

# CS250 CW2 ER, Normalization, SQL, PHP 2020-21

## Database Systems Coursework 2 (worth 20% of CS-250)

Date issued: **26 October 2020**  
Part A Deadline: **23 November 2020 11am**  
Part B Deadline: **30 November 2020 11am**

This coursework is about ER, Normalization, SQL and PHP. It contains two parts, and you must complete all parts.  
**Important!** Please read through this coursework during the *first week* that it is issued and ensure you understand what is required. If you have any questions, please ask – **do not leave this until the last day or two before the deadline**. This coursework contains many materials and is a large piece of work (worth 20%). Do not under-estimate the time you need. Please also read the General Notes and Advice from past students, at the end of this coursework description.

## Marking Scheme

### Part A (100 marks): Multiple choice

Each correct answer will be given marks as stated in the question statement.

### Part B (100 marks): Hand-on (SQL & PHP)

#### Task 1 (60 marks):

Two problems carry 50 (20+30) marks in total. Full marks will be awarded if your SQL statements produce the correct results. 10 Marks will be awarded for the clear presentation and formatting (including comments).

You **must** show the SQL statement(s) and the results obtained from your database (e.g., copy-and-paste the results from a mysql console).

#### Task 2 (40 marks):

You are required to submit your PHP code (in *pdf*) that generates the “Amazon Order History” for any userid.

30 marks will be awarded if your PHP code works as expected:

Up to 10 marks will be awarded if the generated order history look as close as possible to the given ones:

a) Provide screenshots of *three* order history with *different* userids (see specification in Task2).

b) Clear presentation and formatting (including comments for your code).

## Coursework Submission (please read the instructions carefully)


You should submit the following to the Canvas assignment collection page:


- ⇒ Part A: fill the Part A quiz form on Canvas for automatic marking, electronic submission.
- ⇒ Part B: *a* written report (*pdf* only), electronic submission.

**Important!** There is a *10-page limit (A4)* for *PartB*, with a *minimum font size: 10 for Arial font (11 for Times New Roman)*. The minimum font size applies to *all* text in tables, images, screenshots, and handwriting in your pdf. All pdfs will be *printed in 2-up format* for manual marking and feedback. If your text is smaller than the minimum font size, or is incomprehensible on the printed copies, **NO marks** will be given. Submission of non-pdf format (e.g. docx) will **NOT** be marked. *Estimate* for Part B: 3 pages (Task2) + 4 pages (Task3) < 10 pages.

If it is more than 10 pages, your solution *may be wrong*. Check your answers and make good use of whitespace.

Under Canvas > Module > Soft Skills, you should find documents that discuss general formatting and efficient use of space.

 Reasons for page limit.pdf

 Readme! SoftSkills.pdf

You are only allowed to submit *one pdf*. If you have multiple pdfs, please combine them into one (read SoftSkills.pdf).

**Important!** you must use the answer sheet (docx) provided and convert it into *pdf* before submission.

Non pdf format (e.g. doc, docx files) will **NOT** be marked. All pdfs will be **automatically chopped** to the specified page limit, and *no further marks* will be given *beyond* the page limit.

## CW Contributions

You are required to **work in a group of 3 students** in this coursework, and use the provided submission link and answer sheet. Students in the same group will receive the **same marks** unless individual contributions (0-100%, see below) are specified.

The coursework is designed for 1 student to complete in three weeks (we are generous to offer four-five weeks). The intention of group work is for you to achieve a better marks through *discussion* and *sanity checks*. You should follow the steps below:

- Do your own work *individually*,
- THEN** cross check your work with your group in *multiple* meetings (**do not share your work beforehand**),
- If there are discrepancies, discuss, explain and convince one another via discussion and sanity checks,
- Revise the answer appropriately and
- Submit one single piece of work for your group – equal contribution (all 100% contribution).

Alternatively, you may choose to do the work together (e.g. sitting in front of the same machine / via Skype/Zoom) as a group, and practise collaboration and cooperation. It is acceptable. It is your duty to ensure you and your partner contribute equally. All students in the same group will be given the same marks (all 100% contribution). Note however that you may not spot issues with your collaborative solutions, to learn from the others' mistakes, and thus not learn all the material in the course.

### Uneven Contributions

This is a group coursework. You should be aware that some groupmates **may not contribute anything at the end**. You should practise what you have learnt in CS-230 to ensure the team works. You should have regular meetings **early** (start contacting at least **two weeks** before deadline). Before each meeting, some goals must be set (e.g. to discuss certain set of answers). Do not leave things until the last few days – you are warned.

**Caution!** You **should not share** your work before any (physical, Zoom / skype etc) meetings. In each meeting, each member should show the work they have attempted **before** engaging in discussion. Otherwise:

- a) If your group member **does not** respond to contacts at all, never show up nor engage, keep delaying the meetings, their contribution is **0%**. (Each group member is accountable to their active engagements, i.e. not being asked/chased to do so.)
- b) If your group member **does not** provide any work for cross checking (or not meeting agreed deadlines), contribution **0%**.
- c) If your group member only provides some parts of CW2, but **does not** engage in discussion nor meet up, and all their answers look randomly picked (esp. PartA) with no explanation, contribution is **0%**. All students are responsible for **explaining** their solutions with the group **well in advance of the deadline**.
- d) If your group member does some parts of CW2 (but not all), actively engage and discuss in the meetings, their contribution **1-99%**, is negotiated/agreed across all group members. The student(s) who contribute(s) the most get **100%** contribution.

After marking, each student marks will be prorated, reflecting each contributions: CW2 group marks  $\times$  contributions (in %).

**One week before the deadline:** if situations (a-c) arise(s), write an email to the instructor for records, specifying the group number and provide evidences. In the situation (a) above that your partner(s) did not collaborate/respond/meet up/delay meetings, you should further provide **two proofs of attempted communications** (e.g. emails, facebook, txt messages, slack) showing that you have tried to contact/communicate with your partner(s) **at least one week before the deadline**. Your non-collaborating partner(s) will be given **0%** contribution, unless they work with you in the last week. If your non-collaborating partner(s) **accuses** you of the same with evidence, each of your coursework will be individually marked, but all your final marks will be halved because no collaborating/discussing effort is shown - it defeats the purpose of the group work.

**Caution!** CW2 is designed to be completed individually. The group discussion is to help you fix those careless mistakes and for sanity check. You should **NOT** expect your groupmates to provide correct answers or contribute a lot in the group. Instead, you should contribute pro-actively. Even your groupmate does not contribute much, you should still get them to a meeting, and **explain** your answers to them. A simple "why?" from your groupmate would prompt you to rethink/reconfirm your answers. When you **explain** your answers in an alternative way, you learn a new way of thinking to answer the question. This is what "discussion and sanity check" means. This is called "Teaching to learn". (It is more blessed to give than to receive.) **No doubt**, non-contributors (those who do not provide any results for cross checking in meetings) should be given 0% contribution!

### CW2 Group Sign-Up

You **must sign up to your group** on the canvas page **two weeks before the due date**. If you have not signed up to a group **two weeks before the due date** (even if you are partnering with the same students in CW1), the instructor will **randomly assign** you into group. You must then work with your new partner even though you may have been working with someone else. In the exceptional case where you have genuine reasons to do the coursework individually, you must seek approval from the instructor and provide satisfactory evidence (e.g., medical certificate).


If you need help to form a group (e.g. do not want to work with CW1 group or know no one), but would like to be team up with a group earlier, please drop me an email:

- a) If you are looking for a group, email me with the following **title**: CS-250 CW2 team up: XXXXXX
  - b) If your group is looking for one more member, email me with **title**: CS-250 CW2 team up: XXXXXX, YYYYYY
- where XXXXXX and YYYYYY is/are your student id(s).

Once I received similar requests, I will let you know in a first come first serve order. You may then discuss whether to form a group before the team up due date. Note that my reply is only a suggestion – You (not instructor) should sign yourself up on canvas into group. Some students choose to ignore our suggestion or form a different group themselves. It is fine. The policy is that **if you have not joined a group on canvas after the team up due date, you will be randomly assigned a new group**.

## General Notes and Advice

### Late Submission

Standard University and College of Science rules apply for late submission, i.e. **zero** if handed in late. Read  [Readme! SoftSkills.pdf](#)

## Extenuating circumstances

If there are extenuating circumstances, you should follow the CoS procedure (i.e. contact [csadmin@swansea.ac.uk](mailto:csadmin@swansea.ac.uk), not the instructor) for the waiver of these penalties, and you must provide satisfactory evidence. The instructor has **NO right** to authorise late submission. All requests are looked at and authorised by the department individually. The rest of the group should inform the instructor and continue with the submission. The extenuating circumstance will be dealt with centrally, and those affected (student/group) will be informed of the appropriate actions subsequently, according to departmental guidance.

## Academic Integrity and Misconduct

**Important:** You are expected to submit your own work for this coursework.

### Academic Integrity and Academic Misconduct Statement

By submitting this coursework, electronically and/or hardcopy, you state that you fully understand and are complying with the university's policy on Academic Integrity and Academic Misconduct. The policy can be found at: <https://myuni.swansea.ac.uk/academic-life/academic-misconduct/>

On many occasions when working on coursework and projects, it is useful to ask others (the lecturer, other students) for hints or debugging help, or to talk generally about the written problems or programming strategies. Such activity is both acceptable, but you must indicate in your submission any assistance you received. Any assistance received that is not given proper citation will be considered a violation of the policy on plagiarism. In any event, you are responsible for understanding and being able to explain on your own all written and programming solutions that you submit. The instructor will pursue **aggressively** all suspected cases of plagiarism and collusion, and they will be handled through official University channels.

You should **always** keep a record of your own work before the CW marks are finalised. You may need to prove that you have carried out the work yourself or as a group, e.g. in suspected cases of plagiarism.

**Tips:** MySQL / PhpMyAdmin have a utility to **import data** to the database, and this will most likely be the easiest way for you to get the data into the system. If you have any problems with this, please contact me early. **No help will be provided if you leave it until the last week of the coursework deadline.** If you cannot complete any one of the problems, you should at least make some attempt and perhaps explain your approach to getting the solution so that partial marks can be awarded.

## CS-250 CW2 Advice from past years' students

Form group early

Group member is important

*(Gary: don't complaint if you are randomly assigned)*

Read the Lab materials, it helps "A LOT"!

Read loads of examples before trying PHP.

Earn all effort marks to boost your marks.

*(Gary: remember to do the evaluation too : ) )*

Review lecture and tutorial notes.

Lots of problems with XAMPP - start early.

PHP is hard, but good fun.

*(Gary: PHP is fun and is the easiest among all.)*

Work on the presentation early on.

PHP: Define action plan, split task into manageable pieces.

*(Gary: would be a mini SE project. Check other work)*

Ask questions on discussion board early.

Have at least 2 face-to-face meeting with partners.

*(Gary: I expect more than that.)*

Crop essential parts from screenshots to save space...

Make good use of all whitespaces.

*(Gary: Observe the minimum font size limitation.)*

Start the coursework really quickly, by that we mean really really quickly!

*(Gary: this group scores 97%.)*

*In general, you must not under-estimate the CW2 workload – it is 20% of CS250!*

*You may need time to get familiar / revise lecture and tutorial materials before attempting CW2.*

*Further, CW2 PartB requires preparation time (e.g. setup XAMPP, try out MySQL, do Self-Lab1-4).*

*We have covered all these in lectures and tutorials, and you should try them practically (See the Lab Materials folder).*

*If you have skipped any lectures / tutorials, make sure you catch up the materials yourself without delay.*

*If you have not installed XAMPP, remember to try it early. If there are problems, come to XAMPP optional Labs.*

*Act now.*

*(Again, how to eat an elephant?)*

I have tried to provide all the information you require to complete this coursework. If you think some information is missing or not clear, you can either ask me for the information (preferably by email).

Dr Gary KL Tam  
k.l.tam@swansea.ac.uk

# CS250 CW2 ER, Normalization, SQL, PHP 2019-20

---

## Database Systems Coursework 2 (worth 20% of CS-250)

Date issued: **26 October 2020**  
Part A Deadline: **23 November 2020 11am**  
Part B Deadline: **30 November 2020 11am**

This coursework contributes 20% of the assessment for module CS-250. You are given the opportunity to develop practical experience, from ER-diagrams to SQL and PHP implementation. Based on the feedback from past year students, CW2 has been separated into two parts with two deadlines. You should do all of Part A(10%)+Part B (10%).

### Part A: (worth 10% of CS-250)

**Problem 1.** In an ER Diagram, what is used to represent a relationship? (@2)

- a) Diamond
- b) Rectangle
- c) Square
- d) Ellipse

**Problem 2.** What helps you identify a relation in an ER diagram from a requirement statement? (@2)

- a) noun
- b) verb
- c) conjunction
- d) adjective

**Problem 3.** When converting ER diagram to tables, which of the following relationship type(s) allow(s) us to choose the primary key of either participating entities as the primary key of the relation? (@2)

- a) Many-to-many
- b) One-to-one
- c) One-to-many / many-to-one
- d) All of these
- e) None of the above

**Problem 4.** Which is an entity in a database designed for a local convenience store? (@2)

- a) groceries
- b) address
- c) telephone number
- d) opening hours

**Problem 5.** What is the type of relationship between a bus and a route? (@2)

A bus can be running on one route in the morning, and another route in the afternoon. A route consists of many buses.

- a) One-to-one
- b) One-to-many
- c) Many-to-one
- d) Many-to-many
- e) None of the above

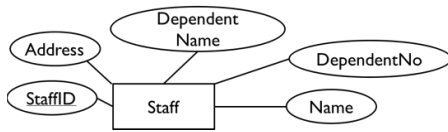
**Problem 6.** Which ER diagram would lead to the following set of table schemas. (@6)

Underlined attribute: primary key. *Italic* attribute: foreign key.

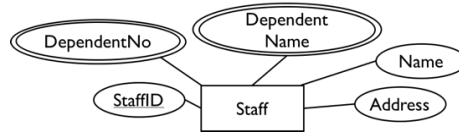
Staff (StaffID, Name, Address)

Dependent (StaffID, DependentNo, *DependentName*) reference (Staff)

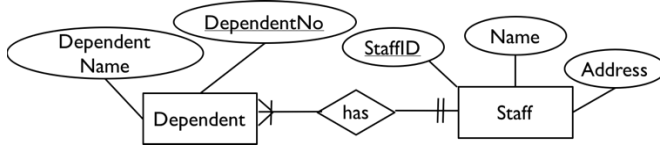
a)



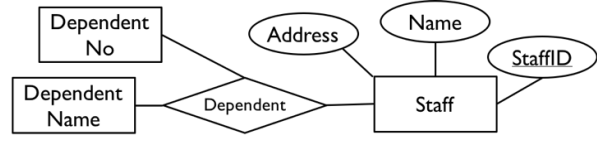
b)



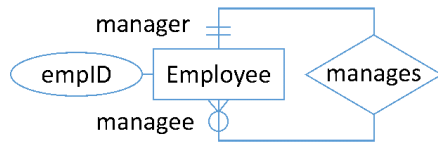
c)



d)



**Problem 7.** With regard to the following ER diagram, which statement(s) is/are true? (@6)



- I) A managee can have many managers.
- II) An employee may not manage anyone.
- III) An employee can be both a manager and a managee at the same time.
- IV) Manages is a binary relationship because there are two roles involved.
- V) Not every employee is managed by someone.
- VI) An employee must manage multiple employees.
- VII) The boss can be an employee, and be identified in the resulting table.
- VIII) The ER diagram can be converted into one table, and created using the SQL below:

```
create table employee (
    EmpID integer not null,
    MgrID integer,
    primary key (EmpID),
    foreign key (MgrID) references employee (EmpID)
);
```

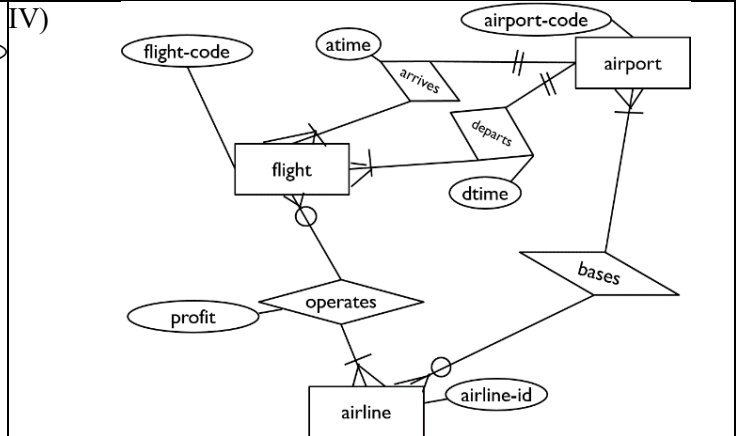
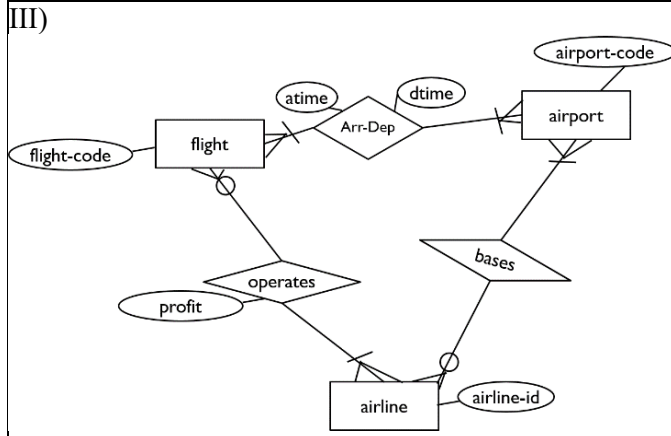
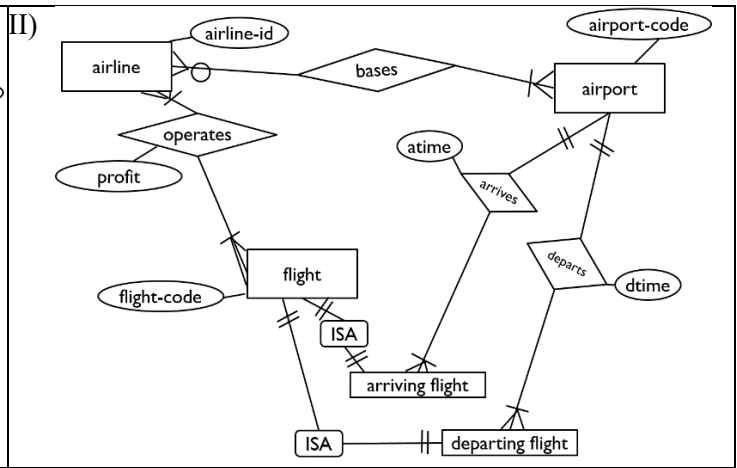
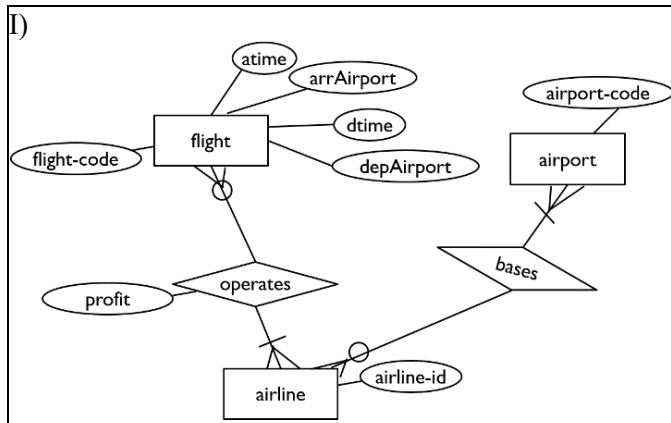
You are given the following requirement, answer problems 8-9.

For each airline, we should store its unique airline-id. For each flight, we should store its unique flight-code. For each airport, we should store its unique airport-code. Each flight must be operated by at least one (can be multiple) airline(s). We have to store the profit that each airline earns by operating the flight (i.e., in case of a joint-operation, there are several profits for the same flight). Every airline must be based in at least one (can be multiple) airport. Each airport may be a base for any number of airlines. For each flight, we should store the departing time and arriving time at its departing airport and arriving airport. We must ensure the airports from which the flights depart/arrive do exist.

**Problem 8.** Select all the ER diagram(s) that capture(s) the above requirement correctly. (@6)

**Hint:** draw the ER diagram and verify your results, then compare and discuss with your team member.





**Problem 9.** Using the answer(s) in problem 8, convert the ER diagram(s) into the fewest number of tables avoiding NULL values. Select the set(s) of tables that correctly capture the above requirement? (@6)

**Hint:** try to convert *all* ER diagrams in problem 8 into tables, then compare and discuss with your team member.

I)

airport (airport-code)

airline (airline-id)

flight (flight-code, airport-code, aAirport-code, atime, dAirport-code, dtime) FK airport(airport-code)

operates (flight-code, airline-id, profit) FK flight(flight-code), FK airline(airline-id)

bases (airline-id, airport-code) FK airline(airline-id), FK airport(airport-code)

II)

airline (airline-id)

airport (airport-code)

flight (flight-code, aAirport-code, atime, dAirport-code, dtime)

operates (flight-code, airline-id, airport-code, profit)

FK flight(flight-code), FK airline(airline-id),  
FK airport(airport-code)

III)

airline (airline-id)

airport (airport-code)

flight (flight-code, aAirport-code, atime, dAirport-code, dtime)

bases (airline-id, airport-code)

operates (flight-code, airline-id, profit)

FK airline(airline-id), FK airport(airport-code)  
FK flight(flight-code), FK airline(airline-id)

IV)

airline (airline-id)

airport (airport-code)

flight (flight-code, aAirport-code, atime, dAirport-code, dtime)

bases (airline-id, airport-code)

operates (flight-code, airline-id, profit)

FK (aAirport-code) ref airport(airport-code),  
FK (dAirport-code) ref airport(airport-code)  
FK airline(airline-id), FK airport(airport-code)  
FK flight(flight-code), FK airline(airline-id)

Answer problems 10-12 using the information below.

There is a relation scheme “ASDA” with the following 12 attributes:

ASDA(storeID<sup>1</sup>, tel#<sup>2</sup>, address<sup>3</sup>, manager<sup>4</sup>, itemCode<sup>5</sup>, itemDesc<sup>6</sup>, unitprice<sup>7</sup>, quantity<sup>8</sup>, TC#<sup>9</sup>, totalpay<sup>10</sup>, typepay<sup>11</sup>, datetime<sup>12</sup>)

Consider the following functional dependencies:

storeID → tel#, address, manager

itemCode → itemDesc, unitprice

TC#, itemCode → quantity

TC# → storeID, datetime, totalpay, typepay

**Problem 10.** Which of the following closure(s) cover(s) all 12 attributes of schema ASDA? (@6)

I) {storeID, itemCode, quantity}<sup>+</sup>

II) {itemCode}<sup>+</sup>

III) {TC#}<sup>+</sup>

IV) {storeID, itemCode}<sup>+</sup>

V) {storeID, itemCode, TC#}<sup>+</sup>

VI) {itemCode, TC#}<sup>+</sup>

VII) {TC#, storeID}<sup>+</sup>

VIII) {itemCode, totalpay, TC#}<sup>+</sup>



TC# means “Till Counter number”.

**Problem 11.** Which of the following set(s) is/are the candidate key(s) of schema ASDA? (@6)

I) {storeID, itemCode, quantity}

II) {itemCode}

III) {TC#}

IV) {storeID, itemCode}

V) {storeID, itemCode, TC#}

VI) {itemCode, TC#}

VII) {TC#, storeID}

VIII) {itemCode, totalpay, TC#}

**Problem 12.** After applying normalization to schema ASDA up to 3NF, the set of tables is: (@6)

Hint: draw a functional dependency diagram.

a) TC#, datetime, totalpay, typepay  
itemCode, itemDesc, unitprice  
storeID, tel#, address, manager  
itemCode, TC#, storeID, quantity

b) TC#, datetime, totalpay, typepay, storeID, tel#, address, manager  
TC#, itemCode, quantity  
itemCode, itemDesc, unitprice

c) TC#, itemCode, quantity  
itemCode, itemDesc, unitprice  
TC#, datetime, totalpay, typepay, storeID  
storeID, tel#, address, manager

d) itemCode, storeID, TC#, quantity  
itemCode, itemDesc, unitprice  
TC#, datetime, totalpay, typepay  
storeID, tel#, address, manager

**Problem 13.** Which operator is used to select values within a range? (@2)

- a) range
- b) in
- c) between
- d) within
- e) with

**Problem 14.** Which SQL command will be used when you post a new message on your facebook account? (@2)

- a) insert
- b) alter
- c) drop
- d) add
- e) delete

**Problem 15.** Which statement is true for a right-outer join? (@2)

- a) It includes all the tuples from both the relations satisfying the join condition along with all the tuples in the left relation that do not have a corresponding tuple in the right relation.
- b) It includes all the tuples from both the relations satisfying the join condition along with all the tuples in the right relation that do not have a corresponding tuple in the left relation.
- c) It selects all the tuples satisfying the join condition along with the tuples for which no tuples from the other relation satisfy the join condition.
- d) It selects all the tuples satisfying the join condition only.

**Problem 16.** Which SQL command deletes all information about a relation from a database? (@2)

- a) delete table
- b) create table
- c) truncate table
- d) drop table
- e) alter table

**Problem 17.** Which is a join condition in the following SQL statement? (@2)

```
SELECT fid
FROM   flight, airline
WHERE  flight.code = 'CX'
      AND flight.fid = airline.fid
```

- a) FROM flight, airline
- b) flight.code = 'CX' AND flight.fid = airline.fid
- c) flight.code = 'CX'
- d) flight.fid = airline.fid
- e) None of the above

**Problem 18.** Which SQL command finds those groups meeting stated conditions? (@2)

- a) order by
- b) group by
- c) where
- d) having
- e) select

**Problem 19.** A student tried to rewrite the following relational algebra expression into SQL

$\Pi_{Lname} (\sigma_{Mid = 'CS-270' \vee Salary \geq 5000} (Lecturer \bowtie Teach))$

```
select Lname
from Lecturer, Teach
where Mid = 'CS-270' and Salary > 5000;
```

List all the problem(s) in the above SQL statement? (@8)

- I) incorrect boolean operator (e.g. and, or, not)
- II) missing join condition
- III) missing unique keyword
- IV) missing project statement
- V) missing intersection statement
- VI) incorrect comparison operator
- VII) missing distinct keyword
- VIII) missing table(s)
- IX) missing select statement
- X) missing having statement



**Problem 20.** You are provided the following list of SQL statements. You try to type these commands sequentially in a MySQL prompt, whilst fixing all issues on the way to completion.

```
1 SHOW DATABASES;
2 DROP DATABASE IF EXISTS cs250exam;
3 CREATE DATABASE IF NOT EXISTS cs250exam;
4 CREATE TABLE prof (
5     pid CHAR(20), name CHAR(20),
6     dept CHAR(20), PRIMARY KEY (pid)
7 );
8 CREATE TABLE teach (
9     pid CHAR(20), cid CHAR(20),
10    year INTEGER,
11    PRIMARY KEY (pid, cid, year),
12    FOREIGN KEY (pid) REFERENCES prof (pid)
13    ON DELETE CASCADE ON UPDATE CASCADE
14 );
15 INSERT INTO teach VALUES ('p1', 'c1', 2011);
16 INSERT INTO prof VALUES ('p1', 'Adam', 'CS');
17 INSERT INTO prof VALUES ('p2', 'Bob', 'EE');
18 INSERT INTO teach VALUES ('p2', 'c1', 2012);
19 SELECT pid FROM prof, teach
20 WHERE prof.pid = teach.pid;
21 DROP TABLE teach;
22 DROP TABLE prof;
```

You are required to execute successfully each SQL statement once. With the fewest number of fixes, indicate all error(s) you encounter? (@6):

- I) ERROR 1008 (HY000): Can't drop database 'cs250exam'; database doesn't exist
- II) ERROR 1046 (3D000): No database selected
- III) ERROR 1146 (42S02): Table 'cs250exam.prof' doesn't exist
- IV) ERROR 1005 (HY000): Can't create table `cs250exam`.`teach` (errno: 150 "Foreign key constraint is incorrectly formed")
- V) ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails
- VI) ERROR 1052 (23000): Column 'pid' in field list is ambiguous
- VII) ERROR 1062 (23000): Duplicate entry 'p1' for key 'PRIMARY'
- VIII) ERROR 1217 (23000): Cannot delete or update a parent row: a foreign key constraint fails

Given the following tables, answer Problems 21-23.

Person		Car			Involve			Accident		
did	name	cid	price	type	did	cid	accid	accid	accyear	dmgamnt
1	David	1	50000	Renault	1	1	1	1	2012	20000
2	Mark	2	4000	Nissan	1	1	2	2	2013	10000
3	Neal	3	6000	Toyota	2	2	3	3	2013	10000
4	Matt	4	10000	Land Rover	4	5	3	4	2014	4000
		5	5000	Land Rover	2	3	4	5	2014	50000
		6	6000	Toyota	3	4	5	6	2015	4500
					4	5	6			
					3	2	3			

**Problem 21.** Which SQL statement returns the following output? (@2)

```
+-----+
|      4500 |
|     10000 |
|     50000 |
+-----+
```

- a) SELECT Avg(dmgamnt) FROM accident WHERE accyear > 2012
- b) SELECT Avg(dmgamnt) FROM accident GROUP BY accyear ORDER BY Avg(dmgamnt)
- c) SELECT Max(dmgamnt) FROM accident WHERE accyear > 2012 GROUP BY accyear ORDER BY Avg(dmgamnt)
- d) SELECT Avg(dmgamnt) FROM accident WHERE accyear > 2012 GROUP BY accyear ORDER BY Avg(dmgamnt) DESC

**Problem 22.** Which of the following SQL statement(s) find the accidents (accid, and its damage cost) where the damage cost is higher than the total price of all cars involved? (@6)

I)	<pre> SELECT accid, dmgamnt FROM   accident A WHERE  dmgamnt &gt; (SELECT Sum(price)                   FROM   car                   WHERE  car.cid IN (SELECT DISTINCT cid                                      FROM   involve                                      WHERE  involve.accid = A.accid)); </pre>
II)	<pre> SELECT A.accid, A.dmgamnt FROM   accident A, car C, involve I WHERE  A.accid = I.accid AND C.cid = I.cid GROUP BY A.accid HAVING A.dmgamnt &gt; Sum(C.price); </pre>
III)	<pre> SELECT accid, dmgamnt FROM   accident A WHERE  dmgamnt &gt; (SELECT Sum(price)                   FROM   involve, car                   WHERE  involve.cid = car.cid                   AND A.accid = involve.accid); </pre>
IV)	<pre> SELECT A.accid, A.dmgamnt FROM   accident A, car C, involve I WHERE  A.accid = I.accid AND C.cid = I.cid GROUP BY A.accid, C.price HAVING A.dmgamnt &gt; C.price; </pre>
V)	<pre> SELECT A.accid, A.dmgamnt FROM   accident A, car C, involve I WHERE  A.accid = I.accid AND c.cid = I.cid GROUP BY A.accid, C.cid HAVING A.dmgamnt &gt; Max(C.price); </pre>
VI)	<pre> SELECT A.accid, A.dmgamnt FROM   accident A, car C, involve I WHERE  C.cid = I.cid GROUP BY A.accid HAVING A.dmgamnt &gt; Min(C.price); </pre>
VII)	<pre> SELECT accid, dmgamnt FROM   accident A WHERE  dmgamnt &gt; All (SELECT price                      FROM   involve, car                      WHERE  involve.cid = car.cid                      AND A.accid = involve.accid); </pre>
VIII)	<pre> SELECT accid, dmgamnt FROM   accident A WHERE  dmgamnt &gt; (SELECT Max(price)                   FROM   car                   WHERE  car.cid IN (SELECT DISTINCT cid                                      FROM   involve                                      WHERE  involve.accid = A.accid)); </pre>

**Problem 23.** What does the following SQL statement do? (@4)

<pre> SELECT c.cid, Count(DISTINCT i.did) FROM   car C, involve I WHERE  C.cid = I.cid GROUP BY C.type HAVING Count(DISTINCT i.did) &gt; 1 </pre>
---------------------------------------------------------------------------------------------------------------------------------------------------

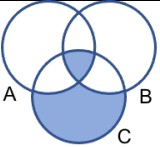
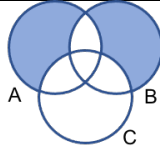
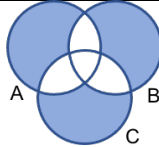
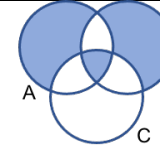
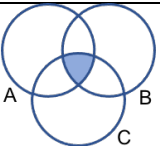
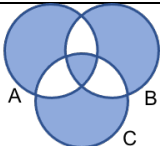
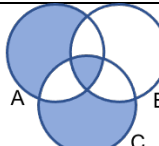
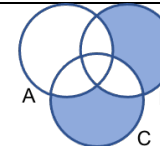
- a) Find the number of different people involved and the id(s) of cars where the total number of different people involved must be great than 1.
- b) Find the number of different people as well as the id(s) of cars, then group the results according to the car's type.
- c) For each car type, find the number of different people and the cars id(s) involved. Only return results where the number of different people involved is more than 1.
- d) For each car type, find i) all the cars (ids) and ii) the different people involved in some accidents. Report only those involving more than 1 person.
- e) None of the answers are correct.

**Problem 24.** Table A, B and C have the same schema  $(x, y)$ . There are eight SQL queries below. If some of these SQL queries express *the same thing*, they form a group. If a SQL expresses totally different thing as others, it form a separate group by itself. There is only one biggest group. Select below all SQL queries in the biggest group. (@6)

I	<pre> SELECT * FROM A WHERE ( x, y ) NOT IN (SELECT * FROM B) AND ( x, y ) NOT IN (SELECT * FROM C) UNION SELECT * FROM B WHERE ( x, y ) NOT IN (SELECT * FROM A) AND ( x, y ) NOT IN (SELECT * FROM C) UNION SELECT * FROM C WHERE ( x, y ) NOT IN (SELECT * FROM A) AND ( x, y ) NOT IN (SELECT * FROM B) </pre>
II	<pre> SELECT * FROM (SELECT * FROM A UNION       SELECT * FROM B UNION       SELECT * FROM C) AS T WHERE ( x, y ) NOT IN (SELECT * FROM A)       AND ( x, y ) NOT IN (SELECT * FROM B)       AND ( x, y ) NOT IN (SELECT * FROM C) UNION (SELECT * FROM A  WHERE ( x, y ) IN (SELECT * FROM B)       AND ( x, y ) IN (SELECT * FROM C)) </pre>
III	<pre> (SELECT * FROM (SELECT * FROM A                 WHERE ( x, y ) NOT IN (SELECT * FROM B)                 UNION                 SELECT * FROM B                 WHERE ( x, y ) NOT IN (SELECT * FROM A)) AS T1  WHERE ( x, y ) NOT IN (SELECT * FROM C)) UNION SELECT C.x, C.y FROM C LEFT OUTER JOIN ( SELECT A.x, A.y   FROM A LEFT OUTER JOIN B ON A.y = B.y AND A.x = B.x   WHERE B.x IS NULL OR B.y IS NULL   UNION   SELECT B.x, B.y   FROM A RIGHT OUTER JOIN B ON A.y = B.y AND A.x = B.x   WHERE A.x IS NULL OR A.y IS NULL ) as T ON C.y = T.y AND C.x = T.x WHERE T.x IS NULL OR T.y IS NULL </pre>
IV	<pre> SELECT * FROM C WHERE ( x, y ) NOT IN (SELECT *                       FROM (SELECT * FROM A UNION SELECT * FROM B) AS T1                       WHERE ( x, y ) NOT IN (SELECT * FROM A                                               WHERE ( x, y ) IN (SELECT * FROM B)))  UNION SELECT * FROM A WHERE ( x, y ) IN (SELECT * FROM B)       AND ( x, y ) NOT IN (SELECT * FROM A UNION SELECT * FROM B)       AND ( x, y ) NOT IN (SELECT * FROM C) </pre>
V	<pre> (SELECT * FROM (SELECT * FROM C                 WHERE ( x, y ) NOT IN (SELECT * FROM A)                 UNION                 SELECT * FROM A                 WHERE ( x, y ) NOT IN (SELECT * FROM C)) AS T1  WHERE ( x, y ) NOT IN (SELECT * FROM B)) UNION </pre>

	<pre> (SELECT * FROM (SELECT * FROM B                   WHERE ( x, y ) NOT IN (SELECT * FROM C)                   UNION                   SELECT * FROM C                   WHERE ( x, y ) NOT IN (SELECT * FROM B)) AS T2 WHERE ( x, y ) NOT IN (SELECT * FROM A)) UNION (SELECT * FROM (SELECT * FROM A                   WHERE ( x, y ) NOT IN (SELECT * FROM B)                   UNION                   SELECT * FROM B                   WHERE ( x, y ) NOT IN (SELECT * FROM A)) AS T3 WHERE ( x, y ) NOT IN (SELECT * FROM C)) </pre>
VI	<pre> SELECT * FROM (SELECT *       FROM (SELECT * FROM A UNION SELECT * FROM B) AS T1       WHERE ( x, y ) NOT IN (SELECT * FROM A                              WHERE ( x, y ) IN (SELECT * FROM B))) AS T2 WHERE ( x, y ) NOT IN (SELECT * FROM C) UNION SELECT * FROM C WHERE ( x, y ) NOT IN (SELECT *                       FROM (SELECT * FROM A UNION SELECT * FROM B) AS T1                       WHERE ( x, y ) NOT IN (SELECT *                                              FROM A                                              WHERE ( x, y ) IN (SELECT * FROM B))) </pre>
VII	<pre> SELECT * FROM A WHERE ( x, y ) NOT IN (SELECT * FROM B) AND ( x, y ) NOT IN (SELECT * FROM C) UNION SELECT * FROM B WHERE ( x, y ) NOT IN (SELECT * FROM C) AND ( x, y ) NOT IN (SELECT * FROM A) UNION SELECT * FROM C WHERE ( x, y ) NOT IN (SELECT * FROM A) AND ( x, y ) NOT IN (SELECT * FROM B) INTERSECT SELECT * FROM A WHERE ( x, y ) IN (SELECT * FROM B) AND ( x, y ) IN (SELECT * FROM C) </pre>
VIII	<pre> SELECT * FROM (SELECT *       FROM (SELECT *             FROM (SELECT * FROM A UNION SELECT * FROM B) AS T1             WHERE (x,y) NOT IN (SELECT * FROM A                                WHERE (x,y) IN (SELECT * FROM B))) AS T2              UNION             SELECT * FROM C) AS T1 WHERE (x,y) NOT IN (SELECT *                   FROM (SELECT *                         FROM (SELECT * FROM A UNION SELECT * FROM B) AS T1                         WHERE (x,y) NOT IN (SELECT * FROM A                                            WHERE (x,y) IN (SELECT * FROM B))) AS T2                   WHERE (x,y) IN (SELECT * FROM C)) </pre>

**Problem 25.** What do these SQL queries (in the biggest group) in **Problem 24** look for? (@4)

a)		b)		c)		d)	
e)		f)		g)		h)	

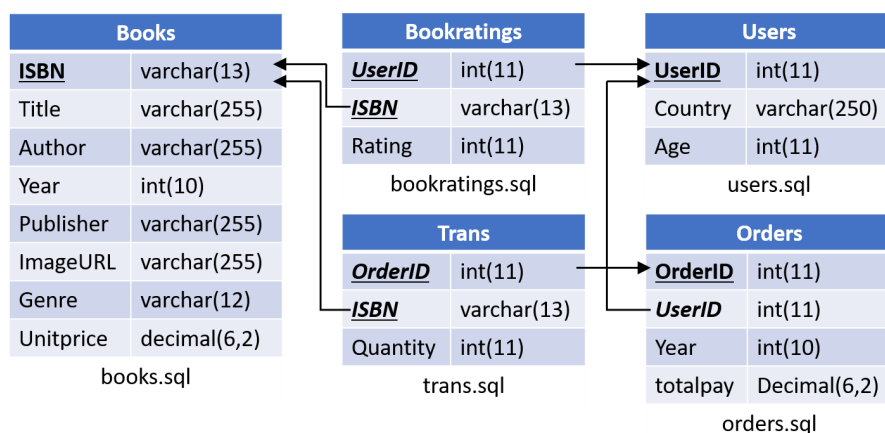
---- End of Part A ----

## B. SQL+PHP (worth 10% of CS-250)



There are two tasks in this part B. You will use SQL knowledge to help Amazon (management team) to answer some simple questions regarding their book sales and ratings database, and to help generate an user order history page.

This part gives you practical opportunities to write SQL queries and full marks can only be obtained if you demonstrate clearly that you have tested the queries on an actual implementation of the database. All test data is randomly generated, and all data is fake - they are for testing purpose only. Data is provided to you in the form of sql files, as explained below:



### Table descriptions

**Books:** Book's details

**Users:** User's details

**Bookratings:** Users' ratings on books

**Orders:** Users' orders on books

**Trans:** Orders' transaction details

All attributes are self-explanatory.

Highlighted attributes are primary keys. Arrows indicates foreign keys and references. No table allows NULL values.

Use DDL SQL statements to implement these tables in MySQL and populate your database with data from the given files (books.sql, bookratings.sql, trans.sql, orders.sql, users.sql) available on Canvas. If you have trouble creating the database / tables, or experience issues when importing data into your tables, please study & redo the lab materials.

### Task 1 (SQL) (@60):

You are required to give one single SQL statement to answer each of the questions. Present your answer concisely:

- Show the SQL statement(s) you used and
- The results obtained from your database (e.g., copy results or screenshots from console / PHPPMyAdmin).

#### Important Notes:

- you **must** use a SQL formatter to tidy up your SQL statements (<http://www.dpriver.com/pp/sqlformat.htm>)
- you **must** observe the font size limit. This applies to all screenshots, figures and tables. Otherwise, marks will be deducted for poor presentation, or no marks will be given if it is incomprehensible.
- You must **NOT** use **natural join SQL keyword** in all the following questions. Otherwise, **NO** marks will be given.
- You should further observe what is required in each of the questions below.

**Question 1.** A **Refund** is a net positive amount between the amount a customer pays and the actual cost of all items s/he ordered. If there is a refund (that is, a customer paid more than s/he should), find the orderid, the customer's userid and the refund. Display your results in decreasing amount of refund values. You should show a table that looks like below:

```
+-----+-----+
| orderid | userid | refund |
+-----+-----+
| ..... | ..... | ..... |
```

Your SQL statement should contain at most one **FROM** keyword. Otherwise, **NO** marks will be given.

**Question 2.** A **booklover** is defined as a user who rates more than 100 books. Define “**AvgRating**” as the average book rating a user gives to all books s/he rates. Find the userid and AvgRating of the booklover(s) who give the highest AvgRating. If there are multiple such booklovers, your query should display them all. You should show a table that looks like below:

```
+-----+-----+
| userid | AvgRating |
+-----+-----+
| .... | ..... |
```

You are not allowed to use **WITH**, **IN**, **MIN**, **MAX**, **ALL**, **ANY**, **SOME** or **LIMIT** keywords to answer this question. Otherwise, **NO** marks will be given.



## Task 2 (SQL + PHP) (@40):

Your final task is to generate an “order history” page that looks as close as possible to the ones below using PHP, *given an userid* (to be submitted from a form textbox). See the two figures below.

- 1) The order history should show the user’s details (userid, country) and order summary (number of orders, books).
- 2) For each order, it should show the year it is placed, the total cost paid, discount/refund (*if there is any*) and the orderid.
- 3) For each order, it should list all book details, including title, ISBN, unitprice, genre, an image of the book (imageURL), and the average rating of the book if it is ever rated by *anyone* (in terms of number of stars).
- 4) If there are two or more copies of the same book in an order, the page should show the quantity (e.g. “2 of” / “3 of”) and their total cost.
- 5) All orders should be listed in descending order of the year placed.
- 6) All books in each order should be listed in ascending alphabetical order of their genre.
- 7) If two orders are placed in the same year, the order of them does not matter.
- 8) If two books share the same genre in an order, the order of them does not matter.

To show that you have done this part, you should submit (in the pdf) the following:

- 1) Syntax highlighted PHP code that you wrote (only one php file, called amazon.php) inside the pdf (within page limit).
- 2) Screenshots of *three* order histories with different *userid*s.

- *All userid*s must be users who made *more than 1 order and bought more than 3 books* in history.

- One of the *userid*s must be a user who got a *discount/refund* in one of his orders.

**Discount** – customer paid less than s/he should.

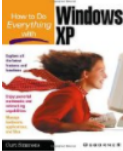



**Refund** – customer paid more than s/he should.

*Note: your SQL must use the set of tables that was created in Task1.*

**Strong Tips:** Don’t waste time on fancy CSS styles. **Nested HTML tables** are enough. Modify from the given PHP examples.

amazon.co.uk  
User Details:  
Userid: 164288  
Country: USA

Orders Summary:  
2 orders placed  
4 books bought


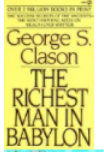


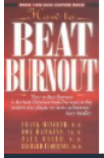
Order placed	Total	Discount	Order ID
2003	£2.00	£9.36	2180
 How to Do Everything with Windows XP 007219300X £11.36 ***** General £11.36			
2000	£64.80		2202
 The Magical Bicycle 0749709391 £16.87 Romance £16.87			
 Feuilles d'herbe 2246404819 £21.12 Romance £21.12			
 Shadow Man (danger.com) 068981433X £26.81 Science £26.81			

amazon  
Australia Brazil Canada China France Germany India Italy Japan Mexico Netherlands Spain United States

Query Userid:

amazon.co.uk  
User Details:  
Userid: 115689  
Country: USA

Orders Summary:  
2 orders placed  
7 books bought

Order placed	Total	Order ID
2000	£78.33	1011
 2 of Head-On/Repossessed 0722538820 £20.65 Adventure £41.30		
 The Richest Man in Babylon 0451165209 £13.91 ***** Fiction £13.91		
 The Silly Gooses 059095735X £13.18 Romance £13.18		
 2 of Diana: The Last Year 0609603183 £4.97 Romance £9.94		
2000	£33.81	1582
 How to Beat Burnout: Help for Men and Women 0802423140 £33.81 Mystery £33.81		

amazon  
Australia Brazil Canada China France Germany India Italy Japan Mexico Netherlands Spain United States

Query Userid:

---- End of CW2 ----