Team 3: Final Project

Simran Bawaskar, Saloni Bhutada, Aman Maheshwari, Reha Patel, Niraj Sai Prasad, Sindhu Swaroop

DAMG 6210 Thursday Class

Business Problem & Solution

In the retail industry, it is vital that companies are able to keep up with the supply and demand on products they are offering. Some companies may choose to keep track of this supply and demand through physical books and paperwork, and over time this can result in insufficient storage space, misplaced orders by human error, and overall customer dissatisfaction.

Our proposed solution demonstrates how a retail company that sells electronic items goes about digitally managing inventory based on orders that are fulfilled and returned. First and foremost, we allow customers to create accounts within our system that store data such as first name, last name, address, etc. This data is used when a customer creates an order with different items to ensure that the right orders are being associated with the right customers. In addition to this, when an order is placed, we will make sure that the correct amount of each item in the order is removed from the inventory. On the other hand, if a customer wants to return an item, we also have the capability to create a return order and readjust the inventory of the item.

Project Database and Github Overview

Github: https://github.com/rehapatel/DAMG6210_Team3 (repo may be private to protect code)

Schema Name: admin

- **Files:** admin_code.sql

Schema Name: customer_owner

- **Tables:** customer, customer_address
- **Views:** customer_view
- **Indexes:** customer_contact_idx
- **Files:** customer_owner_create_table.sql, customer_owner_tables.sql, customer_sp.sql, customer_index.sql, customer_views.sql, customer_address.sql

Schema Name: item_owner

- **Tables:** codes, item, item_inventory, item_reviews
- **Views:** item_reviews_view, item_view
- **Indexes:** item_idx
- **Files:** item_sp.sql, item_owner_create_table.sql, item_index.sql, item_views.sql

Schema Name: order_owner

- **Tables:** order_desc, order_item_detals, order_payment_history
- **Views:** order_desc_item_customer_view, order_desc_item_payment_view
- **Files:** order_owner_sp.sql, order_owner_create_table.sql, order_views.sql

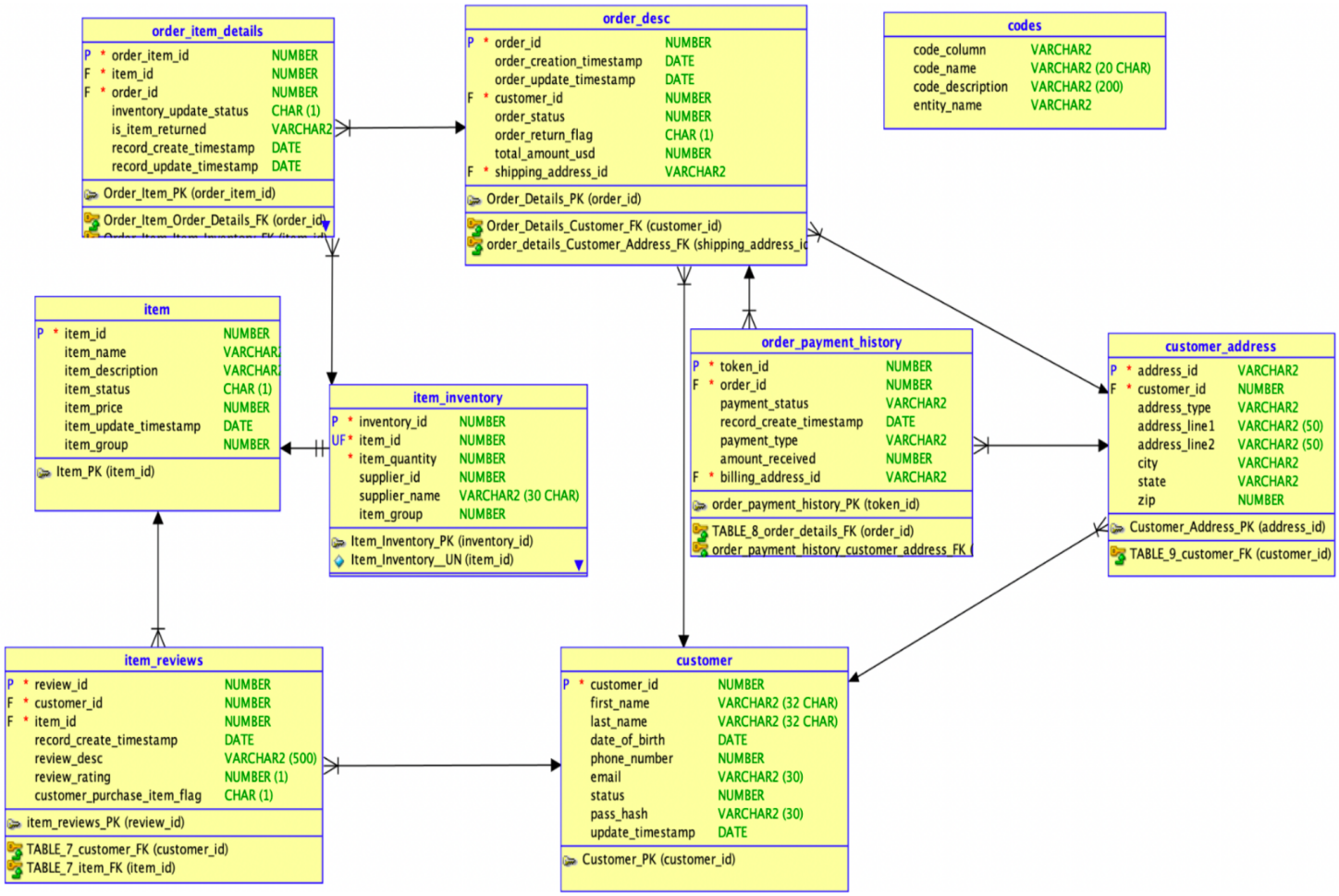
Triggers:

- Item_owner_trigger.sql
- Item_qty_status_trigger.sql
- Order_owner_item_qtn_trigger.sql
- Order_owner_trigger.sql
- customer_trigger.sql

Excel Data Files

- Customer_table.xlsx, Customer_address_list_final.xlsx
- Itemtabledata.xlsx, Iteminventorytable.xlsx, Itemreviewtable.xlsx
- Codes_table_data.xlsx
- order_owner_data.xlsx

Database Model



Instructions to Set Up Admin Components and Tables

-- Connect to your wallet and open the admin_code.sql and run

-- Create new connection for user item_owner with password: 'Itemowner2022'

-- Create new connection for user order_owner with password: 'Orderowner2022'

-- Create new connection for user customer_owner with password: 'Customerowner2022'

-- Open new worksheet for item_owner connection and run the code in file item_owner_create_table.sql -- Open new worksheet for order_owner connection and run the code in file order_owner_create_table.sql

-- Open new worksheet for customer_owner connection and run the code in file customer_owner_create_table.sql

-- Exceptions that you may see upon re-running:

- Admin_code.sql:
    - User may already exist, so we have handled this.
- Owner sql files:
    - Table may already exist so we will not be creating a new table upon every re-run
    - If you are re-running multiple times, rerun the ENTIRE script altogether. Our script will delete all rows before inserting again so that multiple reruns do not fail.

-- Please run the reports SQL code in the respective user worksheets for validation if needed.

Reports

Report 1: Checking to see if there is a specific payment type that is higher and is also used more than others. We allow customers to split purchases on one order across multiple payment types so it is interesting to see the average order price and the total number of orders with each payment type.

```
9  -- Checking to see if there is a specific payment type that is higher and used more than others
10 select unique(payment_type),
11 round(avg(total_amount_usd), 2) as "Average Order Price",
12 count(payment_type) as "Total Number of Orders"
13 from order_owner.order_desc_item_payment_view
14 group by payment_type
15 order by payment_type;
```

Query Result

SQL | All Rows Fetched: 3 in 0.054 seconds

	PAYMENT_TYPE	Average Order Price	Total Number of Orders
1	credit_card	373.35	91
2	debit_card	316.26	44
3	gift_card	308.45	31

Report 2: Checking to see which customer id paid the greatest amount for products. We see that while one customer spent over $1000, many others spent around $200 so we know that our total profits are not coming from one customer.

```
17 -- Checking to see which customer id paid the greatest amount for products
18 select unique(customer_id), round(avg(total_amount_usd), 2) as "Total Amount Paid"
19 from order_owner.order_desc_item_customer_view
20 group by customer_id
21 order by round(avg(total_amount_usd), 2) desc;
```

Query Result

SQL | All Rows Fetched: 56 in 0.128 seconds

	CUSTOMER_ID	Total Amount Paid
1	98	1241.64
2	91	689.95
3	3	424.52
4	70	290.99
5	61	280.99
6	25	279.96
7	99	279.5
8	75	270.99
9	44	254.99
10	62	250.49
11	40	239.99
12	5	225.99
13	65	219.99
14	69	209.99
15	32	209.99
16	23	209.99
17	35	199.98
18	21	199
19	14	194.82
20	56	177.65

Report 3: Checking to see which months had the greatest net profit and the greatest number of orders. With this, we understand that our inventory will drop the most in the months of June, July and August as well as December. We can assume that many people are purchasing electronics during the summer months as well as around Christmas.

```
25 select to_char(order_creation_timestamp, 'YYYY-MM') as "Creation Date",
26 sum(total_amount_usd) as "Total Amount Spent",
27 count(order_creation_timestamp) as "Number of Orders"
28 from order_owner.order_desc
29 group by to_char(order_creation_timestamp, 'YYYY-MM')
30 order by 1;
```

Query Result

SQL | All Rows Fetched: 10 in 0.029 seconds

	Creation Date	Total Amount Spent	Number of Orders
1	2018-01	77.99	1
2	2018-02	219.99	1
3	2018-06	3632.73	20
4	2018-07	4142.66	33
5	2018-08	3231.86	21
6	2018-10	339.77	5
7	2018-11	673.04	6
8	2018-12	1863.85	13
9	2019-01	283.14	6
10	2022-04	5435.76	18

Report 4: Checking to see which items were bought most frequently. This is important to our business because items such as the iPhone 13 Blue 128 GB may need to be restocked more frequently.

```
33 select i.item_id as "Item ID", i.item_name as "Item Name", count(oid.item_id) as "Total Orders"
34 from item_owner.item i
35 join order_owner.order_item_details oid
36 on i.item_id = oid.item_id
37 group by i.item_id, i.item_name
38 order by 3 desc;
```

Query Result

SQL | Fetched 50 rows in 0.048 seconds

	Item ID	Item Name	Total Orders
1	100	iPhone 13 Blue 128 GB	16
2	27	CORSAIR VENGEANCE RGB 16GB (2x8GB) DDR4 3200MHz C16 Desktop Memory - Black	9
3	30	Flipside 300 Backpack (Black)	6
4	3	DENAQ - AC Adapter for TOSHIBA SATELLITE 1700 1710 1715 1730 1735 1750 1755 1955 3000 3005 A100 M30X M35X - Black	6
5	73	Sony Mini Digital Video Cassettes - DVC - 1 Hour	6
6	82	Power Acoustik - Gothic Series 10 Dual-Voice-Coil 2-Ohm Subwoofer - Black"	6
7	69	Yamaha - Micro Component System - White	5
8	25	Niles - SS-4 4-Pair Speaker Selector - Black	5
9	24	Ultimate Ears MEGABLAST Portable Wi-Fi/Bluetooth Speaker with hands-free Amazon Alexa voice control (waterproof) - Graphite Black	5
10	12	Details About Samsung Gear Iconx 2018 Edition Cordfree Fitness Earbuds Black (us Version)	4
11	79	SRS-ZR7 Wireless Speaker	4
12	13	2TB Red 5400 rpm SATA III 3.5 Internal NAS HDD	4
13	97	AudioQuest - GOLDG01R Golden Gate 1m (3.28 ft.) RCA Audio Cable - Red	4
14	9	Corsair Vengeance LPX 16GB (2x8GB) DDR4 DRAM 3000MHz C15 Desktop Memory Kit - Black (CMK16GX4M2B3000C15)	4
15	52	iSimple ISBC01 BluClik Bluetooth Remote Control with Steering Wheel and Dash Mounts	3
16	95	Toshiba - 2.0-Channel Soundbar with 16-Watt Digital Amplifier - Black	3
17	92	Belkin F4U095tt Thunderbolt 3 Express Dock HD with 3.3-Foot Thunderbolt 3 Cable	2
18	14	Details About Panamax Mr4000 8outlets Surge Protector Home Theater Power Line Management	2

Report 5: Checking to see the total amount of returns and total revenue (total purchases - total returns). From this we see that we have a relatively low amount of returns and are still able to have over $40,000 in revenue over the few months we were operating.

```
41  select sum(total_amount_usd) as "Total Amount of Returns"
42  from order_owner.order_desc
43  where order_return_flag = 1;
```

Query Result

SQL | All Rows Fetched: 1 in 0.034 seconds

	Total Amount of Returns
1	-3703.08

```
46  select sum(total_amount_usd) as "Total Revenue"
47  from order_owner.order_desc_item_customer_view;
```

Query Result

SQL | All Rows Fetched: 1 in 0.052 seconds

	Total Revenue
1	40081.85

Report 6: Checking to see who all are the top 10 most frequent buyers. As our business continues to expand, we may want to target them as they are most likely to purchase from us.

```
51  select * from (
52  select c.first_name as Customer_Name, o.customer_id, count(o.customer_id) as frequent_buyers
53  from order_owner.order_desc o
54  join customer_owner.customer c
55  on o.customer_id = c.customer_id
56  where order_status = 1
57  group by c.first_name, o.customer_id
58  order by frequent_buyers desc )
59  where rownum <= 10;
```

Script Output | Query Result

SQL | All Rows Fetched: 10 in 0.053 seconds

	CUSTOMER_NAME	CUSTOMER_ID	FREQUENT_BUYERS
1	James	1	9
2	Kris	10	8
3	Cammy	15	4
4	Josephine	2	4
5	Abel	12	4
6	Veronika	22	3
7	Elly	68	3
8	Lai	88	3
9	Simona	6	3
10	Bernardo	30	3

Report 7: Checking to see who our top returners are. Based on this we see that the same customer does not return items to us more than once.

```
62  -- Checking to understand which customers are returning orders more frequently
63  select * from
64  (select c.first_name, o.customer_id, count(o.customer_id) as returns_count
65  from order_owner.order_desc o
66  join customer_owner.customer c
67  on o.customer_id = c.customer_id
68  where o.order_return_flag = 1
69  group by c.first_name, o.customer_id
70  order by returns_count desc )
71  where rownum <= 5;
```

Script Output | Query Result

SQL | All Rows Fetched: 5 in 0.045 seconds

	FIRST_NAME	CUSTOMER_ID	RETURNS_COUNT
1	Blair	50	1
2	Veronika	22	1
3	Ernie	33	1
4	Josephine	2	1
5	Lai	88	1

Report 8: Checking to understand which areas in MA the most orders coming from. We see that there are multiple addresses in 02128 that have a high number of orders but 4 Jersey Street has the overall highest.

```
75  select * from
76  ( select c.address_line1, c.zip, count(o.customer_id) as Max_orders
77  from CUSTOMER_OWNER.customer_address c
78  join order_owner.order_desc o
79  on c.customer_id = o.customer_id
80  where o.order_status = 1
81  group by c.address_line1, c.zip
82  order by Max_orders desc )
83  where rownum <= 10;
```

Script Output | Query Result

SQL | All Rows Fetched: 10 in 0.056 seconds

	ADDRESS_LINE1	ZIP	MAX_ORDERS
1	4 Jersey Street	02215	9
2	166 Everett Street	02128	8
3	38938 Park Blvd	02128	6
4	Jvue Apartment	02120	5
5	462 Geneva Avenue	02122	4
6	885 Huntington Ave	02128	4
7	255 Condor Street	02125	4
8	Jvue Apartment	02124	3
9	3 Mcauley Dr	02109	3
10	835 South Street	02131	3

Report 9: Checking to see which hour of the day has the greatest number of orders. We see that the greatest number of orders happen later in the night meaning most customers are placing orders later - perhaps after work. If we need to maintain our database, we know to do so during the least popular times.

```
87  select extract(Hour from order_creation_timestamp) as Order_traffic_time,
88  count(order_id) as Num_Orders
89  from order_owner.order_desc
90  where order_status = 1
91  group by extract(Hour from order_creation_timestamp)
92  order by num_orders desc;
```

Script Output × Query Result ×

SQL | All Rows Fetched: 19 in 0.031 seconds

	ORDER_TRAFFIC_TIME	NUM_ORDERS
3	22	10
4	21	10
5	20	9
6	14	9
7	16	6
8	5	5
9	7	4
10	10	4
11	11	4
12	2	3
13	13	2
14	4	2
15	12	2
16	8	2
17	15	2
18	1	2
19	6	2