Monash University

FIT2094 - Databases

MOCK SCHEDULED FINAL ASSESSMENT/EXAM

SAMPLE SOLUTIONS

Author: FIT Database Teaching Team

PART A Relational Model [Total: 10 Marks]

Q1 [3 Marks]
A company wishes to record the following attributes about their employees: employee ID, department number, name, home address, education qualifications and skills which the employee has.

A small sample of data is show below:

Employee ID	**Department Number**	**Employee Name**	**Home Address**	**Qualification**	**Skill**
101	21	Given name: Joe Family name: Bloggs	Street: 12 Wide Rd Town: Mytown Postcode: 1234	Bachelor of Commerce MBA	Project Management Hadoop R
102	13	Given name: Wendy Family name: Xiu	Street: 55 Narrow St Town: Mytown Postcode: 1234	Bachelor of Computer Science Master of IT Doctor of Philosophy	SQL PL/SQL
103	13	Given name: Sarah Family name: Green	Street: 25 High St Rd Town: Mytown Postcode: 1234	Certificate IV in Business Administration	SQL Java Phyton

Use this data to explain the difference between a simple attribute, a composite attribute and a multivalued attribute. Your answer must include examples drawn from this data.

Simple - an attribute which cannot be subdivided eg. employeeid, department number

Composite - an attribute which can be subdivided into additional attributes eg. employee name, home address

Multivalued - an attribute which has many potential values eg. qualification, skill

Q2 [7 Marks]

The following relations represent a publications database:

AUTHOR (author_id, author_firstname, author_lastname)

AUTHOR_PAPER (author_id, paper_id, authorpaper_position)

PAPER (paper_id, paper_title, journal_id)

JOURNAL (journal_id, journal_title, journal_month, journal_year, journal_editor)

* editor in journal references author(author_id) – this is an author acting as the journal editor

Authors write papers which are published in an edition of a journal. Each edition of a journal is assigned a journal id and appoints an editor. A given paper may be authored by several authors, in such cases each author is assigned a position representing their contribution to the paper:

Write the relational algebra for the following queries (your answer must show an understanding of query efficiency):

List of symbols:
project: π, **select**: σ, **join**: ⋈, **left outer join** ⟕, **right outer join** ⟖, **full outer join** ⟗, **intersect** ∩, **union** ∪, **minus** -

(a) Show the paper title, journal title, and month and year of journal publication for all papers published before 2012. (3 marks)

$$R1 = \pi_{\text{journal\_id, journal\_title, journal\_month, journal\_year}}\ (\sigma_{\text{journal\_year} < 2012}\ (\text{JOURNAL}))$$

$$R2 = \pi_{\text{journal\_id, paper\_title}}(\text{PAPER})$$

$$R3 = R1 \bowtie R2$$

$$R = \pi_{\text{paper\_title, journal\_title, journal\_month, journal\_year}}\ (R3)$$

Here R1 could be done in two steps, a select and then a project.

OR

$$\pi_{\text{paper\_title, journal\_title, journal\_month, journal\_ year}}$$

$$($$

$$(\pi_{\text{journal\_id, journal\_title, journal\_month, journal\_year}}\ (\sigma_{\text{journal\_year} < 2012}\ (\text{JOURNAL}))$$

$$\bowtie$$

$$(\pi_{\text{journal\_id, paper\_title}}(\text{PAPER}))$$

$$)$$

(b) Show the names of all authors who have never been listed as first author (authorpaper_position = 1) in any paper. (4 marks)

$R1 = \pi_{\text{author\_id}} (\sigma_{\text{authorpaper\_position} = 1} (\text{AUTHOR\_PAPER}))$

$R2 = \text{AUTHOR} \bowtie R1$

$R3 = \pi_{\text{author\_firstname, author\_lastname}}(R2)$

$R4 = \pi_{\text{author\_firstname, author\_lastname}}(\text{AUTHOR}) - R3$

OR

$$
\begin{array}{l}
\pi_{\text{author\_firstname, author\_lastname}}(\text{AUTHOR}) - (\\
\quad \pi_{\text{author\_firstname, author\_lastname}}(\\
\quad\quad \text{AUTHOR} \\
\quad\quad \bowtie \\
\quad\quad (\pi_{\text{author\_id}} (\sigma_{\text{authorpaper\_position} = 1} (\text{AUTHOR\_PAPER}))) \\
\quad) \\
)
\end{array}
$$

PART B Database Design [Total: 20 Marks]

Q3 [20 marks]

Monash Computing Students Society (MCSS) is one of the student clubs at Monash University.

Students are welcome to join as a member. When a student joins MCSS, a member id is assigned, and the students first name, last name, date of birth, email and phone number will be recorded. This club has an annual membership fee. When a member has paid the membership fee for the current year, the current year is recorded against the year of membership as part of their membership details.

MCSS hosts several events throughout the year. The events are currently categorised into *Professional Events*, *General Events*, and *Social Events*. MCSS would like to be able to add further categories as they develop new events. When an event is scheduled, MCSS assigns an event id to the event. The event date and time, description, location, allocated budget, the ticket price and the discount rate (eg 5%) for members. Some events are organised as free events for members. In this situation, the discount rate is recorded as 100% for members. For all events, only members can purchase the tickets. However, members can buy additional tickets for their friends or family at full price. For each of the sales, the receipt number, number of tickets sold, total amount paid and the member id are recorded.

Some events attract some sponsorships. The sponsor may be an organisation or an individual. The sponsors provide financial support to the event. Some events may have several sponsors. The amount of financial support provided by each sponsor is recorded for the event. Each sponsor is identified by a sponsor id. The name, contact email and sponsor type are also recorded. A sponsor may support several events throughout the year.

For some events such as career night, MCSS may also invite some guest speakers to share their experience. The database records all guests' information, the guests full name, email and phone number are recorded. If a guest comes from an organisation or an individual that provides a sponsorship to any of the MCSS events (does not have to be at the event where the guest speaks), this fact will also be recorded. A guest may be invited to several events.

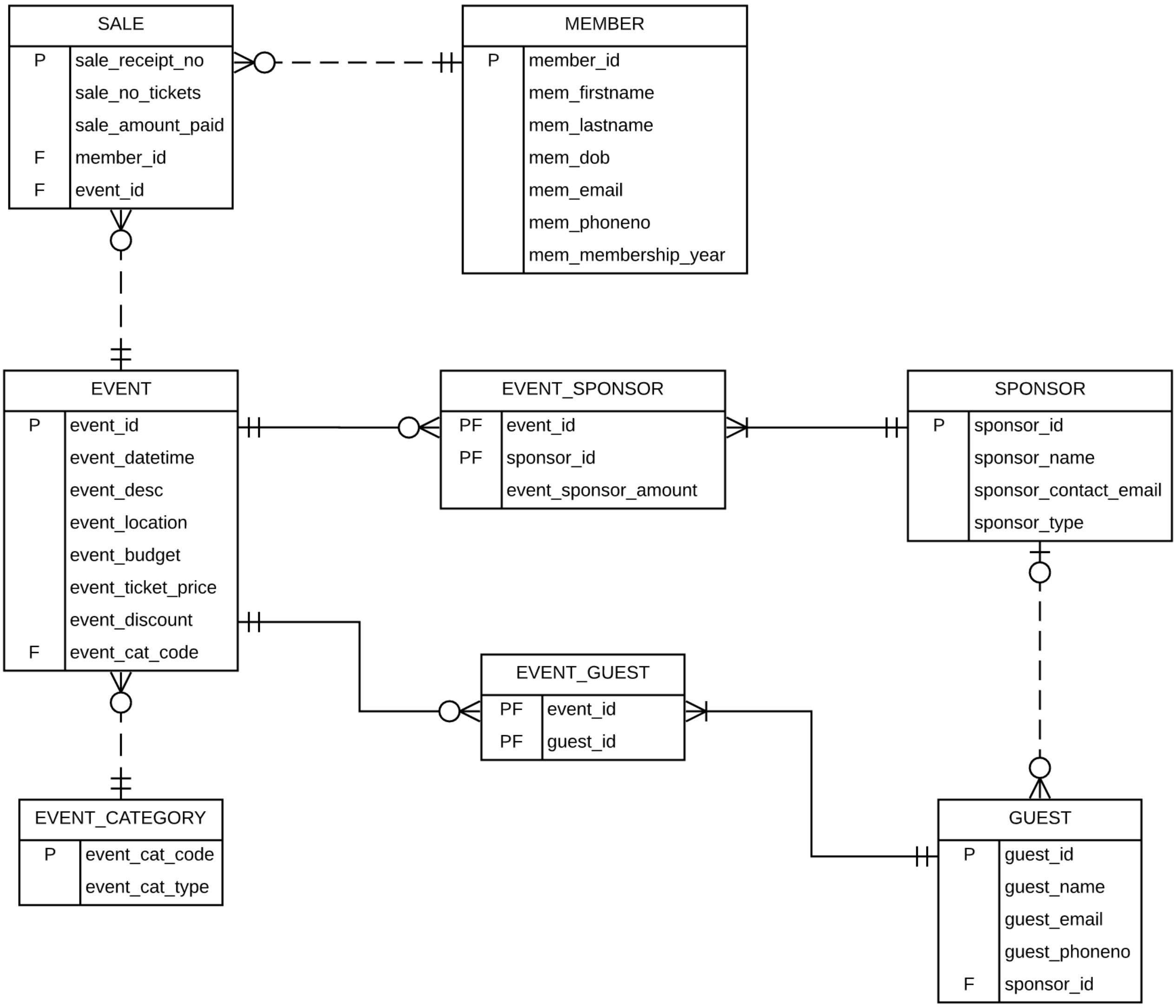
Create ***a logical level diagram using Crow's foot notations*** to represent the "Monash Computing Students Society" data requirements described above. Clearly state any assumptions you make when creating the model.

Please note the following points:

- Be sure to include all relations, attributes and relationships (unnecessary relationships must not be included)
- Identify clearly the Primary Keys (P) and Foreign Keys (F), as part of your design
- In building your model you must conform to FIT2094 modelling requirements
- The following are **NOT required** on your diagram
  - verbs/names on relationship lines
  - indicators (*) to show if an attribute is required or not
  - data types for the attributes

NOTE: This question has been designed such that the model will fit on a single A4 page. You are allowed to use two blank worksheets to draft your model and then submit your final response on ONE page.

Monash Computing Students Society (MCSS) Logical Model



SALE	
P	sale_receipt_no
	sale_no_tickets
	sale_amount_paid
F	member_id
F	event_id

MEMBER	
P	member_id
	mem_firstname
	mem_lastname
	mem_dob
	mem_email
	mem_phoneno
	mem_membership_year

EVENT	
P	event_id
	event_datetime
	event_desc
	event_location
	event_budget
	event_ticket_price
	event_discount
F	event_cat_code

EVENT_SPONSOR	
PF	event_id
PF	sponsor_id
	event_sponsor_amount

SPONSOR	
P	sponsor_id
	sponsor_name
	sponsor_contact_email
	sponsor_type

EVENT_GUEST	
PF	event_id
PF	guest_id

EVENT_CATEGORY	
P	event_cat_code
	event_cat_type

GUEST	
P	guest_id
	guest_name
	guest_email
	guest_phoneno
F	sponsor_id

PART C Normalisation [Total: 10 Marks]

Q3 [10 marks]

The Super Electronics Invoice shown below displays the details of an invoice for the client Alice Paul.

Super Electronics
INVOICE

Client Number:	C3178713	**Invoice No.:**	132
Client Name:	Alice Paul	**Invoice Date:**	02/11/2018
Client Address:	43 High Street, Caulfield, VIC 3162		
Client Phone:	0411 245 718		

ItemID	Item Name	Purchase Price	Expected Delivery Date	Quantity	Cost
316772	Soniq S55UV16B 55"	499.00	2 weeks	1	499.00
452550	Microsoft Surface Pro	1198.00	1-3 weeks	1	1198.00
483041	Delonghi Digital Coffee	299.00	Same Day	2	598.00
				SUB TOTAL:	**$ 2295.00**
				DELIVERY:	**$145.00**
				ORDER TOTAL:	**$2440.00**

Represent this form in UNF. In creating your representation you should note that Super Electronics wish to treat the client name and address as simple attributes. Convert your UNF to first normal form (1NF) and then continue the normalisation to third normal form (3NF). At each normal form show the appropriate dependencies for that normal form, if there are none write "No Dependencies"

Do not add new attributes during the normalisation. Clearly write the relations in each step from the unnormalised form (UNF) to the third normal form (3NF). Clearly, indicate primary keys on all relations from 1NF onwards.

[**10 marks**]

UNF

INVOICE (inv_nbr, inv_date, client_number, client_name, client_address, client_phone, (item_id, item_name, invline_purchaseprice, invline_deliverytime, invline_qtyordered, invline_linecost), inv_subtotal, inv_deliveryfee, inv_ordertotal)

1NF

INVOICE (inv_nbr, inv_date, client_number, client_name, client_address, client_phone, inv_subtotal, inv_deliveryfee, inv_ordertotal)

INVOICE_LINE (inv_nbr, item_id, item_name, invline_purchaseprice, invline_deliverytime, invline_qtyordered, invline_linecost)

Partial Dependencies:

item_id -> item_name

2NF

INVOICE (inv_nbr, inv_date, client_number, client_name, client_address, client_phone, inv_subtotal, inv_deliveryfee, inv_ordertotal)

INVOICE_LINE (inv_nbr, item_id, invline_purchaseprice, invline_deliverytime, invline_qtyordered, invline_linecost)

ITEM (item_id, item_name)

Transitive Dependencies:

client_number -> client_name, client_address, client_phone

3NF

INVOICE (inv_nbr, inv_date, client_number, inv_subtotal, inv_deliveryfee, inv_ordertotal)

CLIENT (client_number, client_name, client_address, client_phone)

INVOICE_LINE (inv_nbr, item_id, invline_purchaseprice, invline_deliverytime, invline_qtyordered, invline_linecost)

ITEM (item_id, item_name)

Full Dependencies:

inv_nbr -> inv_date, client_number, inv_subtotal, inv_deliveryfee, inv_ordertotal

client_number -> client_name, client_address, client_phone

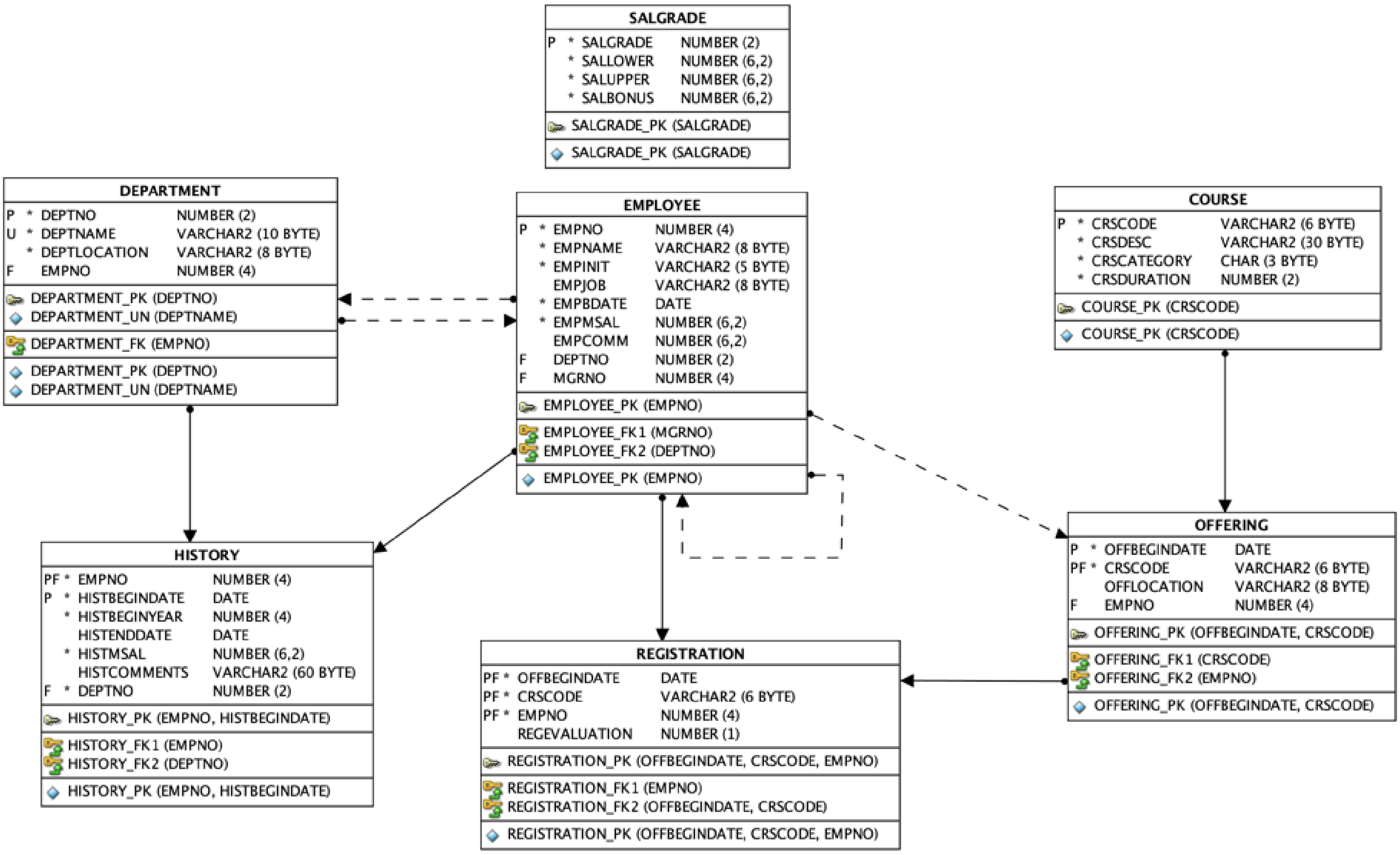
inv_nbr, item_id -> invline_purchaseprice, invline_deliverytime, invline_qtyordered, invline_linecost

item_id -> item_name

PART D SQL [Total: 50 Marks]

Employee System Model and Schema File for Part D

The following relational model depicts an employee system:



The schema file to create these tables is:

```
CREATE TABLE SALGRADE (
  salgrade NUMBER(2)   NOT NULL ,
  sallower NUMBER(6,2)   NOT NULL ,
  salupper NUMBER(6,2)   NOT NULL ,
  salbonus NUMBER(6,2)   NOT NULL   ,
CONSTRAINT salgrade_pk PRIMARY KEY (salgrade),
CONSTRAINT salgrade_chk1 CHECK (sallower >= 0),
CONSTRAINT salgrade_chk2 CHECK (sallower <= salupper));

COMMENT ON COLUMN salgrade.salgrade IS 'Salary Grade';
COMMENT ON COLUMN salgrade.sallower IS 'Salary Lower Limit';
COMMENT ON COLUMN salgrade.salupper IS 'Salary Upper Limit';
COMMENT ON COLUMN salgrade.salbonus IS 'Salary Bonus';
```

```
CREATE TABLE course (
  crscode VARCHAR(6)   NOT NULL ,
  crsdesc VARCHAR(30)   NOT NULL ,
  crscategory CHAR(3)   NOT NULL ,
  crsduration NUMBER(2)   NOT NULL   ,
CONSTRAINT course_pk PRIMARY KEY (crscode),
CONSTRAINT course_chk1 CHECK (crscode = upper(crscode)),
CONSTRAINT course_chk2 CHECK (crscategory in ('GEN','BLD','DSG')));

COMMENT ON COLUMN course.crscode     IS 'Course Code';
COMMENT ON COLUMN course.crsdesc     IS 'Course Description';
COMMENT ON COLUMN course.crscategory IS 'Course Category';
COMMENT ON COLUMN course.crsduration IS 'Course Duration';

CREATE TABLE DEPARTMENT (
  deptno NUMBER(2)   NOT NULL ,
  deptname VARCHAR(10)   NOT NULL ,
  deptlocation VARCHAR(8)   NOT NULL ,
  empno NUMBER(4) ,
CONSTRAINT department_pk PRIMARY KEY (deptno),
CONSTRAINT department_un UNIQUE (deptname),
CONSTRAINT department_chk1 CHECK (deptname = upper(deptname)),
CONSTRAINT department_chk2 CHECK (deptlocation = upper(deptlocation)));

COMMENT ON COLUMN department.deptno       IS 'Department Number';
COMMENT ON COLUMN department.deptname     IS 'Department Name';
COMMENT ON COLUMN department.deptlocation IS 'Location of department';
COMMENT ON COLUMN department.empno        IS 'Employee who manages
department';

CREATE TABLE EMPLOYEE (
  empno NUMBER(4)   NOT NULL ,
  empname VARCHAR(8)   NOT NULL ,
  empinit VARCHAR(5)   NOT NULL ,
  empjob VARCHAR(8)    ,
  empbdate DATE   NOT NULL ,
  empmsal NUMBER(6,2)   NOT NULL ,
  empcomm NUMBER(6,2) ,
  deptno NUMBER(2) ,
  mgrno NUMBER(4) ,
CONSTRAINT employee_pk PRIMARY KEY (empno),
CONSTRAINT employee_fk1  FOREIGN KEY (mgrno)
    REFERENCES EMPLOYEE (empno),
CONSTRAINT employee_fk2 FOREIGN KEY (deptno)
    REFERENCES DEPARTMENT (deptno));

COMMENT ON COLUMN employee.empno     IS 'Employee number';
COMMENT ON COLUMN employee.empname   IS 'Employee name';
COMMENT ON COLUMN employee.empinit   IS 'Employee initials';
COMMENT ON COLUMN employee.empjob    IS 'Employee job';
COMMENT ON COLUMN employee.empbdate  IS 'Employee birthdate';
COMMENT ON COLUMN employee.empmsal   IS 'Employee monthly salary';
COMMENT ON COLUMN employee.empcomm   IS 'Employee commission';
COMMENT ON COLUMN employee.deptno    IS 'Department Number';
COMMENT ON COLUMN employee.mgrno     IS 'Employees manager (empno of
manager)';

ALTER TABLE DEPARTMENT
ADD (CONSTRAINT department_fk FOREIGN KEY (empno)
           REFERENCES employee (empno));

CREATE TABLE HISTORY (
  empno NUMBER(4)   NOT NULL ,
  histbegindate DATE   NOT NULL ,
  histbeginyear NUMBER(4)   NOT NULL ,
  histenddate DATE    ,
  histmsal NUMBER(6,2)   NOT NULL ,
  histcomments VARCHAR(60)    ,
  deptno NUMBER(2)   NOT NULL   ,
CONSTRAINT history_pk PRIMARY KEY (empno, histbegindate),
CONSTRAINT history_chk CHECK (histbegindate < histenddate),
CONSTRAINT history_fk1 FOREIGN KEY (empno)
    REFERENCES EMPLOYEE (empno)
     ON DELETE CASCADE,
CONSTRAINT history_fk2 FOREIGN KEY (deptno)
    REFERENCES DEPARTMENT (deptno));

COMMENT ON COLUMN history.deptno        IS 'Department Number';
COMMENT ON COLUMN history.histbegindate IS 'Date history record begins';
COMMENT ON COLUMN history.histbeginyear IS 'Year history record begins';
COMMENT ON COLUMN history.histenddate   IS 'Date history record ends';
COMMENT ON COLUMN history.histmsal      IS 'Monthly Salary for this
history record';
COMMENT ON COLUMN history.histcomments  IS 'Comments for this history
record';
COMMENT ON COLUMN history.empno         IS 'Employee number';

CREATE TABLE OFFERING (
  offbegindate DATE   NOT NULL ,
  crscode VARCHAR(6)   NOT NULL ,
  offlocation VARCHAR(8) ,
  empno NUMBER(4) ,
CONSTRAINT offering_pk PRIMARY KEY (offbegindate, crscode),
CONSTRAINT offering_fk1 FOREIGN KEY (crscode)
    REFERENCES course(crscode),
CONSTRAINT offering_fk2 FOREIGN KEY (empno)
    REFERENCES EMPLOYEE (empno));

COMMENT ON COLUMN offering.offbegindate IS 'Begin date for offering';
COMMENT ON COLUMN offering.crscode      IS 'Course Code';
COMMENT ON COLUMN offering.offlocation  IS 'Location for offering';
COMMENT ON COLUMN offering.empno        IS 'Employee number for employee
running offering';
```

```
CREATE TABLE REGISTRATION (
  offbegindate DATE   NOT NULL ,
  crscode VARCHAR(6)   NOT NULL ,
  empno NUMBER(4) NOT NULL,
  regevaluation NUMBER(1) ,
CONSTRAINT registration_pk PRIMARY KEY (offbegindate, crscode, empno),
CONSTRAINT resgitration_chk CHECK (regevaluation in (1,2,3,4,5)),
CONSTRAINT registration_fk1 FOREIGN KEY (empno)
    REFERENCES EMPLOYEE (empno),
CONSTRAINT registration_fk2 FOREIGN KEY (offbegindate, crscode)
    REFERENCES OFFERING (offbegindate, crscode));

COMMENT ON COLUMN registration.offbegindate  IS 'Begin date for offering';
COMMENT ON COLUMN registration.crscode       IS 'Course Code';
COMMENT ON COLUMN registration.regevaluation IS 'Grade for course
completed';
COMMENT ON COLUMN registration.empno         IS 'Employee number of
employee completing course';
```

Note in coding your SQL each SELECT, FROM, WHERE, GROUP BY, HAVING and ORDER BY clause **must start on a new line**.

Q5 [10 marks]

The company needs to record a new department. This new department's number will be 10 higher than the highest current department number and will be called EXAM and is located in BOSTON. The employee named KING who has a job as the only company DIRECTOR has been assigned to manage the new EXAM department.

The company has also decided that they wish to record, for each department, the number of employees currently working in the department (the employee count). For new departments the number of employees in the department should be set to 0. For those departments which currently have employees, the employee count should correctly reflect the current number of employees in the department.

Code the SQL statements to modify the database to meet these requirements.

```
INSERT INTO department VALUES (
    (
        SELECT
            MAX(deptno)
        FROM
            department
    ) + 10,
    'EXAM',
    'BOSTON',
    (
        SELECT
            empno
        FROM
            employee
        WHERE
            empname = 'KING'
            AND empjob = 'DIRECTOR'
    )
);

COMMIT;

ALTER TABLE department ADD deptcount NUMBER(3, 0) DEFAULT 0 NOT NULL;

UPDATE department d
SET
    deptcount = (
        SELECT
            COUNT(empno)
        FROM
            employee e
        WHERE
            e.deptno = d.deptno
    );

COMMIT;
```

Q6 [6 marks]

List the employee number, the employee name, the employee job and the yearly salary of all employees that belong to the ‘Sales’ department. The name of the employee must be shown in a column called “Employee Name” and the yearly salary must be shown in the form of $34,200 in a column called “Yearly Salary”. Show the employee with the highest salary first, if two employees have the same salary, order it based on the employee number.
Code the SQL SELECT statement.

```
SELECT
    e.empno,
    empname                             AS "Employee Name",
    empjob,
    to_char(empmsal * 12, '$99,990')        AS "Yearly Salary"
FROM
         employee e
    JOIN department d
    ON e.deptno = d.deptno
WHERE
    upper(deptname) = upper('Sales')
ORDER BY
    empmsal DESC,
    e.empno;
```

Q7 [9 marks]
For each course which has been completed by at least 5 employees, list the course code, the course description and the course duration. The course duration must be shown in a column called “Course Duration” and include the word ‘days’ (e.g. 4 days). Order the output by the course code.
Code the SQL SELECT statement.

```
SELECT
    c.crscode,
    crsdesc,
    crsduration || ' days' as "Course Duration"
FROM
         registration r
    JOIN course c
    ON r.crscode = c.crscode
WHERE
    r.regevaluation IS NOT NULL
GROUP BY
    c.crscode,
    crsdesc,
    crsduration
HAVING
    COUNT(empno) >= 5
ORDER BY
    c.crscode;
```

Q8 [15 marks]

List ALL employees whose total course registrations are less than the average number of registrations for employees who have registered for a course. Note that some employees may repeat a course, this repeat does not count as a different course. In the list, include the employee number, name, date of birth and the number of different courses they have registered for. Order the output by employee number.
Code the SQL SELECT statement.

```
SELECT
    e.empno,
    empname,
    to_char(empbdate, 'dd-Mon-yyyy') AS dob,
    COUNT(DISTINCT crscode) AS crscount
FROM
    employee       e
    LEFT JOIN registration   r ON e.empno = r.empno
GROUP BY
    e.empno,
    empname,
    to_char(empbdate, 'dd-Mon-yyyy')
HAVING
    COUNT(DISTINCT crscode) < (
        SELECT
            AVG(COUNT(DISTINCT crscode))
        FROM
            registration
        GROUP BY
            empno
    )
ORDER BY
    e.empno;
```

Q9 [10 marks]

We wish to develop a php based web page which shows all departments, and their manager's name and monthly salary as shown below:

Here is the incomplete PHP code for the page:

```
<table border =1 width=1000>
      <tr>
        <th width=200><b>Department Number</b></th>
        <th width=200><b>Department Name</b></th>
        <th width=200><b>Department Location</b></th>
        <th width=200><b>Manager</b></th>
        <th width=200><b>Manager's Monthly Salary</b></th>
      </tr>

    <?php
     $query = complete this part!;
      $stmt = oci_parse($conn,$query);
      if (!$stmt) {
        $e = oci_error($conn);
        print "Error on parse of statement:<br>" ;
        print $e['message'] ;
        exit;
      }

      oci_define_by_name($stmt,"DNO",$dno);
      oci_define_by_name($stmt,"DNAME",$dname);
      oci_define_by_name($stmt,"DLOC",$dloc);
      oci_define_by_name($stmt,"MGRNAME",$mgrname);
      oci_define_by_name($stmt,"MGRMSAL",$mgrmsal);

      $r = oci_execute($stmt);
      if (!$r) {
        $e = oci_error($stmt);
        print "Error execute of statement:<br>" ;
        print $e['message'] ;
        exit;
      }

      while (oci_fetch($stmt)) {
        print("
        <tr>
          <td width=200>$dno</td>
          <td width=200>$dname</td>
          <td width=200>$dloc</td>
          <td width=200>$mgrname</td>
          <td width=200>$mgrmsal</td>
        </tr>");
      }

      print ("</table>");
```

Write the missing $query statement for the PHP code above in the answer space below:

```
$query = "SELECT
            d.deptno                          AS dno,
            deptname                          AS dname,
            deptlocation                      AS dloc,
            empinit
            || ' '
            || empname                         AS mgrname,
            to_char(empmsal, '$9990.99')       AS mgrmsal
         FROM
            department d
            JOIN employee e
                        ON e.empno = d.empno
         ORDER BY
            dno"
```

PART F Transaction [Total: 10 Marks]

Q10. [5 marks]

Given two transactions:

T1 – R(X), W(X)

T2 – R(Y), W(Y), R(X), W (X)

Where R(X) means Read(X) and W(X) means Write(X).

(a) If we wish to complete both of these transactions, explain the difference between a *serial* and *non-serial* ordering of these two transactions. Provide an example of each as part of your answer.

(b) What transaction ACID property does a non-serial ordering of these two transactions potentially violate.

(a)

Serial – all of one transaction followed by all of the other

T1 R(X), T1 W(X), T2 R(Y), T2 W(Y), T2 R(X), T2 W(X)

Non-Serial – interleaving of the transactions

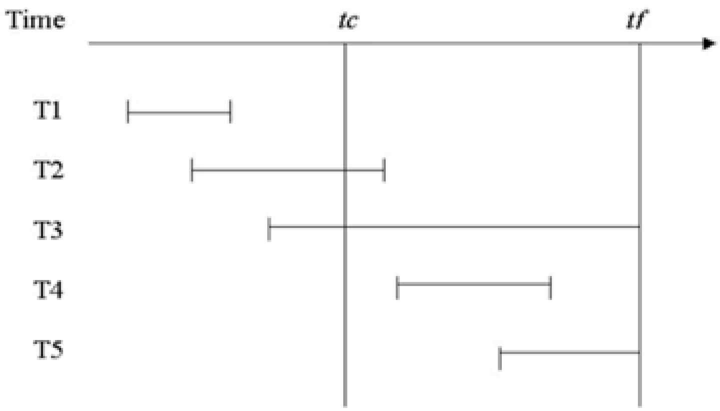
T1 R(X), T2 R(Y), T2 W(Y), T1 W(X), T2 R(X), T2 W(X)

(b)

Isolation or Consistency

Q11 [5 marks]

A *write through* database has five transactions running as listed below (the time is shown horizontally from left to right):



At time *tc* a checkpoint is taken, at time *tf* the database fails due to a power outage.

Explain for each transaction what recovery operations will be needed when the database is restarted and why.

T1 – nothing required, committed before checkpoint

T2 – ROLL FORWARD, committed after checkpoint and before fail

T3 – ROLL BACK, never reached commit

T4 – ROLL FORWARD, started after checkpoint committed before fail

T5 - ROLL BACK, never reached commit