School of Computing and Information Systems

INFO90002

Database Systems and Information Modelling

Practice Exam 1

Semester 1 2022

Reading Time: 15 minutes

Writing Time: 120 minutes (2 Hours)

This exam has 10 pages including this page
ATTEMPT ALL QUESTIONS IN ANY ORDER

Authorised Materials:

None

While you are undertaking this assessment, you MUST NOT

- make use of any messaging or communication technology
- record, screenshot, stream, upload or in any known format duplicate this document
- record, screenshot, stream, upload or in any known format duplicate your solutions
- make use of any world wide web or internet based resources such as wikipedia, github, stackoverflow, google, Weichat or any known search engine / messaging services
- act in a manner that could be regarded as providing assistance to a student who is undertaking this assessment or in the future will be undertaking this assessment
- seek assistance from any other student who is undertaking this assessment or in the future will be undertaking this assessment

Instructions to Students

- The total for this exam is 100 marks
- Attempt **all** 9 questions which are of unequal marks value
- Be sure to number your answers to ensure your questions is marked
- Questions can be answered in any order

- **PLEASE DO NOT USE RED font colour or RED pens.**
- You may revisit/edit your exam answers throughout
- You must not communicate with other students whilst taking this exam, e.g. using messaging, chat rooms or email. **Switch off your communications software on any device, switch off notifications on your smartphone and/or switch off your smartphone**.

IMPORTANT

- **Your file upload must be a single Word document before the elapsed time.**
- **No other document format will be assessed (e.g. Pages, txt, .SQL, etc).**
- **Email submissions <u>will not</u> be assessed.**
- **Every question attempt <u>must</u> be numbered (e.g. Q2C, Q1, Q6) to ensure it is assessed.**
- **The official exam language is English. Sections of the submission in languages other than English will NOT be assessed and will be marked as 0.**
- Before submitting your solution document, check that the diagram(s) is/are readable. It is your responsibility to ensure your answers are readable and make sense to the marker.

The work you submit **must be based on your own knowledge and skills** and without the assistance of any other person. You must not copy directly the text from any material without attribution.

Q1. ER Modelling (20 marks)

Q1. Criterion Classics is an on-demand streaming service that streams old television shows (e.g. 'Skippy') and movies ('Lawrence of Arabia'), including black-and-white movies ('Casablanca') and television shows ("I dream of Lucy"). Currently the company offers 3 subscriptions plans. Each plan has a code (B, S or P), name (Basic, Standard, Premium) and monthly price.

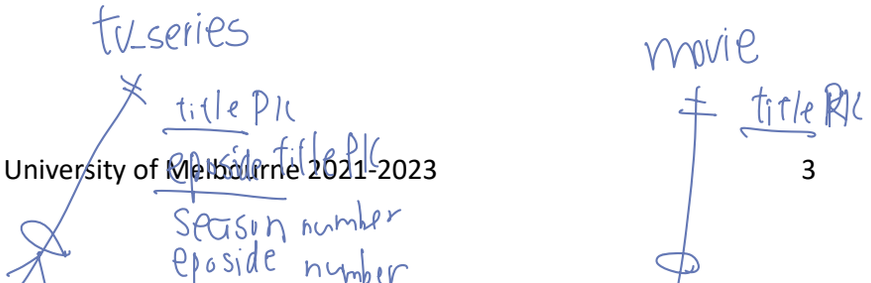
Each Criterion Classics account can have multiple profiles associated with it – for example, a family account could have one profile for the parents and one profile for each child. Profiles of type 'Child' can only view titles classified as 'U' (Universal) and 'G' (General). Each profile is stored with an avatar or digital image. About each account we store a name ('Mary Lawson'), the number of concurrent screens allowed, the day of the month (18th) the account is charged and the subscription plan.

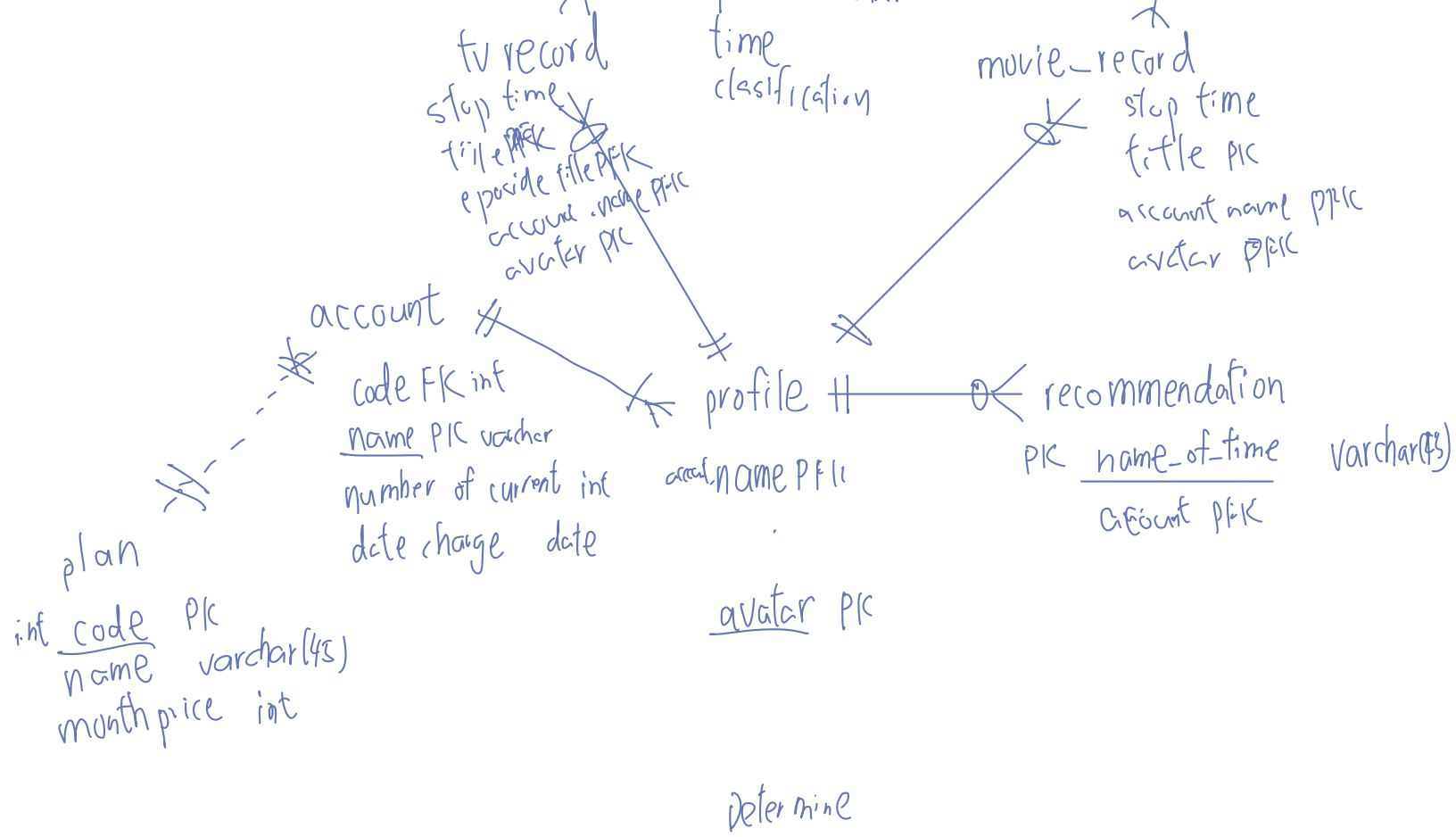
Each movie is an individual title. TV shows can have one or more seasons and up to 24 episodes in a season. Each title of a TV show (e.g. "Breaking Bad") will also have episodes title (e.g. "Ozymandias") as well as season number (5) and episode number (14) as well as its running time 47:35 (minutes and seconds) and classification ('MA').

Criterion Classics stores the length of each title/episode in minutes and seconds, as well as the elapsed time a particular profile stopped watching the title/episode. This is so users can 'continue watching' where they left off if they have to stop watching for any reason. If a user watches the title to the end, the abandon time is equal to the title/episode length time.

By collecting and storing this information, Criterion Classics can make recommendations to other users based on profile data analysis. Criterion Classics makes recommendations about rating appropriate titles to all profiles. Criterion needs to store the name of the title, the date it was suggested to a profile and the number of times (total count) a particular title has been recommended to a user. Using this information and the viewing information of a profile it can determine if its suggested viewing algorithm is working.

Q1. You are asked to model a **physical** Model of the Criterion Classics case study in **Crows Foot** notation. State any assumptions you have made.




tv record
stop time
title PFK
episode title PFK
account name PFK
avatar PFK
time
classification
movie_record
stop time
title PFK
account name PFK
avatar PFK
account
code FK int
name PK varchar
number of current int
date charge date
profile
account name PFK
avatar PK
recommendation
PK name_of_time varchar(45)
account PFK
plan
int code PK
name varchar(45)
month price int
determine

Q2. SQL (20 marks)

This case study is for the SQL model to help contextualise the Physical ER model

Lunch Rider

LunchRider is a new startup in the food delivery business. LunchRider allows people to order lunches from local food vendors and have them delivered by a delivery rider on a bike. When a customer opens the LunchRider app, it first presents a list of local vendors such as cafes, restaurants and snack bars. The customer clicks on a vendor. Then the app shows the meals available from that vendor.

The customer chooses which meals they want from that vendor: for example “2 chili burgers meals" and "1 mango smoothie”. The phone sends this order to the LunchRider server, along with the customer's phone GPS coordinates - the meals will be delivered to this location. Customers can click to “like” a meal and the total number of likes (across all customers) is displayed beside each meal. When a customer’s order is received, LunchRider broadcasts a work offer to riders who are near the customer. Riders see the offer pop up on their app and can press “accept” or “no thanks”.

The chosen rider goes to the vendor, picks up the meals, and delivers them to the customer’s location. We record at what time the rider delivers the order. Payment is automatically deducted from the customer’s credit card and bank transactions are handled by the bank. LunchRiders provide their name, mobile phone number, and date of birth. Whenever a rider is on duty, the rider’s app sends the GPS coordinates about once per minute, allowing us to keep track of each rider’s location.

If someone wants to order a meal, they need to first register as a customer, giving their name, email address, mobile phone number and current credit card. Over time a customer may register more than one credit card. Vendors must register the name and address of their business, including GPS coordinates and a contact email address, and provide the name, price, description (max 1,000 characters) and photo of each meal that they want to sell.

After delivery, the app allows the customer to rate the rider’s service, choosing from 1 star (worst), 2, 3, 4 or 5 stars (best). The app also allows the customer to add the rider to their “favourite” list. LunchRider uses this information to help choose riders for future work offers.

All locations are recorded as a pair of numbers representing latitude and longitude. Latitudes are between -90 and 90 degrees (south pole to north pole) while longitudes are between -180 and 180 degrees (west or east of the prime meridian). LunchRider uses a precision of 4 decimal places, which is about 11 metres at the equator (smaller in Melbourne). For example, the Doug McDonell building at The University of Melbourne is at latitude -37.7989, longitude 144.9627.

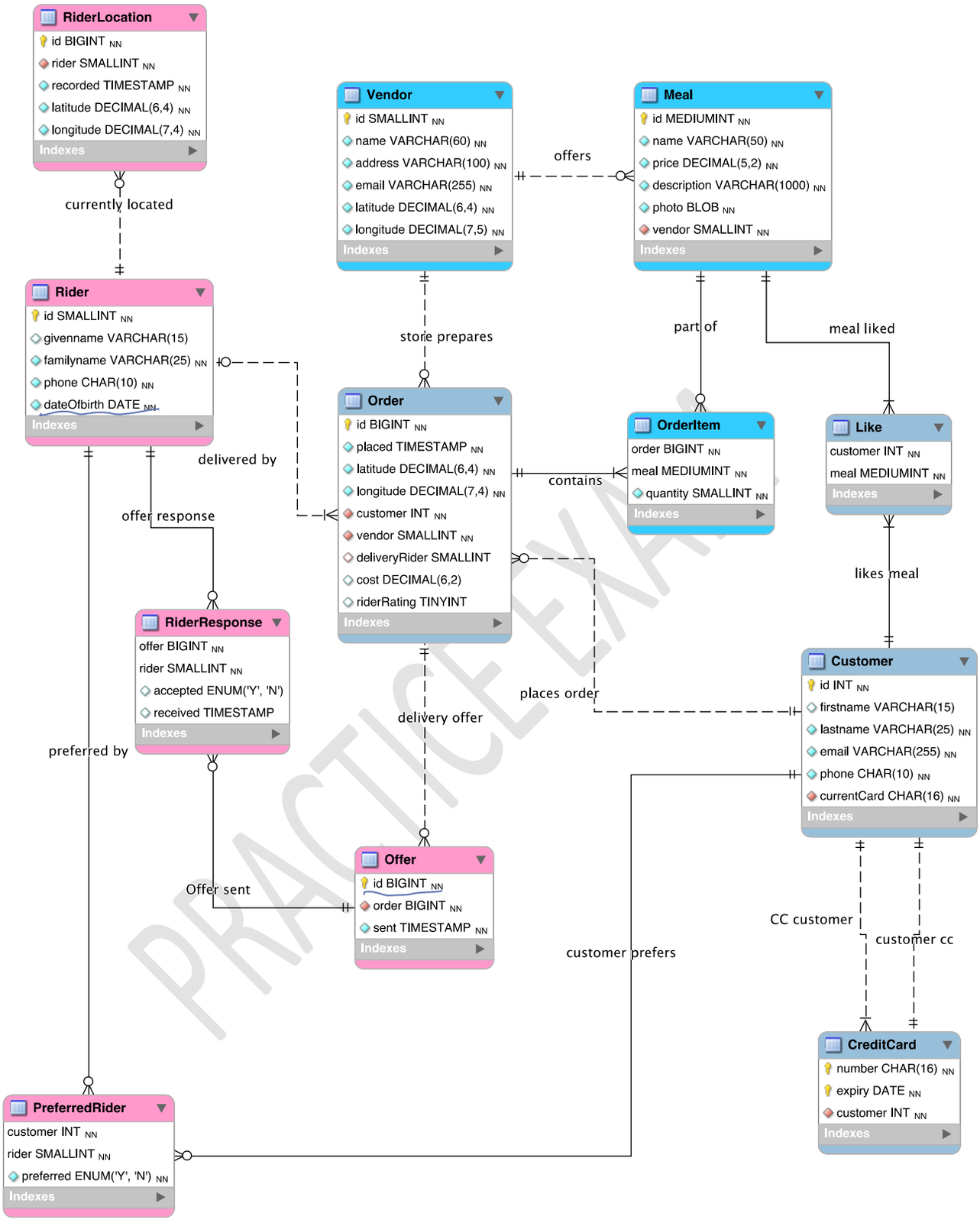


Figure 2: The Lunch Rider Entity Relationship Model

Questions 2A-2G require you to write one single SQL statement per question. Do not use views, temporary tables or inline views. Format code for ease of reading. Ensure user-friendly output by renaming columns where appropriate. Values of column fields are in italics e.g. *Tom Hardy.*
For Example:
Q. List the name and salary of *Jane Grey*

```
SELECT firstname, lastname, salary
FROM  employee
WHERE firstname = 'Jane' and lastname = 'Grey';
```

Q2A. Write the SQL for the following query:
List all Vendor names who offer a meal where the price is more than $60.00 and below $80.00.

(1 marks)

Q2B. List the names of riders born in *1997* have accepted an offer in *August 2020*?

(3 marks)

Q2C. How many customers have ordered and liked the '*Betty Deluxe Burger Meal*' meal ordered from the Melbourne store with coordinates latitude *-37.8156* and longitude *144.9636* ?

(4 marks)

Q2D. List the names of riders who are also customers of the Lunch Rider app (assume the rider uses the same phone number as they would as a customer).

(4 marks)

Q2E. List the vendor names who have at least one ordered meal never liked by any customer.

(3 marks)

Q2F. List the vendor names who have never had any ordered meal liked by a customer.

(3 marks)

Q2G. Name the riders who have received a rating of 4 or above from more than 500 different customers**.**

(5 marks)

Note marks are just indication of difficulty, e.g. 1 mark question is considered easy.

Q2A:

✓
```
select distinct vender.name
from vender
inner join meal
on vender.id = meal.vender
where meal.price ≥ 60 and meal.price ≤ 80
```

Q2B:

```
select rider. familyname
from rider
inner join riderResponse
on rider.id = RiderResponse.rider
where riderResponse.accepted = 'Y' and rider.dateofBirth = 1997
And MonthName (RiderRespone.recieved) = 'August' and Year(RR recieved) = 2020
```

Q2C:

```
select count (customer.id)
from customer
inner join order
inner join vender
on customer.id = order.customer
an vender.id = order.vender
group by customer.id
where vendor.longtitud = 144.9636 and latitude = -37.8156.
And m.name = 'Betty Deluxe Burger Meal'
```

Q2D:

```
select rider.name
from rider
```

```
inner join preferredRide
inner join customer
on preferredRide.rider = rider.id
and preferredRide.customer = customer.id
where ridername = customer.name
```

```
select firstname, lastname
from customer
where exists
      (select *
      from rider
      where Rider.phone = (customer.phone)
```

Q2E

Q2E. List the vendor names who have at least one ordered meal never liked by any customer.

```
SELECT DISTINCT vendor.name
FROM VENDOR
INNER JOIN MEAL
INNER JOIN OrderItem
ON Vendor.ID = Meal.Vendor AND Meal.ID = OrderItem.meal
WHERE Meal.ID NOT IN
     (SELECT meal
     FROM Like);

-- (NOT EXISTS also possible)
```

Q2F. List the vendor names who have never had any ordered meal liked by a customer.

```
SELECT DISTINCT vendor.name
FROM VENDOR
INNER JOIN MEAL
INNER JOIN OrderItem
ON Vendor.ID = Meal.Vendor AND Meal.ID = OrderItem.meal
WHERE Vendor.ID NOT IN
      (SELECT DISTINCT Vendor.id
      FROM Vendor
      INNER JOIN Meal ON Vendor.ID = Meal.Vendor
      INNER JOIN Like ON Like.meal = Meal.id
      );
```

Q2G. Name the riders who have received a rating of 4 or above from more than 500 different customers.

```
SELECT givenname, familyname
FROM Rider INNER JOIN Order
ON Rider.ID = Order.deliveryrider
WHERE Order.riderrating >= 4
GROUP BY Rider.ID
HAVING COUNT(DISTINCT(Order.customer)) > 500;
```

Q2D:

```
select name
from customer
where exist
        (select riderID
         from rider
         where customer.phone = rider.phone)
```

Q2E:

```
select distinct name
from vender
inner join meal
on vender.id = meal.vendor
inner join orderitem
on
where meal.id not in
        (select meal
         from like)
```

Q2F:

```
select distinct name
from vendor
```

inner join meal
inner join orderitem
on
and
where

Q26:

```
select givenname, firstname
from rider
inner join order
inner join customer
on
and
where order.ridercating > 4
group by givenname, firname
having count(customeid) > 500
```

Q3. Normalisation (20 marks)

The table below is part of the medical records for a Veterinarian Clinic

PRACTICE (AnimalID, AnimalName, AnimalType, owner_ID, Owner, Phone, ConsultDateTime, procID, procDescription)

Animal ID	Animal name	Animal Type	Owner_ID	Owner	Phone	ConsultDateTime	ProcID	Description
317	Ralph	Dog	10	Julie Sumner	0409 673-888	13-Oct-22 9:00	101	Annual Checkup
317	Ralph	Dog	10	Julie Sumner	0409 673-888	27-Apr-22 11:15	115	Teeth Clean
317	Ralph	Dog	10	Julie Sumner	0409 673-888	14-Oct-23 9:30	119	3 month Checkup
398	Zeno	Canary	23	Tony Rijks	0408 322-444	21-Jul-23 14:30	105	Parasite treatment
398	Zeno	Canary	23	Tony Rijks	0408 322-444	14-Oct-23 16:00	119	3 month Checkup
441	Panda	Short haired cat	47	Helene Hanff	0419 121-212	24-Apr-23 9:00	715	Initial Consultation
441	Panda	Short haired cat	47	Helene Hanff	0419 121-212	27-Apr-23 9:30	115	Teeth Clean
398	Zeno	Canary	23	Tony Rijks	0408 322-444	1-Mar-24 9:00	001	6 month Checkup

The combination of AnimalID and procID is the candidate key for the relation. The following functional dependencies hold:

- AnimalID → animalName, animalType
- owner_id → owner, phone
- procid → description
- animalID, procID → consultDateTime

Assume more than one procedure can be performed during the consultation.

Q3. Normalise the data to third normal form (3NF). At each stage give an example of an anomaly. Show each stage of normalisation (i.e. 1NF, 2NF, 3NF).

Key: **Underline** primary key *ITALIC* foreign key ***Underline + ITALIC*** primary foreign key

1 NF: PRACTICE (AnimalID, AnimalName, AnimalType, owner_ID, Owner, Phone)
PK

Consultant (ConsultDateTime, procID, procDescription, AnimalID)
PK PK PFK

2NF PRACTICE (AnimalID, AnimalName, AnimalType, owner_ID, Owner, Phone)
PK

consultant (ConsultDateTime (PK), procID (PFK), AnimalID (PFK))

proceducer (procID (PK), procDescription)

3NF PRACTICE (Animal (PK), AnimalName, AnimalType, OwnerID (FK))

Owner (OwnerID (PK), owner, Phone)

consultant (ConsultDateTime (PK), procID (PFK), AnimalID (PFK))

proceducer (procID (PK), procDescription)

In relational database, If we want to get the consistency data, we have to keep updating. But when it's updating, we have no availability to get the data. We have to wait. On the other hand, if we want to access the data all the time, we can use asychronous which update not in the prime time. But the data will not have integrity.

In the NoSQL database, it is basic avaliable, soft sort, and eventually consistene, We get access the data all the time, but we might get the not update data until it has updated.

Q4. NoSQL (4 marks)

Q4. There are trade-offs between the principles of ACID and BASE. Discuss the trade-off between availability and consistency for relational and NoSQL databases. Illustrate your answer using either the example of Facebook or the example of Twitter.

(4 marks)

Q5. Applications (4 marks)

Q5. There are problems with giving end users an SQL interface to access a database. Describe **two** (2) distinct problems, and for each, how providing application software to users solves the problem.

(2+2=4 marks)

Q6. Distributed Databases (4 marks)

Q6. Describe how synchronous updates work and the advantages and disadvantages of this approach.

synchronous update will wait for everynode is ready and then start the processing of update. It's a real-time update. When it's update, other requests or queries need to stop and wait. It can hold the data consistency. However, it'll reduce the availability of data. **(4 marks)**

Q7. Data Warehouses (14 Marks)

Sennheiser Australia wants to track information about the selling of audio equipment (microphones, headphones, turntables) via physical retail stores located throughout Australia. You need to design a data warehouse to report information about sales of audio equipment over time. You need to store the retail store (store ID, Manager's first name, Manager's last name, phone number, address, state), and product (product-id, name, description, category, retail price, wholesale price). The sales managers want to find the number of items sold (e.g. turntables), the revenue and profit. The information needs to be accessible by the retail store, its state, product name (e.g. "Sen-300-II") and for different times (week, month, quarter, half year and year).

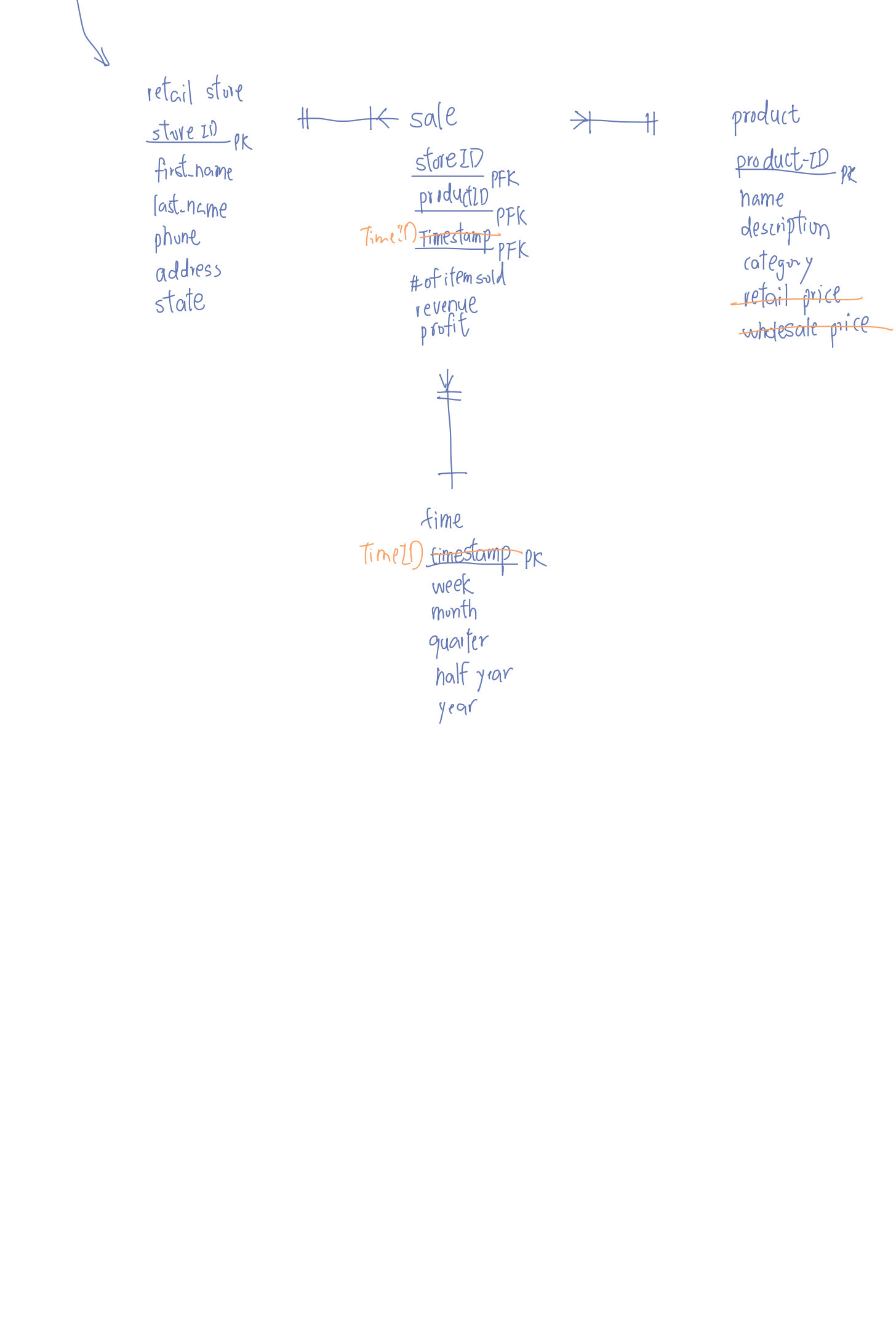
Q7A. Draw a *star schema* to support the design of this data warehouse, showing the attributes in each table. Be sure to denote PK, FK and PFK.

(8 marks)

Q7B. Briefly explain what OLTP and OLAP databases are and demonstrate the difference between these two types of databases.

OLTP: Online transaction processing, it is a transaction database, You can update, delete or select on these database. **(6 marks)**

OLAP: Online analytical processing. This is a information database. It's a schema on read. It record the historical transaction. It's used to analysis the future and make the decision for the future.


retail store
store ID PK
first_name
last_name
phone
address
state
sale
storeID PFK
productID PFK
TimeID timestamp PFK
#of item sold
revenue
profit
product
product-ID PK
name
description
category
retail price
wholesale price
time
TimeID timestamp PK
week
month
quarter
half year
year

Q8. Security and Backups (4 marks)

Q8. What are the advantages and disadvantages of a logical backup when compared with a physical backup of a database?

(4 marks)

END OF EXAM

advantage: faster than physical backup

Don't need the log file

Only backup the partial of data (start backup from the latest one)

When it runs the backup, don't need to shut down the program.

disadvantage: It's is independent from the machine.

It has higher risk to lose data if only do logical backup.