Sales,

### count(distinct orderid) as Total\_Order,

### ROW\_NUMBER() OVER (PARTITION BY rest\_typename

### ORDER BY sum(s.line\_total)

### DESC)AS RANKS,

### ROUND(sum(line\_total)/count(distinct orderid),2)

### AS AveragePerOrder

### FROM sales\_fact S, dim\_restaurants R

### Where s.restaurant\_key = r.restaurant\_key

### AND extract(year from s.order\_date) = &year\_input

### AND rest\_state = UPPER('&state\_input')

### Group by r.rest\_typename,R.rest\_branchid,rest\_name

### )

### WHERE RANKS <= 3

## ORDER BY rest\_typename,Total\_Sales DESC;

## Sample output

## SQL> set linesize 90

## SQL> set pagesize 120

## SQL> break on rest\_typename skip 1 duplicates

## SQL>

## SQL> TTITLE CENTER 'TOP 3 For Each Restaurant Branch Based on Input Year and State'SKIP 1 -

## > CENTER ======================================================================== SKIP 1

## SQL> column TotalSales format 9999999.99

## SQL> column AveragePerOrder format 9999999.99

## SQL> column ranks format 99

## SQL> column rest\_name format a25

## SQL> column rest\_branchid format 99999

## SQL> column TotalQty format 99999

## SQL> column Total\_Order format 9999

## SQL> column rest\_typename format a10

## SQL> column AveragePerOrder HEADING 'Average| Per |Order '

## SQL> column TotalQty HEADING 'Total|QTY'

## SQL> column Total\_Order HEADING 'Total|Order'

## SQL> column rest\_branchid HEADING 'Branch|ID'

## SQL> column rest\_typename HEADING 'Resturant|Type'

## SQL>

## SQL> SELECT rest\_typename,

## 2 rest\_name,

## 3 rest\_branchid,

## 4 Total\_Order,

## 5 TotalQty,

## 6 Total\_Sales,ranks,

## 7 AveragePerOrder

## 8 FROM

## 9 ( SELECT rest\_typename,

## 10 rest\_name, rest\_branchid,

## 11 sum(quantity) as TotalQTY,

## 12 sum(line\_total) as Total\_Sales,

## 13 count(distinct orderid) as Total\_Order,

## 14 ROW\_NUMBER() OVER (PARTITION BY rest\_typename

## 15 ORDER BY sum(s.line\_total)

## 16 DESC)AS RANKS,

## 17 ROUND(sum(line\_total)/count(distinct orderid),2)

## 18 AS AveragePerOrder

## 19 FROM sales\_fact S, dim\_restaurants R

## 20 Where s.restaurant\_key = r.restaurant\_key

## 21 AND extract(year from s.order\_date) = &year\_input

## 22 AND rest\_state = UPPER('&state\_input')

## 23 Group by r.rest\_typename,R.rest\_branchid,rest\_name

## 24 )

## 25 WHERE RANKS <= 3

## 26 ORDER BY rest\_typename,Total\_Sales DESC;

## Enter value for year\_input: 2020

## old 21: AND extract(year from s.order\_date) = &year\_input

## new 21: AND extract(year from s.order\_date) = 2020

## Enter value for state\_input: Wilayah Persekutuan

## old 22: AND rest\_state = UPPER('&state\_input')

## new 22: AND rest\_state = UPPER('Wilayah Persekutuan')

## TOP 3 For Each Restaurant Branch Based on Input Year and State

## ========================================================================

## Average

## Resturant Branch Total Total Per

## Type REST\_NAME ID Order QTY TOTAL\_SALES RANKS Order

## ---------- ------------------------- ------ ----- ------ ----------- ----- -----------

## ARAB ARABELLA RESTAURANT 10386 90 1352 38946.00 1 432.73

## ARAB ARABELLA RESTAURANT 10387 70 884 25140.00 2 359.14

## ARAB SABA RESTAURANT 10003 83 1047 19831.00 3 238.93

## CAFE WHEELER'S CAFE 10273 72 2158 43149.20 1 599.29

## CAFE 3 LEG CAT 10277 98 2235 33174.00 2 338.51

## CAFE CHINA HOUSE 10159 77 1629 26655.70 3 346.18

## CHINESE FEI FAN STEAMBOAT 10310 72 2123 121621.75 1 1689.19

## CHINESE FEI FAN STEAMBOAT 10311 76 2088 114308.04 2 1504.05

## CHINESE MING GARDEN 10251 85 1381 88965.00 3 1046.65

## FAST FOOD 4FINGERS CRISPY CHICKEN 10343 89 1310 64515.32 1 724.89

## FAST FOOD 4FINGERS CRISPY CHICKEN 10341 83 1045 49761.40 2 599.53

## FAST FOOD 4FINGERS CRISPY CHICKEN 10340 64 892 42615.64 3 665.87

## JAPANESE NAUGHTY NURI'S 10528 79 1072 162957.00 1 2062.75

## JAPANESE IKETERU RESTAURANT 10396 83 1240 72266.00 2 870.67

## JAPANESE ZIPANGU 10410 95 1341 69816.00 3 734.91

## KOREAN KOBQ 10078 92 1210 47070.00 1 511.63

## KOREAN SAE MA EUL KOREAN BBQ 10225 94 1363 41863.20 2 445.35

## KOREAN NANDA CHICKEN 10409 89 1241 40996.00 3 460.63

## MAMAK IMPIAN MAJU MAMAK 10315 112 1584 89141.11 1 795.90

## MAMAK IMPIAN MAJU MAMAK 10317 93 1362 76603.72 2 823.70

## MAMAK IMPIAN MAJU MAMAK 10316 92 1377 75000.86 3 815.23

## THAILAND MY ELEPHANT 10026 94 1284 24906.00 1 264.96

## THAILAND MY ELEPHANT 10025 82 1164 19618.50 2 239.25

## THAILAND SUPPER 10481 86 1395 17762.30 3 206.54

## VIETNAMESE PHO STREET 10379 85 1175 19924.43 1 234.41

## VIETNAMESE SAIGON HOUSE CUISINE 10064 70 981 19388.00 2 276.97

## VIETNAMESE SAIGON BOWL 10193 80 1004 15851.30 3 198.14

## WESTERN FARQUHAR MANSION FINE DIN 10486 86 1202 71666.00 1 833.33

## ING & LOUNGE PENANG

## WESTERN THIRTYTWO AT THE MANSION 10470 82 1117 47804.00 2 582.98

## WESTERN LK WESTERN 10043 97 1279 40589.80 3 418.45

## 30 rows selected.

### 3.4.3 Compared Top 10 City sales percentage with Top City

### This report will compare the top restaurant sales based on the restaurant located city sales in 2020. The results will show in food quantity ordered, sales without discount and sales with discount. The last column is the total sales with discount of the city compared with the top city and shows in percentage. This allows ACME Food Delivery (AFD) to assign more riders to that city for delivery service and lesser riders to lower sales city to avoid late delivery and reduce the waiting for riders to wait for the next order. More riders will help to improve the delivery time and avoid dissatisfaction of customers.

### Script/coding

### set linesize 80

### COLUMN PERCENTAGE\_VS\_TOPCITY HEADING 'Percentage| VS |TopCity'

### COLUMN RANKQuantity HEADING 'Rank|Quantity'COLUMN RANKSales HEADING 'Rank|Sales|(Discount)'

### COLUMN Total\_Actual\_Sales\_Discount HEADING 'Sales|(Discount)'

### COLUMN Total\_Sales FORMAT 9999999.99

### COLUMN Total\_Actual\_Sales\_Discoun FORMAT 9999999.99

### Create or Replace view Top\_City\_View\_2020 as

### Select \* from

### (Select rest\_city,

### sum(quantity) Top\_Qty,

### sum(line\_total) Top\_Sales,

### sum(actual\_amount\_paid) Top\_Actual\_Sales

### From sales\_fact s, dim\_restaurants r, dim\_date d

### Where s.restaurant\_key = r.restaurant\_key

### AND d.date\_key = s.date\_key

### AND extract(year from order\_date) = 2020

### Group by rest\_city

### Order by 4 DESC)

### Where rownum<=1;

### Create or Replace view Top10\_City\_View\_2020 as

### Select \* from

### (Select rest\_city,

### sum(s.quantity) Total\_Qty,

### sum(s.line\_total) Total\_Sales,

### sum(s.actual\_amount\_paid) Total\_Actual\_Sales\_Discount

### From sales\_fact s, dim\_restaurants r, dim\_date d

### Where s.restaurant\_key = r.restaurant\_key

### AND d.date\_key = s.date\_key

### AND extract(year from order\_date) = 2020

### Group by rest\_city

### Order by 4 DESC)

### Where rownum<=10;

### column Percentage\_vs\_TopCity format 990.00

### TTITLE CENTER 'VS Top City Sales with Top 10 City in Percentage'SKIP 1 -

### CENTER ===================================================== SKIP 1

### Select A.rest\_city,

### A.Total\_Qty,

### A.Total\_Sales,

### A.Total\_Actual\_Sales\_Discount,

### ROUND(A.Total\_Actual\_Sales\_Discount/B.Top\_Actual\_Sales\*100,2) Percentage\_vs\_TopCity

### from Top10\_City\_View\_2020 A, Top\_City\_View\_2020 B;

## Sample output

## SQL> set linesize 80

## SQL> COLUMN PERCENTAGE\_VS\_TOPCITY HEADING 'Percentage| VS |TopCity'

## SQL> COLUMN RANKQuantity HEADING 'Rank|Quantity'COLUMN RANKSales HEADING 'Rank|Sales|(Discount)'

## SP2-0158: unknown COLUMN option "COLUMN"

## SQL> COLUMN Total\_Actual\_Sales\_Discount HEADING 'Sales|(Discount)'

## SQL> COLUMN Total\_Sales FORMAT 9999999.99

## SQL> COLUMN Total\_Actual\_Sales\_Discoun FORMAT 9999999.99

## SQL>

## SQL>

## SQL> Create or Replace view Top\_City\_View\_2020 as

## 2 Select \* from

## 3 (Select rest\_city,

## 4 sum(quantity) Top\_Qty,

## 5 sum(line\_total) Top\_Sales,

## 6 sum(actual\_amount\_paid) Top\_Actual\_Sales

## 7 From sales\_fact s, dim\_restaurants r, dim\_date d

## 8 Where s.restaurant\_key = r.restaurant\_key

## 9 AND d.date\_key = s.date\_key

## 10 AND extract(year from order\_date) = 2020

## 11 Group by rest\_city

## 12 Order by 4 DESC)

## 13 Where rownum<=1;

## View created.

## SQL>

## SQL>

## SQL> Create or Replace view Top10\_City\_View\_2020 as

## 2 Select \* from

## 3 (Select rest\_city,

## 4 sum(s.quantity) Total\_Qty,

## 5 sum(s.line\_total) Total\_Sales,

## 6 sum(s.actual\_amount\_paid) Total\_Actual\_Sales\_Discount

## 7 From sales\_fact s, dim\_restaurants r, dim\_date d

## 8 Where s.restaurant\_key = r.restaurant\_key

## 9 AND d.date\_key = s.date\_key

## 10 AND extract(year from order\_date) = 2020

## 11 Group by rest\_city

## 12 Order by 4 DESC)

## 13 Where rownum<=10;

## View created.

## SQL>

## SQL> column Percentage\_vs\_TopCity format 990.00

## SQL>

## SQL> TTITLE CENTER 'VS Top City Sales with Top 10 City in Percentage'SKIP 1 -

## > CENTER ===================================================== SKIP 1

## SQL> Select A.rest\_city,

## 2 A.Total\_Qty,

## 3 A.Total\_Sales,

## 4 A.Total\_Actual\_Sales\_Discount,

## 5 ROUND(A.Total\_Actual\_Sales\_Discount/B.Top\_Actual\_Sales\*100,2) Percentage\_vs\_TopCity

## 6 from Top10\_City\_View\_2020 A, Top\_City\_View\_2020 B;

## VS Top City Sales with Top 10 City in Percentage

## =====================================================

## Percentage

## Sales VS

## REST\_CITY TOTAL\_QTY TOTAL\_SALES (Discount) TopCity

## ------------------------------ ---------- ----------- ---------- ----------

## KUALA LUMPUR 192749 4056706.61 3542444.93 100.00

## PETALING JAYA 90173 1780637.61 1557761.24 43.97

## JOHOR BAHRU 74977 1421749.26 1246426.91 35.19

## KLANG 61170 1019339.40 886821.47 25.03

## KUCHING 40864 954367.66 840940.86 23.74

## IPOH 49093 903993.89 786529.86 22.20

## SEREMBAN 15622 332678.04 287908.36 8.13

## SERI KEMBANGAN 8565 270428.43 237153.78 6.69

## SHAH ALAM 9312 164747.15 143609.21 4.05

## KAJANG 8677 159725.61 139424.56 3.94

## 10 rows selected.