Xi'an Jiaotong-Liverpool University
西交利物浦大学

Department of Electrical and Electronic Engineering

CPT103 - Introduction to Databases

Group Project

Leader:

Name : Kenan Duan

Student ID : 1929926

Team member 1:

Name : Fangtao Zhao

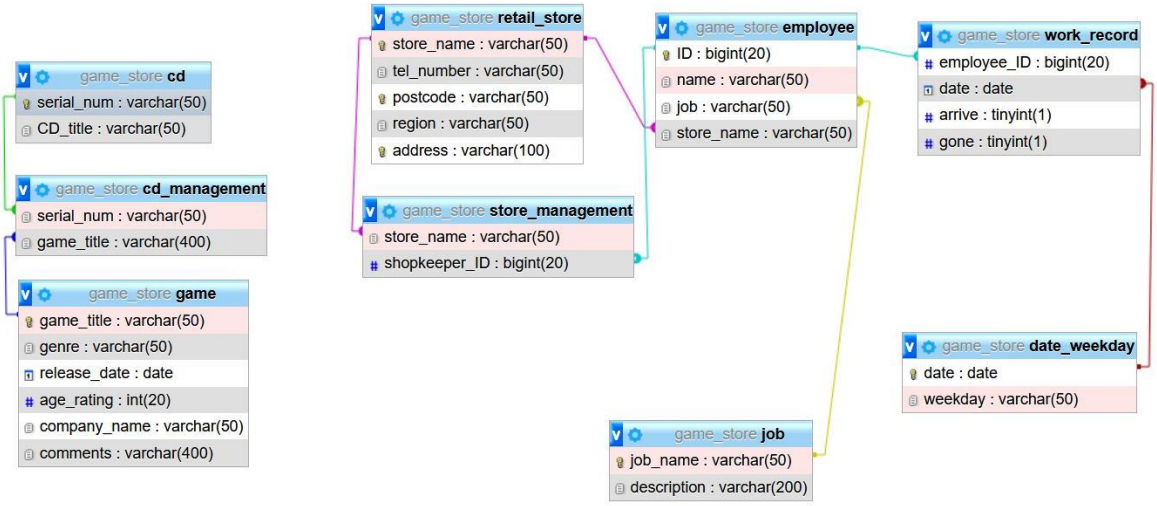
Student ID : 1929009

Team member 2:

Name : Juerui Li

Student ID : 1929654

Section 1: ER Diagram



Section 2: Detailed Specification

Table retail_store

This table is to show the information of each store.

1. store_name: varchar(50) NOT NULL PRIMARY KEY

   According to the requirement, the name of each store should be unique. Therefore, it should not be empty and should be the primary key of this table. The data type of store_name is "varchar" with an upper limit of 50.

2. tel_number: varchar(50) NOT NULL

   Since each store should have a telephone number, it will not be empty. The data type here is varchar with an upper limit of 50. The "varchar" type is used instead of "int" is to deal with the potentially present area codes that would require symbols as prefixes and connectors.

3. postcode: varchar(50) NOT NULL UNIQUE

   Since each address is assigned with a unique postcode, the constraint of postcode is unique and therefore should not be empty. The postcode is a combination of letters and numbers so the data type here is varchar with an upper limit of 7, with two digits of area code and 5 digits of unique characters,..

4. region: varchar(50) NOT NULL

   There are five regions in the kingdom and each store should have a location, thus this attribute should be not be empty. Since the content is letters, the data type here is set to "varchar" with an upper limit of 50. Initially, this is not included in this table since the region of a store can be read from the postcode. However, in subsequent data selection, for cases such as when all the name of stores in the middle of the country are needed, it is much easier to have a region column for the ease of retrieval.

5. address: varchar(100) NOT NULL UNIQUE

   There is no specific requirement about the address but each address should be unique and not empty. The data type of address is "varchar" with a much higher limitation to store

detailed information.

Table store_management

This table is created for the ease of referencing to the employees working at a specific store.

1. store_name: varchar(50) NOT NULL FOREIGN KEY
   The store name is used to identify the stores that employees work. This is a foreign key that contains references to the store_name column from the *retail_store* table.
2. shopkeeper_ID: int(12) NOT NULL FOREIGN KEY
   Each store will be managed by a shopkeeper and will not be repetitive. Therefore, it should not be empty Data type of shopkeeper_ID is "int" with an upper limit of 20. It is also a foreign key that refers to the ID from the *employee* table.

Constraint:

1. fk_sm_rs: This is the constraint for the foreign key that refers the store_name column in this table to the store_name in the table *retail_store*.
2. fk_sm_employee: This is the constraint for the foreign key that refers shopkeeper_ID column in this table to the ID in table *employee*.

Table employee

This table is to show the information of each employee.

1. ID: bigint(20) NOT NULL PRIMARY KEY
   Each employee should have a unique ID number. Therefore, it is suitable to be the primary key in this table. It is stated that the ID should not be empty as well. The data type used is "int" with an upper limit of 12, in line with the length of their ID number..
2. name: varchar(50) NOT NULL
   The name of each employee is not unique but ever present. The data type of name is "varchar" with an upper limit of 50.
3. job: varchar(50) NOT NULL FOREIGN KEY
   As the requirement stated, each person should have a job. Meanwhile, this is a foreign key that refers to the job_name from the *job* table.
4. store_name: varchar(50) NOT NULL FOREIGN KEY
   Since the retail store is managed by one store keeper and a fixed group of staff members, this table is not null but not unique, and references to the store_name from the table *retail_store*.

Constraint:

1. fk_employee_job: This is the constraint for a foreign key that refers the job in table `employee` is a foreign key which refers to job_name in table *job*.
2. fk_employee_rs: This is the constraint for a foreign key that references to the store_name in the table *retail_store* table.

Table job

This table is to show each job and the corresponding description.

1. job_name: varchar(50) NOT NULL PRIMARY KEY
   There are three different jobs and this attribute is suitable for the primary key. The data type of job_name is set to "varchar" with an upper limit of 50.
2. description: varchar(200) NOT NULL
   Every job has a description to show the job content. The upper limit of this attribute is 200 to contain more information.

Table work_record

This table is to show daily attendance of each employee.

1. employee_ID: int(12) NOT NULL FOREIGN KEY
   This attribute is a foreign key which refers ID in table employee to show a relationship between the person and each certain date.
2. date: date NOT NULL FOREIGN KEY
   This attribute is the date record in the database, therefore assigned the "date" data type. Therefore, it should be not empty. This is a foreign key that refers date in table *date_weekday* as well.
3. arrive: boolean NOT NULL
   This is a Boolean attribute to show whether an employee is arrived or not. When it is 1, it means the employee is arrived at the work place on time. When it is 0, it means the employee is not arrived at the work place on time. This will record every day so it will not be empty.
4. leave: boolean NOT NULL
   This is a Boolean attribute to show whether an employee is left or not. When it is 1, it means the employee is left the work place after the required time. When it is 0, it means the employee is left the work place before the required time. This will record every day so it will not be empty.

Constraint:

1. fk_wr_employee: This is the constraint for a foreign key that refers employee_ID column in this table to ID column in table *employee*.
2. fk_wr_dw: This is the constraint for a foreign key that refers the date column in this table to the date column in the table *date_weekday*.

Table date_weekday

This table is to show the day of the week of each date. It can be used to find out the day of the week for each particular date.

1. date: date NOT NULL PRIMARY KEY
   This attribute is the date record in the database. The data type of this attribute is date. Therefore, it should be not empty.
2. weekday: varchar(50) NOT NULL
   This attribute should be one of the "Monday, Tuesday, Wednesday, Thursday, Friday,

Saturday, Sunday".

Table game

This table is about the contents of the game.

1. game_title: varchar(50) NOT NULL PRIMARY KEY
   This is about the title of the game. Since each game has a unique name and identifying games using their names is more natural, it is suitable to assign it as the primary key.
2. genre: varchar(50) NOT NULL
   This is about the genre of the game, and the genre of the game is always present, thus the "not null" definition. The genre of the game is likely to contain various symbols thus the "varchar" type is used.
3. release_date: date NOT NULL
   This is about the release date of the game.
4. age_rating: varchar(20) NOT NULL
   This is about the age rating of the game. This column may contain various characters, symbols and numbers, for instance "ESRB Mature 17+" and "IARC 7+", hence the "varchar" data type is used.
5. company_name: varchar(50) NOT NULL
   Every game has a developing company, but multiple games could have the same developing company, which led us to ditch the unique attribute.
6. comments: varchar(400) NOT NULL
   The comments are potentially long, so an upper limit of 400 characters is set.

Table CD_management

This table is created to index the games that a specific game CD contains.

1. serial_num: varchar(5) NOT NULL FOREIGN KEY
   This column is used to identify the CDs to be indexed, and is the foreign key that refers to the serial_num column in the table *CD*.
2. game_title: varchar(400) NOT NULL FOREIGN KEY
   This column contains an upper limit of 400 characters, for that any CD could contain more than one games, making a much higher limit necessary. This is a foreign key that refers to the game_title column in the table *game*.

Constraint:

1. fk_cm_cd: This is the constraint for a foreign key that refers serial_num column in this table to serial_num column in table *CD*.
2. fk_cm_game: This is the constraint for a foreign key that refers game_title column in this table to game_title column in table *game*.

Table CD

This table is created to index the specification of the CDs released.

1. serial_num varchar(50) NOT NULL PRIMARY KEY
   Since each game contains a unique serial number and easily identifiable with their serial number, which makes it ideal to be the primary key of this table.
2. CD_title varchar(50) NOT NULL
   Each CD has a title that tells the content of the CD. It is decided that the name of each CD is not unique, as complications containing the same number of games and released in the same year might have the same title.

Section 3: Select

a) Kenan Duan 1929926

1. Find the name of the store which is located in the north and order them by the name

select store_name from retail_store where region = "north"
order by store_name ASC

The result:

+ Options

←T→	store_name ▲ 1
Edit Copy Delete	VintageFun1
Edit Copy Delete	VintageFun6

2. Find all names of shopkeeper

select name as shopkeeper_name
from employee where job = "shopkeeper"

The result:

+ Options

←T→	shopkeeper_name
Edit Copy Delete	Alan
Edit Copy Delete	Bob
Edit Copy Delete	Caven
Edit Copy Delete	Davi
Edit Copy Delete	Eurus
Edit Copy Delete	Fata
Edit Copy Delete	Guy

3. Find all names of except shopkeeper

select name as staff_name

from employee where job in("sales staff", "shop cleaner")

The result:

+ Options

←T→				staff_name
☐	Edit	Copy	Delete	Halen
☐	Edit	Copy	Delete	Iris
☐	Edit	Copy	Delete	John
☐	Edit	Copy	Delete	Kaven
☐	Edit	Copy	Delete	Lily
☐	Edit	Copy	Delete	Marry
☐	Edit	Copy	Delete	Neo
☐	Edit	Copy	Delete	Oli
☐	Edit	Copy	Delete	Peter
☐	Edit	Copy	Delete	Queen
☐	Edit	Copy	Delete	Roly
☐	Edit	Copy	Delete	Steven
☐	Edit	Copy	Delete	Tylar
☐	Edit	Copy	Delete	Ueli

4. Find all titles of games which are suitable for people under 15 years old to play and order them by the title

select game_title from game
where age_rating < 15
order by game_title ASC

The result:

+ Options

←T→				game_title
☐	Edit	Copy	Delete	Among us
☐	Edit	Copy	Delete	Civilization VI
☐	Edit	Copy	Delete	Counter-Strike: Global Offensive
☐	Edit	Copy	Delete	Dota2
☐	Edit	Copy	Delete	Fall Guys: Ultimate Knockout
☐	Edit	Copy	Delete	Forza Horizon 4
☐	Edit	Copy	Delete	NBA 2K21

5. Find the name of game which has the earliest year of release

select game_title as `the oldest game` from game
where release_date<= ALL (select release_date from game)

The result:

+ Options

←T→	the oldest game
☐ Edit Copy Delete	Dota2

6. Find the number of games in the store

select count(game_title) as `number of game` from game

The result:

+ Options

number of game
9

7. Find how many games is suitable for minors

select count(game_title) as `number of game` from game where age_rating <18

The result:

+ Options

number of game
8

8. find the difference of year between the earliest game and the latest game

select max(year (release_date))- min(year (release_date)) as `year difference`
from game

The result:

+ Options

year difference
8

9. Find the name of CD which contains game called "Counter-Strike: Global Offensive"

select CD_title from cd right outer join cd_management
on cd.serial_num=cd_management.serial_num
where game_title= "Counter-Strike: Global Offensive"

The result:

+ Options

CD_title
VintageFun 3 in 1 2019
VintageFun 4 in 1 2021

10. Find all names of games contained in the CD called "VintageFun 5 in 1 2020"

```
select game_title from cd right outer join cd_management
on cd.serial_num=cd_management.serial_num
where cd_title ="VintageFun 5 in 1 2020"
```

The result:

+ Options

game_title
Civilization VI
NBA 2K21
Fall Guys: Ultimate Knockout
Forza Horizon 4
Mirror

11. Find the average age rating of games in the CD of B567C

```
select avg(age_rating) as `average age rating`
from cd right outer join cd_management
on cd.serial_num=cd_management.serial_num
where cd_title ="VintageFun 5 in 1 2020"
```

The result:

+ Options

average age rating
10.4000

12. Find the name of employee who punch the clock when arriving while forget doing so when leaving in "2021-5-3"

```
select name from employee right outer join work_record
on employee.ID =work_record.employee_ID
where date ="2021-5-3" and arrive = 1 and gone = 0
```

The result:

+ Options

name
Eurus
Kaven
Oli

13. Find out "2021-5-3" is what day of the week

select weekday from date_weekday
where date ="2021-5-3"

The result:

+ Options

←T→	weekday
☐ Edit Copy Delete	Monday

14. Find all the employee work in the central region

select name from employee right outer join retail_store
on employee.store_name = retail_store.store_name
where region = "central"

The result:

+ Options

name
Eurus
Lily
Steven

15. Find out the number of employees who punch the clock when arriving and leaving on 2021-5-5

select count(name) as `number of employee` from employee right outer join work_record
on employee.ID =work_record.employee_ID
where date ="2021-5-5" and arrive = 1 and gone = 1

The result:

+ Options

number of employee
15

b) Fangtao Zhao 1929009

1. Select stores in the north region, and display their name, address and telephone number. The whole "retail store" table is shown as well for comparation. Two methods are used to achieve this select:

Method 1 (according to the "region" column):

SELECT store_name, tel_number, address FROM retail_store
WHERE (region = "north");

Method 2 (according to the “postcode” column, the first two characters of the postcode is “LN”):

SELECT store_name, tel_number, address FROM retail_store
WHERE postcode LIKE "LN%";

The “retail store” table:

store_name	tel_number	postcode	region	address
VintageFun1	12345	LN12345	north	Oxiford Rd
VintageFun2	23456	LS23456	south	Priory St
VintageFun3	34567	LE34567	east	Aston St
VintageFun4	45678	LW45678	west	Barrack Rd
VintageFun5	56789	LC9ABCD	central	Old Hall Ln
VintageFun6	67890	LNEF0G1	north	Booth St W
VintageFun7	78901	LS23ABC	south	Deane Rd

Select result:

store_name	tel_number	address
VintageFun1	12345	Oxiford Rd
VintageFun6	67890	Booth St W

2. Get all shopkeepers with their ID, name and job description from two tables. (apply “AND”) Part of the “employee” table is also shown for comparation.

SELECT ID, name, description FROM employee, job
WHERE (employee.job = "shopkeeper")
AND (job.job_name = "shopkeeper");

Part of the “employee” table:

ID	name	job	store_name
123456789000	Alan	shopkeeper	VintageFun1
123456789001	Bob	shopkeeper	VintageFun2
123456789002	Caven	shopkeeper	VintageFun3
123456789003	Davi	shopkeeper	VintageFun4
123456789004	Eurus	shopkeeper	VintageFun5
123456789005	Fata	shopkeeper	VintageFun6
123456789006	Guy	shopkeeper	VintageFun7
123456789007	Halen	sales staff	VintageFun1
123456789008	Iris	sales staff	VintageFun2
123456789009	John	sales staff	VintageFun3
123456789010	Kaven	sales staff	VintageFun4
123456789011	Lily	sales staff	VintageFun5
123456789012	Marry	sales staff	VintageFun6
123456789013	Neo	sales staff	VintageFun7
123456789014	Oli	shop cleaner	VintageFun1
123456789015	Peter	shop cleaner	VintageFun2
123456789016	Queen	shop cleaner	VintageFun3

Select result:

ID	name	description
123456789000	Alan	To manage the shop
123456789001	Bob	To manage the shop
123456789002	Caven	To manage the shop
123456789003	Davi	To manage the shop
123456789004	Eurus	To manage the shop
123456789005	Fata	To manage the shop
123456789006	Guy	To manage the shop

3. Select games that the age rating between 10 and 16 with their game title and comments
The whole “game” table is also shown for comparation.

```
SELECT game_title, comments FROM game
    WHERE (age_rating >= 10)
    AND (age_rating <= 16);
```

The “game” table:

game_title	genre	release_date	age_rating	company_name	comments
Among us	casual	2018-11-17	12	Innersloth	Made friends from it and probably will lose them a...
Civilization VI	strategy	2016-10-21	8	Firaxis Games	Civilization is a turn-based strategy game where p...
Counter-Strike: Global Offensive	action	2017-09-16	14	Valve	A popular first person shooter game around the wor...
Dota2	action	2013-07-10	10	Valve	Dota is deep and evolving, and it's never too late...
Fall Guys: Ultimate Knockout	casual	2020-08-04	6	Mediatonic	In the game, you will play a cute cheap Jelly Bean...
Forza Horizon 4	racing	2021-03-10	8	Playground Games	Race, stunt, create and explore: choose your own w...
Mirror	adventure	2018-04-19	18	KAGAMI WORKs	This is a beautiful girl + gem elimination +GALGAM...
NBA 2K21	simulation	2020-09-04	12	2K	NBA 2K21 is the latest title in the world-renowned...
The Witcher 3: Wild Hunt	role play	2015-05-18	16	CD PROJEKT RED	The Witcher is a plot-driven, open-world role-play...

Select result:

game_title	comments
Among us	Made friends from it and probably will lose them a...
Counter-Strike: Global Offensive	A popular first person shooter game around the wor...
Dota2	Dota is deep and evolving, and it's never too late...
NBA 2K21	NBA 2K21 is the latest title in the world-renowned...
The Witcher 3: Wild Hunt	The Witcher is a plot-driven, open-world role-play...

4. Select all shop cleaner with their name and ID shown as cleaner_ID. (apply aliases)

```
SELECT
    ID AS cleaner_ID,
    name
FROM
    employee E
WHERE
    E.job = "shop cleaner";
```

Select result:

cleaner_ID	name
123456789014	Oli
123456789015	Peter
123456789016	Queen
123456789017	Roly
123456789018	Steven
123456789019	Tylar
123456789020	Ueli

5. Select the game developed by Valve and released before 2018. Show their game tile and the release_date.

```
SELECT game_title, release_date FROM game
    WHERE release_date < "2018-01-01"
    AND company_name = "Valve";
```

Select result:

game_title	release_date
Counter-Strike: Global Offensive	2017-09-16
Dota2	2013-07-10

6. Get the average age rating of each genre of game, and show the average age as a new column.

```
SELECT genre, MIN(age_rating) AS average_age
FROM game
Group BY genre
```

Select result:

genre	average_age
action	10
adventure	18
casual	6
racing	8
role play	16
simulation	12
strategy	8

7. Select people who work in the north region. Show their name. (two tables are used)

```
SELECT name FROM employee E
    WHERE E.store_name IN
    (SELECT store_name
      FROM retail_store
      WHERE region = "north")
```

Select result:

name
Alan
Halen
Oli
Fata
Marry
Tylar

8. Select the shopkeepers who work in the central region.

```
SELECT * FROM employee E
    WHERE
    E.job = "shopkeeper"
    AND E.store_name =
    (SELECT store_name
      FROM retail_store
      WHERE region = "central");
```

Select result:

ID	name	job	store_name
123456789004	Eurus	shopkeeper	VintageFun5

9. Select all cleaners except Oli and Peter. Show the ID and name. (apply “NOT IN”)

```
SELECT ID, name FROM employee E
    WHERE E.job = "shop cleaner"
    AND E.name NOT IN ("Oli", "Peter");
```

Select result:

ID	name
123456789016	Queen
123456789017	Roly
123456789018	Steven
123456789019	Tylar
123456789020	Ueli

10. Select the game with the highest age rating. Show all information. (apply “ALL”)

```
SELECT * FROM game
    WHERE age_rating >=
    ALL (
        SELECT age_rating
        FROM game);
```

Select result:

game_title	genre	release_date	age_rating	company_name	comments
Mirror	adventure	2018-04-19	18	KAGAMI WORKs	This is a beautiful girl + gem elimination +GALGAM...

11. Select the game with higher age rating than any game else. Show the game title and age rating. (apply “ANY”)

```
SELECT game_title, age_rating FROM game
    WHERE age_rating >
    ANY (
    SELECT age_rating
    FROM game);
```

Select result:

game_title	age_rating
Among us	12
Civilization VI	8
Counter-Strike: Global Offensive	14
Dota2	10
Forza Horizon 4	8
Mirror	18
NBA 2K21	12
The Witcher 3: Wild Hunt	16

12. Select all retail store with all information with the shopkeeper's ID. (apply "INNER JOIN")

```
SELECT * FROM
    retail_store INNER JOIN store_management
    USING (store_name);
```

Select result:

store_name	tel_number	postcode	region	address	shopkeeper_ID
VintageFun1	12345	LN12345	north	Oxiford Rd	123456789000
VintageFun2	23456	LS23456	south	Priory St	123456789001
VintageFun3	34567	LE34567	east	Aston St	123456789002
VintageFun4	45678	LW45678	west	Barrack Rd	123456789003
VintageFun5	56789	LC9ABCD	central	Old Hall Ln	123456789004
VintageFun6	67890	LNEF0G1	north	Booth St W	123456789005
VintageFun7	78901	LS23ABC	south	Deane Rd	123456789006

13. Get the range of age rating of all games and shown in a new column.

```
SELECT
    MAX(age_rating) - MIN(age_rating)
    AS Range_of_age_rating
    FROM game;
```

Select result:

Range_of_age_rating
12

14. Get the work record of Eurus. Show the weekday as well. (using three tables)

```
SELECT * FROM work_record W
RIGHT OUTER JOIN date_weekday D
    ON W.date = D.date
WHERE employee_ID =
    (SELECT ID FROM employee
    WHERE name = "Eurus");
```

Select result:

employee_ID	date	arrive	gone	date	weekday
123456789004	2021-05-03	1	0	2021-05-03	Monday
123456789004	2021-05-04	1	1	2021-05-04	Tuesday
123456789004	2021-05-05	1	1	2021-05-05	Wednesday
123456789004	2021-05-06	0	1	2021-05-06	Thursday
123456789004	2021-05-07	0	1	2021-05-07	Friday
123456789004	2021-05-11	1	1	2021-05-11	Tuesday
123456789004	2021-05-12	1	1	2021-05-12	Wednesday
123456789004	2021-05-13	0	1	2021-05-13	Thursday

15. Get the average of age rating grouped by genre, and show the total average at the bottom. (apply "UNION")

SELECT game_title, age_rating FROM game
GROUP BY genre
UNION
SELECT
"Total" AS game_title,
AVG(age_rating) AS age_rating
FROM game;

Select result:

game_title	age_rating
Counter-Strike: Global Offensive	14.0000
Mirror	18.0000
Among us	12.0000
Forza Horizon 4	8.0000
The Witcher 3: Wild Hunt	16.0000
NBA 2K21	12.0000
Civilization VI	8.0000
Total	11.5556

c) Juerui Li 1929654

1. **Query for the games that are released newer than 2018.**

```
select * from game
where game.release_date > '2018-01-01';
```

```
select * from game where game.release_date > '2018-01-01'
```

Output | assignment3.game

5 rows | Tx: Auto | DDL | DML | Comma-...d (CSV)

	game_title	genre	release_date	age_rating	comp
1	Among us	casual	2018-11-17	12	Inners
2	Fall Guys: Ultimate Knockout	casual	2020-08-04	6	Mediat
3	Forza Horizon 4	racing	2021-03-10	8	Playgr
4	Mirror	adventure	2018-04-19	18	KAGAMI
5	NBA 2K21	simulation	2020-09-04	12	2K

2. Query for the work record of a specific region on a certain day

```
select employee_ID, arrive, `leave`
from work_record, retail_store, store_management
where retail_store.region = 'south'
and work_record.date = 2021-05-05;
```

```
Select employee_ID, date, arrive, `leave` from work_record where employee_id in (sele

select * from cd_management natural join game where age_rating = '18'
```

Output | assignment3.work_record

2 rows | Tx: Auto | DDL | DML | Comma-...d (C

	employee_ID	date	arrive	`leave`
1	123456789001	2021-05-05	0	0
2	123456789006	2021-05-05	1	1

3. Query for the games that appeared in multiple complications and display how much times they appeared.

```
select *, count(*)
from cd_management
natural join game
group by game_title
having count(*)>1;
```

```
Select *, count(*) from cd_management natural join game group by game_title having

Select * from employee natural join retail_store where retail_store.region = 'north
```

Output | Result 141

4 rows | Comma-...d (CSV)

	game_title	serial_num	genre	release_date
1	Civilization VI	B567C	strategy	2016-10-21
2	Counter-Strike: Global Offensive	A1234	action	2017-09-16
3	Forza Horizon 4	B567C	racing	2021-03-10
4	The Witcher 3: Wild Hunt	A1234	role play	2015-05-18

4. Query for the cleaner staff that works in a specific region.

```
select *
from employee
natural join retail_store
where retail_store.region = 'north'
and job like '%cleaner%';
```

```
select *
from employee
natural join retail_store
where retail_store.region = 'north'
and job like '%cleaner%';
```

Output | Result 168

2 rows | Comma-...d (CSV)

	store_name	ID	name	job	tel_number	postcode	regio
1	VintageFun1	123456789014	Oli	shop cleaner	12345	LN12345	north
2	VintageFun6	123456789019	Tylar	shop cleaner	67890	LNEF0G1	north

5. Query for the game CDs that are not complications.

```
select CD_title
from cd
natural join game
where game.game_title = cd.CD_title
```

```
Select CD_title from cd natural join game where game.game_title = cd.CD_title

Select *, count(*) from cd_management natural join game group by game_title having

Select * from employee natural join retail_store where retail_store.region = 'north
```

Output | at random order.

1 row | Tx: Auto | DDL | DML | Comma-...

	CD_title
1	Forza Horizon 4

6. Query for the shopkeeper that did not show up on a specific day, i.e.: has no arrive or leave records, and group by region.

```
select ID, name, job, store_name
from employee, work_record
where work_record.arrive = '0'
and work_record.`leave` = '0'
and date = '2021-05-07';
```

```
select ID, name, job, store_name from employee, work_record where work_record.arriv

select * from work_record where employee_ID = (select shopkeeper_ID from store_mana
```

Output | assignment3.employee

21 rows | Tx: Auto | DDL | Comma-...d

	ID	name	job	store_name
1	123456789000	Alan	shopkeeper	VintageFun1
2	123456789001	Bob	shopkeeper	VintageFun2
3	123456789002	Caven	shopkeeper	VintageFun3
4	123456789003	Davi	shopkeeper	VintageFun4
5	123456789004	Eurus	shopkeeper	VintageFun5
6	123456789005	Fata	shopkeeper	VintageFun6
7	123456789006	Guy	shopkeeper	VintageFun7
8	123456789007	Halen	sales staff	VintageFun1
9	123456789008	Iris	sales staff	VintageFun2

7. Query for the games that has a specific age rating and appears in specific game complications.

```
select *
from cd_management
natural join game
where age_rating = '18';
```

```
select * from cd_management natural join game where age_rating = '18';

select ID, name, job, store_name from employee, work_record where work_record.arrive
```

Output | Result 144

1 row | Comma-...d (CSV)

	game_title	serial_num	genre	release_date	age_rating
1	Mirror	B567C	adventure	2018-04-19	18

8. Query for the work record of employees that work in a certain area on a certain day.

```
Select employee_ID, date, arrive, `leave`
from work_record
where employee_id in
     (
     select shopkeeper_ID
     from retail_store
     natural join store_management
     where retail_store.region = 'south'
     )
and work_record.date = '2021-05-05';
```

```
Select employee_ID, date, arrive, `leave` from work_record where employee_id in (se
```

Output | assignment3.work_record

2 rows | Tx: Auto | DDL | DML | Comma-...d

	employee_ID	date	arrive	`leave`
1	123456789001	2021-05-05	0	0
2	123456789006	2021-05-05	1	1

9. Query for the shopkeeper and cleaner that came to work in a certain day.

```
select *
from employee
left outer join work_record
     on employee.ID = work_record.employee_ID
where job in ('shopkeeper', 'shop cleaner');
```

```
select * from employee left outer join work_record on employee.ID = work_record.emp

select distinct id, name, job, employee.store_name from employee, retail_store wher
```

Output | Result 149

103 rows | Comma-...d (CSV)

	ID	name	job	store_name	employee_ID	date
1	123456789000	Alan	shopkeeper	VintageFun1	123456789000	2021-05-03
2	123456789000	Alan	shopkeeper	VintageFun1	123456789000	2021-05-04
3	123456789000	Alan	shopkeeper	VintageFun1	123456789000	2021-05-05
4	123456789000	Alan	shopkeeper	VintageFun1	123456789000	2021-05-06
5	123456789000	Alan	shopkeeper	VintageFun1	123456789000	2021-05-07
6	123456789001	Bob	shopkeeper	VintageFun2	123456789001	2021-05-05
7	123456789001	Bob	shopkeeper	VintageFun2	123456789001	2021-05-06
8	123456789001	Bob	shopkeeper	VintageFun2	123456789001	2021-05-07
9	123456789001	Bob	shopkeeper	VintageFun2	123456789001	2021-05-10
10	123456789001	Bob	shopkeeper	VintageFun2	123456789001	2021-05-12
11	123456789001	Bob	shopkeeper	VintageFun2	123456789001	2021-05-14

10. Query for the employees that work in the north region.

```
select distinct id, name, job, employee.store_name
from employee, retail_store
where retail_store.region = 'north'
order by employee.store_name;
```

```
select distinct id, name, job, employee.store_name from employee, retail_store wher
```

Output | assignment3.employee

21 rows | Tx: Auto | DDL | Comma-...d (

	id	name	job	store_name
1	123456789000	Alan	shopkeeper	VintageFun1
2	123456789014	Oli	shop cleaner	VintageFun1
3	123456789007	Halen	sales staff	VintageFun1
4	123456789015	Peter	shop cleaner	VintageFun2
5	123456789008	Iris	sales staff	VintageFun2
6	123456789001	Bob	shopkeeper	VintageFun2
7	123456789016	Queen	shop cleaner	VintageFun3
8	123456789009	John	sales staff	VintageFun3
9	123456789002	Caven	shopkeeper	VintageFun3
10	123456789017	Roly	shop cleaner	VintageFun4
11	123456789010	Kaven	sales staff	VintageFun4

11. Query for the complication that contains more than 3 games.

```
select serial_num, CD_title
from cd
where serial_num in
    (
    select serial_num
    from cd_management
```

```
    group by serial_num
    having count(*) > 3
    );
```

```
select serial_num, CD_title from cd where serial_num in (select serial_num from cd_

select * from employee left outer join work_record on(employee.ID = work_record.emp
```

Output | assignment3.cd

3 rows | Tx: Auto | DDL | DML | Comma-...d (C

	serial_num	CD_title
1	A1234	VintageFun 3 in 1 2019
2	B567C	VintageFun 5 in 1 2020
3	CF89E	VintageFun 4 in 1 2021

12. Query for the employees who should work on Friday.

```
select *
from employee
left outer join work_record on
    (
    employee.ID = work_record.employee_ID
    )
natural join date_weekday
where weekday = 'Friday'
order by work_record.date;
```

```
select *
from employee
left outer join work_record on
    (
    employee.ID = work_record.employee_ID
    )
natural join date_weekday
where weekday = 'Friday'
```

Extensions

Output | Result 169

29 rows | Comma-...d (CSV)

	date	ID	name	job	store_name	employee_ID
1	2021-05-07	123456789000	Alan	shopkeeper	VintageFun1	123456789000
2	2021-05-07	123456789001	Bob	shopkeeper	VintageFun2	123456789001
3	2021-05-07	123456789002	Caven	shopkeeper	VintageFun3	123456789002
4	2021-05-07	123456789003	Davi	shopkeeper	VintageFun4	123456789003
5	2021-05-07	123456789004	Eurus	shopkeeper	VintageFun5	123456789004
6	2021-05-07	123456789005	Fata	shopkeeper	VintageFun6	123456789005
7	2021-05-07	123456789006	Guy	shopkeeper	VintageFun7	123456789006
8	2021-05-07	123456789007	Halen	sales staff	VintageFun1	123456789007
9	2021-05-07	123456789008	Iris	sales staff	VintageFun2	123456789008
10	2021-05-07	123456789009	John	sales staff	VintageFun3	123456789009
11	2021-05-07	123456789010	Kaven	sales staff	VintageFun4	123456789010

13. Query for all the shopkeepers' details.

```
select distinct id, name, job, store_name
```

```
from employee
natural join job
where job = 'shopkeeper'
order by store_name;
```

```
select distinct id, name, job, store_name from employee natural join job where job =

select store_name from store_management group by store_name order by count(*) desc li

select distinct name, ID from employee, work_record where employee.ID = (select emplo
```

Output | assignment3.employee

7 rows | Tx: Auto | DDL | DML | Comma-...d (C

	id	name	job	store_name
1	123456789000	Alan	shopkeeper	VintageFun1
2	123456789001	Bob	shopkeeper	VintageFun2
3	123456789002	Caven	shopkeeper	VintageFun3
4	123456789003	Davi	shopkeeper	VintageFun4
5	123456789004	Eurus	shopkeeper	VintageFun5
6	123456789005	Fata	shopkeeper	VintageFun6
7	123456789006	Guy	shopkeeper	VintageFun7

14. Query for the largest shop (one that has the most employee) in a certain region:

```
select store_name
from store_management
group by store_name
order by count(*) desc
limit 1;
```

```
select store_name from store_management group by store_name order by count(*) desc

select name, ID from employee, work_record where employee.ID = (select employee_ID
```

Output | assignment3.store_management

1 row | Tx: Auto | DDL | DML | Comma-...d (C

	store_name
1	VintageFun1

15. Query for the employee that has the best attendance rating.

```
Select distinct name, ID
from employee, work_record
where employee.ID =
    (
    select employee_ID
    from work_record
    group by employee_ID
```

```
    order by count(employee_ID) desc
    limit 1
    );
```