Table of Contents

Assignment #1: Video Game Store DBMS

Application : Video Game Store DBMS
Description: With the introduction of online orders from in-store purchases, it is crucial for a store to offer both services in order to reach a greater number of customers. This will allow the business to flourish as it increases accessibility for customers to purchase items online when they are busy or unable to come into the store. However, online purchases can occur at any time of the day, or concurrently with in-store purchases. Hence, a DBMS that can accurately track stock, track online transactions, and display information is necessary for the successful management of stores and to better compete with online stores.

There are several functions that can occur in this management system.

Type of Function (Create, Read, Update, or Delete)	**Function Name**	**Description/Purpose**
Create	add_order	Once a purchase has been made, add the order into the system.
Create	add_game	Add a new game into the system
Create	add_console	Add a new console into the system
Read	view_inventory	View the inventory of all the products
Read	view_game	View a specific game in the system and the data related to the game
Read	view_console	View a specific console in the system and the data related to the consoles
Read	view_customer_order	View a specific customer order.
Update	update_customer_order	If the order status changes because of confirmation/in-progress/shipping/delivered then update the order for the customer view.
Update	update_inventory_stock	Increase/decrease stock depending on orders made in store and online
Delete	delete_product	If a product has been discontinued, delete the product from the system

First, the store should show the list of all video games available in the store with the ability to filter between genres. It should have basic details about the game such as a description of the game,

which console it is played on, the publisher, the genre of game, and how many players can play, and the current stock in the store. The store should also satisfy online orders, therefore the application must track each order a customer makes online and allow them to view the status of the orders. This includes information of when the order was made, estimated delivery, the customer who made the order, the product ordered which could either be a game or a console, and the current status of the order (confirmed order, in-progress, shipping, delivered, etc). Additionally, any purchases made from the online/in-store should update the stock of the games. Any stock received from the distributor should also update the stock accordingly. There should be restrictions on orders if there is no stock available for the requested game. With each online transaction, computerized billings and reports of earnings should be possible. This includes the customer's information such as first name, last name, address, postal code, email, phone number.

Requirements of the System:

1. An instance in the game table would have a relationship with the game inventory table to keep track of the games
2. An instance in the game table would have a relationship with the purchase table because a customer would be purchasing a game we're keeping track of selling.
3. An instance in the console table would have a relationship with the console inventory table to keep track of individual consoles in our inventory..
4. An instance of the console table would have a relationship with the purchase table if the order was to include a console being bought. .
5. An instance of the customer table would have a relationship with the order table to keep track of individual customers and their orders.
6. An instance of the order table would be related to the purchase table. If the order is confirmed, then the purchase table instance would be updated alongside the order table.

Game Table	
game_id	(id number) An auto-generated field for a specific game
name	(String) Game name
genre	(String) Genre of the game
price	(Float) Price of the product
platform	(platform_id) Playable on which platforms
min_spec	(minspec_id) Minimum hardware specifications required to run game
rec_spec	(recspec_id) Recommended hardware specifications to run game
esrb_rate	(string) Assigns age and content rating (E for everybody, M for mature)
release_year	(int) The year the game was published
publisher	(string) The company that publishes the game
description	(String) Description of game

Customer Table	
customer_id	(id number) An auto-generated field for unique customers
first_name	(String) First name
last_name	(String) Last name
email	(String) Customer's email
address	(String) Customer's address

Game Inventory Table	
game_id	(id number) Relates to a specific game id
count	(Int) Stock of the individual game
num_sales	(int) How many copies have sold in total
region_sales	(int) Sales for game in each region

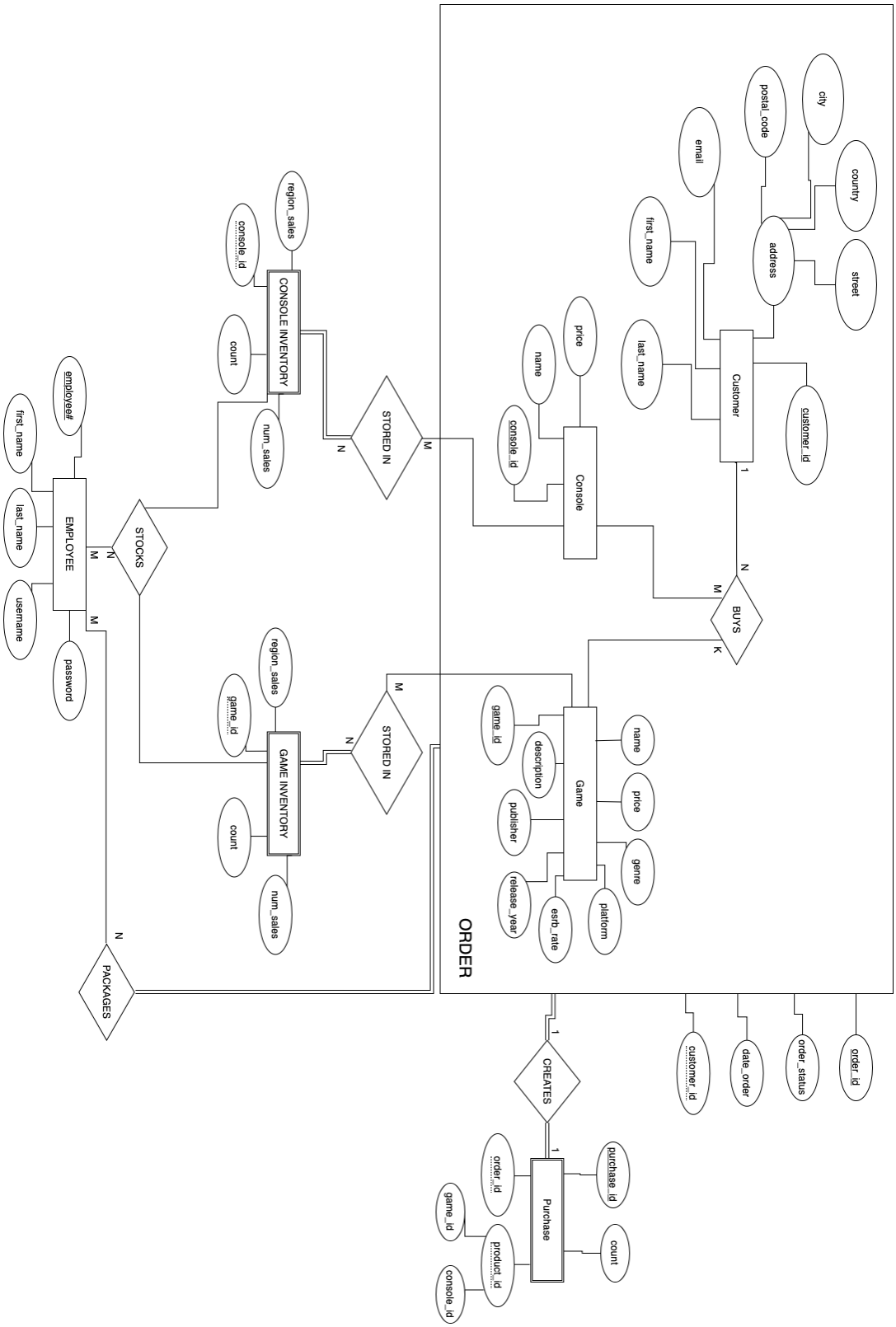
Console Inventory Table	
console_id	(id number) Relates to a specific console
count	(Int) Stock of the individual game console

Purchase Table	
purchase_id	(id number) An auto-generated unique field for product purchased
order_id	(id number) Specifies an order with their id
product_id	(game_id/console_id) Product id
count	(Int) Number of copies

Order Table	
order_id	(id number) An auto-generated unique field for orders
customer_id	(id number) Specifies a customer with their id
date_order	(Date) Date when the order was placed
order_status	(String) Status of the order placed, either (ordered, shipped, delivered, cancelled)
method_of_payment	(String) Type of payment

Console Table	
console_id	(id number) An auto-generated field for a specific console
name	(String) Console name
price	(Float) Price of the product
description	(String) Description of game

Assignment #2: ER



Assignment #3: Schema Design

//CREATE STATEMENTS (atom IDE)

```
CREATE TABLE customer (
  customer_id NUMBER PRIMARY KEY,
  first_name VARCHAR2 (25 char) NOT NULL,
  last_name VARCHAR2 (25 char) NOT NULL,
  email VARCHAR2 (100 char) UNIQUE,
  city VARCHAR2 (100 char) NOT NULL,
  country VARCHAR2 (60 char) NOT NULL,
  street VARCHAR2 (50 char) NOT NULL,
  postal_code VARCHAR2 (8 char) NOT NULL
);
```

```
CREATE TABLE console (
  console_id NUMBER PRIMARY KEY,
  price NUMBER NOT NULL,
  name VARCHAR2 (25 char) NOT NULL
);
```

```
CREATE TABLE game (
   game_id NUMBER PRIMARY KEY,
   price NUMBER NOT NULL,
   name VARCHAR2 (25 char) NOT NULL,
   release_year NUMBER NOT NULL,
   esrb_rate VARCHAR2 (10 char)  NOT NULL,
   publisher VARCHAR2 (50 char) NOT NULL,
   description VARCHAR2(200 char) NOT NULL,
   platform NUMBER NOT NULL,
   FOREIGN KEY (platform) REFERENCES console (console_id)
);
```

```
CREATE TABLE console_inventory_stored_in (
          count NUMBER NOT NULL,
          num_sales NUMBER NOT NULL,
          region_sales NUMBER NOT NULL,
   console_id NUMBER NOT NULL,
   FOREIGN KEY (console_id) REFERENCES console (console_id)
);
```

```
CREATE TABLE game_inventory_stored_in (
        count NUMBER NOT NULL,
        num_sales NUMBER NOT NULL,
        region_sales NUMBER NOT NULL,
   game_id NUMBER NOT NULL,
   FOREIGN KEY (game_id) REFERENCES game (game_id)
);
```

```
CREATE TABLE employee (
        emp_num NUMBER PRIMARY KEY,
        first_name VARCHAR2 (25 char) NOT NULL,
        last_name VARCHAR2 (25 char) NOT NULL,
        username VARCHAR2 (25 char) NOT NULL UNIQUE,
        password VARCHAR2 (25 char) NOT NULL
);
```

```
CREATE TABLE order (
        order_id NUMBER PRIMARY KEY,
        FOREIGN KEY (emp_num) REFERENCES employee (emp_num),
        FOREIGN KEY (customer_id) REFERENCES customer(customer_id),
    date_of_order DATE NOT NULL,
    order_status VARCHAR2(25 char) NOT NULL,
);
```

```
CREATE TABLE purchase (
        purchase_id NUMBER PRIMARY KEY,
        FOREIGN KEY (order_id) REFERENCES order (order_id),
        FOREIGN KEY (game_id) REFERENCES game (game_id),
        FOREIGN KEY (console_id) REFERENCES console (console_id),
        count NUMBER NOT NULL,
);
```

// NOTE: FOREIGN KEY (game_id) and FOREIGN KEY (console_id) should be nullable, as only one of them should be filled in.

INSERT STATEMENTS

INSERT INTO customer (customer_id, first_name, last_name, email, city, country, street, postal_code)
VALUES (111, ‘Baxter’, ‘Seuss’, ‘baxter.seuss@gmail.com’, ‘Toronto’, ‘Canada’, ‘Broadview’, ‘M7K 8L6’);

INSERT INTO console (console_id, price, name)
VALUES (39487, 500, ‘Nintendo Switch’);

INSERT INTO game (game_id, price, name, release_year, platform, esrb_rate, publisher, description)
VALUES (10, 3.14, ‘Over 9000’, 1989, 39487, ‘M’, ‘NT 3.51’, ‘Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.’);

INSERT INTO console_inventory_stored_in (console_id, count, region_sales, num_sales)
VALUES (10, 0, 10240, 16384);

INSERT INTO game_inventory_stored_in (game_id, count, region_sales, num_sales)
VALUES (10, 0, 22000, 194);

INSERT INTO employee (emp_num, first_name, last_name, username, password)
VALUES (234, ‘Mickey’, ‘Mouse’, ‘M.Mouse’, ‘ChocolateCake18!’);

INSERT INTO order(order_id, emp_num, customer_id, date_of_order, order_status)
VALUES (2, 3057809, (SELECT customer_id from customer where first_name = ‘Pie’ AND last_name = ‘JoJo’), 2000-10-07);

INSERT INTO purchase (purchase_id, order_id, game_id, console_id, count) VALUES (1, 2, 4, NULL, 9001);

Assignment #4: Simple Queries/Simple Demo

Queries PART A

CUSTOMER TABLE

//Select all customer in the customer's table
SELECT * FROM customer

	CUSTOMER_ID	FIRST_NAME	LAST_NAME	EMAIL	CITY	COUNTRY	STREET	POSTAL_CODE
1	111	Baxter	Seuss	baxter.seuss@gmail.com	Toronto	Canada	Broadview	M7K8L6
2	112	Michelle	Gao	mvp@gmail.com	Brampton	Canada	Dixie	L3Y56S
3	113	John	Smith	jsmith@gmail.com	Toronto	Canada	Dundas	M9Q1UP

//Show the count of customers that are in each city and group them
SELECT COUNT(customer_id), city
FROM customer
GROUP BY city
ORDER BY COUNT(customer_id) DESC;

	COUNT(CUSTOMER_ID)	CITY
1	2	Toronto
2	1	Brampton

CONSOLE TABLE

//Select all consoles ordered by lowest price to highest price
SELECT * FROM console ORDER BY price ASC;

	CONSOL...	PRICE	NAME
1	39487	500	Nintendo Switch
2	39489	600	Xbox Series X
3	39488	800	PlayStation 5

CONSOLE_INVENTORY_STORED_IN

//select all the consoles we currently have in stock
SELECT * FROM console_inventory_stored_in

	COUNT	NUM_SALES	REGION_SALES	CONSOLE_ID
1	24	16384	10240	39487
2	10	10004	2440	39488
3	5	42332	10023	39489

//Select all consoles that have an inventory more than 9
SELECT * FROM console_inventory_stored_in
WHERE count > 9;

	COUNT	NUM_SALES	REGION_SALES	CONSOLE_ID
1	24	16384	10240	39487
2	10	10004	2440	39488

GAME TABLE

//get all existing games from the table
SELECT * FROM game

	GAME_ID	PRICE	NAME	RELEASE_YEAR	ESRB_RATE	PUBLISHER	DESCRIPTION
1	10	3.14	Over 9000	1989	M	NT 3.51	Duis aute irure dolor in reprehenderit in voluptat
2	11	79.99	SSMB Ultimate	2018	T	Nintendo	It is the fifth installment in the Super Smash Bro
3	12	59.99	Watch Dogs: Legion	2020	M	Ubisoft	It is the third instalment in the Watch Dogs serie
4	13	39.99	Devil May Cry 5	2019	M	Capcom	It is the sixth installment overall and the fifth

//get all games for the Nintendo Switch (the console id is 39487)
SELECT * FROM game WHERE platform = 39487;

	GAM...	PRICE	NAME	RELEASE_YEAR	ESRB_RATE	PUBLISHER	DESCRIPTION
1	10	3.14	Over 9000	1989	M	NT 3.51	Duis aute irure dolor in reprehenderit in voluptate vel
2	11	79.99	SSMB Ultimate	2018	T	Nintendo	It is the fifth installment in the Super Smash Bros. se

GAME_INVENTORY_STORED_IN

//Select all games that have a stock lower than 10
SELECT *
FROM game_inventory_stored_in
WHERE count < 10;

	COUNT	NUM_SALES	REGION_SALES	GAME_ID
1	5	194	22000	10

EMPLOYEE TABLE

//Select all employees' first and last name

SELECT * FROM employee;

	FIRST_NAME	LAST_NAME
1	Mickey	Mouse
2	Donald	Duck
3	Pluto	Dog

//Select the first name of the employee that packed orders post 10/22/2021

SELECT first_name
FROM employee
WHERE emp_num IN (SELECT emp_num FROM customer_order WHERE date_of_order > to_date('10/22/2021', 'mm/dd/yyyy'));

	FIRST_NAME
1	Pluto

CUSTOMER_ORDER TABLE

//Select all orders that are current shipping

SELECT * FROM customer_order WHERE order_status LIKE 'Shipped';

	ORDER_ID	DATE_OF_ORDER	ORDER_STATUS	EMP_NUM	CUSTOMER_ID
1	3	21-10-06	Shipped	235	112
2	4	21-10-21	Shipped	234	111

//Select customer id that currently has an order in progress. This allows us to see the customers who currently have orders without repeats of their id

SELECT DISTINCT customer_id
FROM customer_order
WHERE order_status LIKE 'In-Progress';

	CUSTOMER_ID
1	112

PURCHASE TABLE

//Select all purchase orders

SELECT * FROM purchase;

	PURCHASE_ID	ORDER_ID	GAME_ID	CONSOLE_ID	INVENTORY
1	2	2	12	(null)	1

//==========//

// PART B

CREATE VIEW customer_public AS
SELECT customer_id, first_name, last_name, email, city,country
FROM customer;

```
View CUSTOMER_PUBLIC created.
```

SELECT *
FROM customer_public

```
SELECT *
FROM customer_public
```

Script Output × | Query Result ×

SQL | All Rows Fetched: 3 in 0.035 seconds

	CUSTOMER_ID	FIRST_NAME	LAST_NAME	EMAIL	CITY	COUNTRY
1	111	Baxter	Seuss	baxter.seuss@gmail.com	Toronto	Canada
2	112	Michelle	Gao	mvp@gmail.com	Brampton	Canada
3	113	John	Smith	jsmith@gmail.com	Toronto	Canada

SELECT DISTINCT first_name, last_name, email
FROM customer_public;

```
SELECT DISTINCT first_name, last_name, email
FROM customer_public;
```

Script Output × Query Result ×

SQL | All Rows Fetched: 3 in 0.044 seconds

	FIRST_NAME	LAST_NAME	EMAIL
1	Michelle	Gao	mvp@gmail.com
2	Baxter	Seuss	baxter.seuss@gmail.com
3	John	Smith	jsmith@gmail.com

CREATE VIEW employee_public AS
SELECT emp_num, first_name, username
FROM employee;

```
View EMPLOYEE_PUBLIC created.
```

SELECT *
FROM employee_public

```
SELECT *
FROM employee_public
```

Script Output | Query Result

SQL | All Rows Fetched: 3 in 0.094 seconds

	EMP_NUM	FIRST_NAME	USERNAME
1	234	Mickey	M.Mouse
2	235	Donald	D.Donald
3	236	Pluto	P.Dog

SELECT customer_id, first_name FROM customer
UNION
SELECT emp_num, first_name FROM employee;

	CUSTOMER_ID	FIRST_NAME
1	111	Baxter
2	112	Michelle
3	113	John
4	234	Mickey
5	235	Donald
6	236	Pluto

```
SELECT COUNT(order_id) AS order_number, emp_num AS employee_number
FROM customer_order
INNER JOIN customer
ON customer_order.customer_id = customer.customer_id
GROUP BY emp_num;
```

	ORDER_NUMBER	EMPLOYEE_NUMBER
1	1	235
2	1	236
3	2	234

```
SELECT DISTINCT c.first_name, c.last_name, g.name AS game_title
FROM purchase p, customer_order o, game g, customer c
WHERE p.order_id = o.order_id
   AND o.customer_id = c.customer_id
   AND p.game_id = g.game_id;
```

	FIRST_NAME	LAST_NAME	GAME_TITLE
1	Michelle	Gao	Watch Dogs: Legion

```
SELECT DISTINCT g.game_id, g.name, c.count, c.num_sales AS copies_sold
FROM game g, game_inventory_stored_in c
WHERE g.game_id = c.game_id;
```

	GAME_ID	NAME	COUNT	COPIES_SOLD
1	12	Watch Dogs: Legion	55	1000
2	10	Over 9000	5	194

Assignment #5: Demo of Adv. Queries by Unix Shell Commands

menu.sh

```bash
#!/bin/bash
#export LD_LIBRARY_PATH=/usr/lib/oracle/12.1/client64/lib
MainMenu()
{
    while [ "$CHOICE" != "START" ]
    do
        clear
    echo "==============================================================="
    echo "| Oracle All Inclusive Tool |"
    echo "| Main Menu - Select Desired Operation(s): |"
    echo "| <CTRL-Z Anytime to Enter Interactive CMD Prompt> |"
    echo "---------------------------------------------------------------"
    echo " $IS_SELECTEDM M) View Manual"
    echo " "
    echo " $IS_SELECTED1 1) Drop Tables"
    echo " $IS_SELECTED2 2) Create Tables"
    echo " $IS_SELECTED3 3) Populate Tables"
    echo " $IS_SELECTED4 4) Query Tables"
    echo " "
    echo " $IS_SELECTEDX X) Force/Stop/Kill Oracle DB"
    echo " "
    echo " $IS_SELECTEDE E) End/Exit"
    echo "Choose: "

    read CHOICE
    if [ "$CHOICE" == "0" ]
    then
        echo "Nothing Here"
    elif [ "$CHOICE" == "1" ]
    then
        echo "1"
        bash drop_tables.sh
        Pause
    elif [ "$CHOICE" == "2" ]
    then
        bash create_tables.sh
        Pause
    elif [ "$CHOICE" == "3" ]
    then
        echo "3"
        bash populate_tables.sh
        Pause
    elif [ "$CHOICE" == "4" ]
    then
        echo "4"
        bash queries.sh
        Pause
    elif [ "$CHOICE" == "3" ]
    then
        echo "3"
        bash populate_tables.sh
        Pause
    elif [ "$CHOICE" == "4" ]
    then
        echo "4"
        bash queries.sh
        Pause
    elif [ "$CHOICE" == "E" ]
    then
    exit
    fi
done
}

#--COMMENTS BLOCK--
# Main Program
#--COMMENTS BLOCK--
ProgramStart()
{
    StartMessage
    while [ 1 ]
    do
        MainMenu
    done
}

ProgramStart
```

drop_tables.sh

```bash
#!/bin/sh
#export LD_LIBRARY_PATH=/usr/lib/oracle/12.1/client64/lib
sqlplus64 "m32pham/10228665@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(Host=oracle.scs.ryerson.ca)(Port=1521))(CONNECT_DATA=(SID=orcl)))" <<EOF
DROP TABLE PURCHASE CASCADE CONSTRAINTS;
DROP TABLE CUSTOMER_ORDER CASCADE CONSTRAINTS;
DROP TABLE EMPLOYEE CASCADE CONSTRAINTS;
DROP TABLE GAME_INVENTORY_STORED_IN CASCADE CONSTRAINTS;
DROP TABLE CONSOLE_INVENTORY_STORED_IN CASCADE CONSTRAINTS;
DROP TABLE CONSOLE CASCADE CONSTRAINTS;
DROP TABLE GAME CASCADE CONSTRAINTS;
DROP TABLE CUSTOMER CASCADE CONSTRAINTS;

exit;
EOF
```

create_tables.sh

```
#!/bin/sh
#export LD_LIBRARY_PATH=/usr/lib/oracle/12.1/client64/lib
sqlplus64 "m32pham/10228665@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(Host=oracle
CREATE TABLE customer (
    customer_id NUMBER PRIMARY KEY,
    first_name VARCHAR2 (25 char) NOT NULL,
    last_name VARCHAR2 (25 char) NOT NULL,
    email VARCHAR2 (100 char) UNIQUE,
    city VARCHAR2 (100 char) NOT NULL,
    country VARCHAR2 (60 char) NOT NULL,
    street VARCHAR2 (50 char) NOT NULL,
    postal_code VARCHAR2 (8 char) NOT NULL
);

CREATE TABLE console (
    console_id NUMBER PRIMARY KEY,
    price NUMBER NOT NULL,
    name VARCHAR2 (25 char) NOT NULL
);

CREATE TABLE game (
     game_id NUMBER PRIMARY KEY,
     price NUMBER NOT NULL,
     name VARCHAR2 (25 char) NOT NULL,
     release_year NUMBER NOT NULL,
     esrb_rate VARCHAR2 (10 char)  NOT NULL,
     publisher VARCHAR2 (50 char) NOT NULL,
     description VARCHAR2(200 char) NOT NULL,
     platform NUMBER NOT NULL,
     FOREIGN KEY (platform) REFERENCES console (console_id)
);

CREATE TABLE console_inventory_stored_in (
         count NUMBER NOT NULL,
         num_sales NUMBER NOT NULL,
         region_sales NUMBER NOT NULL,
     console_id NUMBER NOT NULL,
     FOREIGN KEY (console_id) REFERENCES console (console_id)
);

CREATE TABLE game_inventory_stored_in (
         count NUMBER NOT NULL,
         num_sales NUMBER NOT NULL,
         region_sales NUMBER NOT NULL,
     game_id NUMBER NOT NULL,
     FOREIGN KEY (game_id) REFERENCES game (game_id)

CREATE TABLE game_inventory_stored_in (
         count NUMBER NOT NULL,
         num_sales NUMBER NOT NULL,
         region_sales NUMBER NOT NULL,
     game_id NUMBER NOT NULL,
     FOREIGN KEY (game_id) REFERENCES game (game_id)
);

CREATE TABLE employee (
         emp_num NUMBER PRIMARY KEY,
         first_name VARCHAR2 (25 char) NOT NULL,
         last_name VARCHAR2 (25 char) NOT NULL,
         username VARCHAR2 (25 char) NOT NULL UNIQUE,
         password VARCHAR2 (25 char) NOT NULL
);

CREATE TABLE customer_order (
         order_id NUMBER PRIMARY KEY,
     date_of_order DATE NOT NULL,
     order_status VARCHAR2(25 char) NOT NULL,
     emp_num NUMBER NOT NULL,
     customer_id NUMBER NOT NULL,
     FOREIGN KEY (emp_num) REFERENCES employee (emp_num),
         FOREIGN KEY (customer_id) REFERENCES customer(customer_id)
);

CREATE TABLE purchase (
         purchase_id NUMBER PRIMARY KEY,
     order_id NUMBER,
     game_id NUMBER,
     console_id NUMBER NULL,
         count NUMBER NOT NULL,
         FOREIGN KEY (order_id) REFERENCES customer_order (order_id),
         FOREIGN KEY (game_id) REFERENCES game (game_id),
         FOREIGN KEY (console_id) REFERENCES console (console_id)
);
EOF
```

populate_tables.sh

```
#!/bin/sh
#export LD_LIBRARY_PATH=/usr/lib/oracle/12.1/client64/lib
sqlplus64 "m32pham/10228665@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(Host=oracle.scs.ryerson.ca)(Port=1521))(CONNECT_DATA=(SID=orcl)))" <<EOF

INSERT INTO console (console_id, price, name)
VALUES (39487, 500, 'Nintendo Switch');

INSERT INTO console (console_id, price, name)
VALUES (39488, 800, 'PlayStation 5');

INSERT INTO console (console_id, price, name)
VALUES (39489, 600, 'Xbox Series X');

INSERT INTO game (game_id, price, name, release_year, platform, esrb_rate, publisher, description)
VALUES (10, 3.14, 'Over 9000', 1989, 39487, 'M', 'NT 3.51', 'Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.')

INSERT INTO game (game_id, price, name, release_year, platform, esrb_rate, publisher, description)
VALUES (11, 79.99, 'SSMB Ultimate', 2018, 39487, 'T', 'Nintendo', 'It is the fifth installment in the Super Smash Bros. series, succeeding Super Smash Bros. for Nint

INSERT INTO game (game_id, price, name, release_year, platform, esrb_rate, publisher, description)
VALUES (12, 59.99, 'Watch Dogs: Legion', 2020, 39488, 'M', 'Ubisoft', 'It is the third instalment in the Watch Dogs series and the sequel to 2016 Watch Dogs 2.');

INSERT INTO game (game_id, price, name, release_year, platform, esrb_rate, publisher, description)
VALUES (13, 39.99, 'Devil May Cry 5', 2019, 39489, 'M', 'Capcom', 'It is the sixth installment overall and the fifth mainline installment in the Devil May Cry series

INSERT INTO game_inventory_stored_in (game_id, count, region_sales, num_sales)
VALUES (10, 5, 22000, 194);

INSERT INTO game_inventory_stored_in (game_id, count, region_sales, num_sales)
VALUES (12, 55, 842, 1000);

INSERT INTO employee (emp_num, first_name, last_name, username, password)
VALUES (234, 'Mickey', 'Mouse', 'M.Mouse', 'ChocolateCake18!');

INSERT INTO employee (emp_num, first_name, last_name, username, password)
VALUES (235, 'Donald', 'Duck', 'D.Donald', 'quackQuack');

INSERT INTO employee (emp_num, first_name, last_name, username, password)
VALUES (236, 'Pluto', 'Dog', 'P.Dog', 'WoofbarkWoof');

INSERT INTO customer (customer_id, first_name, last_name, email, city, country, street, postal_code)
VALUES (111, 'Baxter', 'Seuss', 'baxter.seuss@gmail.com', 'Toronto', 'Canada', 'Broadview', 'M7K8L6');
```

```
INSERT INTO customer (customer_id, first_name, last_name, email, city, country, street, postal_code)
VALUES (112, 'Michelle', 'Gao', 'mvp@gmail.com', 'Brampton', 'Canada', 'Dixie', 'L3Y56S');

INSERT INTO customer (customer_id, first_name, last_name, email, city, country, street, postal_code)
VALUES (113, 'John', 'Smith', 'jsmith@gmail.com', 'Toronto', 'Canada', 'Dundas', 'M9Q1UP');

INSERT INTO customer_order(order_id, emp_num, customer_id, date_of_order, order_status)
VALUES (2, 234, 112, to_date('10/07/2021', 'mm/dd/yyyy'), 'In-Progress');

INSERT INTO customer_order(order_id, emp_num, customer_id, date_of_order, order_status)
VALUES (3, 235, 112, to_date('10/06/2021', 'mm/dd/yyyy'), 'Shipped');

INSERT INTO customer_order(order_id, emp_num, customer_id, date_of_order, order_status)
VALUES (4, 234, 111, to_date('10/21/2021', 'mm/dd/yyyy'), 'Shipped');

INSERT INTO customer_order(order_id, emp_num, customer_id, date_of_order, order_status)
VALUES (5, 236, 113, to_date('10/25/2021', 'mm/dd/yyyy'), 'Received');

INSERT INTO purchase (purchase_id, order_id, game_id, console_id, count)
VALUES (2, 2, 12, NULL, 1);

EOF

```

queries.sh

```
#!/bin/sh
#export LD_LIBRARY_PATH=/usr/lib/oracle/12.1/client64/lib
sqlplus64 "m32pham/10228665@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(Host=oracle.scs.ryerson.ca)(Port=1521))(CONNECT_DATA=(SID=orcl)))" <<EOF

SELECT first_name
FROM employee
WHERE emp_num IN (SELECT emp_num FROM customer_order WHERE date_of_order > to_date('10/22/2021', 'mm/dd/yyyy'));

SELECT * FROM customer_order WHERE order_status LIKE 'Shipped';

SELECT COUNT(order_id) AS order_number, emp_num AS employee_number
FROM customer_order
INNER JOIN customer
ON customer_order.customer_id = customer.customer_id
GROUP BY emp_num;

SELECT customer_id, first_name FROM customer
    UNION
SELECT emp_num, first_name FROM employee;

SELECT COUNT(order_id) AS order_number, emp_num AS employee_number
FROM customer_order
INNER JOIN customer
ON customer_order.customer_id = customer.customer_id
GROUP BY emp_num;

SELECT DISTINCT c.first_name, c.last_name, g.name AS game_title
FROM purchase p, customer_order o, game g, customer c
WHERE p.order_id = o.order_id
    AND o.customer_id = c.customer_id
    AND p.game_id = g.game_id;

SELECT DISTINCT g.game_id, g.name, c.count, c.num_sales AS copies_sold
FROM game g, game_inventory_stored_in c
WHERE g.game_id = c.game_id;
EOF
```

Unix Shell Implementation

```
================================================================
| Oracle All Inclusive Tool |
| Main Menu - Select Desired Operation(s): |
| <CTRL-Z Anytime to Enter Interactive CMD Prompt> |
----------------------------------------------------------------
  M) View Manual

  1) Drop Tables
  2) Create Tables
  3) Populate Tables
  4) Query Tables

  X) Force/Stop/Kill Oracle DB

  E) End/Exit
Choose:
```

Queries FROM Unix Shell

```
SQL> SQL>   2    3
FIRST_NAME
--------------------------
Pluto

SQL> SQL>
  ORDER_ID DATE_OF_O ORDER_STATUS                 EMP_NUM CUSTOMER_ID
---------- --------- -------------------------- --------- -----------
         3 06-OCT-21 Shipped                        235         112
         4 21-OCT-21 Shipped                        234         111

SQL> SQL>   2    3    4    5
ORDER_NUMBER EMPLOYEE_NUMBER
------------ ---------------
           1             235
           1             236
           2             234

SQL> SQL>   2    3
CUSTOMER_ID FIRST_NAME
----------- --------------------------
        111 Baxter
        112 Michelle
        113 John
        234 Mickey
        235 Donald
        236 Pluto

6 rows selected.

SQL> SQL>   2    3    4    5
ORDER_NUMBER EMPLOYEE_NUMBER
------------ ---------------
           1             235
           1             236
           2             234

SQL> SQL>   2    3    4    5
FIRST_NAME                 LAST_NAME                  GAME_TITLE
-------------------------- -------------------------- --------------------------
Michelle                   Gao                        Watch Dogs: Legion

SQL> SQL>   2    3
   GAME_ID NAME                           COUNT COPIES_SOLD
---------- -------------------------- --------- -----------
        12 Watch Dogs: Legion                55        1000
        10 Over 9000                          5         194

SQL> Disconnected from Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
```

Assignment #6: Normalization of the Database/Functional Dependencies

Console FDs:

	CONSOLE_ID	PRICE	NAME
1	39487	500	Nintendo Switch
2	39488	800	PlayStation 5
3	39489	600	Xbox Series X

Console_ID → {Price, Name}

Console_Inventory_Stored_In FDs:

	CONSOLE_ID	COUNT	NUM_SALES	REGION_SALES
1	39487	24	16384	10240
2	39488	10	10004	2440
3	39489	5	42332	10023

Console_ID → {Count, Num_Sales, Region_Sales}

Customer FDs:

	CUSTOMER_ID	FIRST_NAME	LAST_NAME	EMAIL	CITY	COUNTRY	STREET	POSTAL_CODE
1	111	Baxter	Seuss	baxter.seuss@gmail.com	Toronto	Canada	Broadview	M7K8L6
2	112	Michelle	Gao	mvp@gmail.com	Brampton	Canada	Dixie	L3Y56S
3	113	John	Smith	jsmith@gmail.com	Toronto	Canada	Dundas	M9Q1UP

Customer_Id → {First_Name, Last_Name, Email, City, Country, Street, Postal_Code}

Employee FDs:

	EMP_NUM	FIRST_NAME	LAST_NAME	USERNAME	PASSWORD
1	234	Mickey	Mouse	M.Mouse	ChocolateCake18!
2	235	Donald	Duck	D.Donald	quackQuack
3	236	Pluto	Dog	P.Dog	WoofbarkWoof

Emp_Num → {First_Name, Last_Name, Username, Password}

Game FDs:

	GAME_ID	PLATFORM	NAME	PRICE	PUBLISHER	ESRB_RATE	DESCRIPTION
1	10	39487	Over 9000	3.14	NT 3.51	M	Duis aute irure dolor in reprehenderit in voluptate velit esse cillum
2	11	39487	SSMB Ultimate	79.99	Nintendo	T	It is the fifth installment in the Super Smash Bros. series, succeedi
3	12	39488	Watch Dogs: Legion	59.99	Ubisoft	M	It is the third instalment in the Watch Dogs series and the sequel to
4	13	39489	Devil May Cry 5	39.99	Capcom	M	It is the sixth installment overall and the fifth mainline installmen

Game_Id → {Platform, Name, Price, Publisher, Esrb_rate, description}

Game_Inventory_Stored_In FDs:

	GAME_ID	COUNT	NUM_SALES	REGION_SALES
1	10	5	194	22000
2	12	55	1000	842

Game_Id → {Count, Num_Sales, Region_Sales}

Customer_Order FDs:

	ORDE...	DATE_OF_ORDER	ORDER_STATUS	EMP_NUM	CUSTOMER_ID
1	2	21-10-07	In-Progress	234	112
2	3	21-10-06	Shipped	235	112
3	4	21-10-21	Shipped	234	111
4	5	21-10-25	Received	236	113

Order_Id → {Date_of_Order, Order_status, Emp_Num}

Purchase FDs:

	PURCHAS...	ORDER_ID	GAME_ID	CONSOLE_ID	COUNT
1	2	2	12	(null)	1

Purchase_Id → {Game_ID, Console_ID, Count}

Assignment #7: Normalization/3NF

Console Table

	CONSOLE_ID	PRICE	NAME
1	39487	500	Nintendo Switch
2	39488	800	PlayStation 5
3	39489	600	Xbox Series X

Console_ID → {Price, Name}
1NF: This table has no multivalued (atomic values) attributes or nested relations, all entries in column are the same type and unique column identifiers are used.
2NF: All non-primary attributes are dependent on the primary key. The table does not have a primary key which contains values >1 therefore partial dependencies do not exist.
3NF: All non-primary attributes are determined Only by the primary key in the table therefore transitive dependencies do not exist.

Console Inventory Table

	COUNT	NUM_SALES	REGION_SALES	CONSOLE_ID
1	24	16384	10240	39487
2	10	10004	2440	39488
3	5	42332	10023	39489

Console_ID → {Count, Num_Sales, Region_Sales}
1NF: This table has no multivalued (atomic values) attributes or nested relations, all entries in column are the same type and unique column identifiers are used.
2NF: All non-primary attributes are dependent on the primary key. The table does not have a primary key which contains values >1 therefore partial dependencies do not exist.
3NF: All non-primary attributes are determined Only by the primary key in the table therefore transitive dependencies do not exist.

Game Table

	GAME_ID	PRICE	NAME	RELEASE_YEAR	ESRB_RATE	PUBLISHER	DESCRIPTION
1	10	3.14	Over 9000	1989	M	NT 3.51	Duis aute irure dolor in reprehenderit in voluptate velit e
2	11	79.99	SSMB Ultimate	2018	T	Nintendo	It is the fifth installment in the Super Smash Bros. series
3	12	59.99	Watch Dogs: Legion	2020	M	Ubisoft	It is the third instalment in the Watch Dogs series and the
4	13	39.99	Devil May Cry 5	2019	M	Capcom	It is the sixth installment overall and the fifth mainline

Game_Id → {Platform, Name, Price, Publisher, Esrb_rate, description}
1NF: This table has no multivalued (atomic values) attributes or nested relations, all entries in column are the same type and unique column identifiers are used.
2NF: All non-primary attributes are dependent on the primary key. The table does not have a primary key which contains values >1 therefore partial dependencies do not exist.
3NF: All non-primary attributes are determined Only by the primary key in the table therefore transitive dependencies do not exist.

Game Inventory Table

	COUNT	NUM_SALES	REGION_SALES	GAME_ID
1	5	194	22000	10
2	55	1000	842	12

Game_Id → {Count, Num_Sales, Region_Sales}
1NF: This table has no multivalued (atomic values) attributes or nested relations, all entries in column are the same type and unique column identifiers are used.
2NF: All non-primary attributes are dependent on the primary key. The table does not have a primary key which contains values >1 therefore partial dependencies do not exist.
3NF: All non-primary attributes are determined Only by the primary key in the table therefore transitive dependencies do not exist.

Customer Table

	CUSTOMER_ID	FIRST_NAME	LAST_NAME	EMAIL	CITY	COUNTRY	STREET	POSTAL_CODE
1	111	Baxter	Seuss	baxter.seuss@gmail.com	Toronto	Canada	Broadview	M7K8L6
2	112	Michelle	Gao	mvp@gmail.com	Brampton	Canada	Dixie	L3Y56S
3	113	John	Smith	jsmith@gmail.com	Toronto	Canada	Dundas	M9Q1UP

Customer_Id → {First_Name, Last_Name, Email, City, Country, Street, Postal_Code}
1NF: This table has no multivalued (atomic values) attributes or nested relations, all entries in column are the same type and unique column identifiers are used.
2NF: All non-primary attributes are dependent on the primary key. The table does not have a primary key which contains values >1 therefore partial dependencies do not exist.
3NF: All non-primary attributes are determined Only by the primary key in the table therefore transitive dependencies do not exist.

Employee Table

	EMP_NUM	FIRST_NAME	LAST_NAME	USERNAME	PASSWORD
1	234	Mickey	Mouse	M.Mouse	ChocolateCake18!
2	235	Donald	Duck	D.Donald	quackQuack
3	236	Pluto	Dog	P.Dog	WoofbarkWoof

Emp_Num → {First_Name, Last_Name, Username, Password}
1NF: This table has no multivalued (atomic values) attributes or nested relations, all entries in column are the same type and unique column identifiers are used.
2NF: All non-primary attributes are dependent on the primary key. The table does not have a primary key which contains values >1 therefore partial dependencies do not exist.
3NF: All non-primary attributes are determined Only by the primary key in the table therefore transitive dependencies do not exist.

Customer Order Table

	ORDER_ID	DATE_OF_ORDER	ORDER_STATUS	EMP_NUM	CUSTOMER_ID
1	2	21-10-07	In-Progress	234	112
2	3	21-10-06	Shipped	235	112
3	4	21-10-21	Shipped	234	111
4	5	21-10-25	Received	236	113

Order_Id → {Date_of_Order, Order_status, Emp_Num}
1NF: This table has no multivalued (atomic values) attributes or nested relations, all entries in column are the same type and unique column identifiers are used.
2NF: All non-primary attributes are dependent on the primary key. The table does not have a primary key which contains values >1 therefore partial dependencies do not exist.
3NF: All non-primary attributes are determined Only by the primary key in the table therefore transitive dependencies do not exist.

Purchase Table

	PURCHASE_ID	ORDER_ID	GAME_ID	CONSOLE_ID	COUNT
1	2	2	12	(null)	1

Purchase_Id → {Game_ID, Console_ID, Count}
1NF: This table has no multivalued (atomic values) attributes or nested relations, all entries in column are the same type and unique column identifiers are used.
2NF: All non-primary attributes are dependent on the primary key. The table does not have a primary key which contains values >1 therefore partial dependencies do not exist.
3NF: All non-primary attributes are determined Only by the primary key in the table therefore transitive dependencies do not exist.

Convert a table to 3NF
CUSTOMER TABLE
Before 1NF

Customer ID	First_Name	Last_Name	email_address	address_ID
111	Baxter	Seuss	baxter.seuss@gmail.com	{Toronto, Canada, Broadview, M7L8L6}
112	Michelle	Gao	mvp@gmail.com	{Brampton, Canada, Dixie L3Y56S}
113	John	Smith	jsmith@gmail.com	{Toronto, Canada, Dundas, M9Q1UP}

Since there's multi-values in the address_ID column, this does not satisfy 1NF requirements. Therefore, we will create a separate table for address_ID. To add a new address, we need a customer.

AFTER 1NF
CUSTOMER TABLE

Customer ID	First_Name	Last_Name	email_address	address_ID
111	Baxter	Seuss	baxter.seuss@gmail.com	1111
112	Michelle	Gao	mvp@gmail.com	1121
113	John	Smith	jsmith@gmail.com	1131

Address_ID TABLE

address_id	city	country	street	postal_code
1111	Toronto	Canada	Broadview	M7K BL6
1121	Brampton	Canada	Dixie	L3Y 565
1131	Toronto	Canada	Dundas	M9Q 1UP

Customer Table is now in 2NF, because there are no multi values in the table, and it depends on a single primary key which is the Customer_ID.
{Customer_ID} -> {First_Name, Last_Name, email_address, address_ID}

Making address_ID the foreign key in Customer Table allows us to link each customer to their respective address without multiple values in the previous customer table, thus 2NF. Furthermore, the current Customer table lacks any transitive dependencies, hence the customer table is also 3NF.

Assignment #8: Normalization of 3NF/BCNF Algorithm

Console Table: (Console_Id, Price, Name)

	CONSOLE_ID	PRICE	NAME
1	39487	500	Nintendo Switch
2	39488	800	PlayStation 5
3	39489	600	Xbox Series X

<u>Step 1: Determine functional dependencies</u>
Console_Id -> Price
Console_Id -> Name
<u>Step 2: Find redundancies</u>
We have to get rid of redundant dependencies. However, only console_id is able to get the price, and the name therefore there are no redundancies.
<u>Step 3: Find keys</u>
We only have one candidate key.
{Console_Id}
<u>Step 4: Find relations</u>
R1(Console_Id, Price, Name)

This table is already in BCNF.

Console Inventory Table

	COUNT	NUM_SALES	REGION_SALES	CONSOLE_ID
1	24	16384	10240	39487
2	10	10004	2440	39488
3	5	42332	10023	39489

<u>Step 1: Determine functional dependencies</u>
Console_Id -> Count
Console_Id -> Num_Sales
Console_Id -> Region_Sales

Since no FDs violate BCNF, this table is already in BCNF

Game Table

	GAME_ID	PRICE	NAME	RELEASE_YEAR	ESRB_RATE	PUBLISHER	DESCRIPTION
1	10	3.14	Over 9000	1989	M	NT 3.51	Duis aute irure dolor in reprehenderit in voluptate velit e
2	11	79.99	SSMB Ultimate	2018	T	Nintendo	It is the fifth installment in the Super Smash Bros. series
3	12	59.99	Watch Dogs: Legion	2020	M	Ubisoft	It is the third instalment in the Watch Dogs series and the
4	13	39.99	Devil May Cry 5	2019	M	Capcom	It is the sixth installment overall and the fifth mainline

<u>Step 1: Determine functional dependencies</u>
Game_Id -> Price
Game_Id -> Name
Game_Id -> Release_Year

Game_Id -> ESRB_Rate
Game_Id -> Publisher
Game_Id -> Description

No FDs violate BCNF, thus this table is BCNF.

Game Inventory Table

	COUNT	NUM_SALES	REGION_SALES	GAME_ID
1	5	194	22000	10
2	55	1000	842	12

<u>Step 1: Determine functional dependencies</u>
Game_Id -> Count
Game_Id -> Num_Sales
Game_Id -> Region_Sales

No FD violates BCNF, therefore this table is already BCNF.

Customer Table

	CUSTOMER_ID	FIRST_NAME	LAST_NAME	EMAIL	CITY	COUNTRY	STREET	POSTAL_CODE
1	111	Baxter	Seuss	baxter.seuss@gmail.com	Toronto	Canada	Broadview	M7K8L6
2	112	Michelle	Gao	mvp@gmail.com	Brampton	Canada	Dixie	L3Y56S
3	113	John	Smith	jsmith@gmail.com	Toronto	Canada	Dundas	M9Q1UP

<u>Step 1: Determine functional dependencies</u>
Customer_Id -> First_Name
Customer_Id -> Last_Name
Customer_Id -> Email
Customer_Id -> City
Customer_Id -> Country
Customer_Id -> Street
Customer_Id -> Postal_Code
Address -> City
Address -> Country
Address -> Street
Address -> Postal_Code
<u>Step 2: Find redundancies</u>
There is a redundancy with
Customer_Id -> City
Customer_Id -> Country
Customer_Id -> Street
Customer_Id -> Postal_Code
So we will remove those 4.
<u>Step 3: Find keys</u>
The candidate keys are
{Customer_Id} and {Address}

Step 4: Find relations
R1(Customer_Id, First_Name, Last_Name, Email, Address)
R2(Address, City, Country, Street, Postal_Code

Since neither are subsets of each other, both of these tables are now BCNF.

Employee Table

	EMP_NUM	FIRST_NAME	LAST_NAME	USERNAME	PASSWORD
1	234	Mickey	Mouse	M.Mouse	ChocolateCake18!
2	235	Donald	Duck	D.Donald	quackQuack
3	236	Pluto	Dog	P.Dog	WoofbarkWoof

Step 1: Determine functional dependencies
Emp_Num -> First_Name
Emp_Num -> Last_Name
Emp_Num -> Username
Emp_Num -> Password
Username -> Password
Step 2: Find redundancies
The last FD is redundant, so we can remove that.

Step 3: Find keys
We have one candidate key, because emp_num can be used to find the user_name FD
{Emp_num}
Step 4: Find relations
R1(Emp_num, First_Name, Last_Name, Username, Password).

This table is in BCNF.

Customer Order Table

	ORDER_ID	DATE_OF_ORDER	ORDER_STATUS	EMP_NUM	CUSTOMER_ID
1	2	21-10-07	In-Progress	234	112
2	3	21-10-06	Shipped	235	112
3	4	21-10-21	Shipped	234	111
4	5	21-10-25	Received	236	113

Step 1: Determine functional dependencies
Order_Id -> Date_Of_Order
Order_Id -> Order_Status
Order_Id -> Emp_Num
Order_Id -> Customer_Id

No FDs violate BCNF and hence this table is BCNF.

Purchase Table

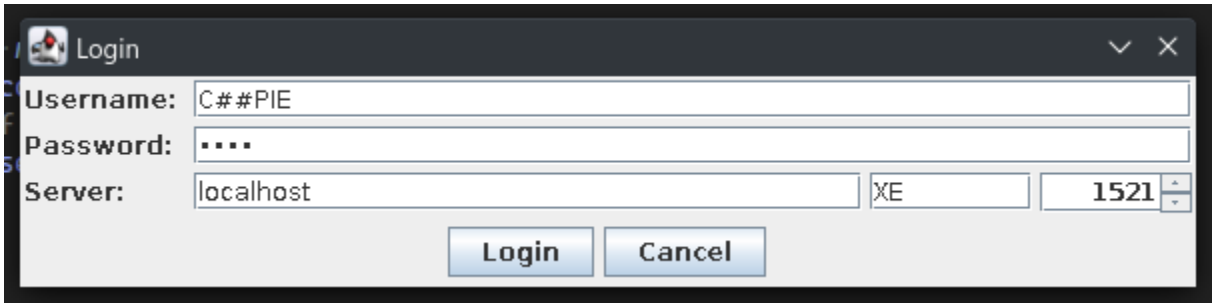
	PURCHASE_ID	ORDER_ID	GAME_ID	CONSOLE_ID	COUNT
1	2	2	12	(null)	1

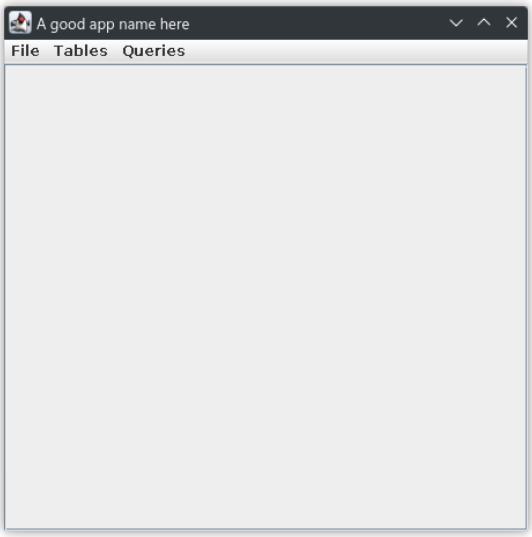
<u>Step 1: Determine functional dependencies</u>

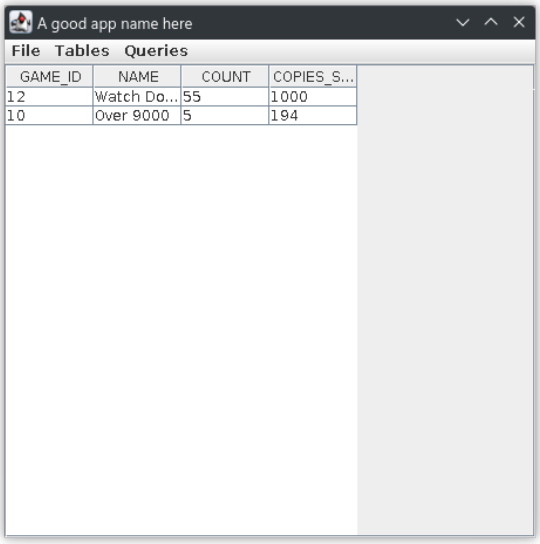
Purchase_Id -> Order_Id
Purchase_Id -> Game_Id
Purchase_Id -> Console_Id
Purchase_Id -> Count

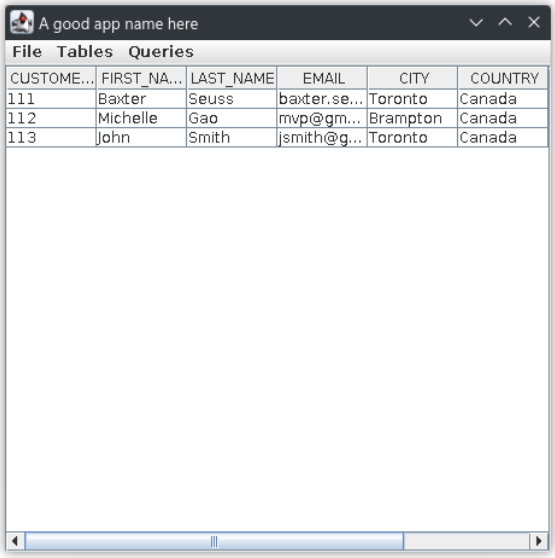
No FDs violate BCNF.

Assignment #9: Java/Web-based UI









Assignment #10: Relational Algebra Notation

Console Table

English	SQL	RA
get all existing consoles organized by ascending price	SELECT * FROM console ORDER BY price ASC;	console

Console Inventory Table

English	SQL	RA
get all existing consoles stock numbers	SELECT * FROM console_inventory_stored_in	console stock number
get all consoles where the stock is above 9	SELECT * FROM console_inventory_stored_in WHERE count > 9;	$\sigma_{count>9}$(console_inventory_stored_in)

Game Table

English	SQL	RA
get all existing games	SELECT * FROM game	game
get all existing games where the platform of the game is 39487(in our scenario, it is the nintendo switch id)	SELECT * FROM game WHERE platform = 39487;	$\sigma_{platform=39487}$(game)

Game Inventory Table

English	SQL	RA
get all existing games where inventory is less than 10	SELECT * FROM game_inventory_stored_in WHERE count < 10;	$\sigma_{count<10}$(game_inventory_stored_in)

Customer Table

English	SQL	RA

get all existing customers	SELECT * FROM customer	customer
Show the count of customers that are in each city and group them	SELECT COUNT(customer_id), city FROM customer GROUP BY city ORDER BY COUNT(customer_id) DESC;	$(\pi_{COUNT(customer\_id),city}(\sigma_{count<10}(customer)))$

<u>Employee Table</u>

English	SQL	RA
get all employees	SELECT * FROM employee;	employee
Select the first name of the employee that packed orders post 10/22/2021	SELECT first_name FROM employee WHERE emp_num IN (SELECT emp_num FROM customer_order WHERE date_of_order > to_date('10/22/2021', 'mm/dd/yyyy'));	$(\pi_{first\_name}(\sigma_{emp\_num=123}(\pi_{emp\_num}(\sigma_{date\_of\_order>to\_date('10/22/2021','mm/dd/yyyy')}(customer\_order))(employee)))$

<u>Customer Order Table</u>

English	SQL	RA
get all existing customer orders that are currently being shipped	SELECT * FROM customer_order WHERE order_status LIKE 'Shipped';	$(\sigma_{order\_status='Shipped'}(customer\_order))$
Select customer id that currently has an order in progress	SELECT DISTINCT customer_id FROM customer_order WHERE order_status LIKE 'In-Progress';	$(\pi_{DISTINCT(customer\_id)}(\sigma_{order\_status='In-Progress'}(customer\_order)))$

<u>Purchases</u>

English	SQL	RA
get all existing purchase orders	SELECT * FROM purchase;	purchase

ADVANCED QUERIES

English	SQL	RA
Get customer id and corresponding id in addition to employees first name and employee number	SELECT customer_id, first_name FROM customer UNION SELECT emp_num, first_name FROM employee;	$(\pi_{customer\_id,first\_name}(\sigma_{customer\_id}(customer))) \cup (\pi_{emp\_num,first\_name}(\sigma_{emp\_num}(employee)))$
Get all existing order number and the employee number that packed that order	SELECT COUNT(order_id) AS order_number, emp_num AS employee_number FROM customer_order INNER JOIN customer ON customer_order.customer_id = customer.customer_id GROUP BY emp_num;	$\rho(order\_number(order\_id),\rho(employee\_number(emp\_num)))(\pi_{COUNT(order\_id)}(\sigma_{customer\_id}(customer\_id)))$
Select customer who's order number and purchase order id match, customer id match, and game id match	SELECT DISTINCT c.first_name, c.last_name, g.name AS game_title FROM purchase p, customer_order o, game g, customer c WHERE p.order_id = o.order_id AND o.customer_id = c.customer_id AND p.game_id = g.game_id;	$\rho(game\_title(g.name))(\pi_{DISTINCT(c.first\_name,\ c.last\_name,\ g.name}(\sigma_{p.order\_id = o.order\_id,\ o.customer\_id = c.customer\_id,\ p.game\_id = g.game\_id}(customer \| games \| order)))$

Conclusion

The document summarizes all the processes taken of our videogame store Database. It utilizes SQL and the fundamentals of database management in order to manage and display the information required to operate a hospital. Many layers of normal forms were explored which were needed to normalize the tables storing information on important components of a video game database such as the
customer and order tables.The SQL queries made during the course allows for easy retrieval for key information. These queries were later rewritten using Relational Algebra notation to yield instances of

relations as output. Nevertheless, this project has given our group the skills required to build any database system.