



MSc Data Science

Coursework Assessment:

CI7320 Database & Data Management

Title:- Database Design

Submitted by:-

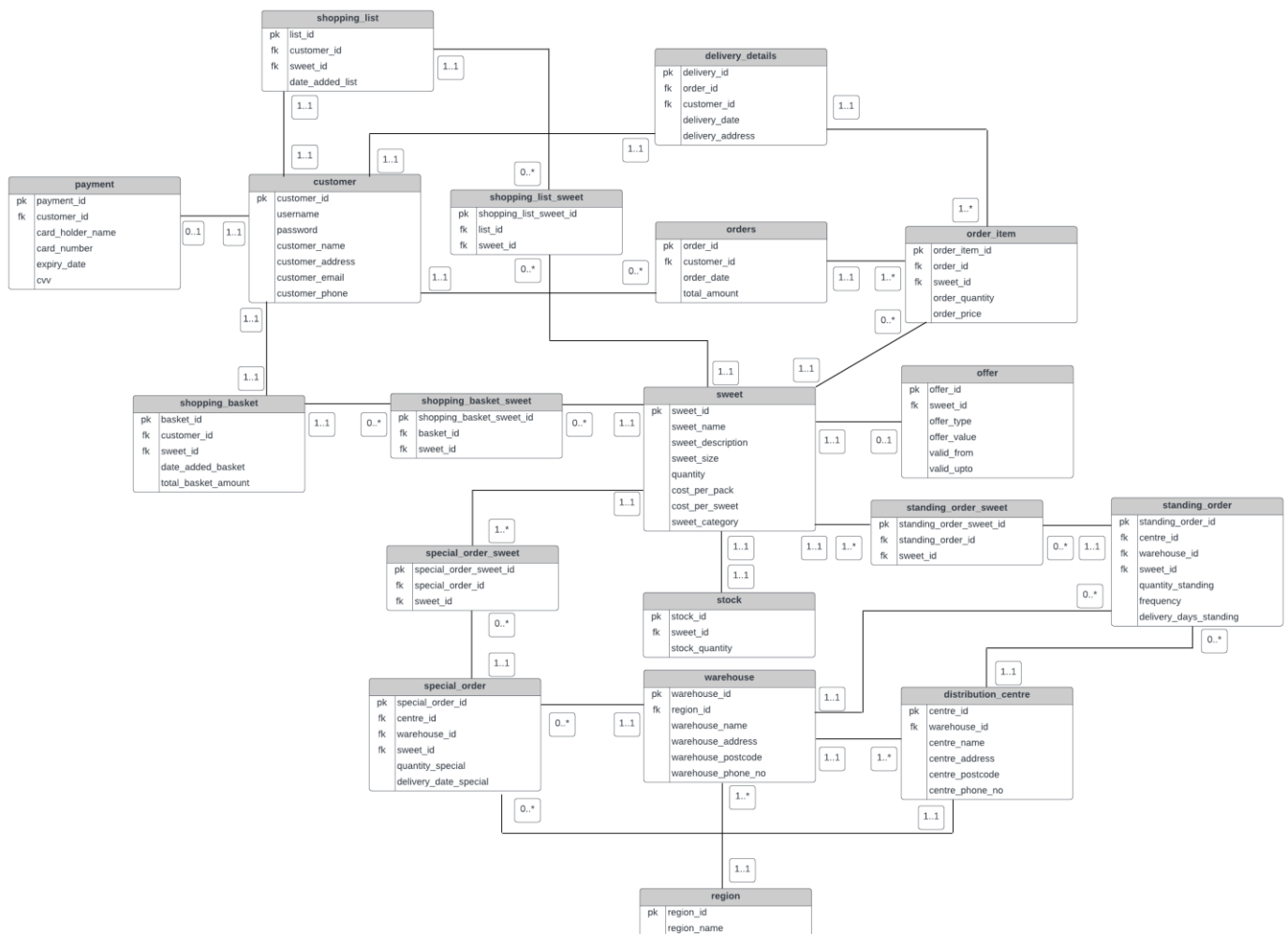
Name:- Jay Santosh Dugad

TABLE OF CONTENTS

Chapter 1. The Entity Relationship diagram with constraints and assumptions.....	3
Chapter 2. Listing of the SQL tables definitions.....	4
1. Customer table.....	4
2. Sweet table.....	4
3. Region table.....	5
4. Warehouse table.....	5
5. Distribution centre table.....	5
6. Orders table.....	5
7. Order item table.....	6
8. Shopping Basket table.....	6
9. Shopping List table.....	6
10. Stock table.....	7
11. Special order table.....	7
12. Standing order table.....	7
13. Delivery details table.....	8
14. Offer table.....	8
15. Payment table.....	8
16. Special order sweet (<i>junction table</i>)	9
17. Standing order sweet (<i>junction table</i>)	9
18. Shopping list sweet (<i>junction table</i>)	9
19. Shopping basket sweet (<i>junction table</i>)	9
Chapter 3. A discussion on the data that was entered with screenshots showing that the multiplicity of the relationships are reflected in the data that was entered.....	10
Chapter 4. Six queries that demonstrate that your database meets the requirements of the system.....	19
Chapter 5. Conclusion. A critical evaluation of your final product and a review of the entire exercise.....	25

CHAPTER 1: ENTITY RELATIONSHIP DIAGRAM

WITH CONSTRAINTS & ASSUMPTIONS



[Link for ER Diagram](#)

In the above ER diagram PK is referred as Primary Key and FK is referred as Foreign Key.

Constraints: -

- Sweet_id, customer_id, warehouse_id, region_id, centre_id, basket_id, list_id, payment_id, delivery_id, order_id, order_item_id, stock_id, offer_id, shopping_list_sweet_id, shopping_basket_sweet_id, standing_order_sweet_id, special_order_sweet_id should be unique and not null.
- In the payment table, card_holder_name, card_number and cvv should be unique.
- Sweets can be stored in multiple sizes and categories.
- Only one offer can be applicable to individual sweet. And one sweet can have only one offer.
- A customer must log in with their username and password before adding items to their shopping basket or list.
- Payment and delivery details are required for order checkout.
- Delivery charges are £5.00 for orders under £50, £10.00 for orders between £50 and £100, and free for orders over £100.
- Items may be delivered separately based on stock availability.

- Delivery is scheduled based on the customer's preferred time during checkout.
- Customers can change their address for delivery before placing order.
- Website accepts only online mode of payment.

Assumptions: -

- Each customer has a unique username and password combination.
- Standing orders have predefined frequencies and delivery days.
- New sweets are promoted on the website before they become available for purchase.
- Regional warehouses keep track of stock levels for each sweet.
- Discounts and offers are applied only to individual sweets, not packs.
- The database system will handle the calculation of delivery dates based on stock availability.
- Customers can add sweets directly to their shopping basket or list from the website interface.
- Shopping basket contents are saved for checkout until the session ends or the order is completed.
- Each order is assigned a unique order ID.
- Payment details include cardholder name, card number, expiry date, and CVV.
- The system maintains historical records of orders for customer tracking purposes.

CHAPTER 2: IMPLEMENTATION

Customer Table

```
CREATE TABLE customer (
    customer_id INT PRIMARY KEY NOT NULL,
    username VARCHAR(50),
    password VARCHAR(50),
    customer_name VARCHAR(100),
    customer_address VARCHAR(255),
    customer_email VARCHAR(100),
    customer_phone VARCHAR(20)
);
```

Sweet Table

```
CREATE TABLE sweet (
    sweet_id INT PRIMARY KEY NOT NULL,
    sweet_name VARCHAR(50),
    sweet_description VARCHAR(50),
    sweet_size VARCHAR(50),
    quantity INT,
```

```
cost_per_pack DECIMAL(10, 2),  
cost_per_sweet DECIMAL(10, 2),  
sweet_category VARCHAR(50)  
);
```

Region Table

```
CREATE TABLE region (  
    region_id INT PRIMARY KEY NOT NULL,  
    region_name VARCHAR(50)  
);
```

Warehouse Table.

```
CREATE TABLE warehouse (  
    warehouse_id INT PRIMARY KEY NOT NULL,  
    region_Cid INT,  
    warehouse_name VARCHAR(50),  
    warehouse_address VARCHAR(100),  
    warehouse_postcode VARCHAR(10),  
    warehouse_phone_no VARCHAR(15),  
    FOREIGN KEY (region_id) REFERENCES region(region_id)  
);
```

Distribution Centre Table

```
CREATE TABLE distribution_centre (  
    centre_id INT PRIMARY KEY NOT NULL,  
    warehouse_id INT,  
    centre_name VARCHAR(50),  
    centre_address VARCHAR(50),  
    centre_postcode VARCHAR(50),  
    centre_phone_no VARCHAR(15),  
    FOREIGN KEY (warehouse_id) REFERENCES warehouse(warehouse_id)  
);
```

Orders Table

```
CREATE TABLE orders (  
    order_id INT PRIMARY KEY NOT NULL,  
    customer_id INT,  
    order_date DATE,
```

```
total_amount DECIMAL(10, 2),  
FOREIGN KEY (customer_id) REFERENCES customer(customer_id)  
);
```

Order Item Table

```
CREATE TABLE order_item (  
    order_item_id INT PRIMARY KEY NOT NULL,  
    order_id INT,  
    sweet_id INT,  
    order_quantity INT,  
    order_price DECIMAL(10, 2),  
    FOREIGN KEY (order_id) REFERENCES orders(order_id),  
    FOREIGN KEY (sweet_id) REFERENCES sweet(sweet_id)  
);
```

Shopping Basket Table

```
CREATE TABLE shopping_basket (  
    basket_id INT PRIMARY KEY NOT NULL,  
    customer_id INT,  
    sweet_id INT,  
    date_added_basket DATE,  
    total_basket_amount DECIMAL(10, 2),  
    FOREIGN KEY (customer_id) REFERENCES customer(customer_id),  
    FOREIGN KEY (sweet_id) REFERENCES sweet(sweet_id)  
);
```

Shopping List Table

```
CREATE TABLE shopping_list (  
    list_id INT PRIMARY KEY NOT NULL,  
    customer_id INT,  
    sweet_id INT,  
    date_added_list DATE,  
    FOREIGN KEY (customer_id) REFERENCES customer(customer_id),  
    FOREIGN KEY (sweet_id) REFERENCES sweet(sweet_id)  
);
```

Stock Table

```
CREATE TABLE stock (  
    stock_id INT PRIMARY KEY NOT NULL,  
    sweet_id INT,  
    stock_quantity INT,  
    FOREIGN KEY (sweet_id) REFERENCES sweet(sweet_id)  
);
```

Special Order Table

```
CREATE TABLE special_order (  
    special_order_id INT PRIMARY KEY NOT NULL,  
    centre_id INT,  
    warehouse_id INT,  
    sweet_id INT,  
    quantity_special INT,  
    delivery_date_special DATE,  
    FOREIGN KEY (centre_id) REFERENCES distribution_centre(centre_id),  
    FOREIGN KEY (warehouse_id) REFERENCES warehouse(warehouse_id),  
    FOREIGN KEY (sweet_id) REFERENCES sweet(sweet_id)  
);
```

Standing Order Table

```
CREATE TABLE standing_order (  
    standing_order_id INT PRIMARY KEY NOT NULL,  
    centre_id INT,  
    warehouse_id INT,  
    sweet_id INT,  
    quantity_standing INT,  
    frequency VARCHAR(20),  
    delivery_days_standing VARCHAR(15) ,  
    FOREIGN KEY (centre_id) REFERENCES distribution_centre(centre_id),  
    FOREIGN KEY (warehouse_id) REFERENCES warehouse(warehouse_id),  
    FOREIGN KEY (sweet_id) REFERENCES sweet(sweet_id)  
);
```

Delivery Details Table

```
CREATE TABLE delivery_details (  
    delivery_id INT PRIMARY KEY NOT NULL,  
    order_id INT,  
    customer_id INT,  
    delivery_date DATE,  
    delivery_address VARCHAR(100),  
    payment_type VARCHAR(50),  
    FOREIGN KEY (order_id) REFERENCES orders(order_id),  
    FOREIGN KEY (customer_id) REFERENCES customer(customer_id)  
);
```

Offer Table

```
CREATE TABLE offer (  
    offer_id INT PRIMARY KEY NOT NULL,  
    sweet_id INT,  
    offer_type VARCHAR(100),  
    offer_value VARCHAR(50),  
    valid_from DATE,  
    valid_upto DATE,  
    FOREIGN KEY (sweet_id) REFERENCES sweet(sweet_id)  
);
```

Payment Table

```
CREATE TABLE payment (  
    payment_id NUMBER PRIMARY KEY NOT NULL,  
    customer_id NUMBER,  
    card_holder_name VARCHAR2(50) NOT NULL,  
    card_number VARCHAR2(20) NOT NULL,  
    expiry_date DATE NOT NULL,  
    cvv NUMBER NOT NULL,  
    FOREIGN KEY (customer_id) REFERENCES customer(customer_id)  
);
```


Special Order Sweet Table (Junction Table)

```
CREATE TABLE special_order_sweet (  
    special_order_sweet_id INT PRIMARY KEY NOT NULL,  
    special_order_id INT,  
    sweet_id INT,  
    FOREIGN KEY (special_order_id) REFERENCES special_order(special_order_id),  
    FOREIGN KEY (sweet_id) REFERENCES sweet(sweet_id)  
);
```

Standing Order Sweet Table (Junction Table)

```
CREATE TABLE standing_order_sweet (  
    standing_order_sweet_id INT PRIMARY KEY NOT NULL,  
    standing_order_id INT,  
    sweet_id INT,  
    FOREIGN KEY (standing_order_id) REFERENCES standing_order(standing_order_id),  
    FOREIGN KEY (sweet_id) REFERENCES sweet(sweet_id)  
);
```

Shopping List Sweet Table (Junction Table)

```
CREATE TABLE shopping_list_sweet (  
    shopping_list_sweet_id INT PRIMARY KEY NOT NULL,  
    list_id INT,  
    sweet_id INT,  
    FOREIGN KEY (list_id) REFERENCES shopping_list(list_id),  
    FOREIGN KEY (sweet_id) REFERENCES sweet(sweet_id)  
);
```

Shopping Basket Sweet Table (Junction Table)

```
CREATE TABLE shopping_basket_sweet (  
    shopping_basket_sweet_id INT PRIMARY KEY NOT NULL,  
    basket_id INT,  
    sweet_id INT,  
    FOREIGN KEY (basket_id) REFERENCES shopping_basket(basket_id),  
    FOREIGN KEY (sweet_id) REFERENCES sweet(sweet_id)  
);
```

CHAPTER 3: MULTIPLICITY OF THE RELATIONSHIP IN THE TABLES

- **Customer Table:** - A customer table consists of customer_id as primary key, username, password, customer_name, customer_address, customer_email, customer_phone. This table consists of 48 rows of data. Here all the data entered is unique and not repeated.

The screenshot shows the Oracle APEX SQL Workshop interface. The 'Object Browser' on the left lists various database objects, with 'CUSTOMER' selected under the 'Tables' category. The main pane displays the 'CUSTOMER' table data with columns: CUSTOMER_ID, USERNAME, PASSWORD, CUSTOMER_NAME, CUSTOMER_ADDRESS, CUSTOMER_EMAIL, and CUSTOMER_PHONE. The table contains 20 rows of data, with the first row selected. The interface includes tabs for Columns, Data, Indexes, Constraints, Grants, Statistics, Triggers, Dependencies, DDL, and Sample Queries. The bottom status bar indicates '1 cells selected' and 'Oracle APEX 23.2.4'.

CUSTOMER_ID	USERNAME	PASSWORD	CUSTOMER_NAME	CUSTOMER_ADDRESS	CUSTOMER_EMAIL	CUSTOMER_PHONE
2	user2	password2	Jane Smith	456 Elm St, Townsville	jane.smith@example.com	+987654321
3	user3	password3	Michael Johnson	789 Oak St, Villagetown	michael.johnson@example.com	+1357924680
4	user4	password4	Emily Brown	321 Maple St, Hamletville	emily.brown@example.com	+9876543210
5	user5	password5	David Lee	654 Pine St, Countryside	david.lee@example.com	+112223333
6	user6	password6	Sarah Garcia	987 Cedar St, Riverside	sarah.garcia@example.com	+4445556666
7	user7	password7	James Martinez	234 Birch St, Mountainview	james.martinez@example.com	+7778889999
8	user8	password8	Jessica Nguyen	567 Spruce St, Lakewood	jessica.nguyen@example.com	+2223334444
9	user9	password9	Daniel Robinson	890 Walnut St, Seaville	daniel.robinson@example.com	+8889990000
10	user10	password10	Lisa Walker	109 Pineapple St, Beachside	lisa.walker@example.com	+6667778888
11	user11	password11	Chris Taylor	321 Orange St, Hilltop	chris.taylor@example.com	+3334445555
12	user12	password12	Amanda Clark	543 Lemon St, Lakeshore	amanda.clark@example.com	+9990001111
13	user13	password13	Kevin Hernandez	765 Grape St, Riverfront	kevin.hernandez@example.com	+1231231234
14	user14	password14	Melissa Carter	987 Banana St, Countryside	melissa.carter@example.com	+4564564567
15	user15	password15	Brian Adams	210 Peach St, Orchard	brian.adams@example.com	+7897897890
16	user16	password16	Ashley Hall	432 Mango St, Valleyview	ashley.hall@example.com	+3213213210
17	user17	password17	Mark King	654 Kiwi St, Hillside	mark.king@example.com	+6546546540
18	user18	password18	Rachel Scott	876 Plum St, Riverside	rachel.scott@example.com	+9879879870
19	user19	password19	Steven Young	210 Avocado St, Lakeside	steven.young@example.com	+7417417410
20	user20	password20	Nicole Allen	543 Passion St, Beachfront	nicole.allen@example.com	+3693693690

- **Sweet Table:** - A sweet table consists of sweet_id as primary key, sweet_name, sweet_description, sweet_size, quantity, cost_per_pack, cost_per_sweet, sweet_category. These table contains total 44 rows of data.

The screenshot shows the Oracle APEX SQL Workshop interface. The 'Object Browser' on the left lists various database objects, with 'SWEET' selected under the 'Tables' category. The main pane displays the 'SWEET' table data with columns: SWEET_ID, SWEET_NAME, SWEET_DESCRIPTION, SWEET_SIZE, QUANTITY, COST_PER_PACK, COST_PER_SWEET, and SWEET_CATEGORY. The table contains 44 rows of data, with the first row selected. The interface includes tabs for Columns, Data, Indexes, Constraints, Grants, Statistics, Triggers, Dependencies, DDL, and Sample Queries. The bottom status bar indicates '1 cells selected' and 'Oracle APEX 23.2.4'.

SWEET_ID	SWEET_NAME	SWEET_DESCRIPTION	SWEET_SIZE	QUANTITY	COST_PER_PACK	COST_PER_SWEET	SWEET_CATEGORY
23456789	Gummy Bears	Assorted flavors	Small	200	3.49	.05	Gummies
34567890	Lollipop	Fruit flavored	Medium	150	1.99	.2	Lollipops
45678901	Caramel Candy	Buttery caramel	Medium	120	4.99	.45	Caramel
56789012	Jelly Beans	Various flavors	Large	180	2.99	.03	Gummies
67890123	Marshmallow	Soft and fluffy	Small	250	2.49	1	Marshmallows
78901234	Toffee	Hard toffee	Medium	180	3.99	.35	Toffee
89012345	Rock Candy	Crystallized sugar	Large	120	4.99	.5	Hard Candy
90123456	Cotton Candy	Fluffy spun sugar	Large	200	1.99	.35	Candy Floss
12341234	Bubble Gum	Bubble flavored	Medium	300	2.49	.02	Gum
23452345	Licorice	Black licorice	Small	180	2.99	.03	Licorice
34563456	Jawbreaker	Layered candy	Large	100	3.99	.04	Hard Candy
45674567	Sour Patch Kids	Sour candy	Medium	150	2.99	.08	Sour Candy
56785678	Fudge	Chocolate fudge	Medium	120	5.99	.5	Fudge
67896789	Hard Candy	Assorted flavors	Small	200	3.49	.05	Hard Candy
78907890	Almond Brittle	Crunchy almond	Medium	150	4.99	.4	Brittle
89018901	Gummy Worms	Fruit flavored	Large	180	2.99	.03	Gummies
90129012	Chocolate Covered Pretzel	Salty and sweet	Medium	250	2.49	.35	Chocolate
11223344	Mint Candy	Refreshing mint	Small	180	1.99	.02	Mints
22334455	Fruit Gushers	Fruit filled	Large	120	4.99	.04	Gummies

- **Region Table:** - A region table consists of region_id as primary key and region_name. There are total 12 regions.

The screenshot shows the Oracle APEX SQL Workshop interface. The 'Object Browser' on the left lists various database objects, with 'REGION' highlighted. The main panel displays the 'REGION' table data under the 'Data' tab. The table has two columns: 'REGION_ID' and 'REGION_NAME'. There are 12 rows of data representing different regions.

REGION_ID	REGION_NAME
10	Wales
12	Northern Ireland
2	North West
3	Yorkshire and The Humber
7	London
8	South East
1	North East
4	East Midlands
5	West Midlands
6	East of England
9	South West
11	Scotland

- **Warehouse Table:** - A warehouse consists of only one and one region whereas a region can have many warehouses [region (1..1)----(1..*) warehouse]

The given figure demonstrates that how region_id 4 and 5 contains multiple warehouse_id which are 3,15 and 4,16.

The screenshot shows the Oracle APEX SQL Workshop interface. The 'Object Browser' on the left lists various database objects, with 'WAREHOUSE' highlighted. The main panel displays the 'WAREHOUSE' table data under the 'Data' tab. The table has seven columns: 'WAREHOUSE_ID', 'REGION_ID', 'WAREHOUSE_NAME', 'WAREHOUSE_ADDRESS', 'WAREHOUSE_POSTCODE', and 'WAREHOUSE_PHONE_NO'. There are 24 rows of data representing different warehouses.

WAREHOUSE_ID	REGION_ID	WAREHOUSE_NAME	WAREHOUSE_ADDRESS	WAREHOUSE_POSTCODE	WAREHOUSE_PHONE_NO
1	2	Sugar Haven Depot	42 Marlborough Road, Breaston	B20 0BB	234-567-8901
2	3	Sweet Bliss Storage	9336 Highfield Road GLASGOW	C30 0CC	345-678-9012
3	4	Candy Cove Warehouse	9688 Victoria Road CHELMSFORD	D40 0DD	456-789-0123
4	5	Honey Haven Distribution	9331 Manor Road COVENTRY	E50 0EE	567-890-1234
5	6	Sugarcane Sanctuary	191 Richmond Road NORTH WEST OXF...	F60 0FF	678-901-2345
6	7	Marshmallow Meadows Warehouse	38 Chester Road TWICKENHAM	G70 0GG	789-012-3456
7	8	Toffee Treasure Trove	87 Park Road BOLTON	H80 0HH	890-123-4567
8	9	Lollipop Land Storage	897 Queensway KIRK WALL	I90 0II	901-234-5678
9	10	Caramel Cove Depot	89 Victoria Street NOTTINGHAM	J10 0JJ	012-345-6789
10	11	Fudge Fantasy Warehouse	5 New Street MANCHESTER	K20 0KK	123-456-7890
11	12	Gummy Grove Distribution Centre	35 Green Lane LEICESTER	L30 0LL	234-567-8901
12	1	Confectionery Corner Warehouse	87 Park Lane STOKE-ON-TRENT	M40 0MM	345-678-9012
15	4	Candyland Warehouse	9861 Broadway BATH	BA72 7EY	555-234-5678
16	5	Jelly Junction Distribution	9379 Grove Road SALISBURY	SP88 3NQ	555-678-9012
17	6	Chocolate Haven Depot	63 Kingsway SHREWSBUR	SY4 4KU	555-345-6789
20	9	Taffy Terrace Depot	99 Main Road TAUNTON	TA87 0ZR	555-654-3210
19	8	Sugary Skies Warehouse	75 Main Street MOTHERWELL	ML99 6ZY	555-456-7890
21	10	Cookie Crate Distribution	84 School Lane WIGAN	WN10 4TD	555-567-8901
24	12	Decadent Delight Dessert	9235 Grove Road SHIFFIELD	S40 3LS	555-789-0123

- **Distribution Centre Table:** - A distribution centre consists of only one and one warehouse whereas a warehouse can have many distributions centre [warehouse (1..1)----(1..*) distribution_centre]

The given figure demonstrates that how warehouse_id 3 and 6 contains more than one centre_id which are 4, 5 and 8,9.

The screenshot shows the Oracle APEX SQL Workshop interface. The 'Object Browser' on the left lists various tables, with 'DISTRIBUTION_CENTRE' selected. The main area displays the table's data in a grid format. The columns are: CENTRE_ID, WAREHOUSE_ID, CENTRE_NAME, CENTRE_ADDRESS, CENTRE_POSTCODE, and CENTRE_PHONE_NO. The data shows that warehouse_id 3 has three associated centres (4, 5, 8, 9) and warehouse_id 6 has two (4, 5).

CENTRE_ID	WAREHOUSE_ID	CENTRE_NAME	CENTRE_ADDRESS	CENTRE_POSTCODE	CENTRE_PHONE_NO
2	2	Nexus Logistics Hub	456 Elm St	DE4 F56	234-567-8921
3	2	PrimePoint Distribution Center	789 Oak St	GH7 89	335-678-9012
4	3	Horizon Logistics Center	101 Pine St	JK1 L01	456-789-0123
5	3	MetroConnect Distribution Hub	202 Maple St	MN2 O02	527-890-1234
6	4	GlobalLink Logistics Center	303 Cedar St	PQ3 R03	628-401-2345
7	5	CityPulse Distribution Hub	404 Birch St	ST4 U04	890-423-4567
8	6	SummitLogistics Center	505 Cedar St	VW5 X05	901-234-6478
9	6	SkyBridge Distribution Center	606 Pine St	YZ6 A06	022-345-6789
10	7	ExpressWay Logistics Hub	707 Maple St	BC7 D07	123-456-7890
11	7	SpeedZone Logistics Hub	808 Oak St	EF8 G08	234-567-1241
12	8	CentralCrest Distribution Center	909 Elm St	H9 J09	322-678-9012
13	9	UrbanSwift Logistics Center	1010 Walnut St	KL10 M10	456-789-0123
14	10	NexusPoint Distribution Hub	1111 Pine St	NO11 P11	537-890-1234
15	11	MetroLink Logistics Hub	1212 Maple St	QR12 S12	678-951-2345
16	11	GlobalGate Distribution Center	1313 Oak St	TU13 V13	756-012-3456
17	12	CityEdge Logistics Center	1414 Elm St	WX14 Y14	890-433-4567
18	13	SummitStream Distribution Hub	1515 Cedar St	YZ15 A15	401-234-5678
19	14	Skyline Logistics Center	1616 Birch St	BC16 D16	012-345-2459
20	15	FrontierLink Distribution Center	1717 Pine St	FF17 G17	121-456-7890

- **Orders Table:** - An order_id consists of only one and one customer whereas a customer can place zero or multiple orders [orders (0..*)----(1..1) customer]

The given figure demonstrates that how customer_id 24 and 2 contains many orders which are order_id 1000069,1000075 and 1000077,1000002.

The screenshot shows the Oracle APEX SQL Workshop interface. The 'Object Browser' on the left lists various tables, with 'ORDERS' selected. The main area displays the table's data in a grid format. The columns are: ORDER_ID, CUSTOMER_ID, ORDER_DATE, and TOTAL_AMOUNT. The data shows that customer_id 24 has three orders (1000069, 1000075, 1000077) and customer_id 2 has two orders (1000002, 1000003).

ORDER_ID	CUSTOMER_ID	ORDER_DATE	TOTAL_AMOUNT
1000069	24	01/07/2024	70
1000075	24	12/18/2023	120
1000076	27	12/29/2023	70
1000063	35	02/12/2024	560
1000064	11	02/14/2024	50
1000066	16	01/10/2024	90.35
1000068	12	01/04/2024	120
1000071	23	12/23/2023	40
1000078	34	12/10/2023	440.2
1000062	41	02/23/2024	220
1000065	12	02/19/2024	220
1000067	27	01/23/2024	80
1000072	45	12/31/2023	220
1000073	32	12/20/2023	200
1000077	2	12/30/2023	400.3
1000080	26	12/25/2023	230
100002	2	02/12/2024	75.5
100003	3	02/13/2024	100
100004	4	02/14/2024	80

- **Order Item Table:** - An order_item_id consists of only one and one sweet_id whereas a sweet_id can be used in zero or multiple order items [order_item (0..1)----(1..1) sweet]

The given figure demonstrates that how sweet_id 90123456 and 23452345 appears in many order_item_id which are 10010, 10011, 10012, 10013, 10018 and 10016, 10017, 10020.

The screenshot shows the Oracle APEX SQL Workshop interface. The 'Object Browser' on the left lists various database objects, with 'ORDER_ITEM' selected. The main panel displays the 'ORDER_ITEM' table data with columns: ORDER_ITEM_ID, ORDER_ID, SWEET_ID, ORDER_QUANTITY, and ORDER_PRICE. The table contains 20 rows of data. The 'SWEET_ID' column shows values 23456789, 34567890, 45678901, 56789012, 67890123, 78901234, 89012345, 90123456, 12341234, 12341234, 23452345, 23452345, 90123456, 12341234, 23452345, 90123456, 12341234, 23452345, 90123456, and 90123456. The 'ORDER_ITEM_ID' column shows values 10001, 10002, 10003, 10004, 10005, 10006, 10008, 10010, 10011, 10012, 10013, 10014, 10015, 10016, 10017, 10018, 10019, 10020, and 10021.

ORDER_ITEM_ID	ORDER_ID	SWEET_ID	ORDER_QUANTITY	ORDER_PRICE
10001	10002	23456789	2	10
10002	10002	34567890	1	15
10003	10002	45678901	3	7
10004	10002	56789012	2	12
10005	10003	67890123	1	20
10006	10003	78901234	4	5
10008	10004	89012345	1	30
10010	10005	90123456	2	18
10011	10006	90123456	1	15
10012	10006	90123456	2	10
10013	10007	90123456	3	8
10014	10007	12341234	1	25
10015	10008	12341234	4	6
10016	10008	23452345	2	12
10017	10009	23452345	1	20
10018	10009	90123456	3	7
10019	10030	12341234	2	18
10020	10030	23452345	1	30
10021	10031	90123456	3	10

- **Shopping Basket Table:** - A shopping basket consists of only one and one customer_id and a customer can have only one shopping basket [shopping_basket (1..1)----(1..1) customer]

The given figure demonstrates that basket_id for every customer_id is different.

The screenshot shows the Oracle APEX SQL Workshop interface. The 'Object Browser' on the left lists various database objects, with 'SHOPPING_BASKET' selected. The main panel displays the 'SHOPPING_BASKET' table data with columns: BASKET_ID, CUSTOMER_ID, SWEET_ID, DATE_ADDED_BASKET, and TOTAL_BASKET_AMOUNT. The table contains 20 rows of data. The 'CUSTOMER_ID' column shows values 2, 3, 4, 5, 6, 7, 18, 39, 10, 11, 22, 33, 34, 45, 26, 47, 8, 19, and 20. The 'BASKET_ID' column shows values 10002, 10003, 10004, 10005, 10006, 10007, 10008, 10009, 10010, 10011, 10012, 10013, 10014, 10015, 10016, 10017, 10018, 10019, and 10020.

BASKET_ID	CUSTOMER_ID	SWEET_ID	DATE_ADDED_BASKET	TOTAL_BASKET_AMOUNT
10002	2	34567890	03/12/2024	15
10003	3	45678901	03/12/2024	30
10004	4	67890123	03/12/2024	20
10005	5	78901234	03/12/2024	40
10006	6	89012345	03/12/2024	35
10007	7	90123456	03/12/2024	18
10008	18	12341234	03/12/2024	28
10009	39	23452345	03/12/2024	22
10010	10	234415	03/12/2024	26
10011	11	234415	03/12/2024	52
10012	22	12341234	03/12/2024	27
10013	33	234415	03/12/2024	23
10014	34	12341234	03/12/2024	19
10015	45	234415	03/12/2024	21
10016	26	12341234	03/12/2024	36
10017	47	234415	03/12/2024	24
10018	8	234415	03/12/2024	29
10019	19	77889910	03/12/2024	31
10020	20	234415	03/12/2024	33

- **Shopping List Table:** - A shopping list consists of only one and one customer_id and a customer can have only one shopping list [shopping_list (1..1)----->(1..1) customer]

The given figure demonstrates that list_id for every customer_id is different, and both are not repeated.

	LIST_ID	CUSTOMER_ID	SWEET_ID	DATE_ADDED_LIST
	1000002	2	34567890	03/11/2024
	1000003	3	45678901	03/10/2024
	1000004	4	67890123	03/01/2024
	1000005	5	78901234	02/12/2024
	1000006	6	89012345	02/12/2024
	1000007	7	90123456	02/22/2024
	1000008	8	12341234	02/13/2024
	1000009	9	23452345	02/14/2024
	1000010	10	233445	02/15/2024
	1000011	11	233445	02/16/2024
	1000012	12	12341234	02/12/2024
	1000013	13	233445	02/12/2024
	1000014	14	12341234	02/12/2024
	1000015	15	233445	02/12/2024
	1000016	16	12341234	03/12/2024
	1000017	17	233445	03/12/2024
	1000018	18	233445	03/12/2024
	1000019	19	77889910	03/12/2024
	1000020	20	2334455	03/12/2024

- **Stock Table:** - A stock_id belongs to only one sweet_id and a sweet_id addresses only one stock_id. [stock (1..1)----->(1..1) sweet]

The given figure demonstrates that stock_id for every sweet_id is different, and both are not repeated.

	STOCK_ID	SWEET_ID	STOCK_QUANTITY
	1	23456789	100
	2	34567890	150
	3	45678901	80
	4	56789012	200
	5	67890123	120
	6	78901234	90
	7	89012345	300
	8	90123456	50
	9	12341234	180
	10	23452345	220
	11	34563456	75
	12	45674567	250
	13	56785678	110
	14	67896789	160
	15	78907890	130
	16	89018901	70
	17	90129012	190
	18	11223344	240
	19	22334455	85

- **Special Order Table:** - A special order can be placed to only one warehouse whereas a warehouse can have multiple special orders. [special_order (0..*)----- (1..1) warehouse]

The given figure demonstrates that different special_order_id like 4, 5, 6 are placed to same warehouse_id that is 2.

SPECIAL_ORDER_ID	CENTRE_ID	WAREHOUSE_ID	SWEET_ID	QUANTITY_SPECIAL	DELIVERY_DATE_SPECIAL
2	2	1	23456789	75	03/16/2024
3	3	1	34567890	100	03/17/2024
4	4	2	45678901	125	03/18/2024
5	5	2	56789012	150	03/19/2024
6	6	2	67890123	175	03/20/2024
7	7	3	78901234	200	03/21/2024
8	8	3	89012345	225	03/22/2024
9	9	4	90123456	250	03/23/2024
10	10	4	12341234	275	03/24/2024
11	11	5	23452345	300	03/25/2024
12	12	6	34563456	325	03/26/2024
13	13	7	45674567	350	03/27/2024
14	14	7	56785678	375	03/28/2024
15	15	8	67896789	400	03/29/2024
16	16	9	78907890	425	03/30/2024
17	17	10	89018901	450	03/31/2024
18	18	11	90129012	475	04/01/2024
19	19	12	11223344	500	04/02/2024
20	20	12	22334455	525	04/03/2024

- **Standing Order Table:** - A standing order can be placed to only one warehouse whereas a warehouse can have multiple standing orders. [standing_order (0..*)----- (1..1) warehouse]

The given figure demonstrates that different standing_order_id like 19, 20 are placed to same warehouse_id that is 12.

STANDING_ORDER_ID	CENTRE_ID	WAREHOUSE_ID	SWEET_ID	QUANTITY_STANDING	FREQUENCY	DELIVERY_DAYS_STANDING
2	2	1	7889900	75	Weekly	Monday
3	3	1	34567890	100	Weekly	Monday
4	4	2	2334445	125	Monthly	Wednesday
5	5	2	7889900	150	Weekly	Monday
6	6	2	67890123	175	Monthly	Wednesday
7	7	3	2334445	200	Monthly	Tuesday
8	8	3	2334445	225	Weekly	Monday
9	9	4	66778819	250	Weekly	Wednesday
10	10	4	12341234	275	Monthly	Tuesday
11	11	5	23452345	300	Weekly	Tuesday
12	12	6	7889900	325	Monthly	Monday
13	13	7	45674567	350	Monthly	Tuesday
14	14	7	66778819	375	Weekly	Wednesday
15	15	8	67896789	400	Weekly	Monday
16	16	9	2334445	425	Monthly	Wednesday
17	17	10	89018901	450	Weekly	Monday
18	18	11	2334445	475	Monthly	Wednesday
19	19	12	7889900	500	Weekly	Saturday
20	20	12	22334455	525	Weekly	Wednesday

- **Delivery Details Table:** - Here a delivery_id can be assigned to only and only one customer and a customer can store only one delivery details at a time i.e., customer can save only one delivery address.[delivery_details(1..1)----(1..1)customer].

In the diagram below every delivery_id is assigned to different customer_id. Both are unique for each data.

DELIVERY_ID	ORDER_ID	CUSTOMER_ID	DELIVERY_DATE	DELIVERY_ADDRESS
1	100003	2	04/02/2024	444 Maple St
2	100005	3	02/04/2024	789 Oak St, Villagetown
3	100003	4	02/05/2024	321 Maple St, Hamletville
4	100003	5	02/06/2024	654 Pine St, Countryside
5	100003	6	02/07/2024	987 Cedar St, Riverside
6	100003	7	02/08/2024	234 Birch St, Mountainview
7	100003	8	02/09/2024	567 Spruce St, Lakeside
8	100003	9	02/10/2024	890 Walnut St, Seaville
9	100003	10	02/11/2024	109 Pineapple St, Beachside
10	100003	11	02/12/2024	555 Walnut St
11	100003	12	02/13/2024	666 Cedar St
12	100003	13	02/14/2024	777 Spruce St
13	100003	14	02/15/2024	888 Pine St
14	100003	15	02/16/2024	999 Oak St
15	100003	16	02/17/2024	111 Elm St
16	100003	17	02/18/2024	432 Mango St, Valleyview
17	100003	18	02/19/2024	654 Kiwi St, Hillside
18	100003	19	02/20/2024	444 Walnut St
19	100003	20	02/21/2024	555 Cedar St

- **Offer Table:** - An offer can be placed to only one sweet and a sweet can have zero or one offers. [offer (0..1)----(1..1) sweet]

The given figure demonstrates that different offer_id are applied to different sweet_id. Here both sweet_id and offer_id are unique.

OFFER_ID	SWEET_ID	OFFER_TYPE	OFFER_VALUE	VALID_FROM	VALID_UPTO
1	23456789	Discount	10%	03/01/2024	03/10/2024
2	56789012	BOGO	Buy One Get One Free	03/06/2024	03/12/2024
3	12341234	Discount	15%	03/05/2024	03/12/2024
4	67896789	Discount	20%	03/22/2024	03/29/2024
5	22334455	BOGO	Buy One Get One Free	03/23/2024	03/30/2024
6	77889900	Discount	25%	03/14/2024	03/21/2024
7	4556677	BOGO	Buy One Get One Free	03/15/2024	03/22/2024
8	1223344	Discount	30%	03/16/2024	03/23/2024
9	44556677	BOGO	Buy One Get One Free	03/13/2024	03/23/2024
11	33445566	BOGO	Buy One Get One Free	03/10/2024	03/20/2024

- **Payment Table:** - Here a customer can store only zero or one payment detail and one payment detail can belong to only one customer [payment(0..1)----(1..1)customer].

The screenshot shows the Oracle APEX SQL Workshop interface. The left sidebar contains an 'Object Browser' with a search bar and a list of database objects. The 'PAYMENT' table is selected and highlighted in green. The main area displays the 'Data' tab for the 'PAYMENT' table, showing a grid of 20 rows. The columns are: PAYMENT_ID, CUSTOMER_ID, CARD_HOLDER_NAME, CARD_NUMBER, EXPIRY_DATE, and CVV. The data shows various payment records for different customers.

PAYMENT_ID	CUSTOMER_ID	CARD_HOLDER_NAME	CARD_NUMBER	EXPIRY_DATE	CVV
1	2	Jane Smith	1234567890123456	12/31/2024	123
2	3	Michael Johnson	9876543210987654	09/30/2023	456
3	4	Emily Brown	4567890123456789	06/30/2025	789
4	5	David Lee	5555666677778888	10/31/2025	246
5	6	Sarah Garcia	2222333344445555	04/30/2025	135
6	7	James Martinez	9999888877776666	08/31/2024	987
7	8	Jessica Nguyen	4567412871100151	08/31/2024	871
8	9	Daniel Robinson	7072125879750114774	02/28/2025	167
9	10	Lisa Walker	4818846897532814	01/31/2026	072
10	11	Chris Taylor	63194394629975906	08/11/2026	265
11	12	Amanda Clark	1125053801921035359	05/21/2026	465
12	13	Kevin Hernandez	37005356467600546602	09/21/2026	356
13	14	Melissa Carter	1234651671038724138	10/23/2026	732
14	15	Brian Adams	1145636684055842343	11/25/2027	237
15	16	Ashley Hall	3616340727402184706	11/12/2028	100
16	17	Mark King	5286452918764152	10/21/2027	987
17	18	Rachel Scott	6243249851264467553	02/11/2028	889
18	19	Steven Young	4032182948422753	03/31/2025	911
19	20	Nicole Allen	55677467810108581	04/30/2025	077

- **Special Order Sweet (Junction Table):** - This is junction table 2 many-to-many relation tables [special_order(0..*)----(1..*)sweet]. This table consists primary key of both the tables special_order as well as sweet.

The screenshot shows the Oracle APEX SQL Workshop interface. The left sidebar contains an 'Object Browser' with a search bar and a list of database objects. The 'SPECIAL_ORDER_SWEET' table is selected and highlighted in green. The main area displays the 'Data' tab for the 'SPECIAL_ORDER_SWEET' table, showing a grid of 20 rows. The columns are: SPECIAL_ORDER_SWEET_ID, SPECIAL_ORDER_ID, and SWEET_ID. The data shows various junction records between special orders and sweets.

SPECIAL_ORDER_SWEET_ID	SPECIAL_ORDER_ID	SWEET_ID
8	9	90123456
12	13	45674567
17	18	90129012
18	19	11223344
20	21	33445566
24	25	77889900
25	26	88990011
29	30	45566777
6	7	78901234
10	11	23456789
11	12	34567890
15	16	78907890
16	17	89018901
19	20	22334455
22	23	55667788
26	27	99001122
28	29	34455666
32	33	89900011
34	35	17777777

- **Standing Order Sweet (Junction Table):** - This is junction table 2 many-to-many relation tables [standing_order(0.*)----(1.*)sweet]. This table consists primary key of both the tables standing_order as well as sweet.

STANDING_ORDER_SWEET_ID	STANDING_ORDER_ID	SWEET_ID
1	2	7889900
2	3	34567890
3	4	2334415
4	5	7889900
5	6	67890123
6	7	2334415
7	8	2334415
8	9	66778819
9	10	12341234
10	11	23452345
11	12	7889900
12	13	45678567
13	14	66778819
14	15	67896789
15	16	2334415
16	17	89018901
17	18	2334415
18	19	7889900
19	20	27854455

- **Shopping List Sweet (Junction Table):** - This is junction table 2 many-to-many relation tables [shopping_list(0.*)----(0.*)sweet]. This table consists primary key of both the tables shopping_list as well as sweet.

SHOPPING_LIST_SWEET_ID	LIST_ID	SWEET_ID	
1	1000002	34567890	
2	1000003	45678901	
3	1000004	67890123	
4	1000005	78901234	
5	1000006	89012345	
6	1000007	90123456	
7	1000008	12341234	
8	1000009	23452345	
9	1000010	2334415	
10	1000011	2334415	
11	1000012	12341234	
12	1000013	2334415	
13	1000014	12341234	
14	1000015	2334415	
15	1000016	12341234	
16	1000017	2334415	
17	1000018	2334415	
18	1000019	77889910	
19	1000020	7334415	

- **Shopping Basket Sweet (Junction Table):** - This is junction table 2 many-to-many relation tables [shopping_basket(0.*)----(0.*)sweet]. This table consists primary key of both the tables shopping_basket as well as sweet.

SHOPSING_BASKET_SWEET_ID	BASKET_ID	SWEET_ID
2	10002	34567890
3	10003	45678901
4	10004	67890123
5	10005	78901234
6	10006	89012345
7	10007	90123456
8	10008	12341234
9	10009	23452345
10	10010	2334415
11	10011	2334415
12	10012	12341234
13	10013	2334415
14	10014	12341234
15	10015	2334415
16	10016	12341234
17	10017	2334415
18	10018	2334415
19	10019	77889910
20	10020	7334415

CHAPTER 4: SQL QUERIES

Query 1) This query calculates delivery charge based on the order's amount and displays delivery fee and total amount after adding delivery fee. It retrieves information about each customer's id, customer's name, order's id, order's amount who has ordered.

SELECT

customer.customer_id, customer.customer_name,

order_id,

SUM(total_amount) AS total_amount,

CASE

WHEN SUM(total_amount) < 50 THEN 5.00

WHEN SUM(total_amount) >= 50 AND SUM(total_amount) < 100 THEN 10.00

ELSE 0

END AS delivery_charge,

SUM(total_amount) +

CASE

WHEN SUM(total_amount) < 50 THEN 5.00

WHEN SUM(total_amount) >= 50 AND SUM(total_amount) < 100 THEN 10.00

```

ELSE 0

END AS total_amount_with_delivery_charge

FROM

customer

JOIN

orders ON orders.customer_id = customer.customer_id

GROUP BY

customer.customer_id, customer.customer_name, order_id

```

Query: -

APEX

App Builder

SQL Workshop

Team Development

Gallery

Search

JD Jay Dugad

k2544616

SQL Commands

Language

SQL

Rows

100

Clear Command

Find Tables

Save

Run

```

1 SELECT
2   customer.customer_id,
3   customer.customer_name,
4   order_id,
5   SUM(total_amount) AS total_amount,
6   CASE
7     WHEN SUM(total_amount) < 50 THEN 5.00
8     WHEN SUM(total_amount) >= 50 AND SUM(total_amount) < 100 THEN 10.00
9     ELSE 0
10  END AS delivery_charge,
11  SUM(total_amount) +
12  CASE
13    WHEN SUM(total_amount) < 50 THEN 5.00
14    WHEN SUM(total_amount) >= 50 AND SUM(total_amount) < 100 THEN 10.00
15    ELSE 0
16  END AS total_amount_with_delivery_charge
17 FROM
18   customer
19 JOIN
20   orders ON orders.customer_id = customer.customer_id
21 GROUP BY
22   customer.customer_id,
23   customer.customer_name,
24   order_id;
25

```

Results

Explain

Describe

Saved SQL

History

CUSTOMER_ID	CUSTOMER_NAME	ORDER_ID	TOTAL_AMOUNT	DELIVERY_CHARGE	TOTAL_AMOUNT_WITH_DELIVERY_CHARGE
43	Tiffany Perez	1000072	220	0	220
16	Ashley Hall	100015	95	10	105
40	Justin Scott	100053	80	10	90
26	Stephanie Turner	100055	135	0	135

k2544616@kingston.ac.uk

k2544616

en

Copyright © 1999, 2023, Oracle and/or its affiliates.

Oracle APEX 23.2.4

Output: -

Results

Explain

Describe

Saved SQL

History

CUSTOMER_ID	CUSTOMER_NAME	ORDER_ID	TOTAL_AMOUNT	DELIVERY_CHARGE	TOTAL_AMOUNT_WITH_DELIVERY_CHARGE
43	Tiffany Perez	1000072	220	0	220
16	Ashley Hall	100015	95	10	105
40	Justin Scott	100053	80	10	90
26	Stephanie Turner	100055	135	0	135
21	Matthew Wright	100160	115	0	115
14	Melissa Carter	1000079	220	0	220
32	Christina White	1000073	200	0	200
3	Michael Johnson	100003	100	0	100
9	Daniel Robinson	100009	120	0	120
29	Patrick Lee	100018	130	0	130
10	Lisa Walker	100019	60	10	70
33	Joseph Garcia	100022	120	0	120
7	James Martinez	100026	170	0	170
9	Daniel Robinson	100028	140.3	0	140.3
49	Vanessa Wright	100035	115	0	115
5	David Lee	100044	280	0	280
10	Lisa Walker	100061	140	0	140
41	Angela Baker	1000062	220	0	220

k2544616@kingston.ac.uk

k2544616

en

Copyright © 1999, 2023, Oracle and/or its affiliates.

Oracle APEX 23.2.4

Query 2) This query displays the sweets which were in shopping list but were not in the shopping basket while making an order.

SELECT

```
shopping_list.sweet_id,  
sweet.sweet_name,  
sweet.sweet_description,  
customer.customer_id,  
customer.customer_name
```

FROM

```
shopping_list
```

LEFT JOIN

```
shopping_basket ON shopping_list.sweet_id = shopping_basket.sweet_id
```

JOIN

```
sweet ON shopping_list.sweet_id = sweet.sweet_id
```

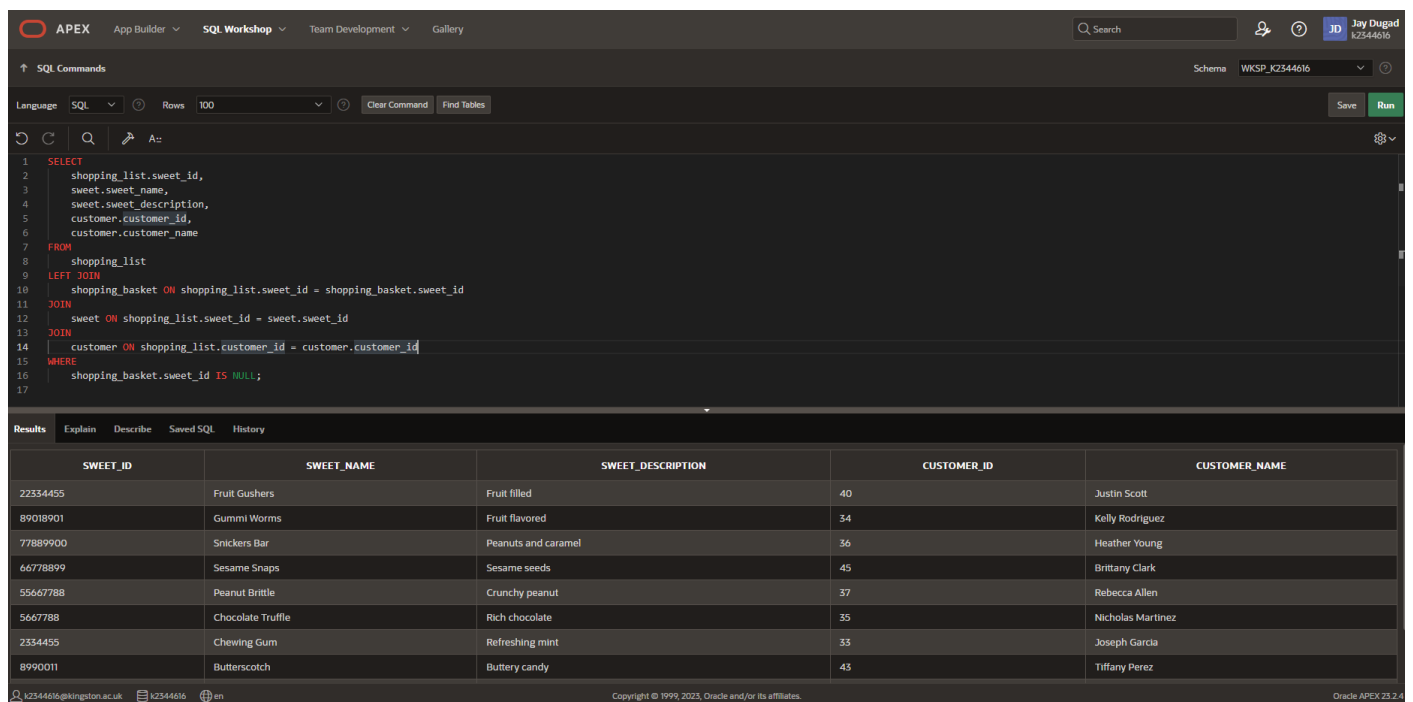
JOIN

```
customer ON shopping_list.customer_id = customer.customer_id
```

WHERE

```
shopping_basket.sweet_id IS NULL;
```

Output: -



The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command window contains the following query:

```
1 SELECT  
2   shopping_list.sweet_id,  
3   sweet.sweet_name,  
4   sweet.sweet_description,  
5   customer.customer_id,  
6   customer.customer_name  
7 FROM  
8   shopping_list  
9 LEFT JOIN  
10  shopping_basket ON shopping_list.sweet_id = shopping_basket.sweet_id  
11 JOIN  
12  sweet ON shopping_list.sweet_id = sweet.sweet_id  
13 JOIN  
14  customer ON shopping_list.customer_id = customer.customer_id  
15 WHERE  
16  shopping_basket.sweet_id IS NULL;  
17
```

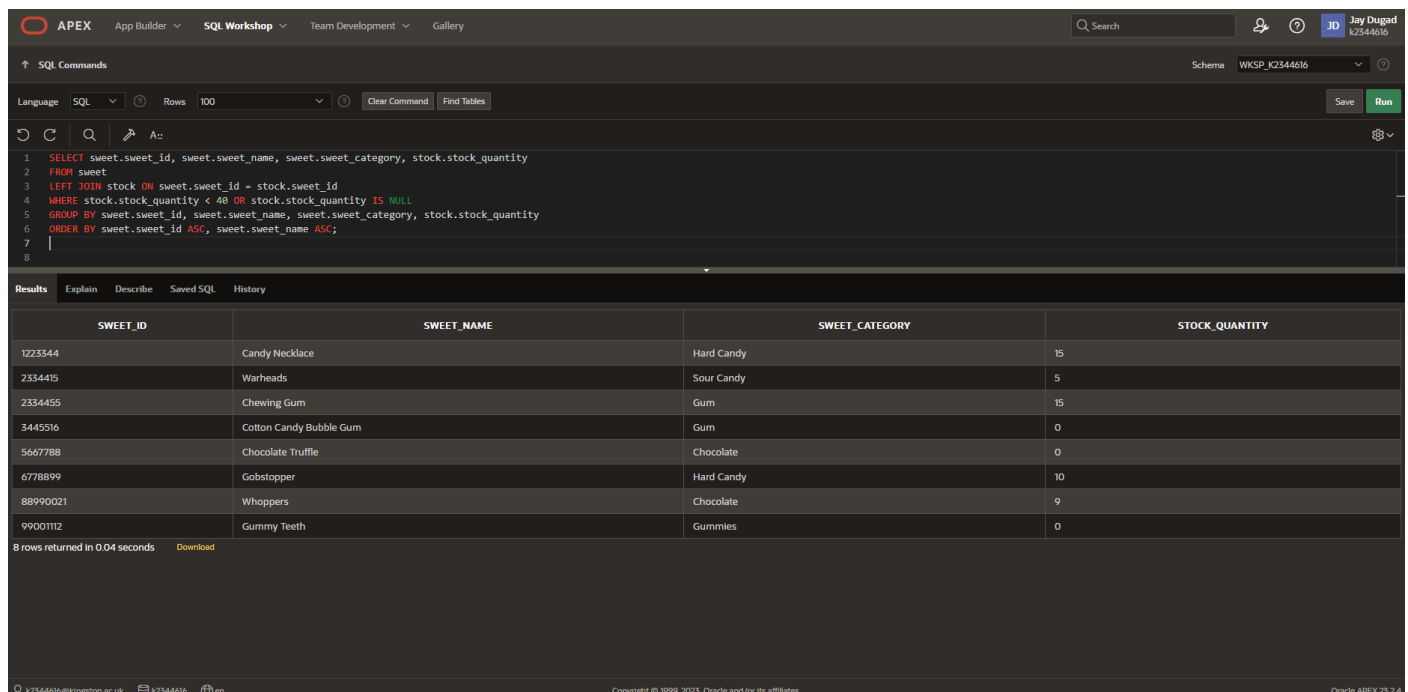
The Results window displays the following data:

SWEET_ID	SWEET_NAME	SWEET_DESCRIPTION	CUSTOMER_ID	CUSTOMER_NAME
22334455	Fruit Gushers	Fruit filled	40	Justin Scott
89018901	Gummi Worms	Fruit flavored	34	Kelly Rodriguez
77889900	Snickers Bar	Peanuts and caramel	36	Heather Young
66778899	Sesame Snaps	Sesame seeds	45	Brittany Clark
55667788	Peanut Brittle	Crunchy peanut	37	Rebecca Allen
5667788	Chocolate Truffle	Rich chocolate	35	Nicholas Martinez
2334455	Chewing Gum	Refreshing mint	33	Joseph Garcia
8990011	Butterscotch	Buttery candy	48	Tiffany Perez

Query 3) Retrieves sweet id, sweet name, sweet category and stock quantity who are soon going to be out of stock.

```
SELECT sweet.sweet_id, sweet.sweet_name, sweet.sweet_category, stock.stock_quantity
FROM sweet
LEFT JOIN stock ON sweet.sweet_id = stock.sweet_id
WHERE stock.stock_quantity < 40 OR stock.stock_quantity IS NULL
GROUP BY sweet.sweet_id, sweet.sweet_name, sweet.sweet_category, stock.stock_quantity
ORDER BY sweet.sweet_id ASC;
```

Output: -



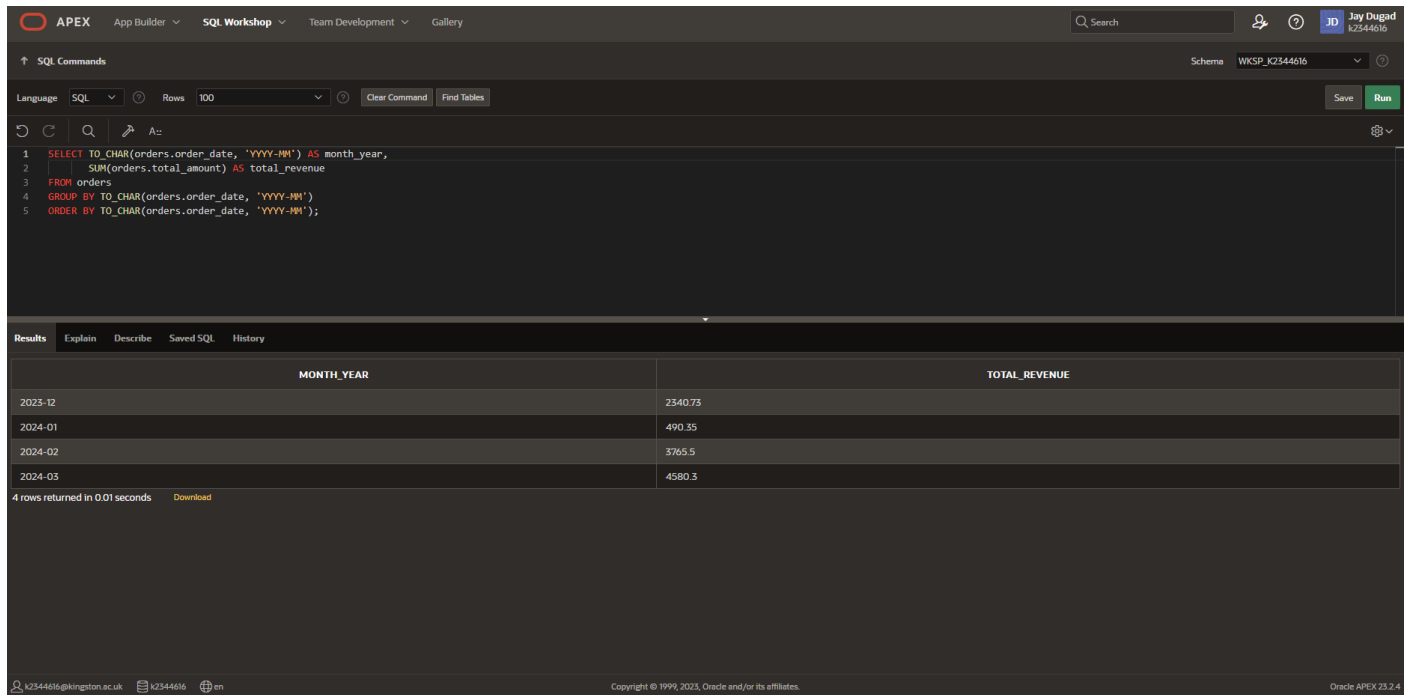
The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command is entered in the editor and executed. The results are displayed in a table with 4 columns: SWEET_ID, SWEET_NAME, SWEET_CATEGORY, and STOCK_QUANTITY. The table contains 8 rows of data.

SWEET_ID	SWEET_NAME	SWEET_CATEGORY	STOCK_QUANTITY
1223344	Candy Necklace	Hard Candy	15
2334415	Warheads	Sour Candy	5
2334455	Chewing Gum	Gum	15
3445516	Cotton Candy Bubble Gum	Gum	0
5667788	Chocolate Truffle	Chocolate	0
6778899	Gobstopper	Hard Candy	10
88990021	Whoppers	Chocolate	9
99001112	Gummy Teeth	Gummies	0

Query 4) This query generates and displays the total revenue of every month and year by using total amount of all the orders.

```
SELECT TO_CHAR(orders.order_date, 'YYYY-MM') AS month_year,
       SUM(orders.total_amount) AS total_revenue
FROM orders
GROUP BY TO_CHAR(orders.order_date, 'YYYY-MM')
ORDER BY TO_CHAR(orders.order_date, 'YYYY-MM');
```

Output: -



The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command area contains the following query:

```
1 SELECT TO_CHAR(orders.order_date, 'YYYY-MM') AS month_year,
2        SUM(orders.total_amount) AS total_revenue
3 FROM orders
4 GROUP BY TO_CHAR(orders.order_date, 'YYYY-MM')
5 ORDER BY TO_CHAR(orders.order_date, 'YYYY-MM');
```

The Results tab displays the following data:

MONTH_YEAR	TOTAL_REVENUE
2023-12	2340.73
2024-01	490.35
2024-02	3765.5
2024-03	4580.3

4 rows returned in 0.01 seconds

Query 5) This query identifies the top 5 best-selling sweet category.

SELECT sweet_id, sweet_category, total_orders

FROM (

SELECT sweet.sweet_id, sweet.sweet_category, COUNT(*) AS total_orders

FROM order_item

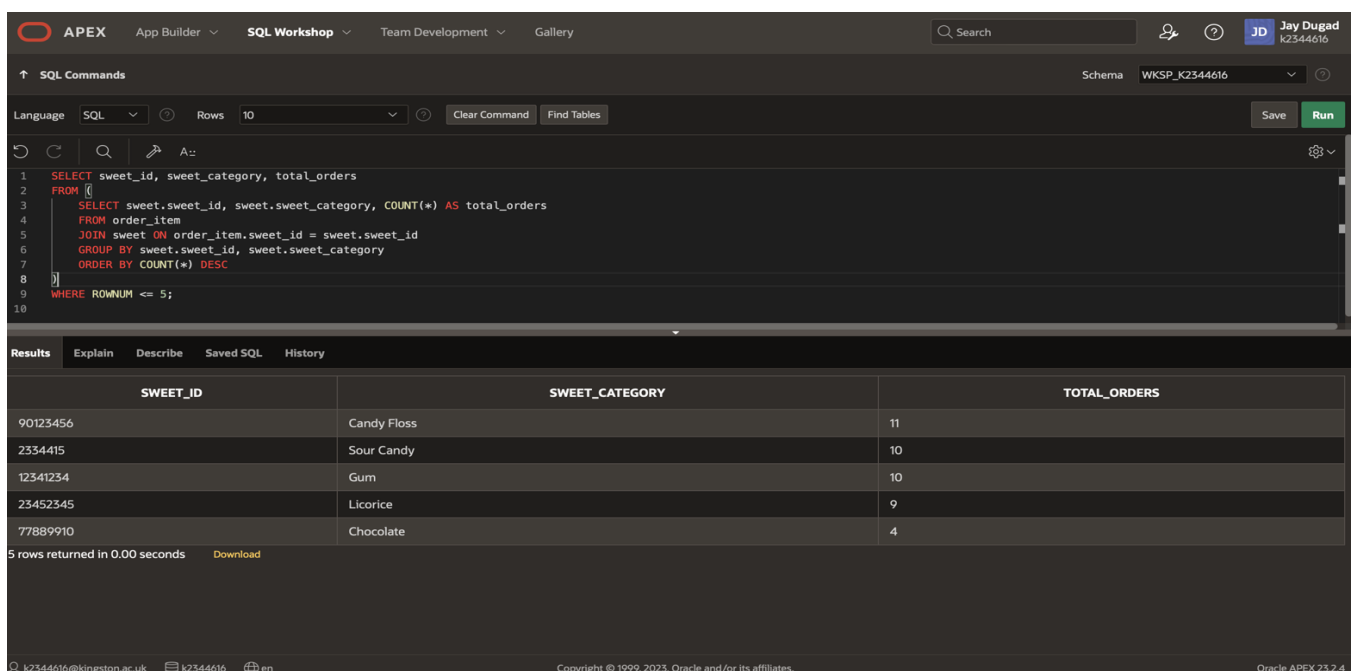
JOIN sweet ON order_item.sweet_id = sweet.sweet_id

GROUP BY sweet.sweet_id, sweet.sweet_category

ORDER BY COUNT(*) DESC

)

WHERE ROWNUM <= 5;



The screenshot shows the Oracle APEX SQL Workshop interface. The SQL command area contains the following query:

```
1 SELECT sweet_id, sweet_category, total_orders
2 FROM (
3     SELECT sweet.sweet_id, sweet.sweet_category, COUNT(*) AS total_orders
4     FROM order_item
5     JOIN sweet ON order_item.sweet_id = sweet.sweet_id
6     GROUP BY sweet.sweet_id, sweet.sweet_category
7     ORDER BY COUNT(*) DESC
8 )
9 WHERE ROWNUM <= 5;
```

The Results tab displays the following data:

SWEET_ID	SWEET_CATEGORY	TOTAL_ORDERS
90123456	Candy Floss	11
2334415	Sour Candy	10
12341234	Gum	10
23452345	Licorice	9
77889910	Chocolate	4

5 rows returned in 0.00 seconds

Query 6) This query shows the sweets that have active offers right now.

SELECT

sweet.sweet_id, sweet.sweet_name, offer.valid_from, offer.valid_upto

FROM

sweet

JOIN

```
offer ON sweet.sweet_id = offer.sweet_id
```

WHERE

`SYSDATE BETWEEN offer.valid_from AND offer.valid_upto;`

APEX

App Builder

SQL Workshop

Team Development

Gallery

Search

JD

Jay Dugad

k2344616

SQL Commands

SchemaWKSP_K2344616

LanguageSQLRows100Clear CommandFind TablesSaveRun

1SELECT

2sweet.sweet_id, sweet.sweet_name, offer.valid_from, offer.valid_upto

3FROM

4sweet

5JOIN

6offer ON sweet.sweet_id = offer.sweet_id

7WHERE

8SYSDATE BETWEEN offer.valid_from AND offer.valid_upto;

Results

ExplainDescribeSaved SQLHistory

SWEET_ID	SWEET_NAME	VALID_FROM	VALID_UPTO
33445566	Candy Corn	03/10/2024	03/20/2024
77889900	Snickers Bar	03/14/2024	03/21/2024
4556677	Candy Buttons	03/15/2024	03/22/2024
1223344	Candy Necklace	03/16/2024	03/23/2024
44556617	Candy Buttons	03/13/2024	03/23/2024

5 rows returned in 0.01 secondsDownload

k2344616@kingston.sc.uk

k2344616

en

Copyright © 1999, 2023, Oracle and/or its affiliates.

Oracle APEX 23.2.3

CHAPTER 5: CONCLUSION

- BSS's database structure is carefully designed to meet the operational requirements of the company. Tables pertaining to client confectionery orders, delivery, shopping lists, customer information, and other information are efficiently arranged. This methodical approach, which covers everything from stocking candy to handling client demands, is a reflection of the company's workflow and procedures. The data model ensures practical use and integrity inside the database by incorporating many data types and restrictions, including NOT NULL, PRIMARY KEY, and FOREIGN KEY
- Additionally, the database makes it easier to manage candy-related data, such as prices, sizes, descriptions, and stock levels, allowing BSS to effectively handle and preserve crucial data. Additionally, it improves browsing and search experiences for users by tracking offers and promotions and streamlining the segmentation of candies into categories. This feature draws customers by offering an enjoyable online buying experience, which incentivizes purchases.
- The database not only enhances browsing and buying experiences but also encourages users involvement with tools like shopping lists and baskets that help users easily bookmark products and finish transactions. Furthermore, the integration of user authentication guarantees secure entry to customized services like order histories and shopping lists, augmenting the general customer experience and contentment. BSS can meet website and database needs while optimizing performance and increasing user engagement thanks to our all-inclusive approach to database administration.

Review of the entire exercise: -

- I enjoyed working on this coursework since it allowed me to gain real-world experience creating and managing databases. Doing this assignment made me better suited for employment. I also learned how to handle large amounts of data and work with the relationships between various tables from this project.
- My ability to solve problems has improved as a result of working with complex data structures and developing effective database designs. I am now able to identify problems and come up with the best solutions. Asking doubts to classmates and professors throughout the assignment improved my communication abilities, allowing me to explain complex database ideas in simple terms and lead fruitful conversations on database architecture and use.
- Participating in coursework also offered me a chance for ongoing education and skill enhancement, encouraging a lifelong learning mindset and a dedication to remaining current with database technology and processes. My study has given me the thorough experience, knowledge, and confidence I need to explore professional prospects in database management roles, making me a competitive candidate in the job market.