Video Game Retailer Transaction Database

Designed by Kevin Bruce

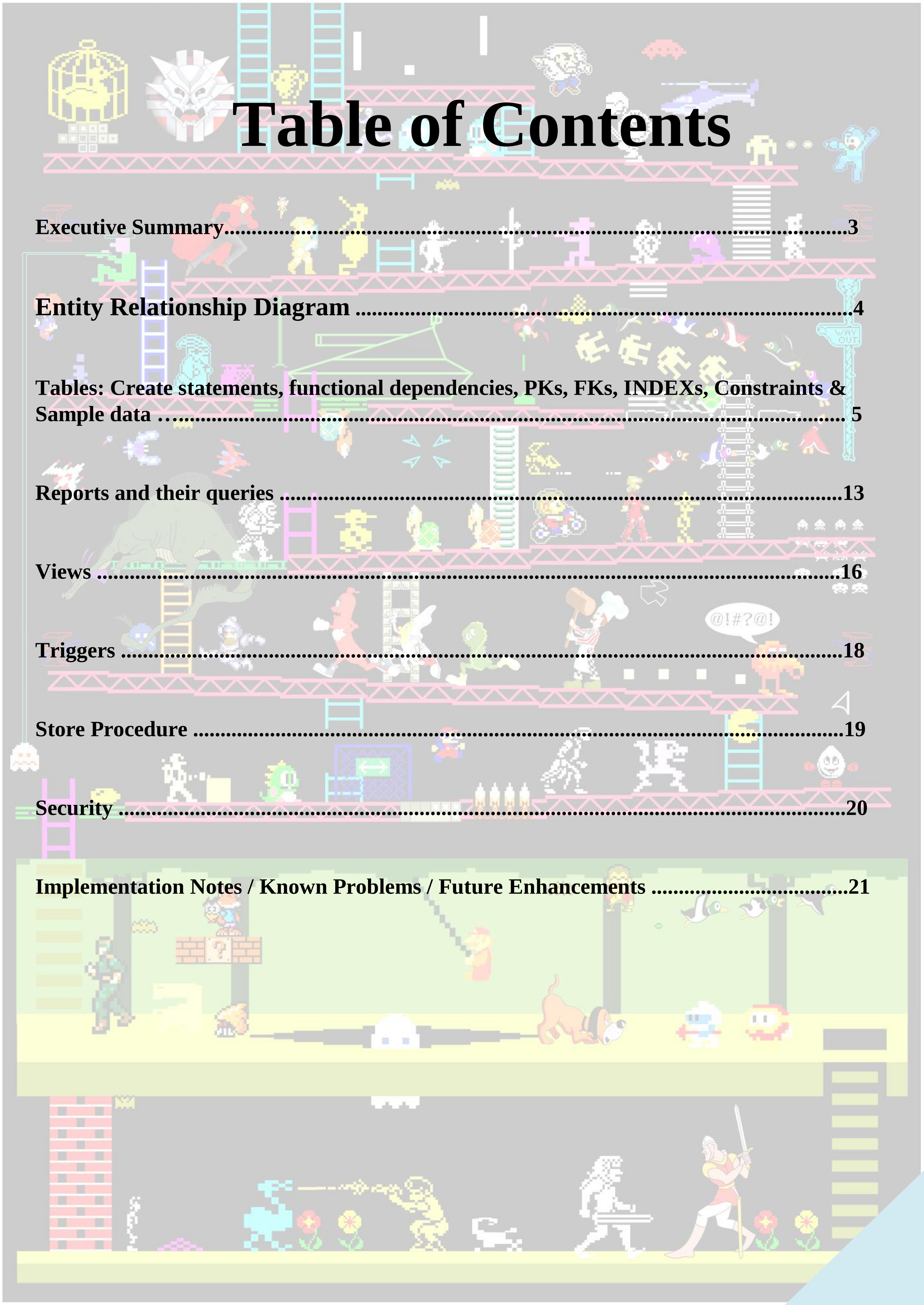
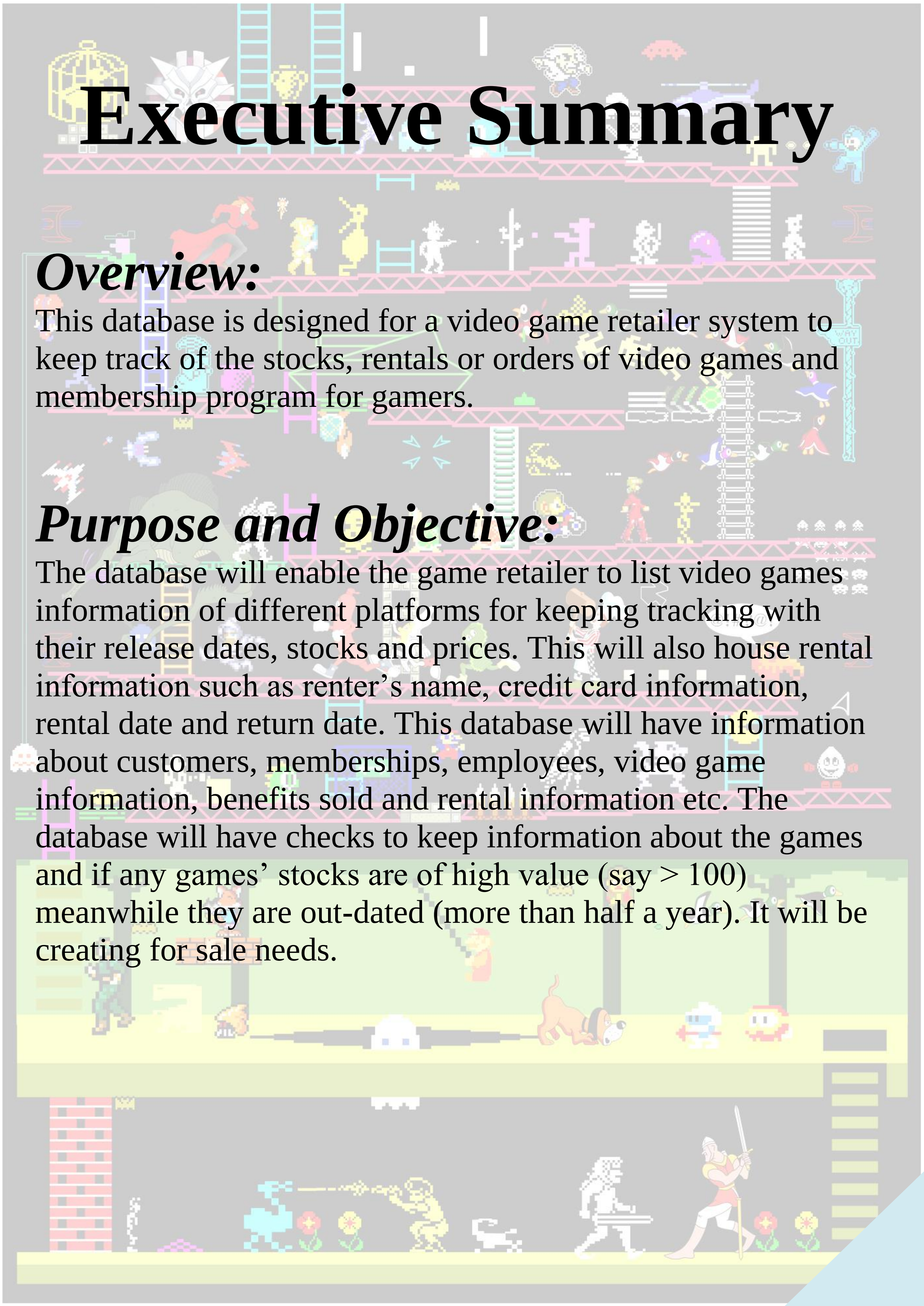


Table of Contents



Executive Summary

Overview:

This database is designed for a video game retailer system to keep track of the stocks, rentals or orders of video games and membership program for gamers.

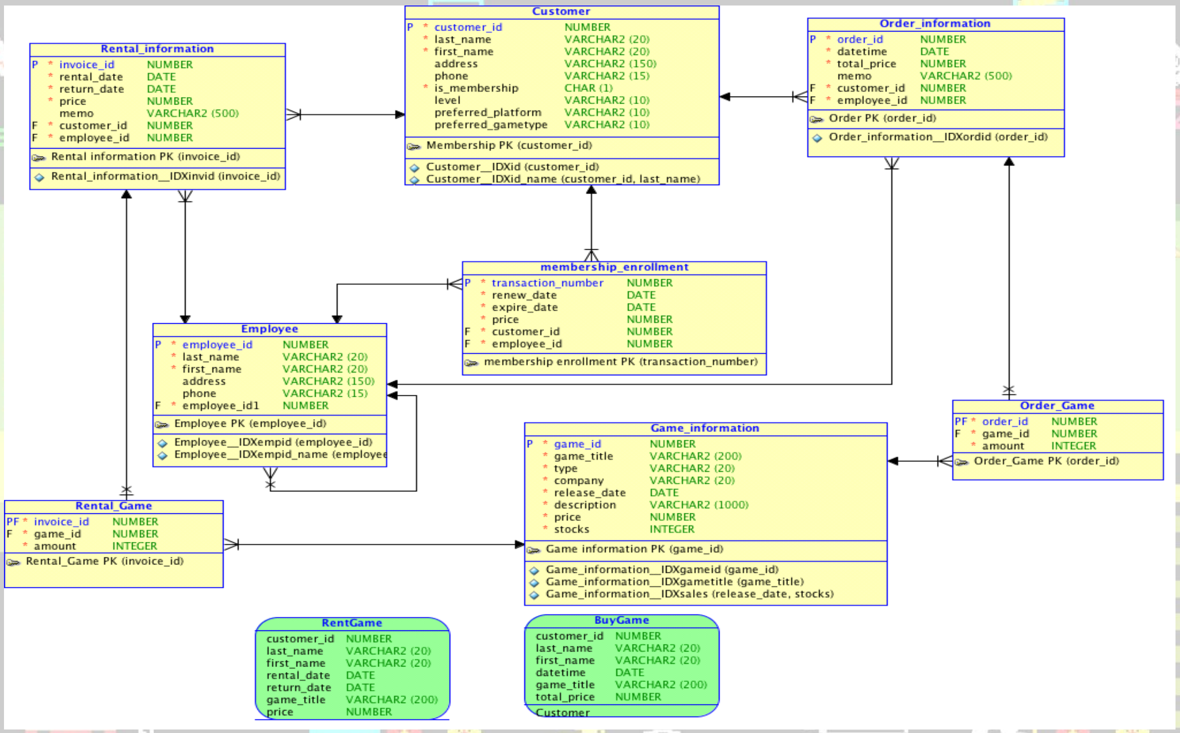
Purpose and Objective:

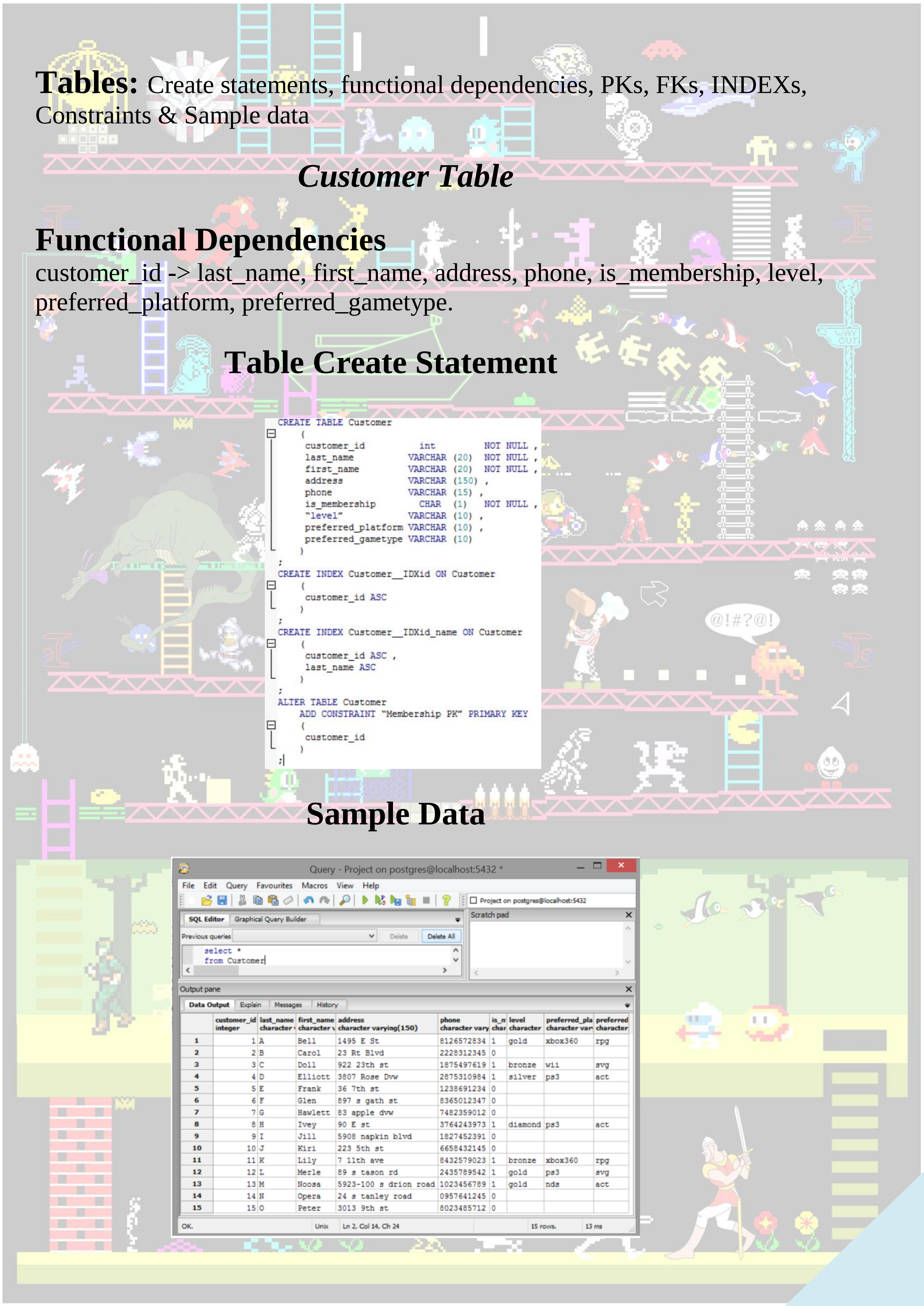
The database will enable the game retailer to list video games information of different platforms for keeping tracking with their release dates, stocks and prices. This will also house rental information such as renter's name, credit card information, rental date and return date. This database will have information about customers, memberships, employees, video game information, benefits sold and rental information etc. The database will have checks to keep information about the games and if any games' stocks are of high value (say > 100) meanwhile they are out-dated (more than half a year). It will be creating for sale needs.

Entity Relationship Diagram

Major Entities and their descriptions

- Customer: customer_id, last_name, first_name, address, phone, is_membership, level, preferred_platform, preferred_gametype.
- Membership_enrollment: transaction_number, renew_date, expire_date, price, customer_id, employee_id.
- Employee: employee_id, last_name, first_name, address, phone, manager_id.
- Game_information: game_id, game_title, type, company, release date, description, price, stocks.
- Rental_information: invoice_id, rental_date, return_date, price, memo, customer_id, employee_id.
- Rental_game: invoice_id, game_id, amount.
- Order_information: order_id, datetime, total price, memo, custom_id, employee_id.
- Order_game: order_id, game_id, amount.





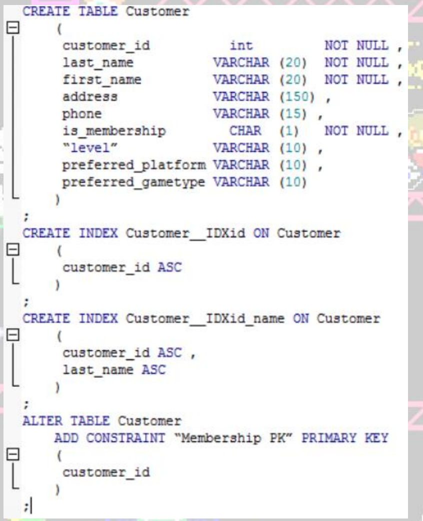
Tables: Create statements, functional dependencies, PKs, FKs, INDEXs, Constraints & Sample data

Customer Table

Functional Dependencies

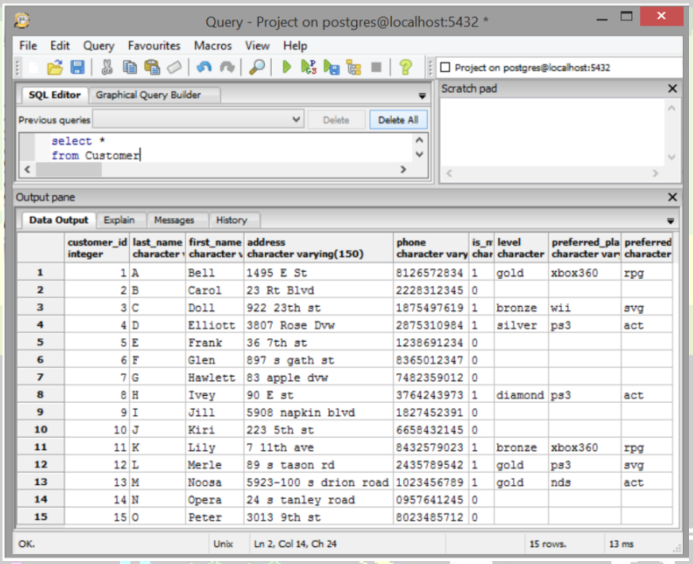
customer_id -> last_name, first_name, address, phone, is_membership, level, preferred_platform, preferred_gametype.

Table Create Statement



```
CREATE TABLE Customer
    (
     customer_id          int          NOT NULL ,
     last_name          VARCHAR (20)   NOT NULL ,
     first_name         VARCHAR (20)   NOT NULL ,
     address            VARCHAR (150) ,
     phone              VARCHAR (15) ,
     is_membership        CHAR  (1)    NOT NULL ,
     "level"            VARCHAR (10) ,
     preferred_platform VARCHAR (10) ,
     preferred_gametype VARCHAR (10)
    )
;
CREATE INDEX Customer__IDXid ON Customer
    (
     customer_id ASC
    )
;
CREATE INDEX Customer__IDXid_name ON Customer
    (
     customer_id ASC ,
     last_name ASC
    )
;
ALTER TABLE Customer
    ADD CONSTRAINT "Membership PK" PRIMARY KEY
    (
     customer_id
    )
;
```

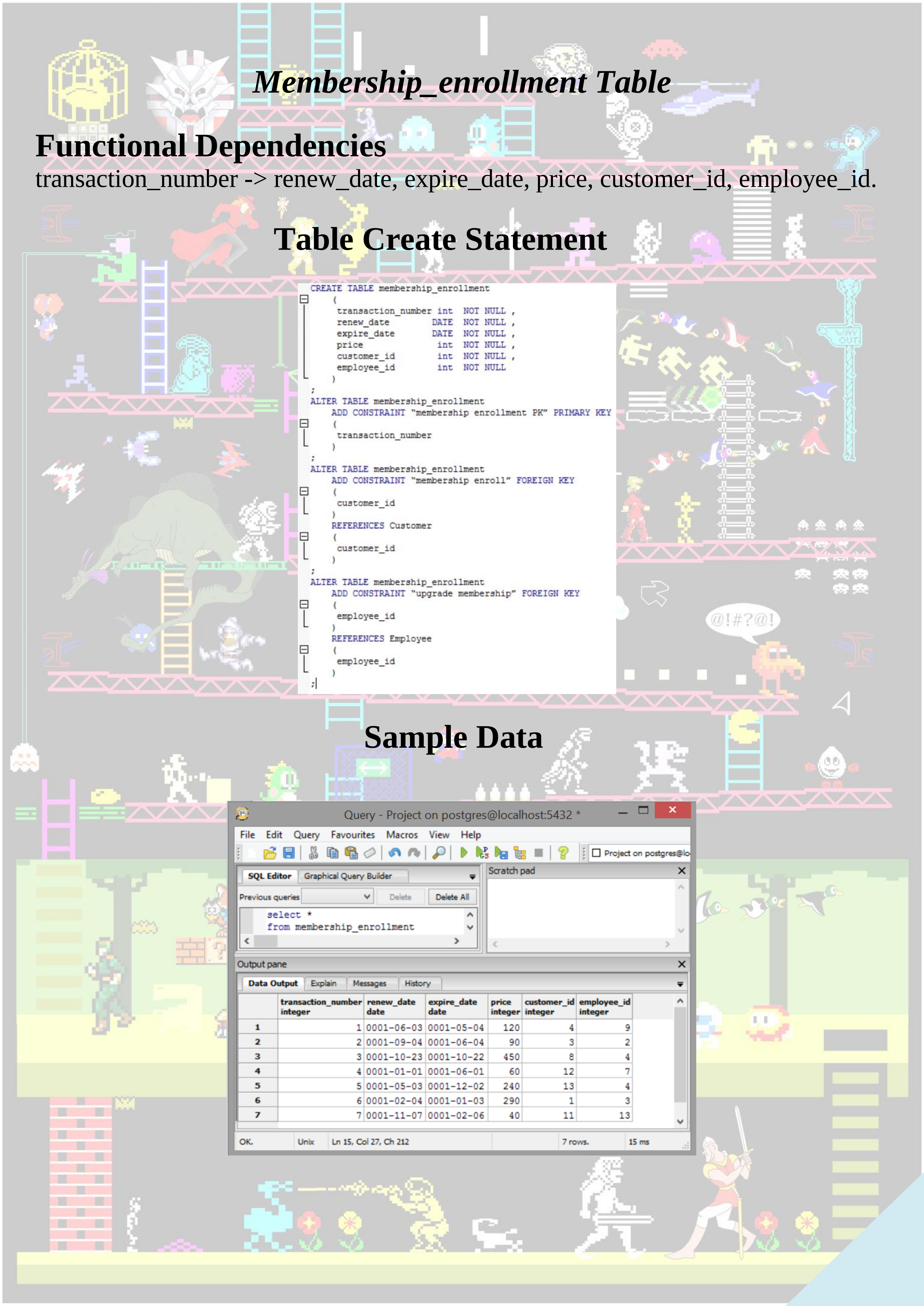
Sample Data


Query - Project on postgres@localhost:5432 *
File Edit Query Favourites Macros View Help
Project on postgres@localhost:5432
SQL Editor
Graphical Query Builder
Scratch pad
Previous queries
Delete
Delete All
Output pane
Data Output
Explain
Messages
History

```
select *
from Customer
```

```
    customer_id last_name  first_name address                 phone        is_m level    preferred_pla preferred
    integer     character  character  character varying(150)  character vary char character character var character
1             1 A          Bell       1495 E St               8126572834   1    gold     xbox360       rpg
2             2 B          Carol      23 Rt Blvd              2228312345   0
3             3 C          Doll       922 23th st             1875497619   1    bronze   wii           svg
4             4 D          Elliott    3807 Rose Dvw           2875310984   1    silver   ps3           act
5             5 E          Frank      36 7th st               1238691234   0
6             6 F          Glen       897 s gath st           8365012347   0
7             7 G          Hawlett    83 apple dvw            7482359012   0
8             8 H          Ivey       90 E st                 3764243973   1    diamond  ps3           act
9             9 I          Jill       5908 napkin blvd        1827452391   0
10           10 J          Kiri       223 5th st              6658432145   0
11           11 K          Lily       7 11th ave              8432579023   1    bronze   xbox360       rpg
12           12 L          Merle      89 s tason rd           2435789542   1    gold     ps3           svg
13           13 M          Noosa      5923-100 s drion road   1023456789   1    gold     nds           act
14           14 N          Opera      24 s tanley road        0957641245   0
15           15 O          Peter      3013 9th st             8023485712   0
```

```
OK.    Unix    Ln 2, Col 14, Ch 24    15 rows.    13 ms
```

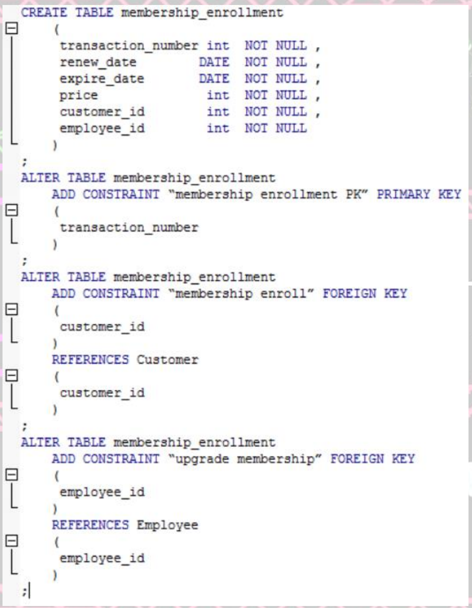


Membership_enrollment Table

Functional Dependencies

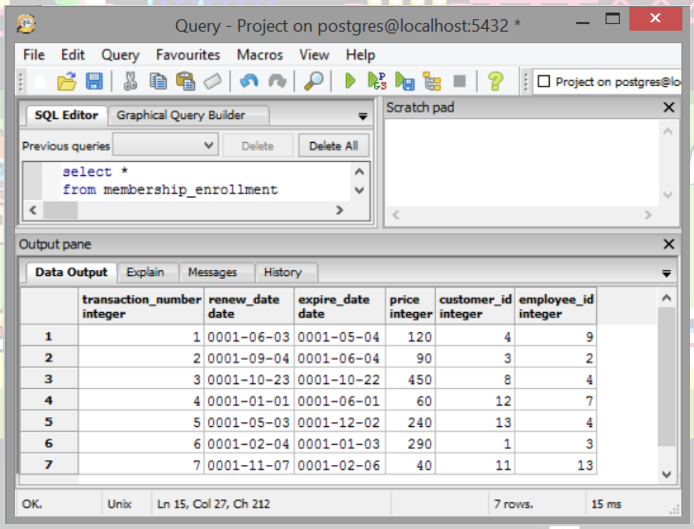
transaction_number -> renew_date, expire_date, price, customer_id, employee_id.

Table Create Statement



```
CREATE TABLE membership_enrollment
    (
     transaction_number int  NOT NULL ,
     renew_date        DATE  NOT NULL ,
     expire_date       DATE  NOT NULL ,
     price              int  NOT NULL ,
     customer_id        int  NOT NULL ,
     employee_id        int  NOT NULL
    )
;
ALTER TABLE membership_enrollment
    ADD CONSTRAINT "membership enrollment PK" PRIMARY KEY
    (
     transaction_number
    )
;
ALTER TABLE membership_enrollment
    ADD CONSTRAINT "membership enroll" FOREIGN KEY
    (
     customer_id
    )
    REFERENCES Customer
    (
     customer_id
    )
;
ALTER TABLE membership_enrollment
    ADD CONSTRAINT "upgrade membership" FOREIGN KEY
    (
     employee_id
    )
    REFERENCES Employee
    (
     employee_id
    )
;
```

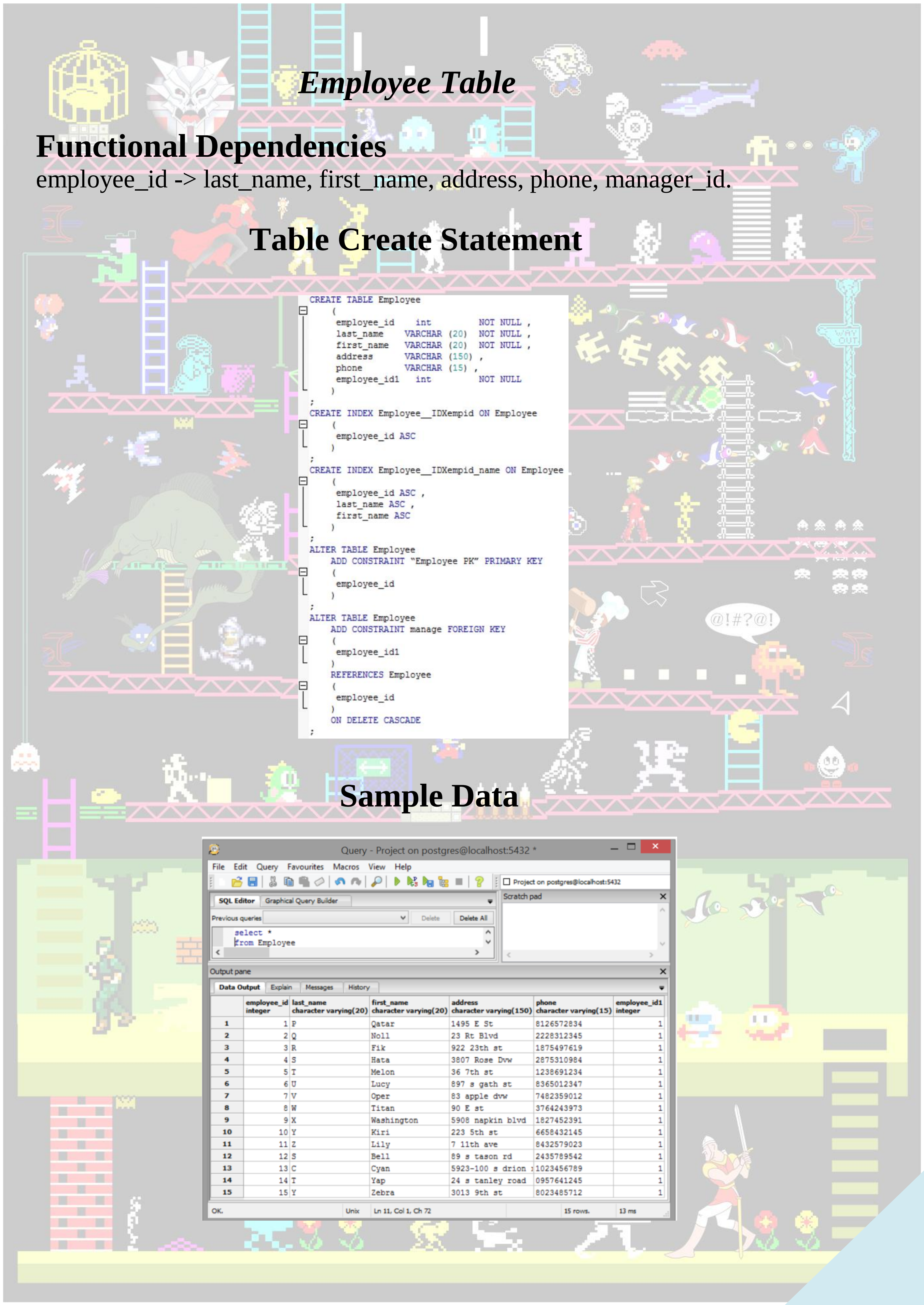
Sample Data



```
select *
from membership_enrollment
```

	transaction_number integer	renew_date date	expire_date date	price integer	customer_id integer	employee_id integer
1	1	0001-06-03	0001-05-04	120	4	9
2	2	0001-09-04	0001-06-04	90	3	2
3	3	0001-10-23	0001-10-22	450	8	4
4	4	0001-01-01	0001-06-01	60	12	7
5	5	0001-05-03	0001-12-02	240	13	4
6	6	0001-02-04	0001-01-03	290	1	3
7	7	0001-11-07	0001-02-06	40	11	13

```
OK.    Unix    Ln 15, Col 27, Ch 212    7 rows.    15 ms
```

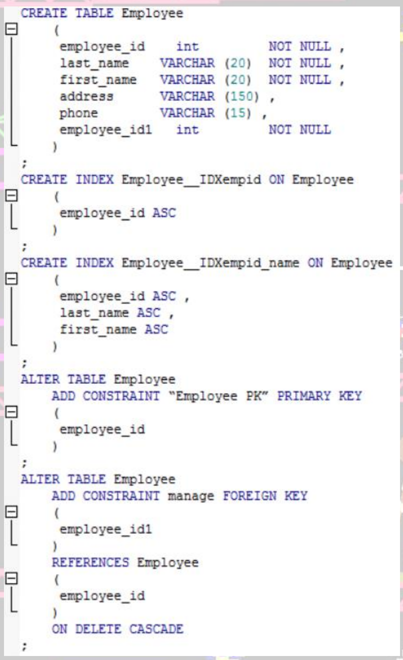


Employee Table

Functional Dependencies

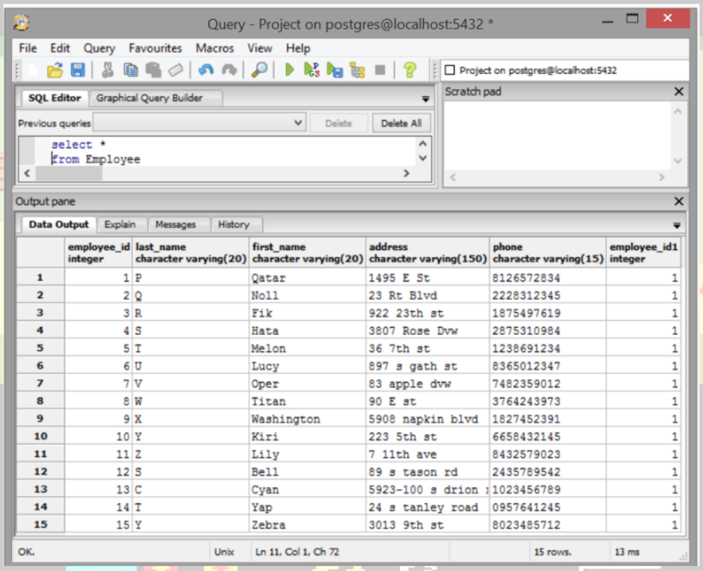
employee_id -> last_name, first_name, address, phone, manager_id.

Table Create Statement



```
CREATE TABLE Employee
     (
      employee_id    int           NOT NULL ,
      last_name    VARCHAR (20)  NOT NULL ,
      first_name   VARCHAR (20)  NOT NULL ,
      address      VARCHAR (150) ,
      phone        VARCHAR (15) ,
      employee_id1   int           NOT NULL
     )
;
CREATE INDEX Employee__IDXempid ON Employee
     (
      employee_id ASC
     )
;
CREATE INDEX Employee__IDXempid_name ON Employee
     (
      employee_id ASC ,
      last_name ASC ,
      first_name ASC
     )
;
ALTER TABLE Employee
    ADD CONSTRAINT "Employee PK" PRIMARY KEY
     (
      employee_id
     )
;
ALTER TABLE Employee
    ADD CONSTRAINT manage FOREIGN KEY
     (
      employee_id1
     )
    REFERENCES Employee
     (
      employee_id
     )
    ON DELETE CASCADE
;
```

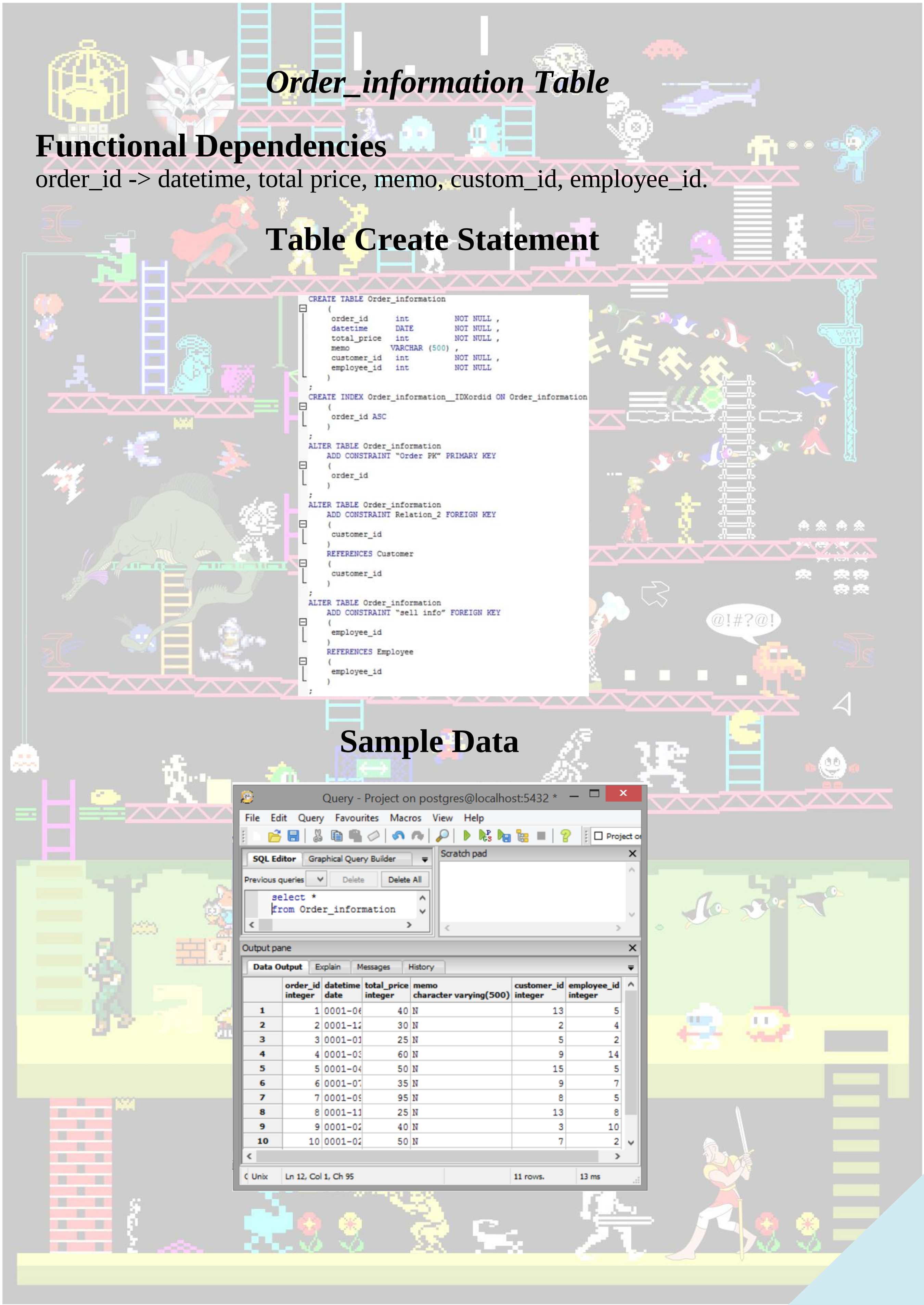
Sample Data


Query - Project on postgres@localhost:5432 *
File Edit Query Favourites Macros View Help
SQL Editor
Graphical Query Builder
Previous queries
Delete
Delete All
Project on postgres@localhost:5432
Scratch pad
Output pane
Data Output
Explain
Messages
History

```
select *
from Employee
```

	employee_id integer	last_name character varying(20)	first_name character varying(20)	address character varying(150)	phone character varying(15)	employee_id1 integer
1	1	P	Qatar	1495 E St	8126572834	1
2	2	Q	Noll	23 Rt Blvd	2228312345	1
3	3	R	Fik	922 23th st	1875497619	1
4	4	S	Hata	3807 Rose Dvw	2875310984	1
5	5	T	Melon	36 7th st	1238691234	1
6	6	U	Lucy	897 s gath st	8365012347	1
7	7	V	Oper	83 apple dvw	7482359012	1
8	8	W	Titan	90 E st	3764243973	1
9	9	X	Washington	5908 napkin blvd	1827452391	1
10	10	Y	Kiri	223 5th st	6658432145	1
11	11	Z	Lily	7 11th ave	8432579023	1
12	12	S	Bell	89 s tason rd	2435789542	1
13	13	C	Cyan	5923-100 s drion :	1023456789	1
14	14	T	Yap	24 s tanley road	0957641245	1
15	15	Y	Zebra	3013 9th st	8023485712	1

OK. Unix Ln 11, Col 1, Ch 72 15 rows. 13 ms

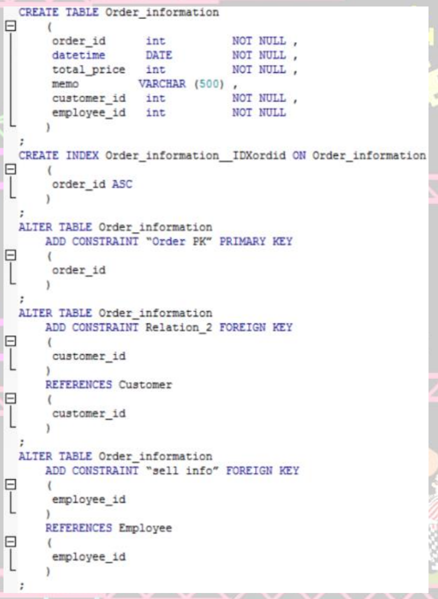


Order_information Table

Functional Dependencies

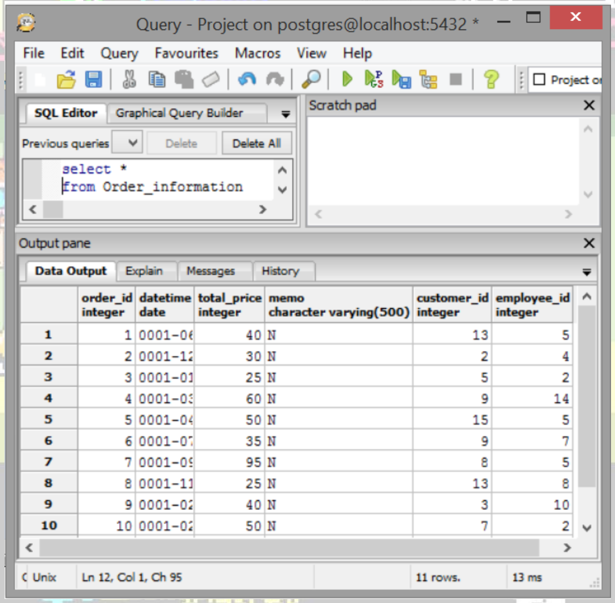
order_id -> datetime, total price, memo, custom_id, employee_id.

Table Create Statement



```
CREATE TABLE Order_information
    (
     order_id      int            NOT NULL ,
     datetime      DATE           NOT NULL ,
     total_price   int            NOT NULL ,
     memo         VARCHAR (500) ,
     customer_id   int            NOT NULL ,
     employee_id   int            NOT NULL
    )
;
CREATE INDEX Order_information__IDXordid ON Order_information
    (
     order_id ASC
    )
;
ALTER TABLE Order_information
    ADD CONSTRAINT "Order PK" PRIMARY KEY
    (
     order_id
    )
;
ALTER TABLE Order_information
    ADD CONSTRAINT Relation_2 FOREIGN KEY
    (
     customer_id
    )
    REFERENCES Customer
    (
     customer_id
    )
;
ALTER TABLE Order_information
    ADD CONSTRAINT "sell info" FOREIGN KEY
    (
     employee_id
    )
    REFERENCES Employee
    (
     employee_id
    )
;
```

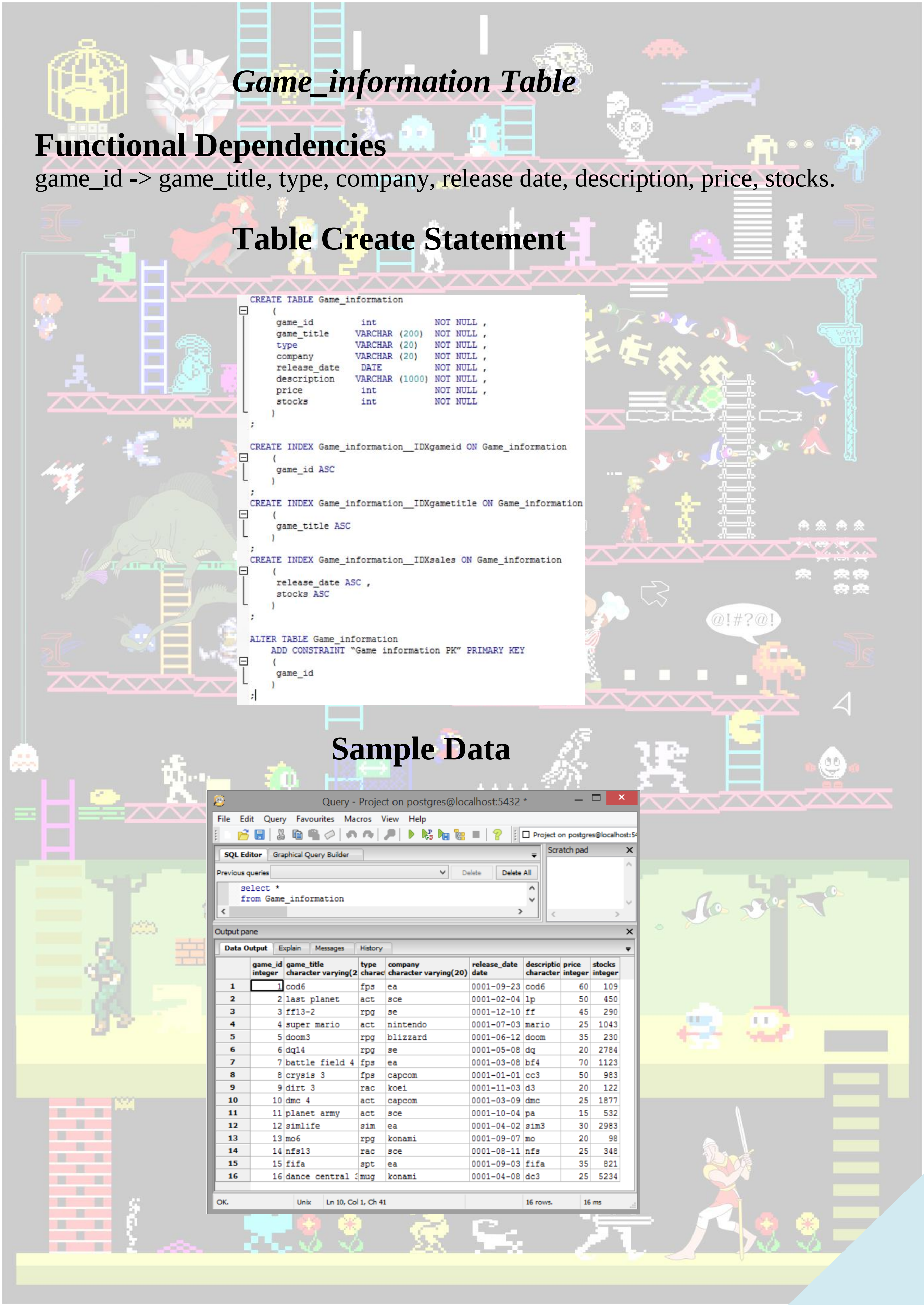
Sample Data



```
select *
from Order_information
```

	order_id integer	datetime date	total_price integer	memo character varying(500)	customer_id integer	employee_id integer
1	1	0001-06	40	N	13	5
2	2	0001-12	30	N	2	4
3	3	0001-01	25	N	5	2
4	4	0001-03	60	N	9	14
5	5	0001-04	50	N	15	5
6	6	0001-07	35	N	9	7
7	7	0001-09	95	N	8	5
8	8	0001-11	25	N	13	8
9	9	0001-02	40	N	3	10
10	10	0001-02	50	N	7	2

```
Unix    Ln 12, Col 1, Ch 95    11 rows.    13 ms
```

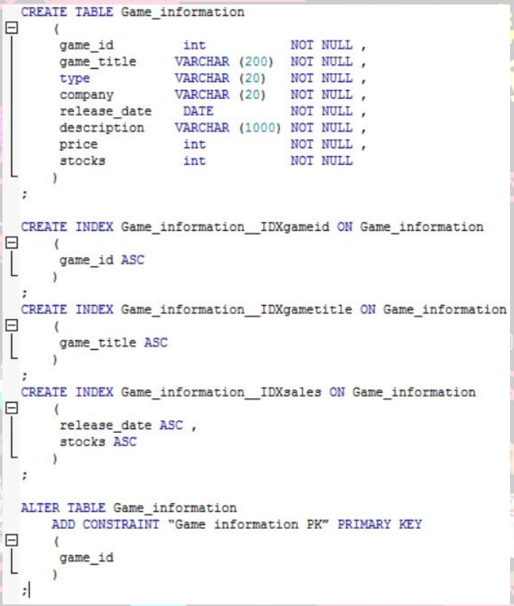


Game_information Table

Functional Dependencies

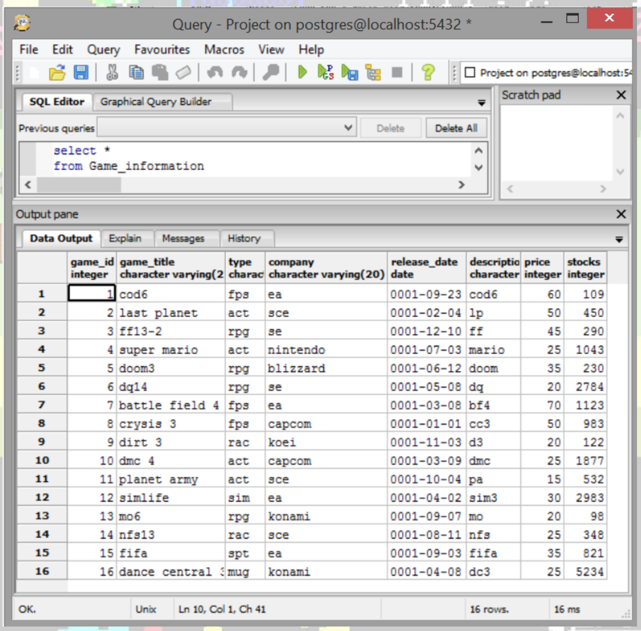
game_id -> game_title, type, company, release date, description, price, stocks.

Table Create Statement



```
CREATE TABLE Game_information
    (
     game_id          int            NOT NULL ,
     game_title       VARCHAR (200)  NOT NULL ,
     type             VARCHAR (20)   NOT NULL ,
     company          VARCHAR (20)   NOT NULL ,
     release_date      DATE          NOT NULL ,
     description      VARCHAR (1000) NOT NULL ,
     price             int           NOT NULL ,
     stocks            int           NOT NULL
    )
;

CREATE INDEX Game_information__IDXgameid ON Game_information
    (
     game_id ASC
    )
;
CREATE INDEX Game_information__IDXgametitle ON Game_information
    (
     game_title ASC
    )
;
CREATE INDEX Game_information__IDXsales ON Game_information
    (
     release_date ASC ,
     stocks ASC
    )
;

ALTER TABLE Game_information
    ADD CONSTRAINT "Game information PK" PRIMARY KEY
    (
     game_id
    )
;
```

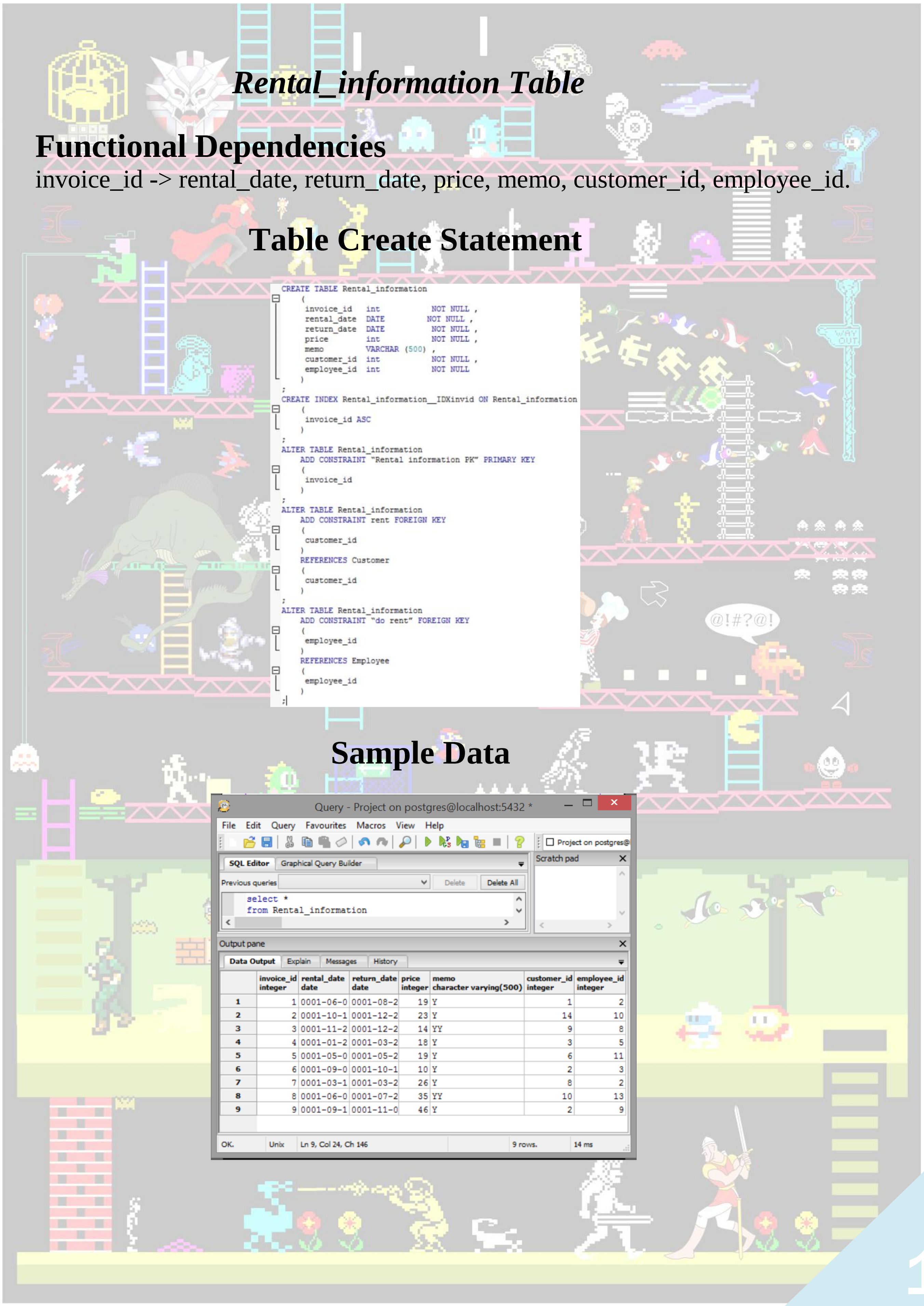
Sample Data


Query - Project on postgres@localhost:5432 *
File Edit Query Favourites Macros View Help
Project on postgres@localhost:5432
SQL Editor
Graphical Query Builder
Previous queries
Delete
Delete All
Scratch pad
Output pane
Data Output
Explain
Messages
History

```
select *
from Game_information
```

	game_id integer	game_title character varying(2	type charac	company character varying(20)	release_date date	descriptio character	price integer	stocks integer
1	1	cod6	fps	ea	0001-09-23	cod6	60	109
2	2	last planet	act	sce	0001-02-04	lp	50	450
3	3	ff13-2	rpg	se	0001-12-10	ff	45	290
4	4	super mario	act	nintendo	0001-07-03	mario	25	1043
5	5	doom3	rpg	blizzard	0001-06-12	doom	35	230
6	6	dq14	rpg	se	0001-05-08	dq	20	2784
7	7	battle field 4	fps	ea	0001-03-08	bf4	70	1123
8	8	crysis 3	fps	capcom	0001-01-01	cc3	50	983
9	9	dirt 3	rac	koei	0001-11-03	d3	20	122
10	10	dmc 4	act	capcom	0001-03-09	dmc	25	1877
11	11	planet army	act	sce	0001-10-04	pa	15	532
12	12	simlife	sim	ea	0001-04-02	sim3	30	2983
13	13	mo6	rpg	konami	0001-09-07	mo	20	98
14	14	nfs13	rac	sce	0001-08-11	nfs	25	348
15	15	fifa	spt	ea	0001-09-03	fifa	35	821
16	16	dance central 3	mug	konami	0001-04-08	dc3	25	5234

OK. Unix Ln 10, Col 1, Ch 41 16 rows. 16 ms

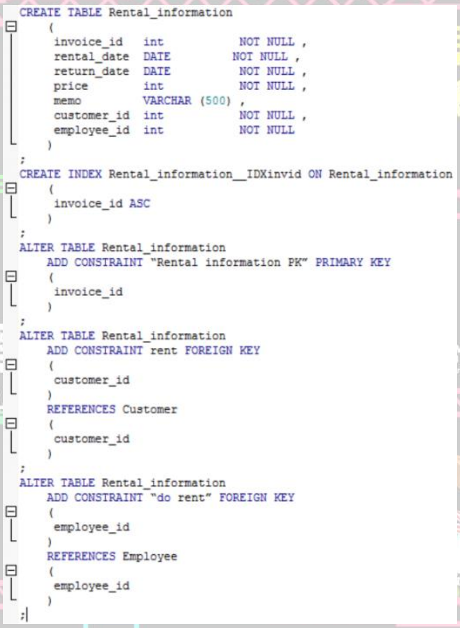


Rental_information Table

Functional Dependencies

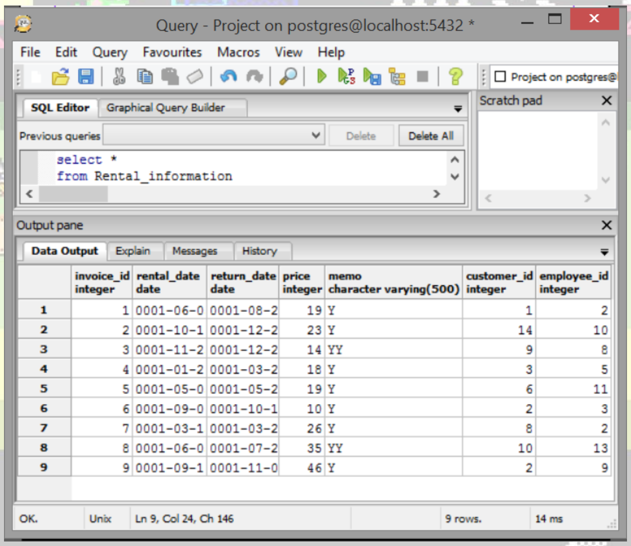
invoice_id -> rental_date, return_date, price, memo, customer_id, employee_id.

Table Create Statement



```
CREATE TABLE Rental_information
    (
     invoice_id   int            NOT NULL ,
     rental_date  DATE          NOT NULL ,
     return_date  DATE           NOT NULL ,
     price        int            NOT NULL ,
     memo         VARCHAR (500) ,
     customer_id  int            NOT NULL ,
     employee_id  int            NOT NULL
    )
;
CREATE INDEX Rental_information__IDXinvid ON Rental_information
    (
     invoice_id ASC
    )
;
ALTER TABLE Rental_information
    ADD CONSTRAINT "Rental information PK" PRIMARY KEY
    (
     invoice_id
    )
;
ALTER TABLE Rental_information
    ADD CONSTRAINT rent FOREIGN KEY
    (
     customer_id
    )
    REFERENCES Customer
    (
     customer_id
    )
;
ALTER TABLE Rental_information
    ADD CONSTRAINT "do rent" FOREIGN KEY
    (
     employee_id
    )
    REFERENCES Employee
    (
     employee_id
    )
;
```

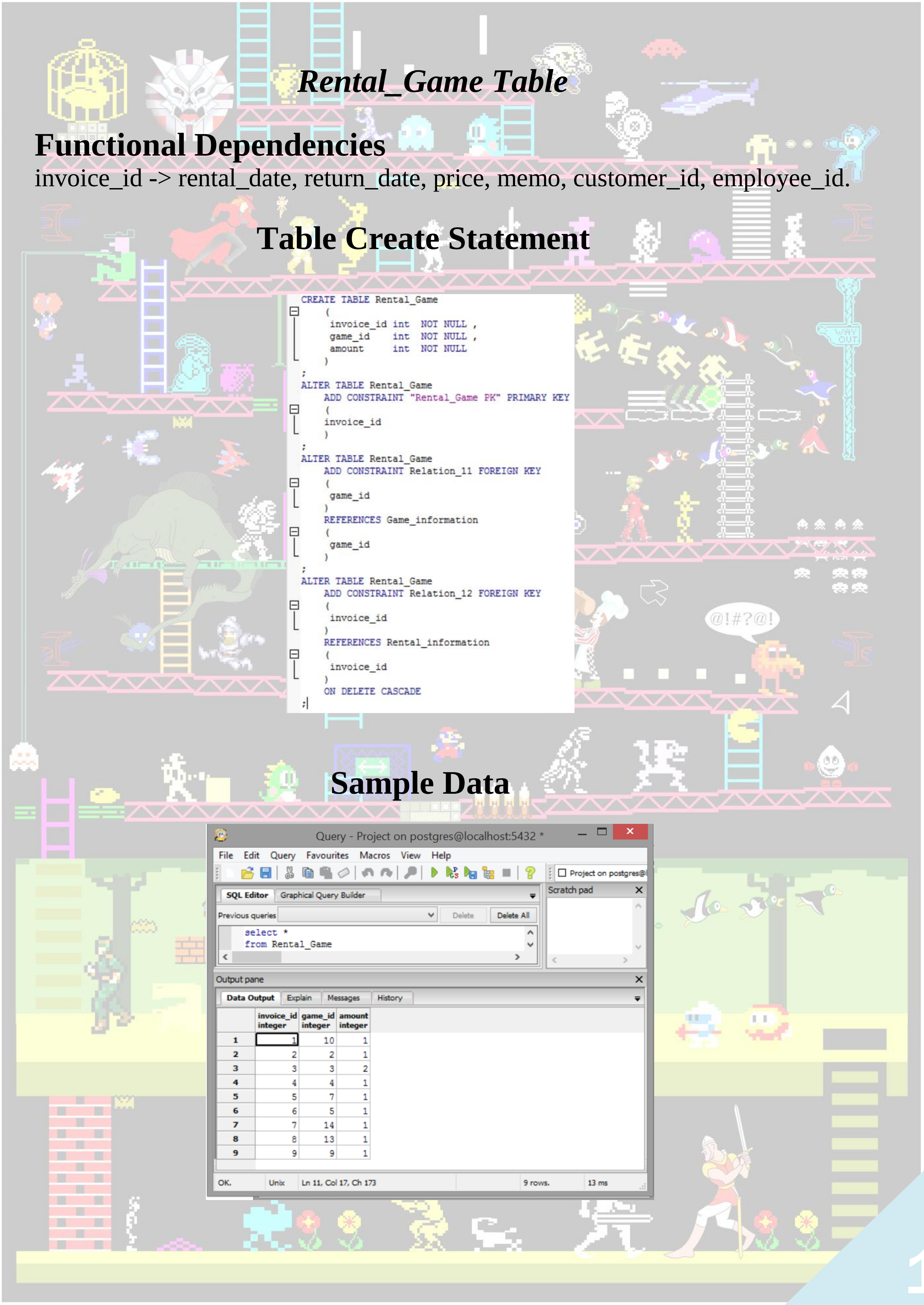
Sample Data



```
select *
from Rental_information
```

	invoice_id integer	rental_date date	return_date date	price integer	memo character varying(500)	customer_id integer	employee_id integer
1	1	0001-06-0	0001-08-2	19	Y	1	2
2	2	0001-10-1	0001-12-2	23	Y	14	10
3	3	0001-11-2	0001-12-2	14	YY	9	8
4	4	0001-01-2	0001-03-2	18	Y	3	5
5	5	0001-05-0	0001-05-2	19	Y	6	11
6	6	0001-09-0	0001-10-1	10	Y	2	3
7	7	0001-03-1	0001-03-2	26	Y	8	2
8	8	0001-06-0	0001-07-2	35	YY	10	13
9	9	0001-09-1	0001-11-0	46	Y	2	9

```
OK.    Unix    Ln 9, Col 24, Ch 146        9 rows.    14 ms
```

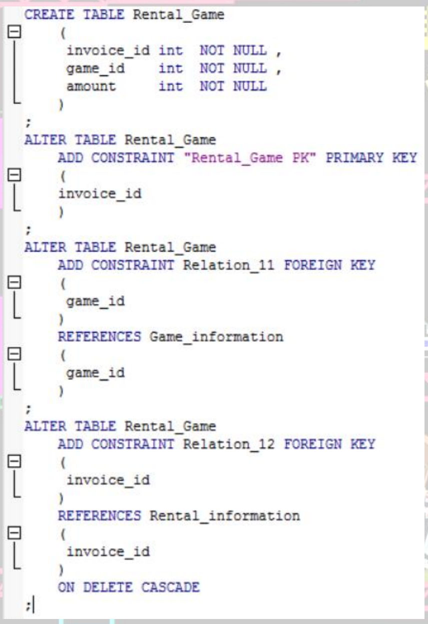


Rental_Game Table

Functional Dependencies

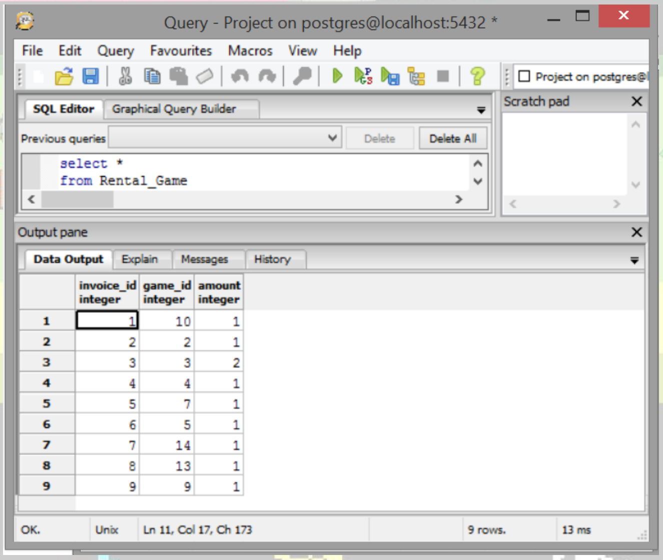
invoice_id -> rental_date, return_date, price, memo, customer_id, employee_id.

Table Create Statement



```
CREATE TABLE Rental_Game
    (
     invoice_id int  NOT NULL ,
     game_id    int  NOT NULL ,
     amount     int  NOT NULL
    )
;
ALTER TABLE Rental_Game
    ADD CONSTRAINT "Rental_Game PK" PRIMARY KEY
    (
    invoice_id
    )
;
ALTER TABLE Rental_Game
    ADD CONSTRAINT Relation_11 FOREIGN KEY
    (
     game_id
    )
    REFERENCES Game_information
    (
     game_id
    )
;
ALTER TABLE Rental_Game
    ADD CONSTRAINT Relation_12 FOREIGN KEY
    (
     invoice_id
    )
    REFERENCES Rental_information
    (
     invoice_id
    )
    ON DELETE CASCADE
;
```

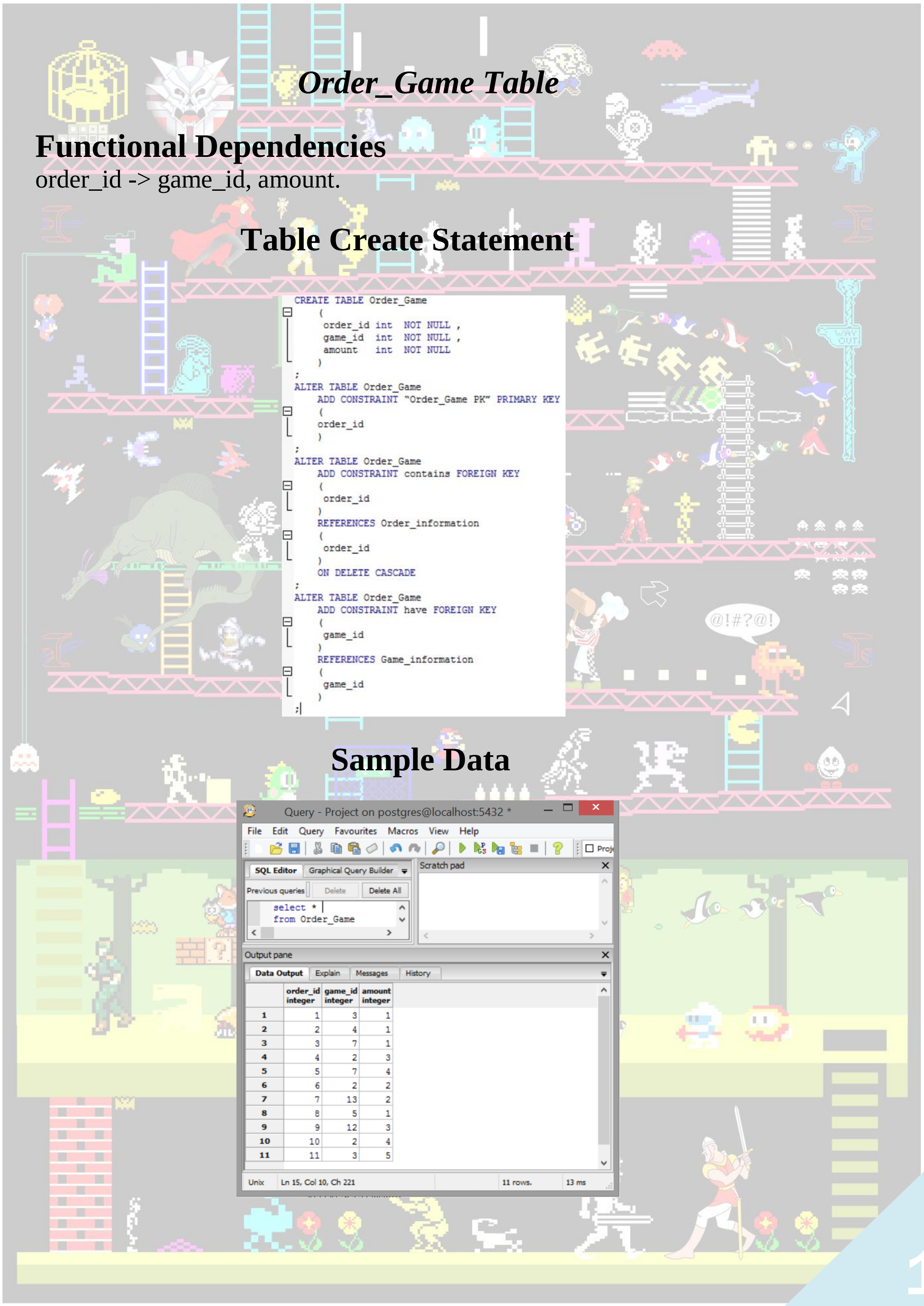
Sample Data



```
select *
from Rental_Game
```

	invoice_id integer	game_id integer	amount integer
1	1	10	1
2	2	2	1
3	3	3	2
4	4	4	1
5	5	7	1
6	6	5	1
7	7	14	1
8	8	13	1
9	9	9	1

OK. Unix Ln 11, Col 17, Ch 173 9 rows. 13 ms

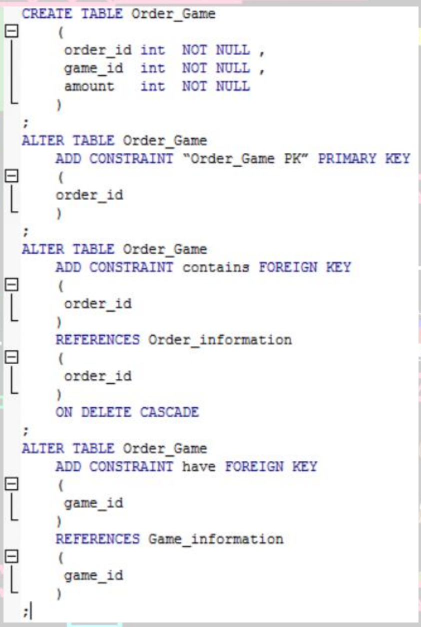


Order_Game Table

Functional Dependencies

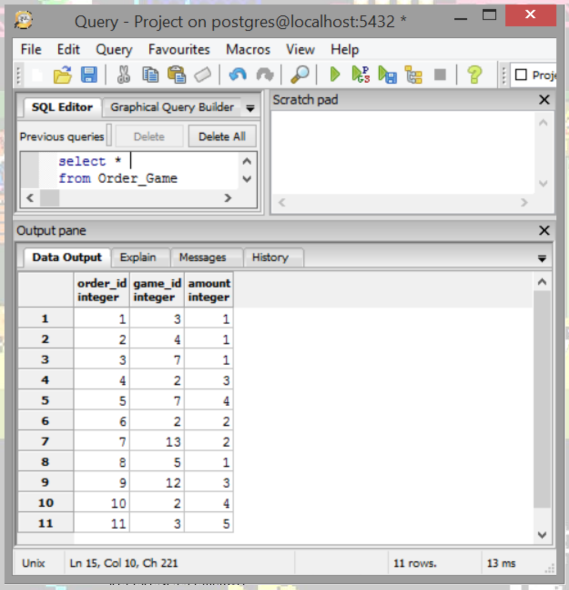
order_id -> game_id, amount.

Table Create Statement



```
CREATE TABLE Order_Game
    (
     order_id int  NOT NULL ,
     game_id  int  NOT NULL ,
     amount   int  NOT NULL
    )
;
ALTER TABLE Order_Game
    ADD CONSTRAINT "Order_Game PK" PRIMARY KEY
    (
    order_id
    )
;
ALTER TABLE Order_Game
    ADD CONSTRAINT contains FOREIGN KEY
    (
     order_id
    )
    REFERENCES Order_information
    (
     order_id
    )
    ON DELETE CASCADE
;
ALTER TABLE Order_Game
    ADD CONSTRAINT have FOREIGN KEY
    (
     game_id
    )
    REFERENCES Game_information
    (
     game_id
    )
;
```

Sample Data



```
select *
from Order_Game
```

	order_id integer	game_id integer	amount integer
1	1	3	1
2	2	4	1
3	3	7	1
4	4	2	3
5	5	7	4
6	6	2	2
7	7	13	2
8	8	5	1
9	9	12	3
10	10	2	4
11	11	3	5

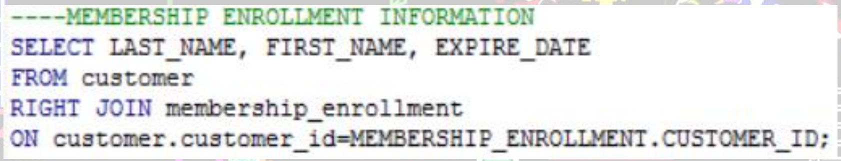
Unix   Ln 15, Col 10, Ch 221   11 rows.   13 ms

Reports and their queries

MEMBERSHIP ENROLLMENT INFORMATION

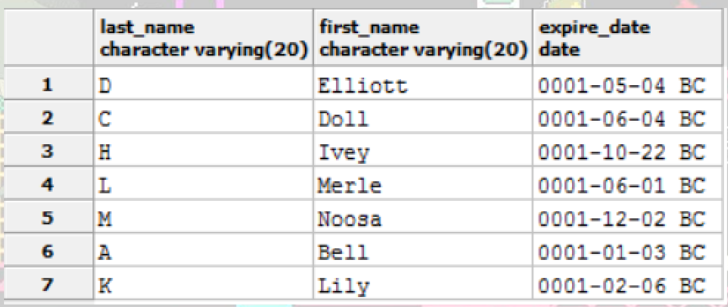
Purpose: Displays all records for customers who have a membership, and the date it will expire. References order_information table.

Code



```
----MEMBERSHIP ENROLLMENT INFORMATION
SELECT LAST_NAME, FIRST_NAME, EXPIRE_DATE
FROM customer
RIGHT JOIN membership_enrollment
ON customer.customer_id=MEMBERSHIP_ENROLLMENT.CUSTOMER_ID;
```

Sample

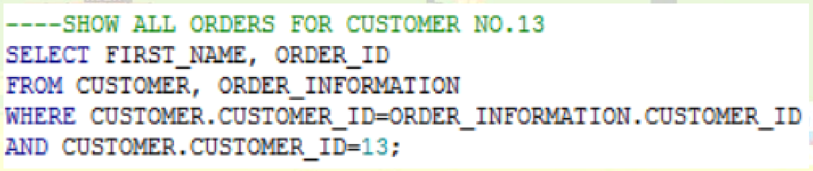


	last_name character varying(20)	first_name character varying(20)	expire_date date
1	D	Elliott	0001-05-04 BC
2	C	Doll	0001-06-04 BC
3	H	Ivey	0001-10-22 BC
4	L	Merle	0001-06-01 BC
5	M	Noosa	0001-12-02 BC
6	A	Bell	0001-01-03 BC
7	K	Lily	0001-02-06 BC

CUSTOMER NO.13

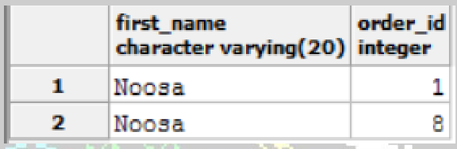
Purpose: Displays all records on orders made by customer no. 13. References customer and order_information table.

Code

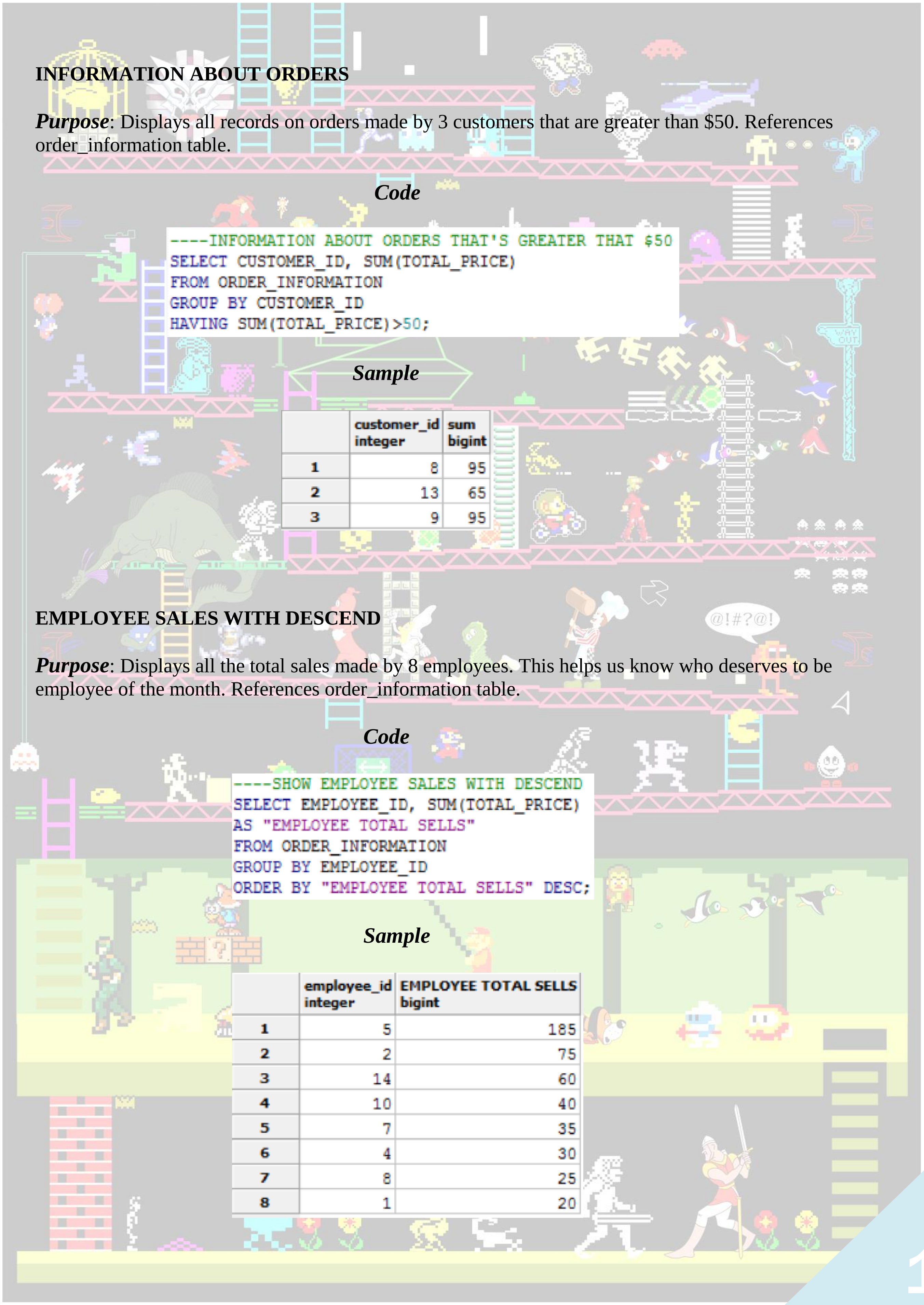


```
----SHOW ALL ORDERS FOR CUSTOMER NO.13
SELECT FIRST_NAME, ORDER_ID
FROM CUSTOMER, ORDER_INFORMATION
WHERE CUSTOMER.CUSTOMER_ID=ORDER_INFORMATION.CUSTOMER_ID
AND CUSTOMER.CUSTOMER_ID=13;
```

Sample



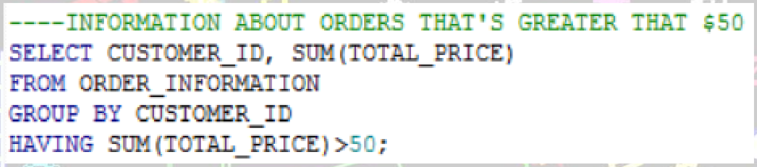
	first_name character varying(20)	order_id integer
1	Noosa	1
2	Noosa	8



INFORMATION ABOUT ORDERS

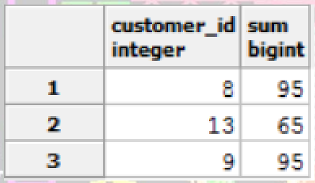
Purpose: Displays all records on orders made by 3 customers that are greater than $50. References order_information table.

Code



```
----INFORMATION ABOUT ORDERS THAT'S GREATER THAT $50
SELECT CUSTOMER_ID, SUM(TOTAL_PRICE)
FROM ORDER_INFORMATION
GROUP BY CUSTOMER_ID
HAVING SUM(TOTAL_PRICE)>50;
```

Sample

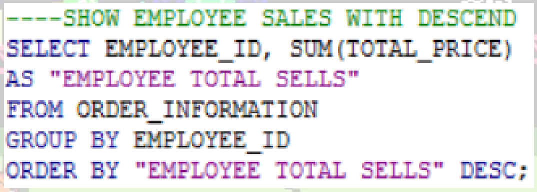


	customer_id integer	sum bigint
1	8	95
2	13	65
3	9	95

EMPLOYEE SALES WITH DESCEND

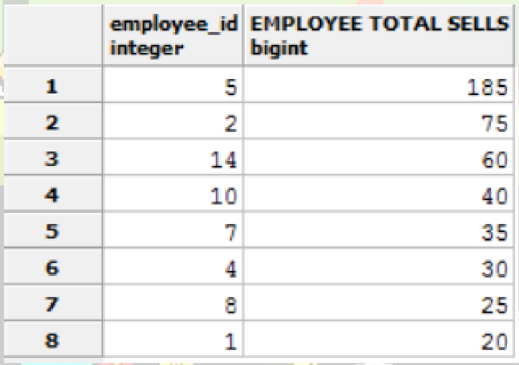
Purpose: Displays all the total sales made by 8 employees. This helps us know who deserves to be employee of the month. References order_information table.

Code

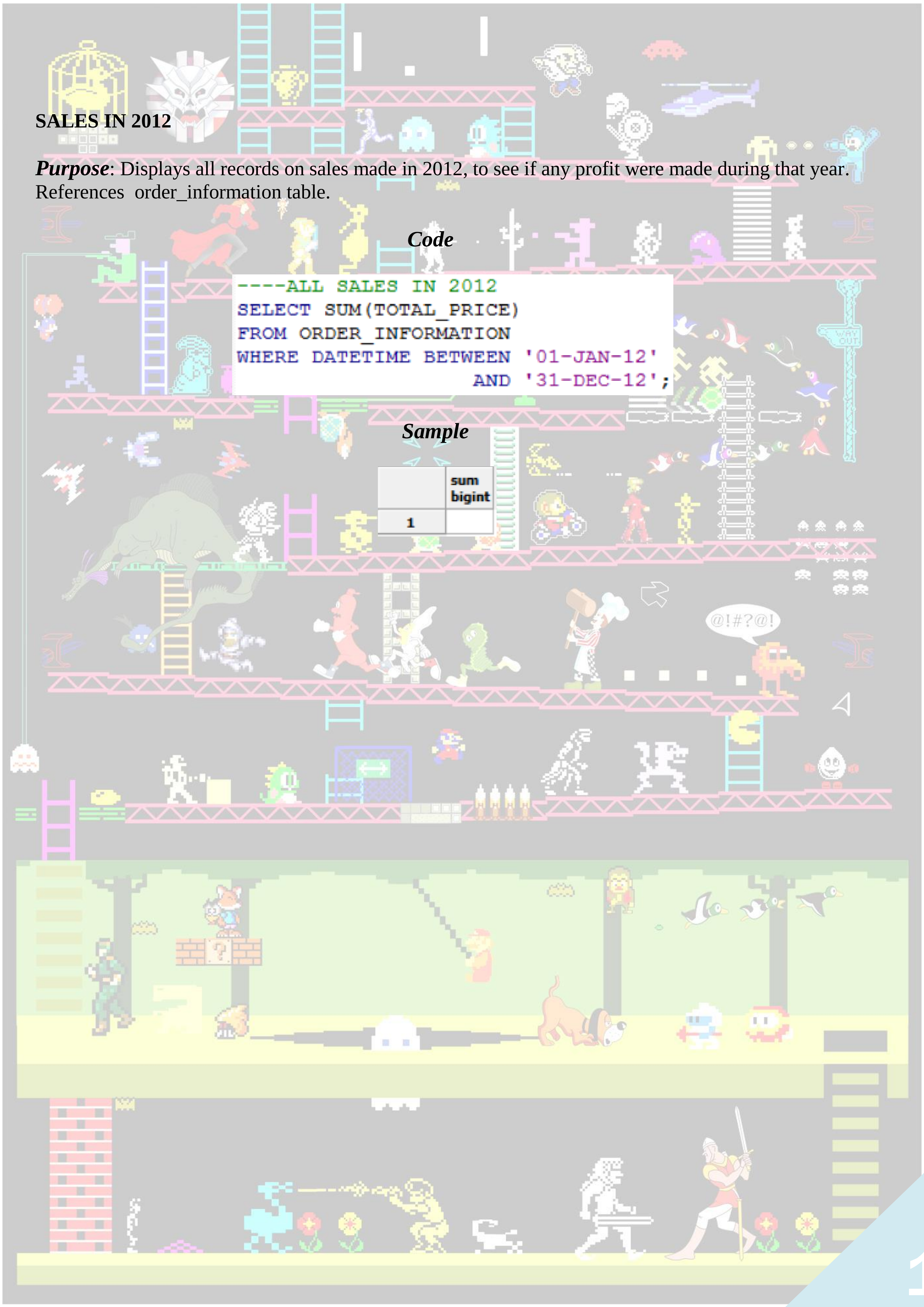


```
----SHOW EMPLOYEE SALES WITH DESCEND
SELECT EMPLOYEE_ID, SUM(TOTAL_PRICE)
AS "EMPLOYEE TOTAL SELLS"
FROM ORDER_INFORMATION
GROUP BY EMPLOYEE_ID
ORDER BY "EMPLOYEE TOTAL SELLS" DESC;
```

Sample



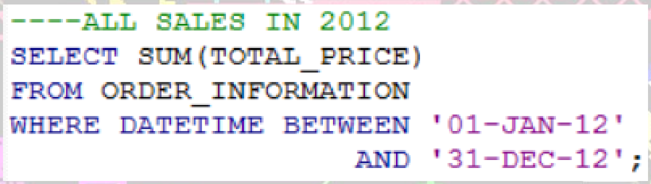
	employee_id integer	EMPLOYEE TOTAL SELLS bigint
1	5	185
2	2	75
3	14	60
4	10	40
5	7	35
6	4	30
7	8	25
8	1	20



SALES IN 2012

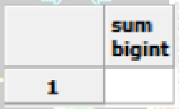
Purpose: Displays all records on sales made in 2012, to see if any profit were made during that year. References order_information table.

Code

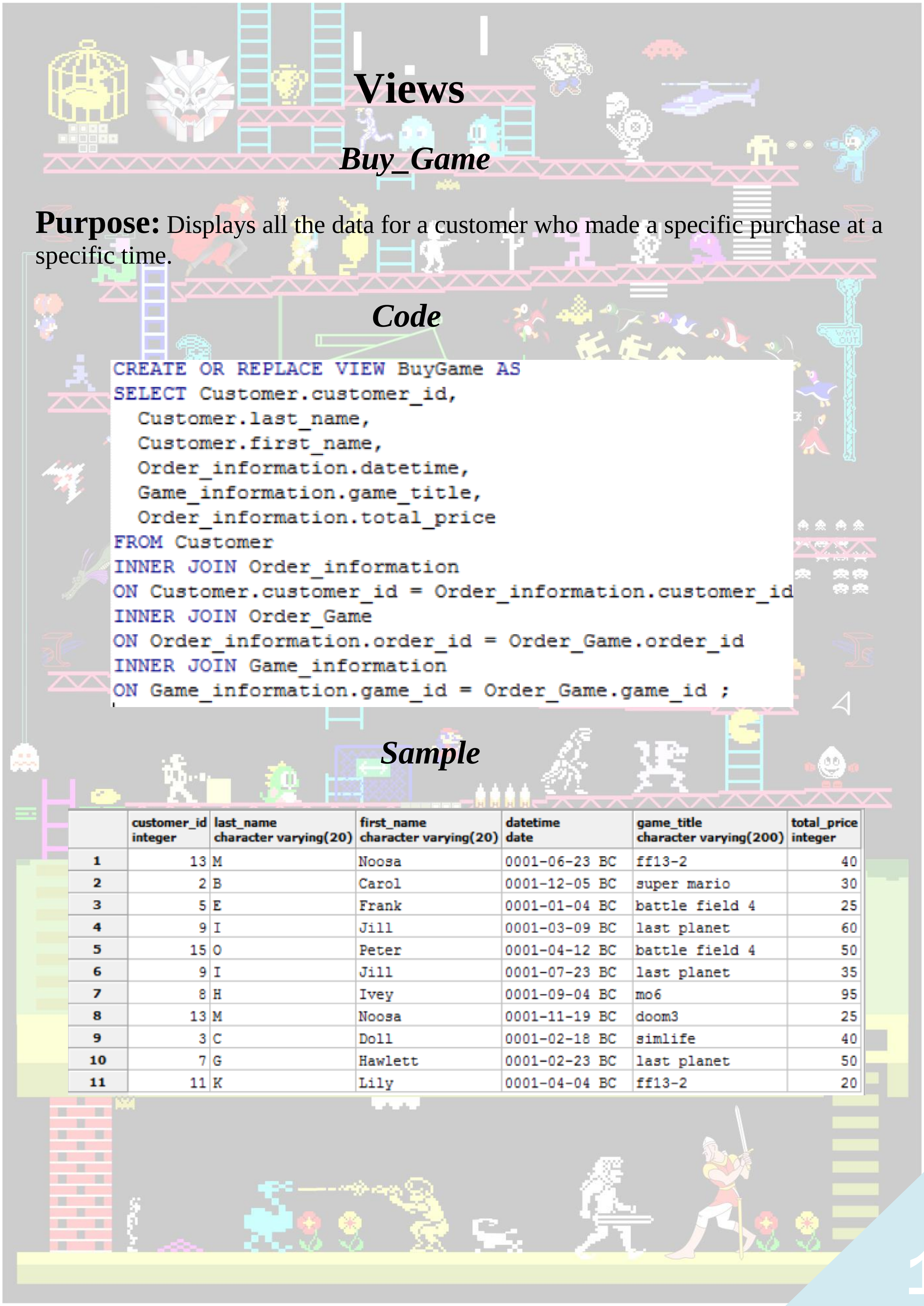


```
----ALL SALES IN 2012
SELECT SUM(TOTAL_PRICE)
FROM ORDER_INFORMATION
WHERE DATETIME BETWEEN '01-JAN-12'
                   AND '31-DEC-12';
```

Sample



	sum bigint
1	

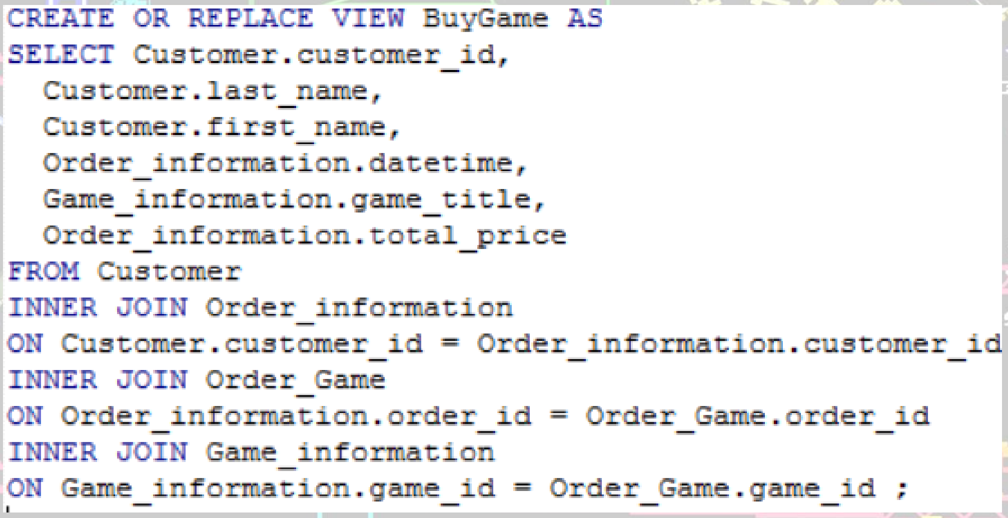


Views

Buy_Game

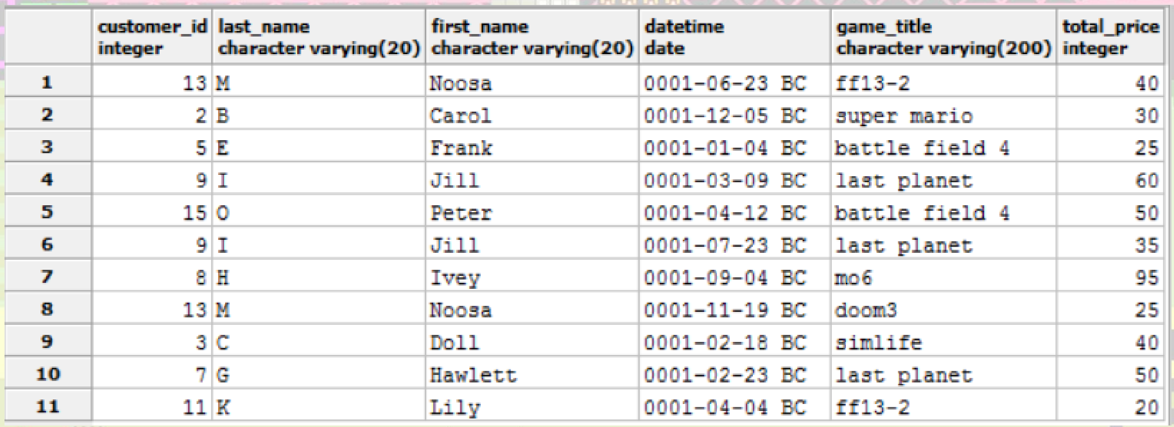
Purpose: Displays all the data for a customer who made a specific purchase at a specific time.

Code

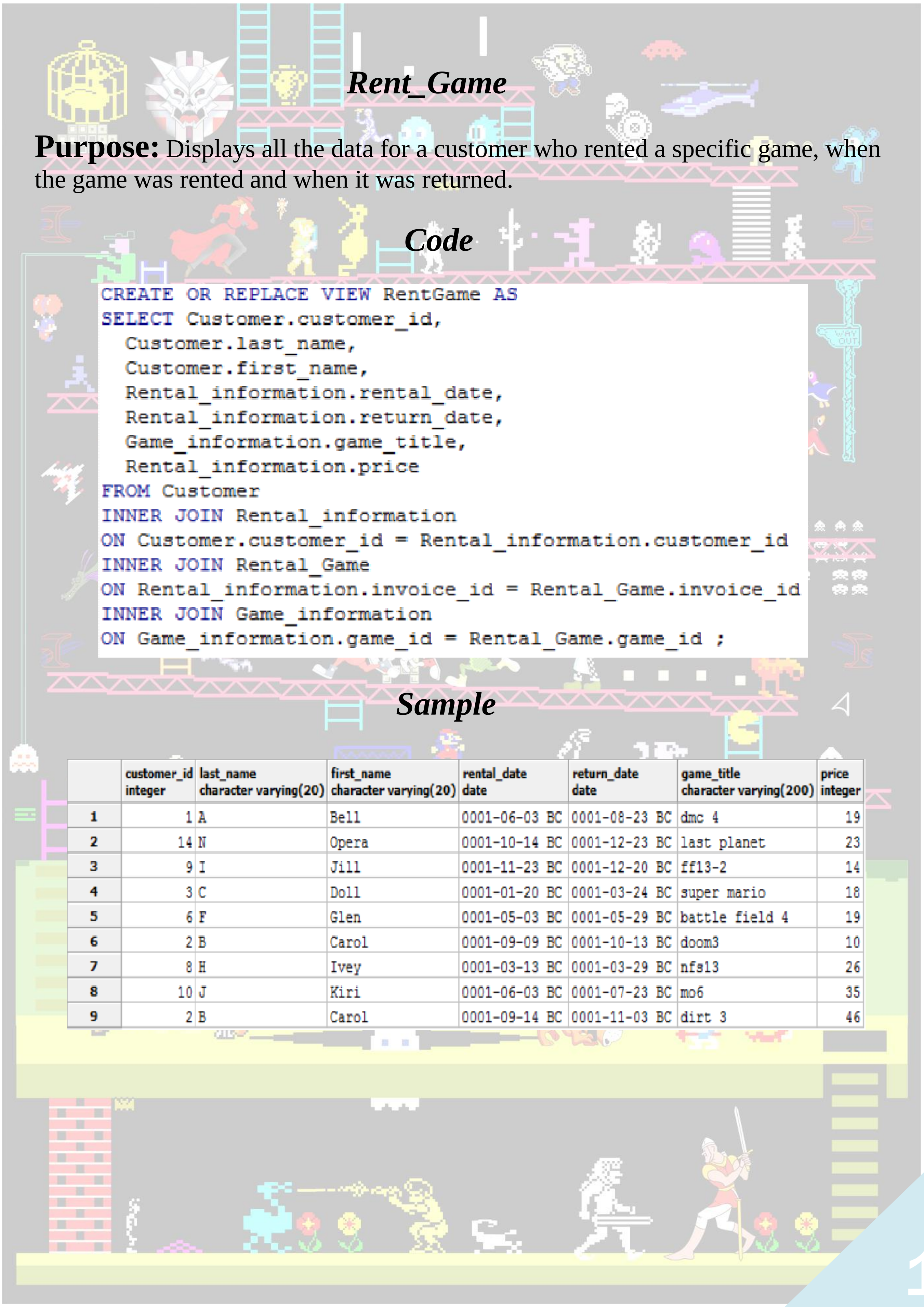


```
CREATE OR REPLACE VIEW BuyGame AS
SELECT Customer.customer_id,
  Customer.last_name,
  Customer.first_name,
  Order_information.datetime,
  Game_information.game_title,
  Order_information.total_price
FROM Customer
INNER JOIN Order_information
ON Customer.customer_id = Order_information.customer_id
INNER JOIN Order_Game
ON Order_information.order_id = Order_Game.order_id
INNER JOIN Game_information
ON Game_information.game_id = Order_Game.game_id ;
```

Sample



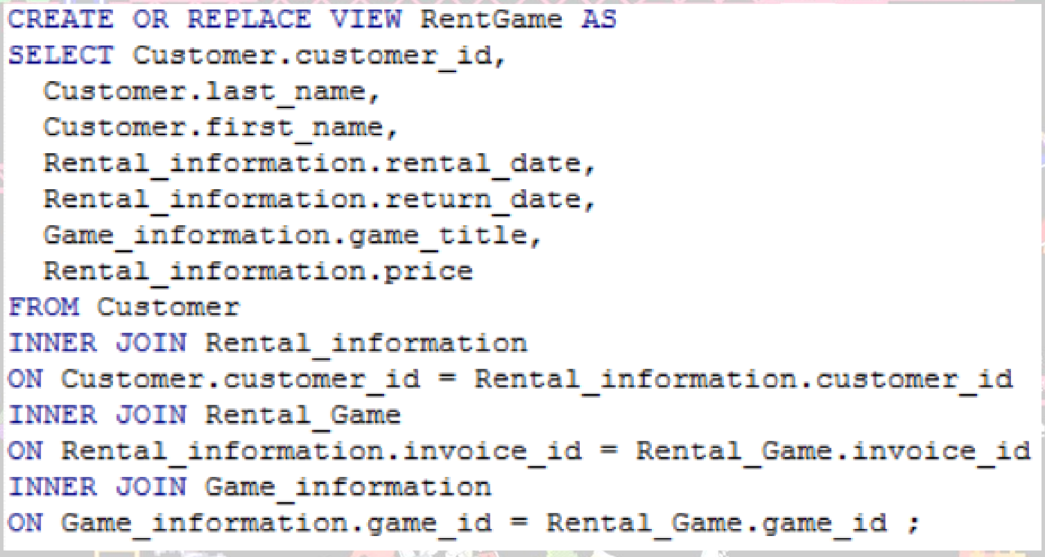
	customer_id integer	last_name character varying(20)	first_name character varying(20)	datetime date	game_title character varying(200)	total_price integer
1	13	M	Noosa	0001-06-23 BC	ff13-2	40
2	2	B	Carol	0001-12-05 BC	super mario	30
3	5	E	Frank	0001-01-04 BC	battle field 4	25
4	9	I	Jill	0001-03-09 BC	last planet	60
5	15	O	Peter	0001-04-12 BC	battle field 4	50
6	9	I	Jill	0001-07-23 BC	last planet	35
7	8	H	Ivey	0001-09-04 BC	mo6	95
8	13	M	Noosa	0001-11-19 BC	doom3	25
9	3	C	Doll	0001-02-18 BC	simlife	40
10	7	G	Hawlett	0001-02-23 BC	last planet	50
11	11	K	Lily	0001-04-04 BC	ff13-2	20



Rent_Game

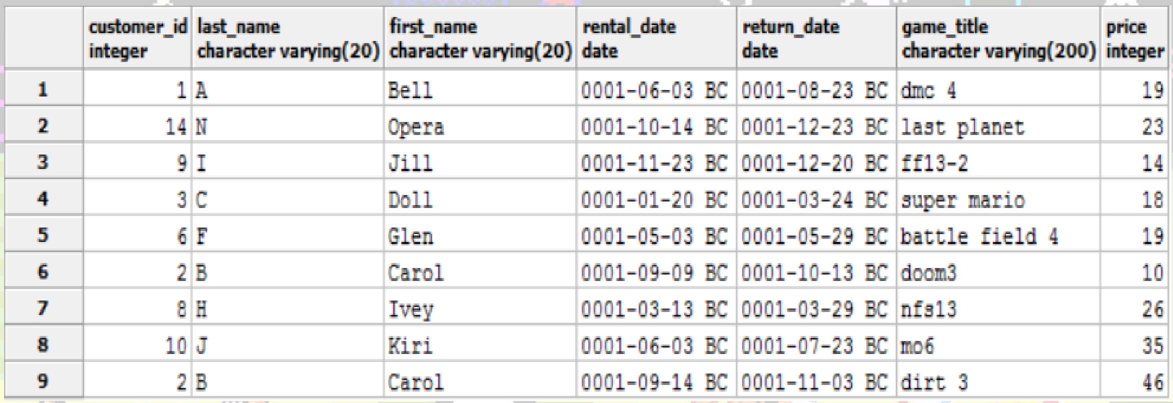
Purpose: Displays all the data for a customer who rented a specific game, when the game was rented and when it was returned.

Code



```
CREATE OR REPLACE VIEW RentGame AS
SELECT Customer.customer_id,
  Customer.last_name,
  Customer.first_name,
  Rental_information.rental_date,
  Rental_information.return_date,
  Game_information.game_title,
  Rental_information.price
FROM Customer
INNER JOIN Rental_information
ON Customer.customer_id = Rental_information.customer_id
INNER JOIN Rental_Game
ON Rental_information.invoice_id = Rental_Game.invoice_id
INNER JOIN Game_information
ON Game_information.game_id = Rental_Game.game_id ;
```

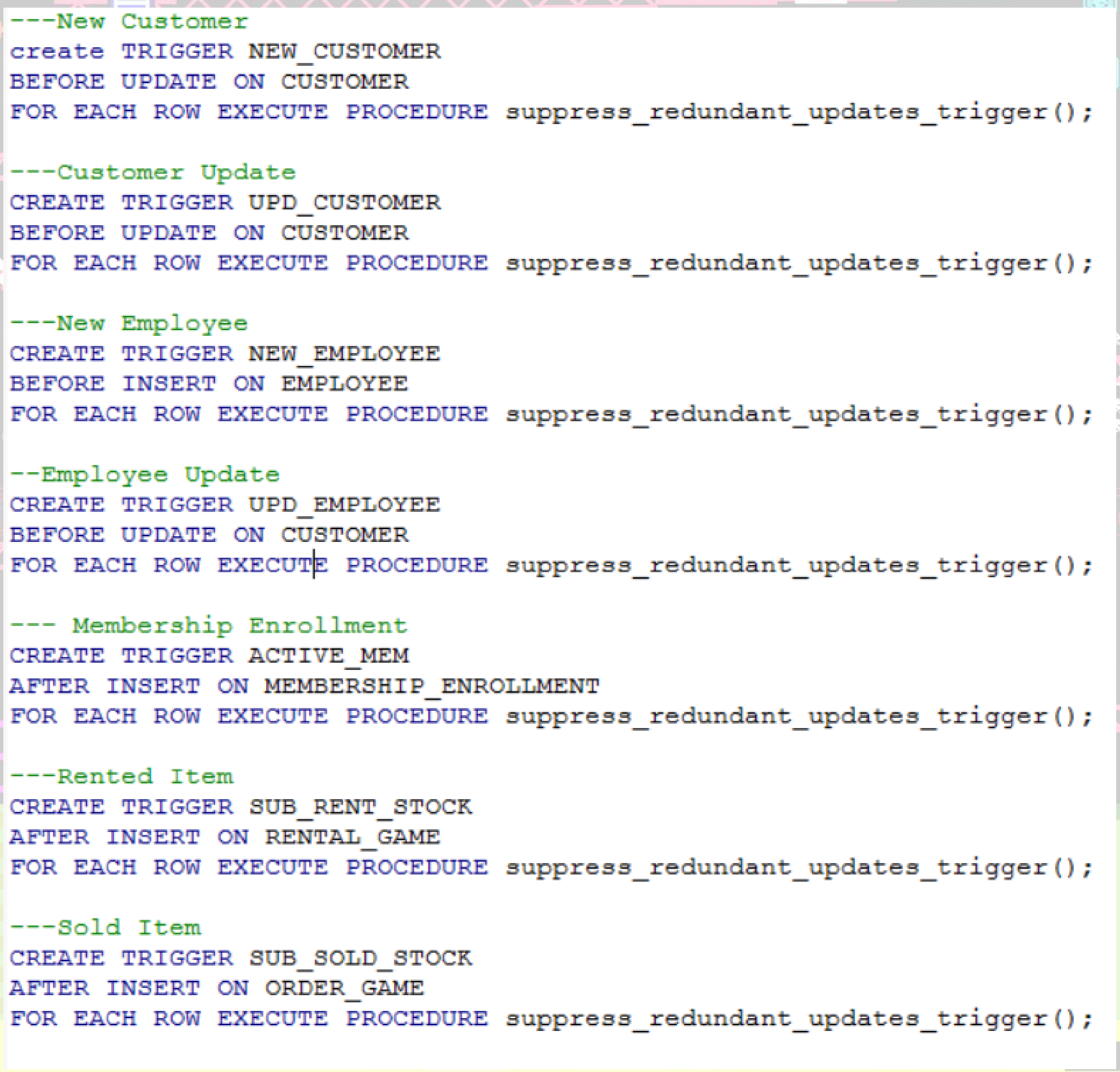
Sample



	customer_id integer	last_name character varying(20)	first_name character varying(20)	rental_date date	return_date date	game_title character varying(200)	price integer
1	1	A	Bell	0001-06-03 BC	0001-08-23 BC	dmc 4	19
2	14	N	Opera	0001-10-14 BC	0001-12-23 BC	last planet	23
3	9	I	Jill	0001-11-23 BC	0001-12-20 BC	ff13-2	14
4	3	C	Doll	0001-01-20 BC	0001-03-24 BC	super mario	18
5	6	F	Glen	0001-05-03 BC	0001-05-29 BC	battle field 4	19
6	2	B	Carol	0001-09-09 BC	0001-10-13 BC	doom3	10
7	8	H	Ivey	0001-03-13 BC	0001-03-29 BC	nfs13	26
8	10	J	Kiri	0001-06-03 BC	0001-07-23 BC	mo6	35
9	2	B	Carol	0001-09-14 BC	0001-11-03 BC	dirt 3	46

Triggers

The triggers below gives a good idea of the actors involved and what is needed for a transaction to be made. Also benefits apply to those who have a membership. That is if you are a serious gamer!

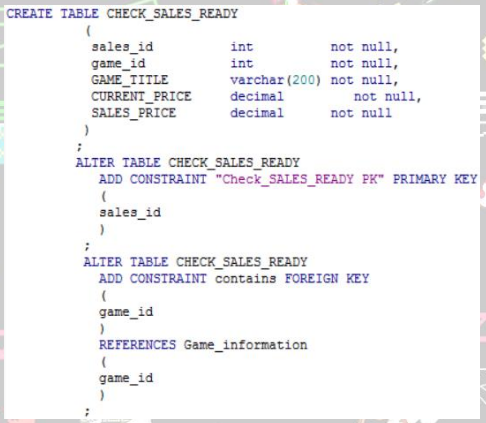


```
---New Customer
create TRIGGER NEW_CUSTOMER
BEFORE UPDATE ON CUSTOMER
FOR EACH ROW EXECUTE PROCEDURE suppress_redundant_updates_trigger();

---Customer Update
CREATE TRIGGER UPD_CUSTOMER
BEFORE UPDATE ON CUSTOMER
FOR EACH ROW EXECUTE PROCEDURE suppress_redundant_updates_trigger();

---New Employee
CREATE TRIGGER NEW_EMPLOYEE
BEFORE INSERT ON EMPLOYEE
FOR EACH ROW EXECUTE PROCEDURE suppress_redundant_updates_trigger();

--Employee Update
CREATE TRIGGER UPD_EMPLOYEE
BEFORE UPDATE ON CUSTOMER
FOR EACH ROW EXECUTE PROCEDURE suppress_redundant_updates_trigger();

--- Membership Enrollment
CREATE TRIGGER ACTIVE_MEM
AFTER INSERT ON MEMBERSHIP_ENROLLMENT
FOR EACH ROW EXECUTE PROCEDURE suppress_redundant_updates_trigger();

---Rented Item
CREATE TRIGGER SUB_RENT_STOCK
AFTER INSERT ON RENTAL_GAME
FOR EACH ROW EXECUTE PROCEDURE suppress_redundant_updates_trigger();

---Sold Item
CREATE TRIGGER SUB_SOLD_STOCK
AFTER INSERT ON ORDER_GAME
FOR EACH ROW EXECUTE PROCEDURE suppress_redundant_updates_trigger();
```

Store Procedure

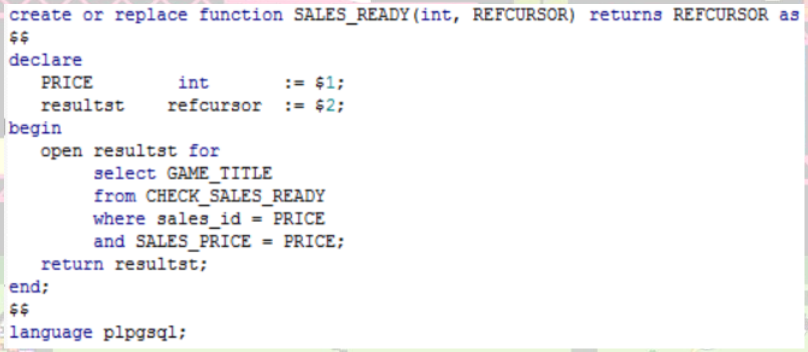
Helping your customers save a few dollars can always lead to customer satisfaction. The store procedure shown below is exactly what we intended to implement because our customers deserve the best service possible. The stored procedure returns all the video games that are on sale ranging from $70 and below. Furthermore we know that money is well spent when items go on sale.

Table Create Statement

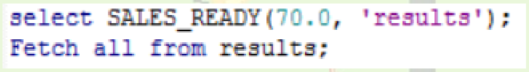


```
CREATE TABLE CHECK_SALES_READY
          (
           sales_id          int          not null,
           game_id           int          not null,
           GAME_TITLE        varchar(200) not null,
           CURRENT_PRICE     decimal         not null,
           SALES_PRICE       decimal      not null
          )
         ;
         ALTER TABLE CHECK_SALES_READY
            ADD CONSTRAINT "Check_SALES_READY PK" PRIMARY KEY
            (
            sales_id
            )
          ;
          ALTER TABLE CHECK_SALES_READY
            ADD CONSTRAINT contains FOREIGN KEY
            (
            game_id
            )
            REFERENCES Game_information
            (
            game_id
            )
          ;
```

Code

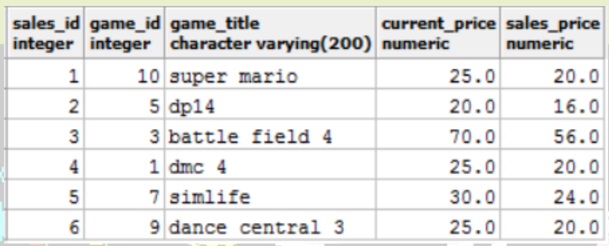


```
create or replace function SALES_READY(int, REFCURSOR) returns REFCURSOR as
$$
declare
   PRICE        int       := $1;
   resultst    refcursor  := $2;
begin
   open resultst for
        select GAME_TITLE
        from CHECK_SALES_READY
        where sales_id = PRICE
        and SALES_PRICE = PRICE;
   return resultst;
end;
$$
language plpgsql;
```

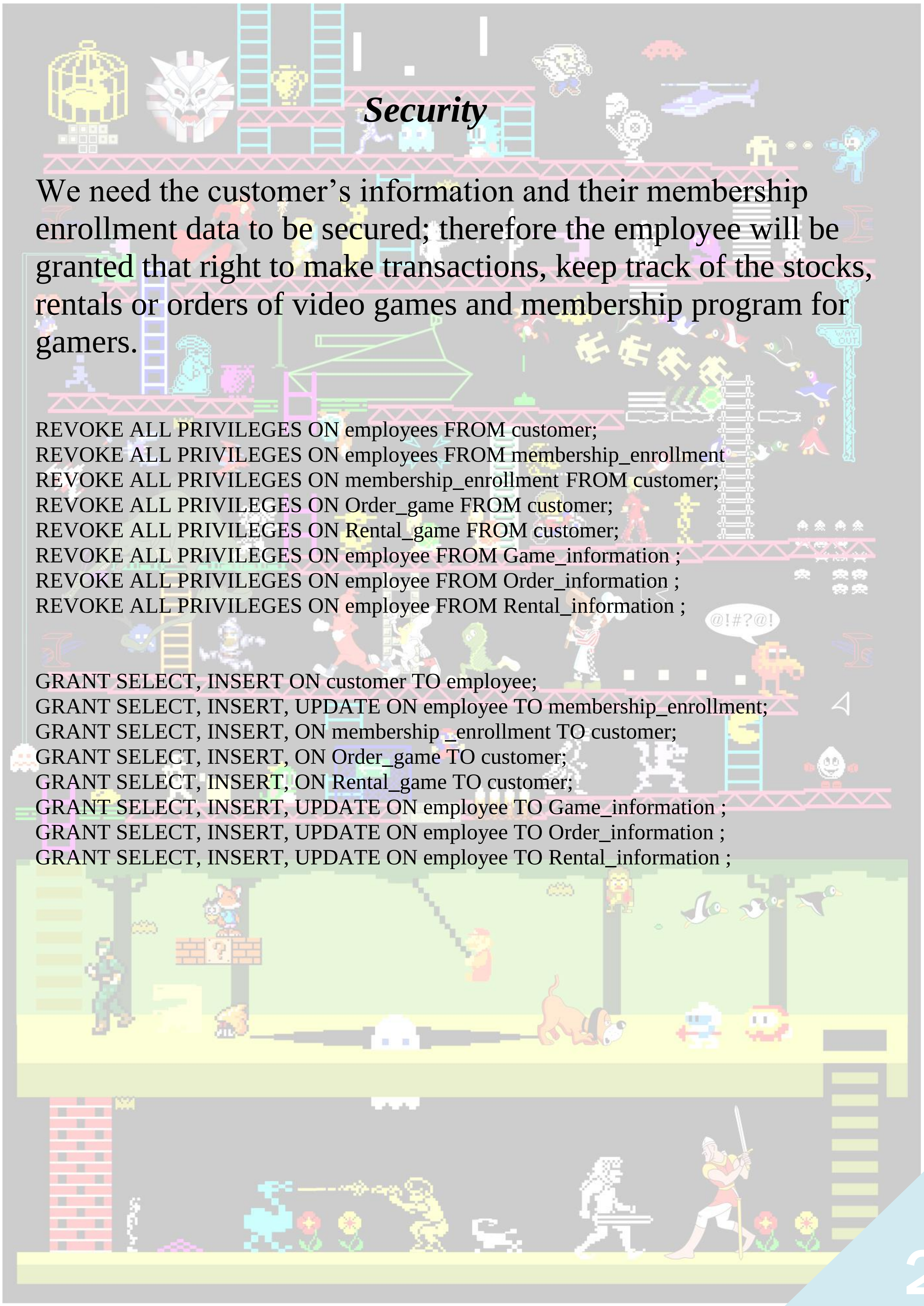


```
select SALES_READY(70.0, 'results');
Fetch all from results;
```

Sample



sales_id integer	game_id integer	game_title character varying(200)	current_price numeric	sales_price numeric
1	10	super mario	25.0	20.0
2	5	dp14	20.0	16.0
3	3	battle field 4	70.0	56.0
4	1	dmc 4	25.0	20.0
5	7	simlife	30.0	24.0
6	9	dance central 3	25.0	20.0

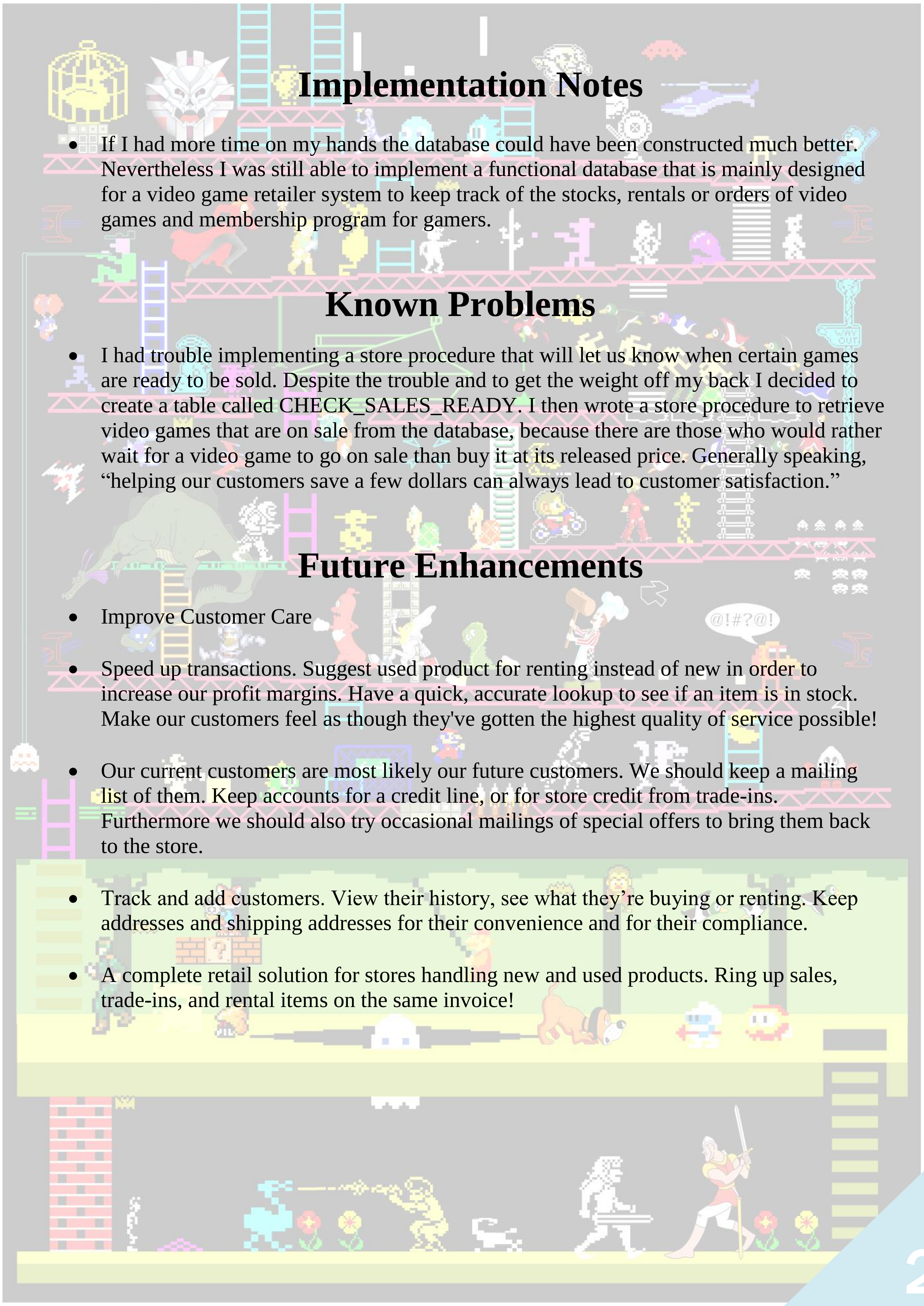


Security

We need the customer’s information and their membership enrollment data to be secured; therefore the employee will be granted that right to make transactions, keep track of the stocks, rentals or orders of video games and membership program for gamers.

```
REVOKE ALL PRIVILEGES ON employees FROM customer;
REVOKE ALL PRIVILEGES ON employees FROM membership_enrollment
REVOKE ALL PRIVILEGES ON membership_enrollment FROM customer;
REVOKE ALL PRIVILEGES ON Order_game FROM customer;
REVOKE ALL PRIVILEGES ON Rental_game FROM customer;
REVOKE ALL PRIVILEGES ON employee FROM Game_information ;
REVOKE ALL PRIVILEGES ON employee FROM Order_information ;
REVOKE ALL PRIVILEGES ON employee FROM Rental_information ;
```

```
GRANT SELECT, INSERT ON customer TO employee;
GRANT SELECT, INSERT, UPDATE ON employee TO membership_enrollment;
GRANT SELECT, INSERT, ON membership _enrollment TO customer;
GRANT SELECT, INSERT, ON Order_game TO customer;
GRANT SELECT, INSERT, ON Rental_game TO customer;
GRANT SELECT, INSERT, UPDATE ON employee TO Game_information ;
GRANT SELECT, INSERT, UPDATE ON employee TO Order_information ;
GRANT SELECT, INSERT, UPDATE ON employee TO Rental_information ;
```



Implementation Notes

- If I had more time on my hands the database could have been constructed much better. Nevertheless I was still able to implement a functional database that is mainly designed for a video game retailer system to keep track of the stocks, rentals or orders of video games and membership program for gamers.

Known Problems

- I had trouble implementing a store procedure that will let us know when certain games are ready to be sold. Despite the trouble and to get the weight off my back I decided to create a table called CHECK_SALES_READY. I then wrote a store procedure to retrieve video games that are on sale from the database, because there are those who would rather wait for a video game to go on sale than buy it at its released price. Generally speaking, "helping our customers save a few dollars can always lead to customer satisfaction."

Future Enhancements

- Improve Customer Care

- Speed up transactions. Suggest used product for renting instead of new in order to increase our profit margins. Have a quick, accurate lookup to see if an item is in stock. Make our customers feel as though they've gotten the highest quality of service possible!

- Our current customers are most likely our future customers. We should keep a mailing list of them. Keep accounts for a credit line, or for store credit from trade-ins. Furthermore we should also try occasional mailings of special offers to bring them back to the store.

- Track and add customers. View their history, see what they're buying or renting. Keep addresses and shipping addresses for their convenience and for their compliance.

- A complete retail solution for stores handling new and used products. Ring up sales, trade-ins, and rental items on the same invoice!