SCHOOL OF COMPUTER SCIENCES

UNIVERSITI SAINS MALAYSIA

CMT221/CMM222: Database Organization and Design

Semester 1, Academic Session: 2021/2022

System Implementation

Group 16

Case Study 26: Delish Enterprise Canteen Management Database System

Name	Matric No.	USM Email	Module	Role	Core/Minor
Shindujaah Jaya Kumar	152818	shindujaahj@student.usm.my	Canteen and Stalls	Leader	Core
Thineshkumar Saravanan	152771	thineshsaravanan@student.usm.my	Stocks	Member	Core
Ilanthtamil Jayasangar	154831	ilanthtamil@student.usm.my	Employees	Member	Core
Kavindhren Visvanthan	153694	kavindhren@student.usm.my	Customers	Member	Core

Date of Submission

6 February 2022

1.0 Business Rules and Partial ERDs

[Present the updated business rules by also taking into account of the requirements given in Section 2.0 and partial ERDs for each module. Please highlight modifications that you have made. Include the business rule for Section 2.0. Note that some of your business rules and partial ERDs may change because of the requirements in Section 2.0.]

Module 1: Canteen and Stalls – Shindujaah Jaya Kumar

- One canteen manages one or many stalls. One stall operates in only one canteen.
- Each stall offers many menu items. One menu item is available in only one stall.
- Each menu item contains many ingredients. One ingredient can be found in many menu items.
- Each menu item is classified into one cuisine type. One cuisine type is a class of one menu item. [Section 2.0]

Module 2: Stocks – Thineshkumar Saravanan

- One canteen manages many storerooms. One storeroom is managed by one canteen.
- One storeroom keeps many stocks. One stock is kept in many storerooms.
- One supplier supplies many stocks. One stock is supplied by only one supplier.
- One stock is stored in many shelves. One shelf stores only one stock item.
- One storeroom contains many shelves. One shelf is contained in only one storeroom.

Module 3: Employees – Ilanthtamil Jayasangar

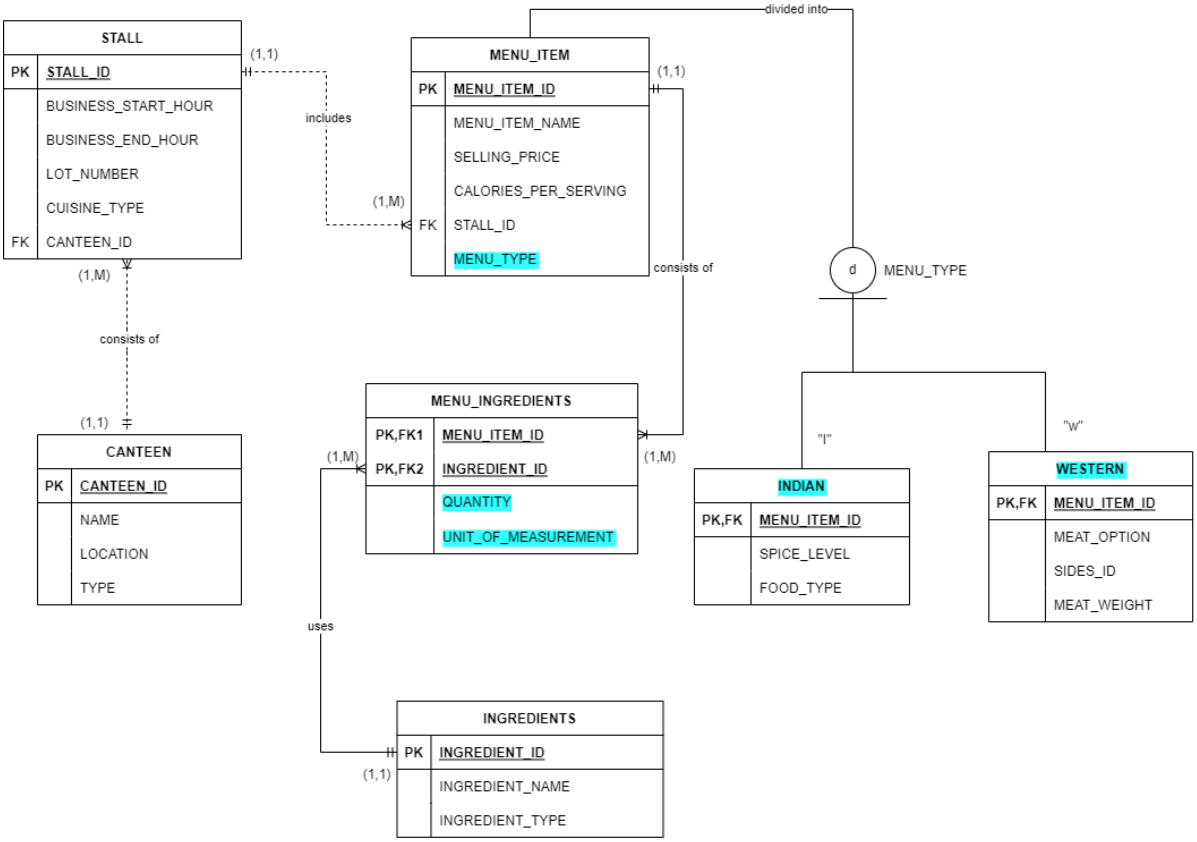
- Each staff hired at one canteen. Each canteen hires one or many staff.
- Each staff has been appointed to one or many positions. Each position is appointed to one or many staffs.
- Each staff given one or many work records. Each work record is given to one or many staff.
- Each canteen is managed by one or many staff. Each staff manages one canteen.

Module 4: Orders – Kavindhren Visvanathan

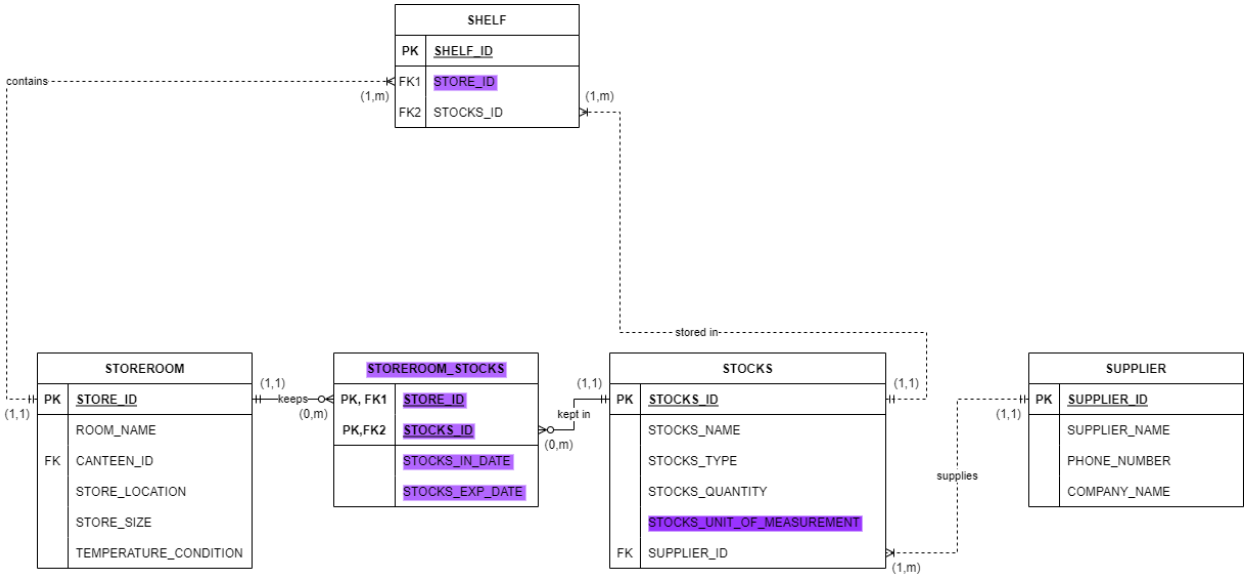
- A customer account can add many top-ups. A top-up credit can be added to one customer account.
- A customer account can be used to pay many orders. An order can be paid by one customer account.
- An order can consist of many ordered items. An ordered item can be on many orders.
- An ordered item can be in one menu item. A menu item can be in many ordered items.

Partial ERD:

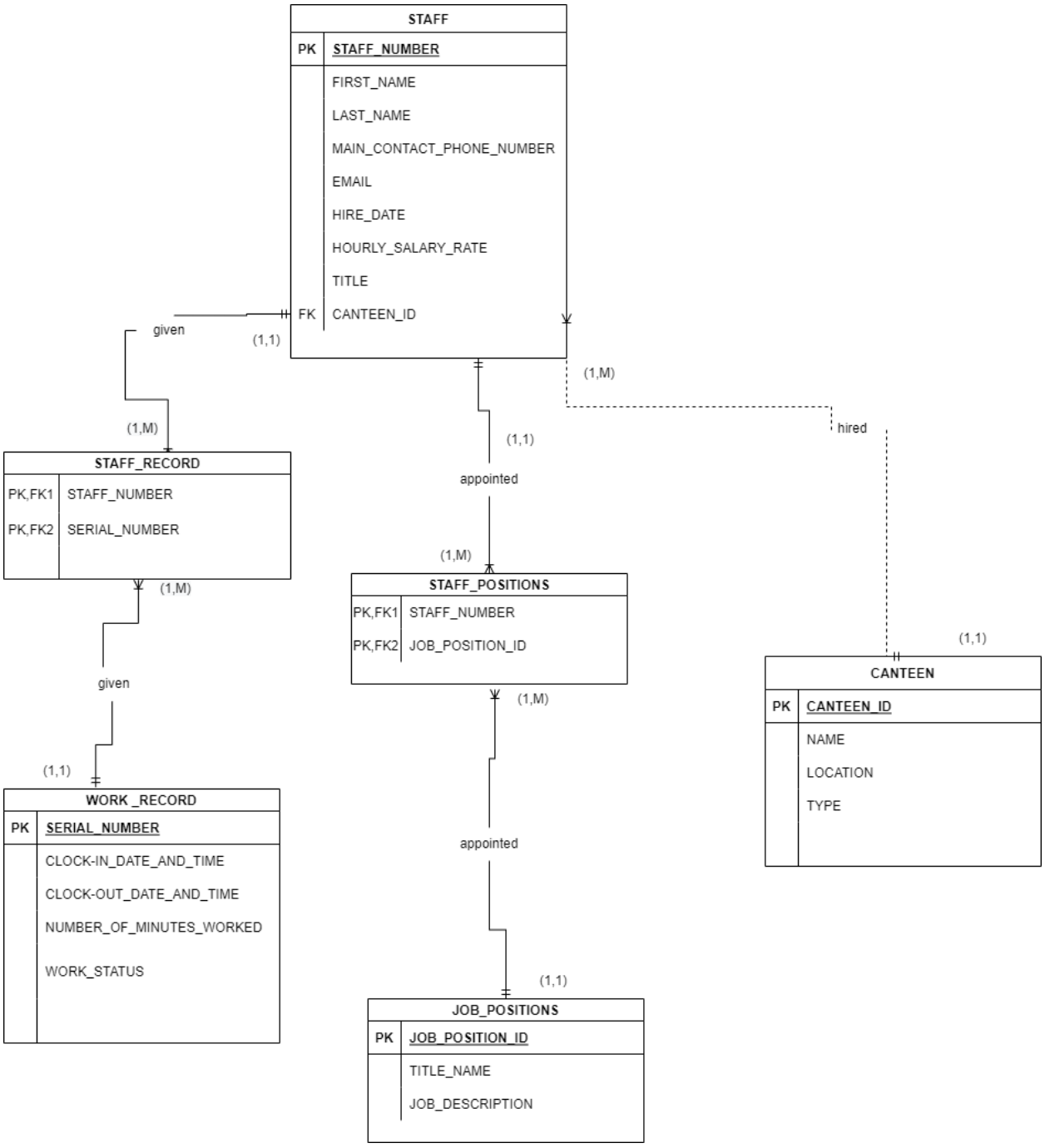
Module 1: Canteen and Stalls



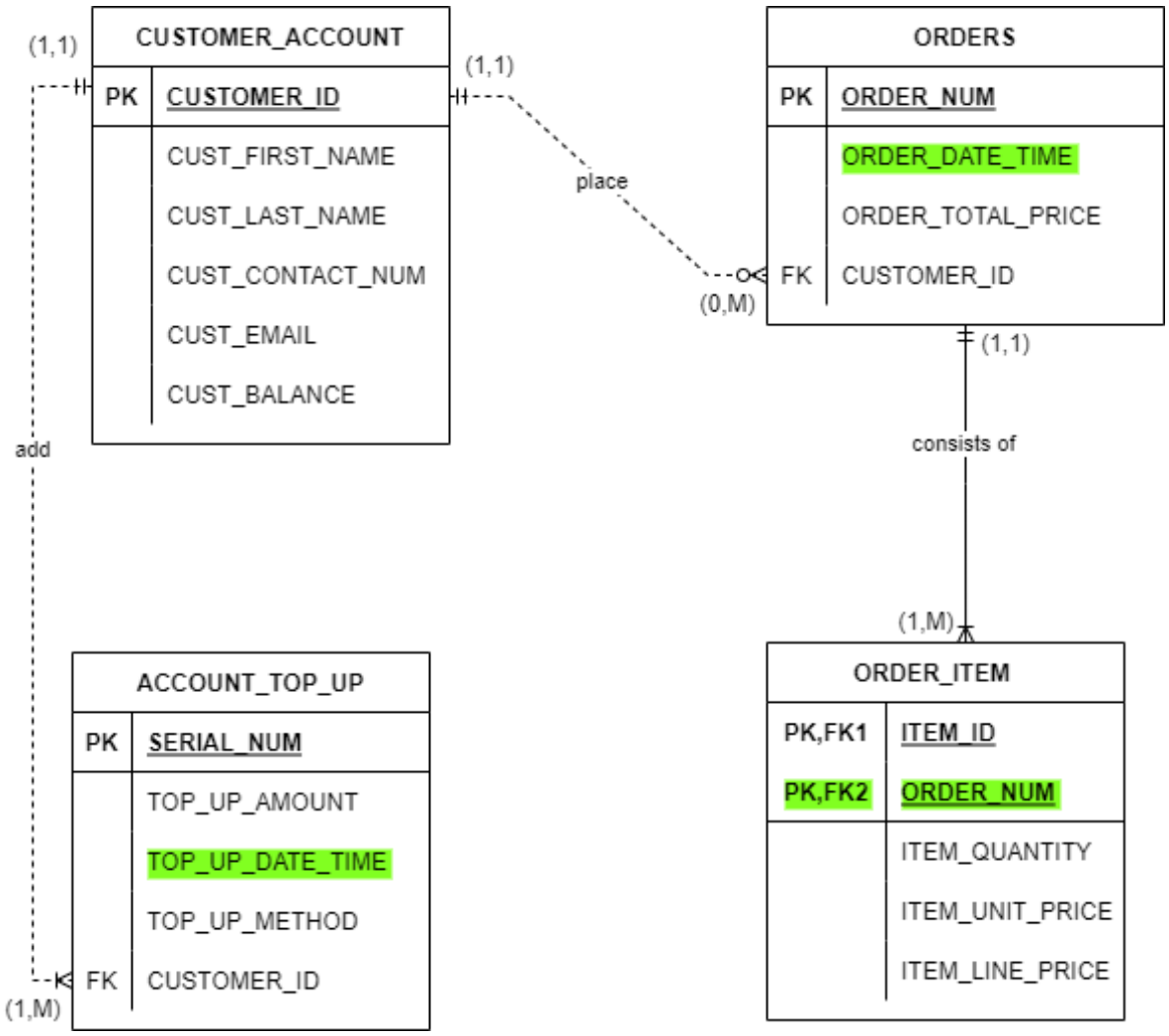
Module 2: Stocks



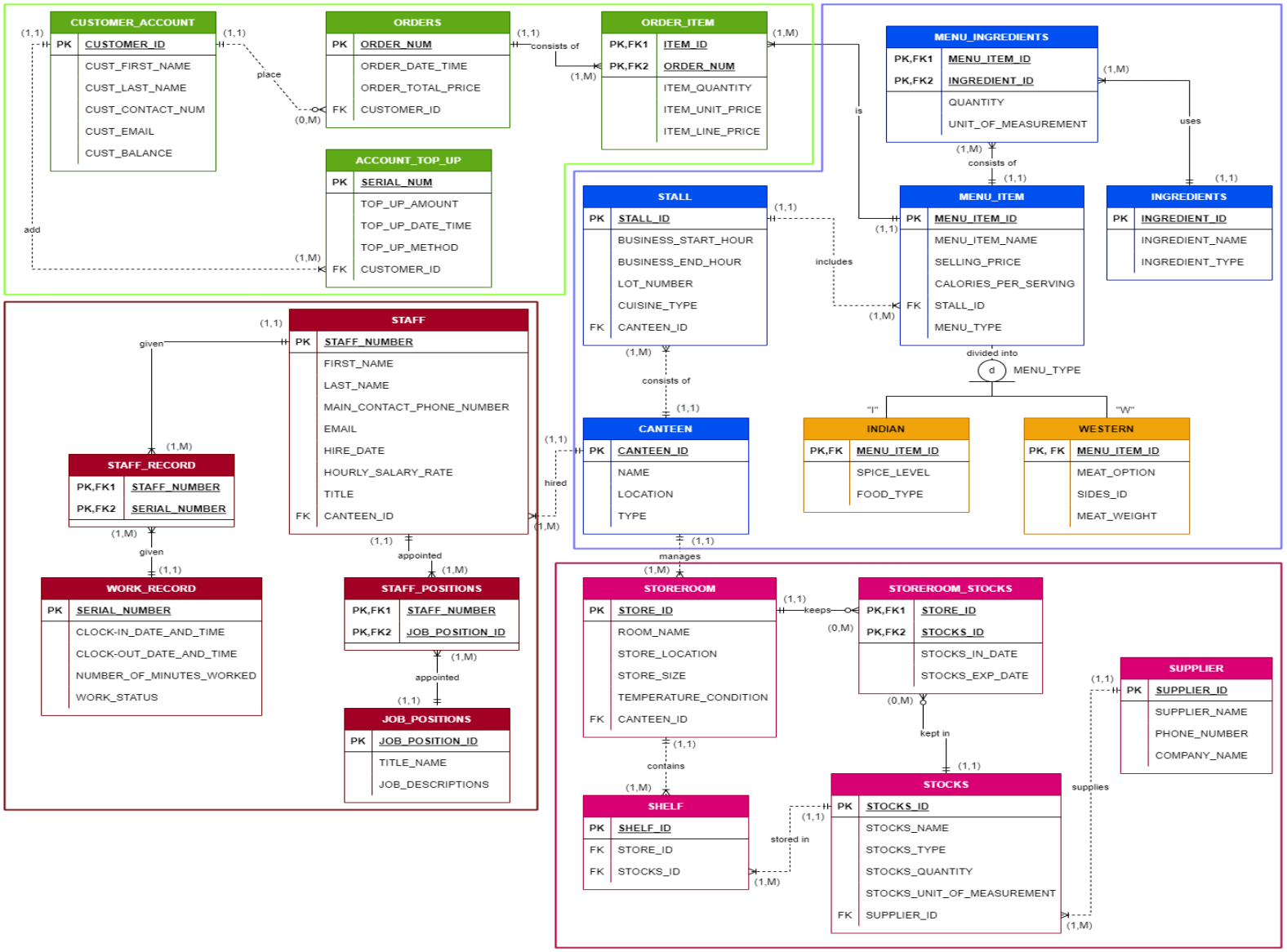
Module 3: Employees



Module 4: Orders



2.0 Extended ERD



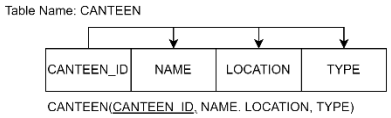
3.0 Normalization

[Specify the highest normal form all modules should achieve and justify why the normal form is selected. Explain if each table/relation from your ERD has achieved the selected normal form using the dependency diagram. If a table is not in the desired normal form, show the normalization steps. Do not need to start from 1NF if the tables from your ERD are already in higher normal form.]

The highest normal form achieved by all modules are 3NF. This is because all tables has reached the desired highest normal form which is 3NF because there is no transitive dependency and composite primary key/ primary key can be used to determine other attributes.

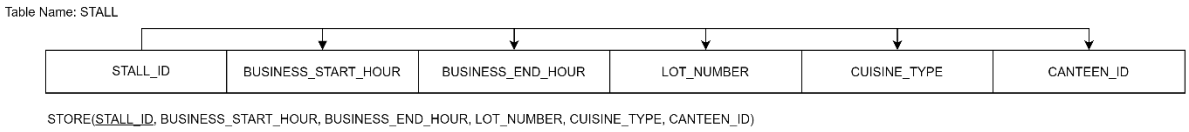
Module 1: Canteen and Stalls – Shindujaah Jaya Kumar

CANTEEN



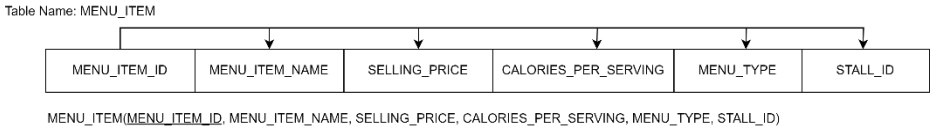
- The table is already in 1NF since all key attributes are defined, no repeating groups and all the attributes are dependent on the primary key.
- The table is already in 2NF too since it has no partial dependency.
- The table is currently in 3NF since it doesn't have any transitive dependency.

STALL



- The table is already in 1NF since all key attributes are defined, no repeating groups and all the attributes are dependent on the primary key.
- The table is already in 2NF too since it has no partial dependency.
- The table is currently in 3NF since it doesn't have any transitive dependency.

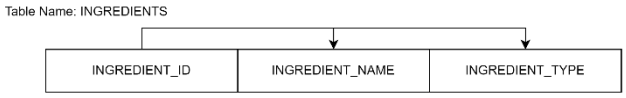
MENU_ITEM



MENU_ITEM(MENU_ITEM_ID, MENU_ITEM_NAME, SELLING_PRICE, CALORIES_PER_SERVING, MENU_TYPE, STALL_ID)

- The table is already in 1NF since all key attributes are defined, no repeating groups and all the attributes are dependent on the primary key.
- The table is already in 2NF too since it has no partial dependency.
- The table is currently in 3NF since it doesn't have any transitive dependency.

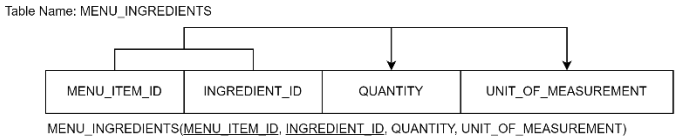
INGREDIENTS



INGREDIENTS(INGREDIENT_ID, INGREDIENT_NAME, INGREDIENT_TYPE)

- The table is already in 1NF since all key attributes are defined, no repeating groups and all the attributes are dependent on the primary key.
- The table is already in 2NF too since it has no partial dependency.
- The table is currently in 3NF since it doesn't have any transitive dependency.

MENU_INGREDIENTS

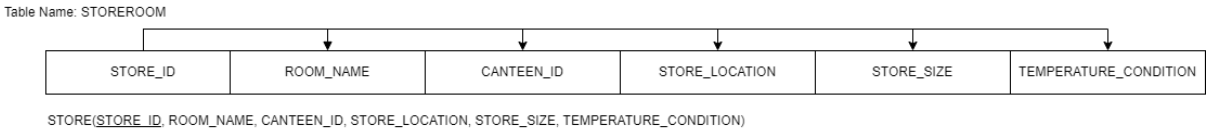


MENU_INGREDIENTS(MENU_ITEM_ID, INGREDIENT_ID, QUANTITY, UNIT_OF_MEASUREMENT)

- The table is already in 1NF since all key attributes are defined, no repeating groups and all the attributes are dependent on the primary key.
- The table is already in 2NF too since it has no partial dependency.
- The table is currently in 3NF since it doesn't have any transitive dependency.

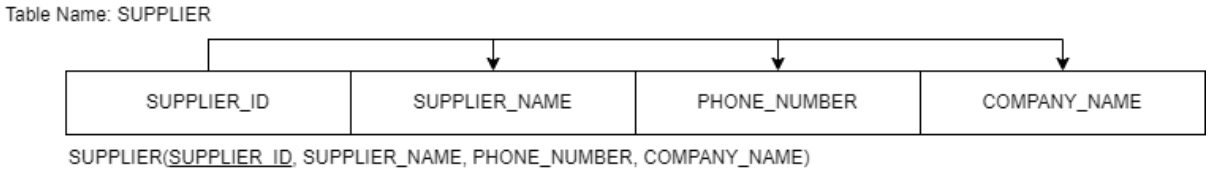
Module 2: Stocks – Thineshkumar Saravanan

STOREROOM



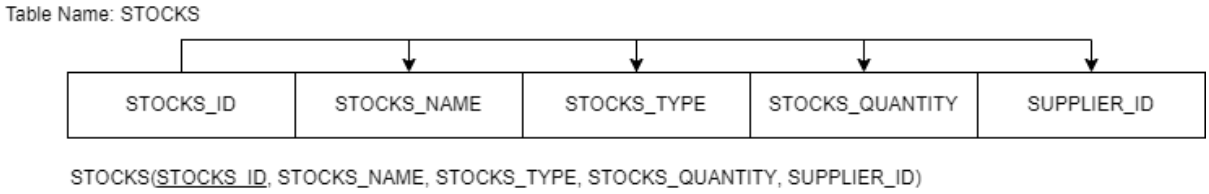
- The table is already in 1NF since all key attributes are defined, no repeating groups and all the attributes are dependent on the primary key.
- The table is already in 2NF too since it has no partial dependency.
- The table is currently in 3NF since it doesn't have any transitive dependency.

SUPPLIER



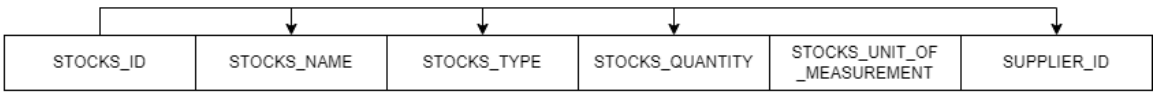
- The table is already in 1NF since all key attributes are defined, no repeating groups and all the attributes are dependent on the primary key.
- The table is already in 2NF too since it has no partial dependency.
- The table is currently in 3NF since it doesn't have any transitive dependency.

STOCKS



- The table is already in 1NF since all key attributes are defined, no repeating groups and all the attributes are dependent on the primary key.
- The table is already in 2NF too since it has no partial dependency.
- The table is currently in 3NF since it doesn't have any transitive dependency.

Table Name: STOCKS

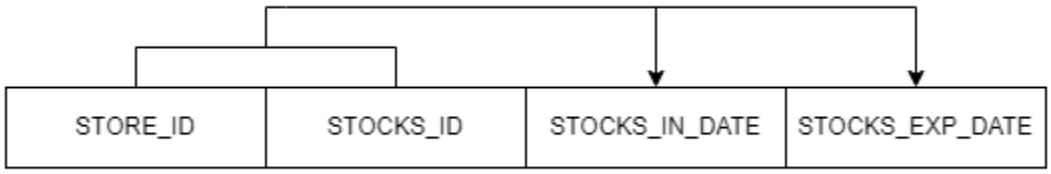


STOCKS(STOCKS_ID, STOCKS_NAME, STOCKS_TYPE, STOCKS_QUANTITY, STOCKS_UNIT_OF_MEASUREMENT, SUPPLIER_ID)

- To improve the database design, a new attribute called STOCKS_UNIT_OF_MEASUREMENT is added to the table to identify the unit for every STOCKS_QUANTITY.

STOREROOM_STOCKS

Table Name: STOREROOM_STOCKS

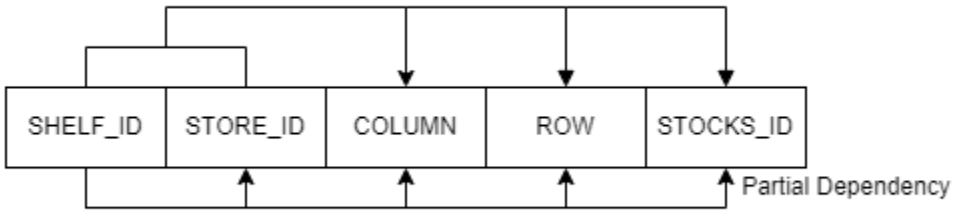


STOREROOM_STOCKS(STORE_ID, STOCKS_ID, STOCKS_IN_DATE, STOCKS_EXP_DATE)

- The table is already in 1NF since all key attributes are defined, no repeating groups and all the attributes are dependent on the primary key.
- The table is already in 2NF too since it has no partial dependency.
- The table is currently in 3NF since it doesn't have any transitive dependency.

SHELF

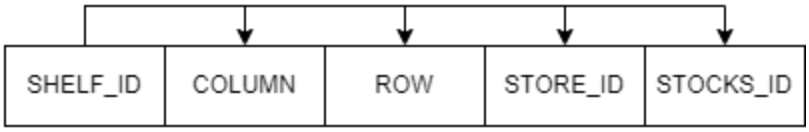
Table Name: SHELF



Shelf(SHELF_ID, STORE_ID, COLUMN, ROW, STOCKS_ID)

- The table is still in 1NF, because the composite key that consist of SHELF_ID and STORE_ID will form repeating groups.
- Although all the key attributes are defined, the primary key is not a candidate key.

Table Name: SHELF



SHELF(SHELF_ID, COLUMN, ROW, STORE_ID, STOCKS_ID)

- The table is already in 1NF since all key attributes are defined, no repeating groups and all. the attributes are dependent on the primary key.
- The table is already in 2NF too since it has no partial dependency.
- The table is currently in 3NF since it doesn't have any transitive dependency.

Table Name: SHELF

SHELF(SHELF_ID, STORE_ID, STOCKS_ID)

- The attributes ROW and COLUMN are redundant because according to the business requirement, a stock item is stored in multiple shelves but one shelf stores only one stock item.
- Therefore, no matter which row or column is chosen, the STOCKS_ID will be the same if the SHELF_ID is the same.

Module 3: Employees – Ilanthtamil Jayasangar

STAFF

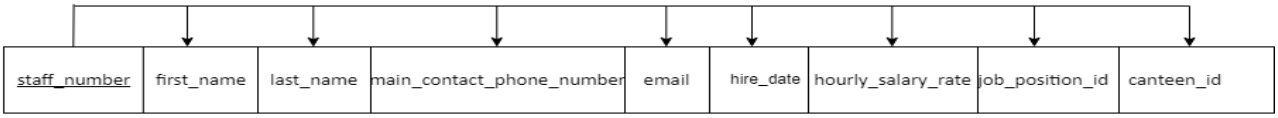


TABLE NAME:STAFF

STAFF(staff_number first_name last_name main_contact_phone_number email hire_date hourly_salary_rate job_position_id canteen_id)

- This STAFF table already in 3NF as it does not have repeating groups, partial dependencies, and transitive dependency.

WORK_RECORD

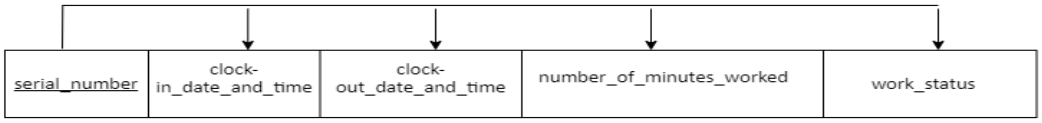


TABLE NAME :WORK_RECORD

WORK_RECORD(serial_number clock-in_date_and_time clock-out_date_and_time number_of_minutes_worked work_status)

- This WORK_RECORD table already in 3NF as it does not have repeating groups, partial dependencies and transitive dependency.

JOB_TITLES

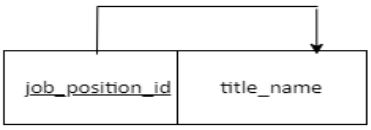


TABLE NAME :JOB TITLES

JOB TITLES (job_position_id,title_name)

- The JOB_POSITIONS table initially have transitive dependency.
- Then, we remove the transitive dependency and make the table to two separate tables which is called JOB_TITLES and JOB_POSITION_DESCRIPTION.
- Now, the tables are in 3NF.

JOB_POSITION_DESCRIPTION

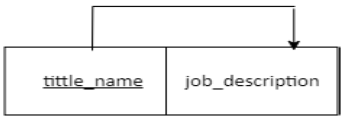


TABLE NAME:JOB_POSITION_DESCRIPTION

JOB_POSITION_DESCRIPTION(tittle_name,job_description)

STAFF_RECORD

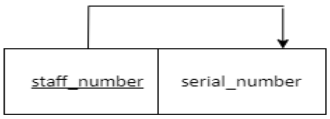


TABLE NAME:STAFF_RECORD

STAFF_RECORD(staff_number serial_number)

- This STAFF_RECORD table already in 3NF as it does not have repeating groups, partial dependencies and transitive dependency.

STAFF_POSITIONS

TABLE NAME : STAFF_POSITION

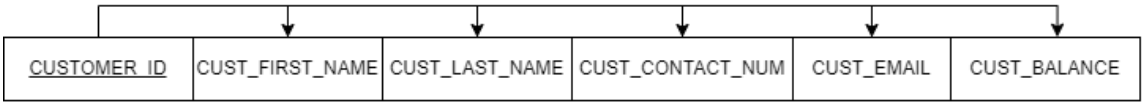
STAFF_POSITION(staff_number job_position_id)

- This STAFF_ POSITIONS table already in 3NF as it does not have repeating groups, partial dependencies and transitive dependency.

Module 4: Orders – Kavindhren Visvanathan

CUSTOMER_ACCOUNT

Table name: CUSTOMER_ACCOUNT

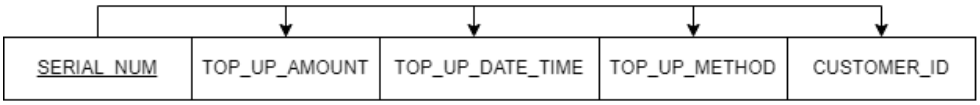


CUSTOMER_ACCOUNT (CUSTOMER_ID, CUST_FIRST_NAME, CUST_LAST_NAME, CUST_CONTACT_NUM, CUST_EMAIL, CUST_BALANCE)

- CUSTOMER_ACCOUNT table has reached 1NF because it doesn't have repeating groups, it has a primary key and it is in a table format.
- CUSTOMER_ACCOUNT table has reached 2NF because there are no attributes (columns) that depend on only part of a multi-part key which removes the possibilities of having partial dependency. Each of these non-key attributes are fully functionally dependent on the primary key (PK).
- CUSTOMER_ACCOUNT table has reached the desired highest normal form which is 3NF because there is no transitive dependency and primary key (PK) can be used to determine other attributes.

ACCOUNT_TOP_UP

Table name: ACCOUNT_TOP_UP



ACCOUNT_TOP_UP (SERIAL_NUM, TOP_UP_AMOUNT, TOP_UP_DATE_TIME, TOP_UP_METHOD, CUSTOMER_ID)

- ACCOUNT_TOP_UP table has reached 1NF because it doesn't have repeating groups, it has a primary key and it is in a table format.
- ACCOUNT_TOP_UP table has reached 2NF because there are no attributes (columns) that depend on only part of a multi-part key which removes the possibilities of having partial dependency. Each of these non-key attributes are fully functionally dependent on the primary key (PK).
- ACCOUNT_TOP_UP table has reached the desired highest normal form which is 3NF because there is no transitive dependency and primary key (PK) can be used to determine other attributes.

ORDERS

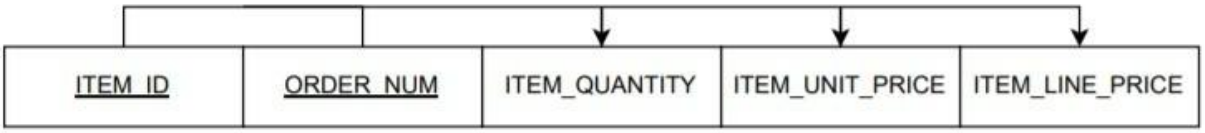
Table name: ORDERS

ORDER (ORDER_NUM, ORDER_DATE_TIME, ORDER_TOTAL_PRICE, CUSTOMER_ID)

- ORDERS table has reached 1NF because it doesn't have repeating groups, it has a primary key and it is in a table format.
- ORDERS table has reached 2NF because there are no attributes (columns) that depend on only part of a multi-part key which removes the possibilities of having partial dependency. Each of these non-key attributes are fully functionally dependent on the primary key (PK).
- ORDERS table has reached the desired highest normal form which is 3NF because there is no transitive dependency and primary key (PK) can be used to determine other attributes.

ORDER_ITEM

Table name: ORDER_ITEM



ORDER_ITEM (ITEM_ID, ORDER_NUM, ITEM_QUANTITY, ITEM_UNIT_PRICE, ITEM_LINE_PRICE)

- ORDER_ITEM table has reached 1NF because it doesn't have repeating groups, it has a composite primary key and it is in a table format.
- ORDER_ITEM table has reached 2NF because there are no attributes (columns) that depend on only part of a multi-part key which removes the possibilities of having partial dependency. Each of these non-key attributes are fully functionally dependent on the composite primary key.
- ORDER_ITEM table has reached the desired highest normal form which is 3NF because there is no transitive dependency and composite primary key can be used to determine other attributes.

4.0 Data Dictionary

Table Name	Attribute Name	Contents	Data Type	Format	Range	Required	PK or FK	FK Referenced Table
CANTEEN	CANTEEN_ID	ID of the canteens	VARCHAR(50)	C10XXX	50	Yes	PK	-
	NAME	Name of the canteens	VARCHAR(30)		30	Yes	-	-
	LOCATION	Location of the canteens	VARCHAR(250)		250	Yes	-	-
	TYPE	Canteen's type	VARCHAR(10)		10	Yes	-	-
STALL	STALL_ID	ID of the stalls	VARCHAR(10)	S1XX	10	Yes	PK	-
	BUSINESS_START_HOUR	Business start hour of the stall	VARCHAR(10)		10	Yes	-	-
	BUSINESS_END_HOUR	Business end hour of the stall	VARCHAR(10)		10	Yes	-	-
	LOT_NUMBER	Lot number of the stall	VARCHAR(10)	LXXX	10	Yes	-	-
	CUISINE_TYPE	Type of cuisine sold in the stall	VARCHAR(20)		20	Yes	-	-
	CANTEEN_ID	ID of the canteens	VARCHAR(10)		10	Yes	FK	CANTEEN
MENU_ITEM	MENU_ITEM_ID	ID of the menu item	VARCHAR(10)	X1XXX	10	Yes	PK	-
	MENU_ITEM_NAME	Name of the menu item	VARCHAR(50)		50	Yes	-	-

	SELLING_PRICE	Selling price of the menu item	NUMBER(5,2)		5,2	Yes	-	-
	CALORIES_PER_SERVING	Calories_Per serving of the menu item	NUMBER		-	Yes	-	-
	STALL_ID	ID of the menu item	VARCHAR(10)		10	Yes	FK	STALL
	MENU_TYPE	Type of the menu item	CHAR(1)		1	Yes	-	-
INGREDIENTS	INGREDIENT_ID	ID of the ingredient	VARCHAR(10)	IGXXX	10	Yes	PK	-
	INGREDIENT_NAME	Name of the ingredient	VARCHAR(50)		50	Yes	-	-
	INGREDIENT_TYPE	Type of the ingredient	VARCHAR(10)		10	Yes	-	-
MENU_INGREDIENTS	MENU_ITEM_ID	ID of the menu item	VARCHAR(10)	X1XXX	10	Yes	PK,FK	INGREDIENTS
	INGREDIENT_ID	ID of theingredient	VARCHAR(10)	IGXXX	10	Yes	PK,FK	MENU_ITEM
	QUANTITY	Quantity of the ingredient	NUMBER(5,2)		5,2	Yes	-	-
	UNIT_OF_MEASUREMENT	Unit of measurement of the ingredient	VARCHAR(20)		20	Yes	-	-
STOREROOM	STORE_ID	Store ID	VARCHAR(10)	SRMXXX		Yes	PK	-
	ROOM_NAME	Store name	VARCHAR(30)				-	-
	CANTEEN_ID	Canteen ID that the store belongs to	VARCHAR(10)			Yes	FK	CANTEEN

	STORE_LOCATION	Where the store is located	VARCHAR(30)				-	-
	STORE_SIZE	Store size	INTEGER				-	-
	TEMPERATURE CONDITION	Temperature of the store	VARCHAR(30)		‘Room Temperature’ or ‘Chilled’		-	-
SUPPLIER	SUPPLIER_ID	Supplier ID	VARCHAR(10)	SUPXX		Yes	PK	-
	SUPPLIER_NAME	Supplier name	VARCHAR(30)				-	-
	PHONE_NUMBER	Supplier phone number	VARCHAR(20)				-	-
	STOCKS_NAME	Stocks name delivered	VARCHAR(45)				-	-
STOCKS	STOCKS_ID	Stocks ID	VARCHAR(10)	STOXX		Yes	PK	-
	STOCKS_NAME	Stocks name	VARCHAR(45)				-	-
	STOCKS_TYPE	Type of stocks	VARCHAR(20)				-	-
	STOCKS_QUANTITY	Stocks quantity	INTEGER				-	-
	STOCKS_UNIT_OF_MEASUREMENT	Measurement unit of the stocks	VARCHAR(20)		‘kilogram’, ‘litre’ or ‘unit’		-	-
	SUPPLIER_ID	Supplier ID	VARCHAR(10)	SUPXX		Yes	FK	SUPPLIER
STOREROOM_STOCKS	STORE_ID	Store ID	VARCHAR(10)	SRMXX		Yes	PK, FK	STORE

	STOCKS_ID	Stocks ID	VARCHAR(10)	STOXXX		Yes	PK, FK	STOCKS
	STOCKS_IN_DATE	Imported date	DATE				-	-
	STOCKS_EXP_DATE	Expiry date	DATE				-	-
SHELF	SHELF_ID	Shelf ID	VARCHAR(10)	SHFXXX		Yes	PK	
	STORE_ID	Store ID	VARCHAR(10)	SRMXXX		Yes	FK	STORE
	STOCKS_ID	Stocks ID	VARCHAR(10)	STOXXX		Yes	FK	STOCKS
STAFF	staff_number	Staff's number	VARCHAR(10)	TEXT	10	Yes	PK	-
	first_name	Staff's first name	VARCHAR(30)	TEXT	30	Yes	-	-
	last_name	Staff's last name	VARCHAR(30)	TEXT	30	Yes	-	-
	main_contact_phone_number	Staff's phone number	NUMBER(11)	NUMBER	11	Yes	-	-
	email	Staff email	VARCHAR2(30)	TEXT	30	Yes	-	-
	hire_date	Staff hire date	DATE	DD/MM/YYYY	-	Yes	-	-
	title	Staff title	VARCHAR(30)	TEXT	30	Yes	-	-
	hourly_salary_rate	Staff's salary date per hour	NUMBER(5,2)	NUMBER	5,2	Yes	-	-
	job_position_id	Staff's job position id	VARCHAR(10)	TEXT	10	Yes	FK	-
	canteen_id	Staff's canteen id	VARCHAR(3)	TEXT	10	Yes	FK	CANTEEN

CANTEEN	canteen_id	Canteen ID	VARCHAR(50)	TEXT	50	YES	PK	-
	name	Canteen name	VARCHAR(30)	TEXT	30	YES	-	-
	location	Canteen location	VARCHAR(250)	TEXT	250	YES	-	-
	type	Canteen type	VARCHAR(10)	TEXT	10	YES	-	-
WORK_RECORD	serial_number	Staff serial number	VARCHAR(10)	TEXT	10	Yes	PK	-
	staff_number	Staff number	NUMBER(10)	NUMBER	10	Yes	-	-
	clock-in_date_and_time	Staff clock in date and time	TIMESTAMP	MM/DD/YYYY HH:MI:SS:FF	-	Yes	-	-
	clock-out_date_and_time	Staff clock out date and time	TIMESTAMP	MM/DD/YYYY HH:MI:SS:FF	-		-	-
	number_of_minutes_worked	Staff number of minutes worked	NUMBER(10,2)	NUMBER	10,2	Yes	-	-
	work_status	Staff work status	VARCHAR(10)	TEXT	10	Yes	-	-
STAFF_RECORD	staff_number	Staff number	VARCHAR(10)	TEXT	10	Yes	PK	STAFF

	serial_number	Staff serial number	VARCHAR(10)	TEXT	10	Yes	PK	WORK RECORD
JOB_TITLES	job_position_id	Staff job titles	VARCHAR(10)	TEXT	10	Yes	PK	-
	title_name	Staff title name	VARCHAR(30)	TEXT	30	Yes	-	JOB_POSITION_DESCRIPTIONS
JOB_POSITION_DESCRIPTION	title_name	Staff title name	VARCHAR(30)	TEXT	30	Yes	PK	-
	job_description	Staff job description	VARCHAR(100)	TEXT	100	Yes	-	-
STAFF_POSITIONS	Staff_number		VARCHAR (10)	TEXT	10	Yes	PK	STAFF
	Job_positions_id		VARCHAR (10)	TEXT	10	Yes	PK	JOB TITLES
CUSTOMER_ACCOUNT	CUSTOMER_ID	Customer's ID	NUMBER(6,0)		6	Yes	PK	-
	CUST_FIRST_NAME	Customer's first name	VARCHAR(15)			Yes	-	-
	CUST_LAST_NAME	Customer's last name	VARCHAR(15)			Yes	-	-
	CUST_CONTACT_NUM	Customer's contact number	VARCHAR(13)	XXX-XXXXXXX		Yes	-	-
	CUST_EMAIL	Customer's email	VARCHAR(30)				-	-
	CUST_BALANCE	Account balance	NUMBER(7,2)				-	-
ACCOUNT_TOP_UP	SERIAL_NUM	Top up's serial number	NUMBER(6,0)		6	Yes	PK	-

	TOP_UP_AMOUNT	Top up amount	NUMBER(6,2)			Yes	-	-
	TOP_UP_DATE_TIME	Date and time of top up purchased	TIMESTAMP(0)	YYYY-MM-DD HH24:MI:SS			-	-
	TOP_UP_METHOD	Top up purchase method	VARCHAR(30)			Yes	-	-
	CUSTOMER_ID	Customer's ID	NUMBER(6,0)		6	Yes	FK	CUSTOMER_ACCOUNT
ORDERS	ORDER_NUM	Order's code number	NUMBER(6,0)		6	Yes	PK	-
	ORDER_DATE_TIME	Date and time of order placed	TIMESTAMP(0)	YYYY-MM-DD HH24:MI:SS			-	-
	ORDER_TOTAL_PRICE	Total price of the order	NUMBER(6,2)			Yes	-	-
	CUSTOMER_ID	Customer's ID	NUMBER(6,0)			Yes	FK	CUSTOMER_ACCOUNT
ORDER_ITEM	ITEM_ID	Ordered item's ID	VARCHAR(10)			Yes	PK, FK	MENU_ITEM
	ITEM_QUANTITY	Amount of ordered item	NUMBER(10,0)			Yes	-	-
	ITEM_UNIT_PRICE	Ordered item's unit prices	NUMBER(5,2)				-	-
	ITEM_LINE_PRICE	Ordered item's line price	NUMBER(7,2)				-	-
	ORDER_NUM	Order's code number	NUMBER(6,0)		6	Yes	PK, FK	ORDER

5.0 Database Implementation

5.1 DDL

[SQL commands to create and implement the data structure and objects. Also include the use of triggers and stored procedures for data manipulations (if any).]

CANTEEN TABLE

```
CREATE TABLE CANTEEN
(
   CANTEEN_ID VARCHAR(10) PRIMARY KEY,
   CANTEEN_NAME VARCHAR(20) NOT NULL,
   CANTEEN_LOCATION VARCHAR(20) NOT NULL,
   CANTEEN_TYPE VARCHAR(20) NOT NULL
);
```

STALL TABLE

```
CREATE TABLE STALL
(
   STALL_ID VARCHAR(10) PRIMARY KEY,
   BUSINESS_START_HOUR VARCHAR(10) NOT NULL,
   BUSINESS_END_HOUR VARCHAR(10) NOT NULL,
   LOT_NUMBER VARCHAR(10) NOT NULL,
   CUISINE_TYPE VARCHAR(20) NOT NULL,
   CANTEEN_ID VARCHAR(10) NOT NULL REFERENCES CANTEEN(CANTEEN_ID)
);
```

MENU_ITEM TABLE

```
CREATE TABLE MENU_ITEM
(
   MENU_ITEM_ID VARCHAR(10) PRIMARY KEY,
   MENU_ITEM_NAME VARCHAR(50) NOT NULL,
   SELLING_PRICE NUMBER(5,2) NOT NULL,
   CALORIES_PER_SERVING INTEGER,
   STALL_ID VARCHAR(10) NOT NULL REFERENCES STALL(STALL_ID),
   MENU_TYPE CHAR(1) NOT NULL
);
```

INGREDIENTS TABLE

```
CREATE TABLE INGREDIENTS
(
   INGREDIENT_ID VARCHAR(10) PRIMARY KEY,
   INGREDIENT_NAME VARCHAR(50) NOT NULL,
   INGREDIENT_TYPE VARCHAR(10) NOT NULL
);
```

MENU_INGREDIENTS TABLE

```
CREATE TABLE MENU_INGREDIENTS
(
   MENU_ITEM_ID VARCHAR(10) NOT NULL,
   INGREDIENT_ID VARCHAR(10)NOT NULL,
   QUANTITY NUMBER(5,2) NOT NULL,
   UNIT_OF_MEASUREMENT VARCHAR(20) NOT NULL,
   CONSTRAINT MENU_INGREDIENTS_PK PRIMARY KEY(MENU_ITEM_ID,
INGREDIENT_ID),
   CONSTRAINT MENU_INGREDIENTS_FK1 FOREIGN KEY(MENU_ITEM_ID)
      REFERENCES MENU_ITEM(MENU_ITEM_ID) ON DELETE CASCADE,
   CONSTRAINT MENU_INGREDIENTS_FK2 FOREIGN KEY(INGREDIENT_ID)
      REFERENCES INGREDIENTS(INGREDIENT_ID) ON DELETE CASCADE
);
```

WESTERN TABLE [SECTION 2.0]

```
CREATE TABLE WESTERN
(
   MENU_ITEM_ID VARCHAR(10),
   MEAT_OPTION VARCHAR(20),
   SIDES_ID VARCHAR(30),
   MEAT_WEIGHT VARCHAR(20),
   CONSTRAINT WESTERN_PK PRIMARY KEY(MENU_ITEM_ID)
);
```

INDIAN TABLE [SECTION 2.0]

```
CREATE TABLE INDIAN
(
   MENU_ITEM_ID VARCHAR(10),
   SPICE_LEVEL VARCHAR(20),
   FOOD_TYPE VARCHAR(30),
   CONSTRAINT INDIAN_PK PRIMARY KEY(MENU_ITEM_ID)
);
```

STOREROOM TABLE

```
CREATE TABLE STOREROOM
(
   STORE_ID VARCHAR(10),
   ROOM_NAME VARCHAR(30) NOT NULL,
   CANTEEN_ID VARCHAR(10) NOT NULL,
   STORE_LOCATION VARCHAR(30) NOT NULL,
   STORE_SIZE INTEGER NOT NULL,
   TEMPERATURE_CONDITION VARCHAR(30),
   CONSTRAINT STORE_PK PRIMARY KEY(STORE_ID),
   CONSTRAINT STORE_FK FOREIGN KEY(CANTEEN_ID)
      REFERENCES CANTEEN(CANTEEN_ID) ON DELETE CASCADE
);
```

SUPPLIER TABLE

```
CREATE TABLE SUPPLIER
(
   SUPPLIER_ID VARCHAR(10),
   SUPPLIER_NAME VARCHAR(30),
   PHONE_NUMBER VARCHAR(20),
   COMPANY_NAME VARCHAR(45),
   CONSTRAINT SUPPLIER_PK PRIMARY KEY(SUPPLIER_ID)
);
```

STOCKS TABLE

```
CREATE TABLE STOCKS
(
   SROCKS_ID VARCHAR(10) NOT NULL,
   STOCKS_NAME VARCHAR(45) NOT NULL,
   STOCKS_TYPE VARCHAR(20) NOT NULL,
   STOCKS_QUANTITY VARCHAR(45) ,
   SUPPLIER_ID VARCHAR(10) NOT NULL,
   CONSTRAINT STOCKS_PK PRIMARY KEY(STOCKS_ID),
   CONSTRAINT STOCKS_FK FOREIGN KEY(SUPPLIER_ID)
      REFERENCES SUPPLIER(SUPPLIER_ID) ON DELETE CASCADE
);
```

SHELF TABLE

```
CREATE TABLE SHELF
(
   SHELF_ID VARCHAR(10) NOT NULL,
   STORE_ID VARCHAR(10) NOT NULL,
   STOCKS_ID VARCHAR(10) NOT NULL,
   CONSTRAINT SHELF_PK PRIMARY KEY(SHELF_ID),
   CONSTRAINT SHELF_STORE_FK FOREIGN KEY(STORE_ID)
      REFERENCES STOREROOM(STORE_ID) ON DELETE CASCADE,
   CONSTRAINT SHELF_STOCKS_FK FOREIGN KEY(STOCKS_ID)
      REFERENCES STOCKS(STOCKS_ID) ON DELETE CASCADE
);
```

STOREOOM_STOCKS TABLE

```
CREATE TABLE STOREOOM_STOCKS
(
   STORE_ID VARCHAR(10) NOT NULL,
   STOCKS_ID VARCHAR(10)NOT NULL,
   STOCKS_IN_DATE DATE,
   STOCKS_EXP_DATE DATE,
   CONSTRAINT STORE_STOCKS_PK PRIMARY KEY(STORE_ID, STOCKS_ID),
   CONSTRAINT STORE_STOCKS_FK1 FOREIGN KEY(STORE_ID)
      REFERENCES STOREROOM(STORE_ID) ON DELETE CASCADE,
   CONSTRAINT STORE_STOCKS_FK2 FOREIGN KEY(STOCKS_ID)
      REFERENCES STOCKS(STOCKS_ID) ON DELETE CASCADE
);
```

JOB_POSITION_DESCRIPTIONS TABLE

```
CREATE TABLE JOB_POSITION_DESCRIPTIONS (
title_name VARCHAR(30) NOT NULL,
job_description VARCHAR(100) NOT NULL,
PRIMARY KEY (title_name)
);
```

JOB_TITLES TABLE

```
CREATE TABLE JOB_TITLES (
job_position_id VARCHAR(10) NOT NULL,
title_name VARCHAR(30) NOT NULL,
PRIMARY KEY (job_position_id),
FOREIGN KEY (title_name) REFERENCES JOB_POSITION_DESCRIPTIONS (title_name)
);
```

STAFF TABLE

```
CREATE TABLE STAFF(
staff_number VARCHAR(10) NOT NULL,
first_name VARCHAR2(30) NOT NULL,
last_name VARCHAR2(30) NOT NULL,
main_contact_phone_number VARCHAR(11) NOT NULL,
email VARCHAR2(30) NOT NULL,
hire_date DATE NOT NULL,
hourly_salary_rate NUMBER(5,2) NOT NULL,
title VARCHAR2(30) NOT NULL,
job_position_id  VARCHAR(10)  NOT NULL,
canteen_id VARCHAR(3) NOT NULL,
PRIMARY KEY (staff_number),
FOREIGN KEY (canteen_id) REFERENCES Canteen (canteen_id)
);
```

WORK_RECORD TABLE

```
CREATE TABLE WORK_RECORD(
serial_number VARCHAR(10) NOT NULL,
clock_in_date_and_time TIMESTAMP (2) NOT NULL,
clock_out_date_and_time TIMESTAMP (2) NOT NULL,
number_of_minutes_worked NUMBER(10, 2) NOT NULL,
work_status VARCHAR(10) NOT NULL,
PRIMARY KEY (serial_number)
);
```

STAFF_RECORD TABLE

```
CREATE TABLE STAFF_RECORD(
staff_number VARCHAR(10) NOT NULL,
serial_number VARCHAR(10) NOT NULL,
PRIMARY KEY (staff_number, serial_number),
FOREIGN KEY (staff_number) REFERENCES Staff (staff_number),
FOREIGN KEY (serial_number) REFERENCES Work_Record (serial_number)
);
```

STAFF_POSITIONS TABLE

```
CREATE TABLE STAFF_POSITIONS(
staff_number VARCHAR(10) NOT NULL,
job_position_id  VARCHAR(10) NOT NULL,
PRIMARY KEY (staff_number, job_position_id),
FOREIGN KEY (staff_number) REFERENCES Staff (staff_number),
FOREIGN KEY (job_position_id) REFERENCES Job_Titles (job_position_id)
);
```

CUSTOMER_ACCOUNT TABLE

```
CREATE TABLE CUSTOMER_ACCOUNT
(
   CUSTOMER_ID NUMBER(6,0) NOT NULL,
   CUST_FIRST_NAME VARCHAR(15) NOT NULL,
   CUST_LAST_NAME VARCHAR(15) NOT NULL,
   CUST_CONTACT_NUM VARCHAR(13) NOT NULL UNIQUE,
   CUST_EMAIL VARCHAR(30),
   CUST_BALANCE NUMBER(7,2),
   CONSTRAINT CUSTOMER_ID_PK PRIMARY KEY(CUSTOMER_ID)
);
```

ACCOUNT_TOP_UP_TABLE

```
CREATE TABLE ACCOUNT_TOP_UP
(
   SERIAL_NUM NUMBER(6,0) NOT NULL,
   TOP_UP_AMOUNT NUMBER(6,2) NOT NULL,
   TOP_UP_DATE_TIME TIMESTAMP(0) DEFAULT CURRENT_TIMESTAMP,
   TOP_UP_METHOD VARCHAR(30) NOT NULL,
   CUSTOMER_ID NUMBER(6,0) NOT NULL,
   CONSTRAINT SERIAL_NUM_PK PRIMARY KEY(SERIAL_NUM),
   CONSTRAINT ATP_CUSTOMER_ID_FK FOREIGN KEY (CUSTOMER_ID) REFERENCES
CUSTOMER_ACCOUNT (CUSTOMER_ID) ON DELETE CASCADE
);
```

ORDERS TABLE

```
CREATE TABLE ORDERS
(
   ORDER_NUM NUMBER(6,0)NOT NULL,
   ORDER_DATE_TIME TIMESTAMP(0) DEFAULT CURRENT_TIMESTAMP,
   ORDER_TOTAL_PRICE NUMBER(6,2) DEFAULT 0 NOT NULL,
   CUSTOMER_ID NUMBER(6,0) NOT NULL,
   CONSTRAINT ORDER_NUM_PK PRIMARY KEY(ORDER_NUM),
   CONSTRAINT O_CUSTOMER_ID_FK FOREIGN KEY (CUSTOMER_ID) REFERENCES
CUSTOMER_ACCOUNT (CUSTOMER_ID) ON DELETE CASCADE
);
```

ORDER_ITEM TABLE

```
CREATE TABLE ORDER_ITEM
(
```

```
   ITEM_ID VARCHAR(10)NOT NULL,
   ITEM_QUANTITY NUMBER(10,0) NOT NULL,
   ITEM_UNIT_PRICE NUMBER(5,2),
   ITEM_LINE_PRICE NUMBER(7,2),
   ORDER_NUM NUMBER(6,0)NOT NULL,
   CONSTRAINT ITEM_ID_ORDER_NUM_PK PRIMARY KEY (ITEM_ID, ORDER_NUM),
   CONSTRAINT ITEM_ID_FK FOREIGN KEY (ITEM_ID) REFERENCES MENU_ITEM
(MENU_ITEM_ID) ON DELETE CASCADE ,
   CONSTRAINT ORDER_NUM_FK FOREIGN KEY (ORDER_NUM) REFERENCES ORDERS
(ORDER_NUM) ON DELETE CASCADE
);
```

SEQUENCE AND TRIGGER FOR CUSTOMER_ACCOUNT

```
CREATE SEQUENCE CUST_ID_SEQ
START WITH 51 INCREMENT BY 1 NOCACHE;

CREATE TRIGGER CUST_ACCOUNT_INSERT
  BEFORE INSERT ON CUSTOMER_ACCOUNT
  FOR EACH ROW
BEGIN
  SELECT CUST_ID_SEQ.NEXTVAL
  INTO :NEW.CUSTOMER_ID
  FROM DUAL;
END;
```

SEQUENCE AND TRIGGER FOR TOP_UP_ACCOUNT

```
CREATE SEQUENCE SERIAL_NUM_SEQ
START WITH 100 INCREMENT BY 1 NOCACHE;

CREATE TRIGGER SERIAL_NUM_INSERT
  BEFORE INSERT ON ACCOUNT_TOP_UP
  FOR EACH ROW
BEGIN
  SELECT SERIAL_NUM_SEQ.NEXTVAL
  INTO :NEW.SERIAL_NUM
  FROM DUAL;
END;
```

SEQUENCE AND TRIGGER FOR ORDERS

```
CREATE SEQUENCE ORDER_NUM_SEQ
START WITH 111 INCREMENT BY 1 NOCACHE;
```

```
CREATE TRIGGER ORDER_NUM_INSERT
  BEFORE INSERT ON ORDERS
  FOR EACH ROW
BEGIN
  SELECT ORDER_NUM_SEQ.NEXTVAL
  INTO :NEW.ORDER_NUM
  FROM DUAL;
END;
```

DATE TIME TRIGGER FOR ACCOUNT_TOP_UP

```
CREATE OR REPLACE TRIGGER DATE_TIME_TRG
   BEFORE INSERT OR UPDATE ON ACCOUNT_TOP_UP
   FOR EACH ROW
   BEGIN
        :NEW.TOP_UP_DATE_TIME := SYSTIMESTAMP;
END;
```

DATE TIME TRIGGER FOR ODERS

```
CREATE OR REPLACE TRIGGER DATE_TIME_ORDERS_TRG
   BEFORE INSERT OR UPDATE ON ORDERS
   FOR EACH ROW
   BEGIN
        :NEW.ORDER_DATE_TIME := SYSTIMESTAMP;
END;
```

ADD CUSTOMER BALANCE TRIGGER

```
CREATE OR REPLACE TRIGGER TR_ACCOUNT_TOP_UP_AFTER_INSERT
AFTER INSERT ON ACCOUNT_TOP_UP FOR EACH ROW
BEGIN
    UPDATE CUSTOMER_ACCOUNT
    SET CUST_BALANCE = NVL(CUST_BALANCE,0) + :NEW.TOP_UP_AMOUNT
    WHERE CUSTOMER_ID = :NEW.CUSTOMER_ID;
END;
```

BIG TRIGGER

```
CREATE OR REPLACE TRIGGER ORDER_ITEM_AFTER_IDU BEFORE INSERT OR
UPDATE OR DELETE
ON ORDER_ITEM FOR EACH ROW
BEGIN
    IF INSERTING THEN
```

```
      SELECT SELLING_PRICE INTO :NEW.ITEM_UNIT_PRICE FROM MENU_ITEM WHERE
MENU_ITEM_ID = :NEW.ITEM_ID;
      :NEW.ITEM_LINE_PRICE := :NEW.ITEM_QUANTITY * :NEW.ITEM_UNIT_PRICE;
      UPDATE ORDERS SET ORDER_TOTAL_PRICE = NVL(ORDER_TOTAL_PRICE,0) +
:NEW.ITEM_LINE_PRICE
      WHERE ORDER_NUM = :NEW.ORDER_NUM;
   ELSIF UPDATING THEN
      SELECT SELLING_PRICE INTO :NEW.ITEM_UNIT_PRICE FROM MENU_ITEM WHERE
MENU_ITEM_ID = :NEW.ITEM_ID;
      :NEW.ITEM_LINE_PRICE := :NEW.ITEM_QUANTITY * :NEW.ITEM_UNIT_PRICE;
      UPDATE ORDERS SET ORDER_TOTAL_PRICE = NVL(ORDER_TOTAL_PRICE,0) +
:NEW.ITEM_LINE_PRICE - :OLD.ITEM_LINE_PRICE
      WHERE ORDER_NUM = :NEW.ORDER_NUM;
   ELSIF DELETING THEN
      UPDATE ORDERS SET ORDER_TOTAL_PRICE = NVL(ORDER_TOTAL_PRICE,0) -
:OLD.ITEM_LINE_PRICE
      WHERE ORDER_NUM = :OLD.ORDER_NUM;
   END IF;
END;
```

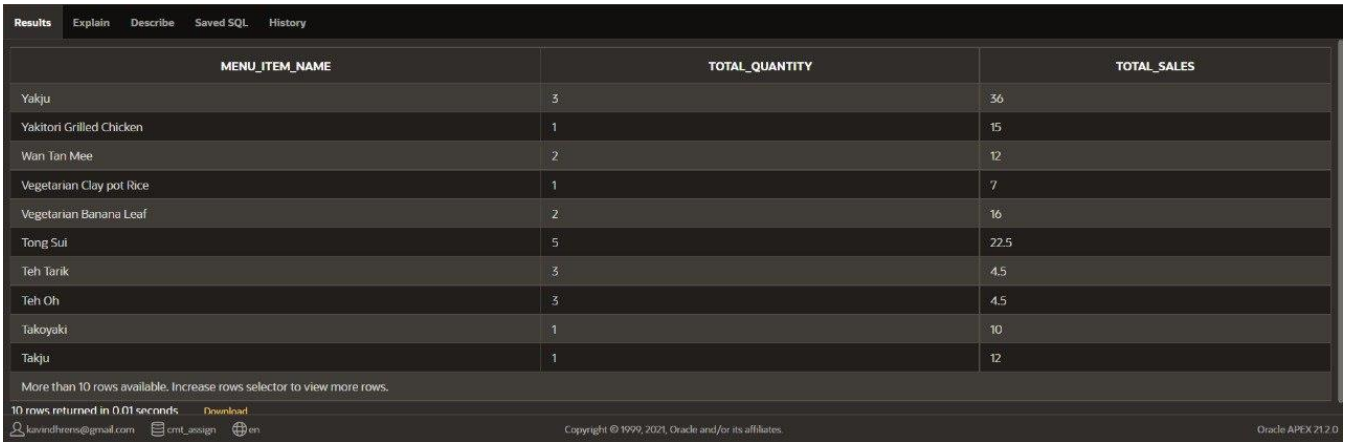
MINUS CUSTOMER BALANCE TRIGGER

```
CREATE OR REPLACE TRIGGER tr_order_after_insert
AFTER INSERT OR UPDATE ON orders FOR EACH ROW
BEGIN
   IF INSERTING THEN
      UPDATE customer_account SET cust_balance = NVL(cust_balance,0) -
:NEW.ORDER_TOTAL_PRICE
      WHERE customer_id = :NEW.customer_id;
   ELSIF UPDATING THEN
      UPDATE customer_account SET cust_balance = NVL(cust_balance,0) -
:NEW.ORDER_TOTAL_PRICE + :OLD.ORDER_TOTAL_PRICE
      WHERE customer_id = :NEW.customer_id;
   END IF;
END;
```

5.2 DML

[Write the SQL statement based on Section 3.0 of the case study and design an Oracle APEX user-generated report using the SQL statement (include a clear screenshot of the report design in run mode).]

```
SELECT
   MENU_ITEM_NAME,
   SUM(ITEM_QUANTITY) TOTAL_QUANTITY,
   SUM(ITEM_LINE_PRICE) TOTAL_SALES
FROM
   (( ORDER_ITEM
   FULL JOIN  MENU_ITEM ON ORDER_ITEM.ITEM_ID = MENU_ITEM.MENU_ITEM_ID)
   FULL JOIN ORDERS ON ORDER_ITEM.ORDER_NUM = ORDERS.ORDER_NUM)
WHERE
    ORDER_DATE_TIME BETWEEN '01-JAN-2020' AND '31-DEC-2020'
GROUP BY
   MENU_ITEM_NAME
HAVING
   SUM(ITEM_LINE_PRICE ) < 500
ORDER BY
   MENU_ITEM_NAME DESC;
```



Results | Explain | Describe | Saved SQL | History

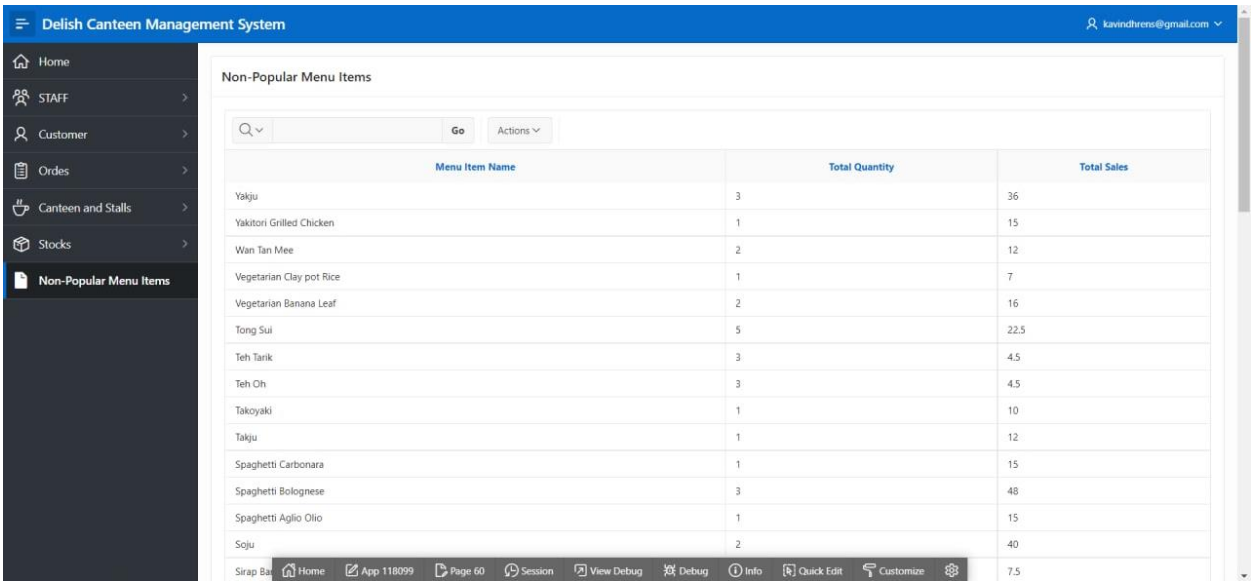
MENU_ITEM_NAME	TOTAL_QUANTITY	TOTAL_SALES
Yakju	3	36
Yakitori Grilled Chicken	1	15
Wan Tan Mee	2	12
Vegetarian Clay pot Rice	1	7
Vegetarian Banana Leaf	2	16
Tong Sui	5	22.5
Teh Tarik	3	4.5
Teh Oh	3	4.5
Takoyaki	1	10
Takju	1	12

More than 10 rows available. Increase rows selector to view more rows.

10 rows returned in 0.01 seconds Download

kavindhrens@gmail.com cmt_assign en Copyright © 1999, 2021, Oracle and/or its affiliates. Oracle APEX 21.2.0

Screenshot 1



Screenshot 2

6.0 Reflection

[Write a reflection based on your project experience. Can include project problems and pitfalls, how you overcome the problems and what did you learn from the database project.]

Delish Enterprise Canteen Management Database System was assigned to our group as a case study, and the first thing that came to mind was the canteen management that a few members of our group completed last semester. The process of this database project, beginning with discussing, planning, dividing the task, designing, and implementing our database, presented us with an excellent opportunity to gather useful knowledge about the building of a database application as well as work on our first database project. The members of this project conducted their own study for their module and managed to produce the best results possible despite the fact that it took a lot of energy, a lot of knowledge and research, and continual asking about the unclear procedure along the way. To be honest, having such a project allocated to students will undoubtedly provide an opportunity to improve on what we have learnt in our lectures and lab sessions. We were able to gain a better comprehension of this course throughout the way, as there were instances when we didn't understand some aspects of our labs and tutorials. As a result of executing such a project, we were able to gain expertise, and as the phrase goes, practise makes perfect, we realised and attempted to improve the database implementation after gaining a better understanding. Our group members were able to contribute fully to their assigned module, despite flaws such as misunderstanding and making mistakes while working on the project, which were properly addressed by all through communication and discussions as needed. Throughout the course of this project, we submitted four deliverables: the system planning report, the system design report, the system implementation report, and the system demo.

We would also want to express our gratitude to Dr. Jasy Liew and Mr. Ying Hao for their suggestions and assistance along the way, since the explanation provided enabled us to offer a more promising database application. The essentials of this course were covered, particularly when it came to database architecture, creating SQL queries, creating Entity Relationship Diagrams, and using Oracle APEX for our project. The learning process was visible, as were the problems encountered, but we managed to pull it off with a lot of internal and external guidance, as well as the backing of good team members.

Project Problems

1. Cannot increment staff number automatically with JavaScript. So, user need to enter the staff number manually each time.
2. Cannot automatically calculate number of minutes worked using JavaScript. So, user have to calculate number of minutes worked manually.
3. There is no built-in version control and all components must be edited through the web interface.
4. Cannot add external components other than the stuffs from oracle apex and there are only limited plugins to enhance the application.
5. The homepage in ORACLE automatically deleted without any prompt and wasn't able to retrieve, and realized several groups faced the same problem.

7.0 System Demo

Short Demo URL: https://youtu.be/hnGClTPsW3Y

Oracle APEX Cloud Login Details

- **Workspace: CMT_ASSIGN**
- **Username: kavindhrens@gmail.com**
- **Password: cmt221usm**
- **App name: Delish Enterprise Canteen Management Database System**