SCENARIO

Run Monash (RM) is a running carnival which is held on separate dates at various Monash campuses during different seasons (Summer, Autumn, Winter and Spring) of the year. The carnival naming convention that Run Monash uses is RM <season name> Series <campus name> <year>. So, for example, a carnival to be held during the Autumn season at the Clayton campus in 2024 will be named RM Autumn Series Clayton 2024.

Anyone can attend an RM Carnival; the carnivals are open to the public as well as Monash staff and students. A carnival is run on a particular date, in a particular location, and only lasts for one day. RM only runs one carnival on any particular date. During a carnival, a range of event types are offered from the following list (only some may be offered):

- Marathon 42.2 Km
- Half Marathon 21.1 Km
- 10 Km Run
- 5 Km Run
- 3 Km Community Run/Walk

Run Monash expects to offer around 15 - 30 such events across all carnivals in a given year.

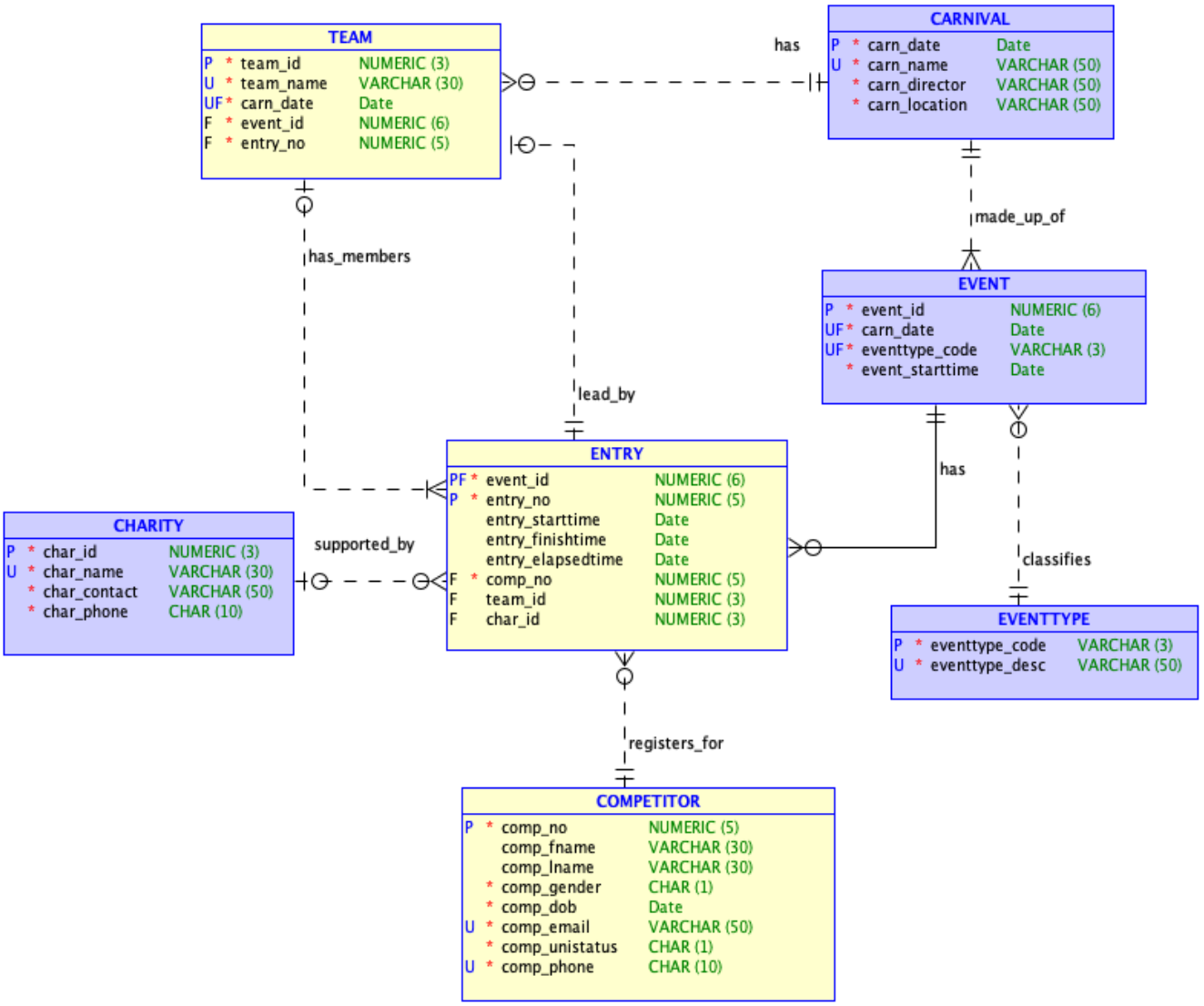
When a competitor first registers for Run Monash, they are assigned a unique competitor number. They are required to supply a unique phone number and a unique email address for contact purposes (no other competitor can be assigned the same email or phone number).

When a carnival is being offered, Run Monash contacts all registered competitors and provides details of the carnival date and what events are on offer. Competitors can only enter one event at a particular carnival (this is enforced by application logic). Every entry in an event is assigned an entry number. The entry numbers are reused in each event. As a result, each event starts numbering entries from one. Run Monash also, on the carnival day, using official timing devices, record the entrants' starting and finishing times. Immediately after the event has been completed, all entrants who have completed the event (i.e. have a finishing time) are assigned a calculated elapsed time.

Teams are identified by a unique team name that the team manager must select when they first create the team. The team manager can then add/invite other competitors from the carnival to join their team. Team members can be any entered competitor in any event in the carnival. Team names are unique only within a given carnival. A given team name may be reused by different competitors in a different carnival, since teams are recreated for each carnival.

Individual competitors in a carnival (entrants) may nominate a charity for which they will raise funds, although not all entrants in the carnival will do so. All charities for which funds can be raised must first be approved by Run Monash.

A model to represent this system has been developed:



You must ensure that any activities you carry out in the database to complete the assignment tasks conform to the requirements of the model displayed above.

The schema/insert file for creating this model (rm-schema-insert.sql) is available in the archive ass2_student.zip. This file partially creates the Run Monash tables and populates several of them (those shown in purple on the supplied model above). Please read this schema carefully and be sure you understand the various data requirements.

IMPORTANT points for you to observe when completing this assignment are:

1. The ass2-student.zip archive also contains seven script files to code your answers in. **You MUST ensure these files are regularly pushed to the GitLab server so that a clear development history is available for the marker to verify your work (a minimum of fourteen pushes are required - 2 pushes per file).** In each file, you ***must*** fill in the header details with your name and student ID before beginning work. Your SQL script files **must not include any SPOOL or ECHO commands**. Although you might include such commands when testing your work, **they must be removed before submission** (a -10 mark grade penalty will be applied if your documents contain spool or echo commands).
2. You are free to make assumptions if needed. However, ***your assumptions must align with the details here and in the Ed Assignment 2 forum*** and must be clearly documented (see the required submission files).
3. When handling dates with SQL, the default date format must not be assumed; you must use ***the TO_DATE and TO_CHAR functions where appropriate***.
4. **ANSI joins must be used** where the joining of tables is required.
5. In completing the following tasks, you must ***design your test data so that you always get output for the queries specified below*** - this may require you to add further data as you move through completing the required tasks. Such extra data MUST be added as part of Task 2 (i.e. as part of your test data load). ***Queries that are correct but do not produce any output ("no rows selected" message) using your test data will lose 50% of the marks allocated***. So, you should carefully check your test data and ensure it thoroughly validates your SQL queries.

Steps for working on Assignment 2

1. Download the Assignment 2 Required Files zip archive (ass2-student.zip) from Moodle.
2. Extract the zip archive and place the contained files in your local repository in the folder:
   /Assignments/Ass2
   Do not add the zip archive to your local repo.
3. Examine the extracted files, i.e. read carefully through them and ensure you understand their content.
4. In each supplied script, fill in the header details with your name and student ID. Then, add, commit and push them to the FITGitLab server.
5. Run rm-schema-insert.sql from the supplied zip archive to set up the initial state of the database.
6. Write your answer for each task in its respective file (e.g. write your answer for task 1 in T1-rm-schema.sql and so on).
7. Save, add, commit and push the file/s **regularly** while working on the assignment.
8. Finally, when you have completed all tasks, separately run each SQL or MongoDB ***as a script (not as individual statements) and ensure there are no errors***. Upload all required files from your local repository to Moodle. Check that the files you have uploaded are the correct files (download them from Moodle into a temporary folder and check that they are correct). After you are sure they are correct, submit your assignment.

****

For all assignment tasks, **your final script must run as a script without errors, except for SQL errors generated by the DROP TABLE/DROP SEQUENCE statements**. **Any task's script that runs with an error will receive a maximum grade of half of the task's available marks -1**. For example, if your task 1 script runs with an error, regardless of the code contained, your maximum grade will be 15/2 => 7.5 - 1 = 6.5 marks. This will be applied even if the error is simply a forgotten semicolon. Thus, **please carefully check that your final scripts for all tasks run without error**.

In arriving at your solutions for Assignment 2 you are ONLY permitted to use the SQL/NoSQL structures and syntax that have been covered within this unit:

- **Topic 6 Workshop and Applied 7 - Creating & Populating the Database**
- **Topic 7 Workshop and Applied 8 - Insert, Update, Delete (DML) and Transaction Management**
- **Topic 8 Workshop and Applied 9 - SQL Part I - Basic**
- **Topic 9 Workshop and Applied 10 - SQL Part II- Intermediate**
- **Topic 10 Workshop and Applied 11 - SQL Part III - Advanced**
- **Topic 11 Workshop and Applied 12 - Non-Relational Database**

SQL/NoSQL syntax and commands outside of the covered work will NOT be accepted/marked.

You must NOT use PL/SQL commands such as BEGIN/END nor SQL structures such as WITH since these were not covered in the unit. Views must not be used in completing these tasks.

You must also keep up to date with the Ed Assignment 2 forum where further clarifications may be posted. Please **ensure you do not publicly post anything that includes your reasoning, logic, or any part of your work to this forum**; doing *so violates Monash plagiarism/collusion rules* and has significant academic penalties. Attend a consultation session or use a private Ed post to raise such questions.

****

GIT STORAGE

Your work for these tasks MUST be saved in your local working directory (repo) in the Assignment/Ass2 folder and **regularly pushed to the FIT GitLab server to build a clear history of the development of your approach**. ***Any work outside this folder will not be marked***. A minimum of fourteen pushes to the FIT GitLab server is required (2 pushes per solution script). Please note that fourteen pushes are a minimum; we expect significantly more in practice. All commits must include a **meaningful commit message** that clearly describes what the particular commit is about and **must be correctly assigned to your valid GitLab author**.

You must regularly check that your pushes have been successful by logging in to the FIT GitLab server's web interface; you must not simply assume they are working. Before submission via Moodle, you must log in to the GitLab server's web interface and ensure your submission files are present and their names are unchanged.

****

TASK 1: DDL (15 marks)

For this task, you are required to add to **T1-rm-schema.sql**, the CREATE TABLE and CONSTRAINT definitions that are missing from the supplied partial schema script in the positions indicated by the comments in the script. You must use the default delete rule (i.e. no action/restrict) for all foreign keys.

The table below provides details of the meaning of the attributes in the three missing tables. You **MUST use exactly the same relation and attribute names and data types/sizes as shown in the data model** above to name the tables and attributes that you add. The attributes **must be in the same order as shown in the model**. These new DDL commands *must be hand-coded, not generated in any manner (generated code will not be marked)*.

Table name	Attribute name	Meaning
COMPETITOR		
	comp_no	Unique identifier for a competitor
	comp_fname	Competitor's first name
	comp_lname	Competitor's last name
	comp_gender	Competitor's gender ('M' for male, 'F' for female, or 'U' for 'Undisclosed')
	comp_dob	Competitor's date of birth
	comp_email	Competitor's email - unique for each competitor
	comp_unistatus	Competitor is a university student or staff ('Y' for Yes or 'N' for No)
	comp_phone	Competitor's phone number - unique for each competitor
ENTRY		
	entry_no	Entry number (unique only within an event)
	entry_starttime	The entrant's start time (time only), stored using the format of hh24:mi:ss
	entry_finishtime	The entrant's finish time (time only), stored using the format of hh24:mi:ss
	entry_elapsedtime	The time the entrant took to complete the event, stored using the format of hh24:mi:ss (e.g. 01:25:30 for 1 hour 25 minutes and 30 seconds)
TEAM		
	team_id	Team identifier (unique)
	team_name	Team name

To test your code, you must first run the provided script rm-schema-insert.sql to create the purple supplied tables.

TASK 2: Populate Sample Data (20 marks)

Before proceeding with Task 2, you must ensure you have run the file rm-schema-insert.sql (which **must not be edited in any way**, followed by the extra definitions that you added in Task 1 above (T1-rm-schema.sql).

Load the COMPETITOR, ENTRY, and TEAM tables with **your own test data** using the **T2-rm-insert.sql**. Add (write) SQL commands to **T2-rm-insert.sql** that will insert *as a minimum (i.e. you may and should insert more)* the following sample data:

(i) 15 COMPETITOR entries
- Have at least 5 competitors who are Monash student/staff
- Have at least 5 competitors who are not Monash student/staff

(ii) 30 ENTRY entries
- Included at least 10 competitors
- Included at least 6 events from 3 different carnivals
- Have at least 5 competitors who join more than 2 events
- Have at least 2 uncompleted entries (registration only)

(iii) 5 TEAM entries
- Have at least 2 teams with more than 2 members
- At least one team name is used in two different carnivals (eg, the team name 'Coyotes' is used in two different carnivals).

In adding this data, you must ensure that the test data thoroughly tests the model as supplied to ensure your schema is correct (you are not required to submit or code fail tests; all insert statements must execute correctly).

Your inserted data must conform to the following rules:

(i) Treat all the data you add as a single transaction since you are setting up the initial test state for the database.

(ii) The primary key values for this data should be hardcoded values (i.e. NOT make use of sequences). If the primary key attribute's data type is numeric, it must consist of values below 100.

(iii) The data added must be sensible, e.g. entry finish times should be after entry start times with a sensible elapsed time.

For this task **ONLY,** Task 2, you may manually look up/calculate and include values for the loaded tables/data directly where required. However, you can still use SQL to get any non-key values if you wish. You may make use of AI & Generative AI tools to assist you with the generation of this data; however, if you do so, you must clearly acknowledge the source following Monash Guidelines. Place the acknowledgement in the "Comments for your marker" section at the top of T2-rm-insert.sql.

In carrying out Task 2, you must not modify any data or add any further data to the tables populated by the rm-schema-insert.sql script. Design your test data to get output for the SQL scripts/queries specified below - this may require you to add further data as you complete the required tasks.

TASK 3: DML (15 marks)

Your answers for this task (Task 3) must be placed in the supplied SQL Script **T3-rm-dm.sql**

For this and *all subsequent Tasks*, **you are NOT permitted to**:

- manually look up a value in the database, obtain its primary key, or manually obtain the highest/lowest value in a column,
- manually calculate values external to the database, e.g. on a calculator and then use such values in your answers. ***Any necessary calculations must be carried out as part of your SQL code*** or
- assume any particular contents in the database - rows in a table are potentially in a constant state of change

Your answers must recognise that you are dealing with only a very small sample snapshot of a multiuser database; as such, you must operate on the basis that there will be ***more data in all of the database tables than you currently have access to. Thus, data will be in a constant state of change. Your answers must work regardless of the extra quantity of this extra "real" data and the fact that multiple users will operate in the tables simultaneously. You must consider this aspect when writing SQL statements.***

For any following SQL tasks, **your SQL must correctly manage transactions and use sequences to generate new primary keys for numeric primary key values** (a new primary key value cannot be hardcoded as a number or value).

You must ONLY use the data provided in the questions' text.

For Task 3, you must complete the following sub-tasks in the same order they are listed. Where you have been supplied with a string in italics, such as *RM AUTUMN SERIES CLAYTON 2025*, you may search in the database using the string *as listed*. Where a particular case for a word is provided, *you must only use that case (same spacing, case, etc) in your SQL code*. When a name is supplied, you may break the name into first name and last name. For example, *Farrel Grazier* can be split into Farrel and Grazier; again, note that the case must be maintained as it was supplied. **Failure to adhere to these requirements, such as changing the case of a provided string, will result in a grade penalty**.

(a) Oracle sequences will be implemented in the database to insert records for the COMPETITOR and TEAM tables.

Provide the CREATE SEQUENCE statements to create sequences that could be used to provide primary key values for the COMPETITOR and TEAM tables. These sequences must start at 100 and increment by 5. Immediately before the create sequence commands, place appropriate DROP SEQUENCE commands so that the sequences will be dropped before being created if they exist. Please note that these are the ONLY sequences that can be introduced and used in Task 3.

[1 mark]

Tasks 3b - 3d are potentially related questions. Where appropriate, you can use the information given in any of the parts to answer any question e.g. use the information in (b) to help answer (c) and/or (d).

(b) Record two competitors named *Keith Rose* (phone number: *0422141112*) and *Jackson Bull* (phone number: *0422412524*). Both of them are Monash students.

Both of them have indicated that they would like to participate in *RM WINTER SERIES CAULFIELD 2025* carnival and enter the *10 km run* event. For this carnival, Keith will support the *Salvation Army* charity, and Jackson will raise funds to support the *RSPCA* charity.

Keith also indicated that both of them would like to form a team called *Super Runners* for the carnival. Run Monash staff check and confirm that the team name is not currently in use for this carnival. So they inform Keith that such a team can be created and note Keith as the team leader and Jackson as a member of the team.

Make these changes to the data in the database. You may make up sensible data for the rest of the attributes. The full set of actions to satisfy (b) must either all be recorded in the database or none of the changes should be recorded.

[8 marks]

(c) One day later, *Jackson Bull* indicated that he would like to downgrade his event for the *RM WINTER SERIES CAULFIELD 2025* carnival from the *10 km run* to the *5 km run* and change his supported charity from *RSPCA* to *Beyond Blue*.

Make these changes to the data in the database.

[2 marks]

(d) One week later, *Keith Rose* indicated that due to personal schedule conflict, he would like to withdraw from the *RM WINTER SERIES CAULFIELD 2025* carnival. As a consequence, the *Super Runners* team should be disbanded and removed from the system. *Jackson Bull* will still participate in the carnival as an individual runner.

Keith Rose also indicated that he is looking forward to competing in the next 2025 carnival.

Make these changes to the data in the database.

[4 marks]

TASK 4: DATABASE MODIFICATIONS (10 marks)

Your answers for these tasks (Task 4) must be placed in the supplied SQL script **T4-rm-mods.sql**

The required changes must be made to the "live" database (the database *after* you have completed tasks 1, 2 and 3, in which other users must be assumed to be active). You MUST not edit and execute your schema file again. Before completing the work below, please ensure you have completed tasks 1, 2 and 3 above.

In completing this task, you **must**:

- if you need to add new columns, tables or related constraints, follow the naming conventions used in the data models and schema file which have been provided,
- provide column comments for any new columns that you add, and
- correctly manage any transactions used as part of your solution

(a) Run Monash wants to store the number of completed events for each competitor. Add a new attribute to the database to meet this requirement. You must populate the new attribute based on the current data in the database at the time you add the new attribute.

As part of your solution, provide appropriate select and desc statements to show the changes you have made. Select to show any data changes that have occurred and desc tablename e.g. desc customer to show any table structural changes.

[3 marks]

(b) Some competitors have indicated that they would like to support more than one charity in future carnivals. They also wish to indicate the percentage (0 to 100) of the total funds that they raise that will go to each charity.

For example, *Jackson Bull* supports the *RSPCA and Beyond Blue* charities in the *RM WINTER SERIES CAULFIELD 2025* carnival. 70% of the funds raised will be donated to the RSPCA, and the remaining 30% will be donated to *Beyond Blue*.

Change the database to satisfy this requirement. As part of your solution, provide appropriate select and desc statements to show the changes you have made. Select to show any data changes that have occurred and desc tablename, e.g. desc customer to show any table structural changes.

[7 marks]

TASK 5: SQL QUERIES (20 marks)

Your answers for this task (Task 5) must be placed in the supplied SQL script **T5-rm-select.sql.**

If you need to add further data for this task, you must return and add it as part of task 2 and then rerun rm-schema-insert.sql and all tasks (tasks 1, 2, 3 and 4) before running task 5.

You can only code **a single select statement for each question** below. Note that an SQL select statement begins with the SELECT keyword and ends with a semicolon (;) - within this statement, the SELECT keyword can be used multiple times.

Where you have been supplied with a string in *italics*, you must search the database using the provided string precisely as supplied. Where you need to show a full name, you must not have any extra spaces (e.g., leading space or extra space in the middle of the name).

Please remember VIEWS or PL/SQL (including anonymous blocks BEGIN…END) **must not be used**. **Your SQL code used must be restricted to the syntax covered in your unit.**

(a) Prepare a report that lists the team details for all completed carnivals, where the most popular team name/s (the team name/s used most often across all completed carnivals) have been used.

Each row of the listing must include, for a most popular team name:
- the team name,
- the date of the carnival in which the team name was used,
- the team leader's first name and last name for that carnival, as a single column called teamleader, and
- the number of members in the team for that carnival

Order the output by the team name and within the team name by the carnival date.

Typical output would have the form shown below (you are required to use the **format, output positions and column headings** as shown below). Clearly, you will have different data.

TEAM_NAME	CARNIVAL_DATE	TEAMLEADER	TEAM_NO_MEMBERS
Champions	22-Sep-2024	Rob De Costella	4
Champions	05-Oct-2024	Dan Chu	2

[5 marks]

(b) Prepare a report to show for each type of event, which has elapsed times, who the current record holder is (the runner with the shortest time for all such event types).

Each row of the listing must include
- the event type description (the event),
- the carnival in which the record was run showing the carnival name, day and date run,
- the record time,
- the competitor who ran this time, showing their competitor number and name, and
- their age in years at the time they ran this record.

Order the output by event type description (the event) and within an event by competitor number.

Typical, possibly incomplete, output would have the form shown below (you are required to use the **format, output positions and column headings** as shown below). Clearly, you will have different data.

Event	Carnival	Current Record	Competitor No and Name	Age at Carnival
10 Km Run	RM Spring Series Clayton 2024 held Sun 22-Sep-2024	00:55:01	00002 Rob De Costella	64
21.1 Km Half Marathon	RM Summer Series Caulfield 2025 held Sun 02-Feb-2025	02:43:10	00017 Rose	53
3 Km Community Run/Walk	RM Autumn Series Clayton 2025 held Sat 15-Mar-2025	00:13:22	00013 Williams	16
42.2 Km Marathon	RM Autumn Series Clayton 2025 held Sat 15-Mar-2025	03:06:35	00016 Rose	54
5 Km Run	RM Spring Series Caulfield 2024 held Sat 05-Oct-2024	00:28:05	00020 Nguyen Nguyen	26

[5 marks]

(c) Prepare a report to show the number of entries for every Run Monash carnival (this includes future carnivals) and the percentage of these entries in each event by carnival. For example, a carnival run on the 10th May 2025 may have had 100 entries, of these 100, 25 entered the 21.1 Km, 25 entered the 10 km run, and 50 entered the 5 Km run. The 3 Km Run and 42.2 Km Marathon were not offered. You may assume that if an event has no entries, then it was not offered. For the 10th May 2025 carnival, 25% of the carnival entries were for the 21.2 Km Half Marathon, 25% were for the 10 Km Run, and the remaining 50% entered the 5 Km Run. This report is concerned with entries; the entry is counted even if the runner did not show up or did not finish.

Each row of the listing must include

- the carnival name,
- the carnival date,
- the event description (the event type description),
- the number of entries in this event in this carnival (right-aligned) - if the event has no entries, "Not offered" must be shown (left-aligned), see below, and
- the percentage of all entries for this carnival for this event (shown as a whole number) right-aligned, see below

Order the output by carnival date and within a carnival by the number of entries, descending. If two events in the same carnival have the same number of entries, then sort them by the event type description (the event).

Typical incomplete output would have the form shown below (you are required to use the **format, output positions and column headings** as shown below). Clearly, you will have different data.

Carnival Name	Carnival Date	Event Description	Number of Entries	% of Carnival Entries
RM Spring Series Clayton 2024	22 Sep 2024	21.1 Km Half Marathon	Not offered	
RM Spring Series Clayton 2024	22 Sep 2024	3 Km Community Run/Walk	Not offered	
RM Spring Series Clayton 2024	22 Sep 2024	42.2 Km Marathon	Not offered	
RM Spring Series Clayton 2024	22 Sep 2024	10 Km Run	4	67
RM Spring Series Clayton 2024	22 Sep 2024	5 Km Run	2	33
RM Spring Series Caulfield 2024	05 Oct 2024	21.1 Km Half Marathon	Not offered	
RM Spring Series Caulfield 2024	05 Oct 2024	3 Km Community Run/Walk	Not offered	
RM Spring Series Caulfield 2024	05 Oct 2024	42.2 Km Marathon	Not offered	
RM Spring Series Caulfield 2024	05 Oct 2024	5 Km Run	5	71
RM Spring Series Caulfield 2024	05 Oct 2024	10 Km Run	2	29

In completing this task, you may wish to use an SQL CROSS join - this is the SQL version of what we covered as a CROSS join in Relational Algebra. The syntax is

select * from TABLEA CROSS JOIN TABLEB;

Remember, this is SQL. There are many ways of obtaining the required output; you may decide not to use a cross join, it is entirely up to you.

[10 marks]

TASK 6: MongoDB (15 marks)

Your answers for this task (Task 6) must be placed in the supplied sql file **T6-rm-json.sql** and the supplied MongoDB script file **T6-rm-mongo.mongodb.js.**

(a) Write an SQL statement in **T6-rm-json.sql** to generate a *collection* of JSON documents using the following structure/format from the Run Monash tables (code/run this after task 5). Each document in the collection represents a team and contains the list of the team members and their entry details.
Notes:
- Where you need to show a full name, you must not have any extra spaces (e.g., leading space or extra space in the middle of the name)
- If a field is null, it must be printed as "-"


```
[
   {
          "_id": 1,
          "carn_name": "RM Spring Series Clayton 2024",
          "carn_date": "22-Sep-2024",
          "team_name": "Champions",
          "team_leader": {
              "name": "Rob De Costella",
              "phone": "0422888999",
              "email": "rob@gmail.com"
          },
          "team_no_of_members": 4,
          "team_members": [
              {
                   "competitor_name": "Jane Ryan",
                   "competitor_phone": "0453243132",
                   "event_type": "5 Km Run",
                   "entry_no": 2,
                   "starttime": "09:31:04",
                   "finishtime": "10:02:22",
                   "elapsedtime": "00:31:18"
               },
               {
                   "competitor_name": "Cathy Freeman",
                   "competitor_phone": "0422666777",
                   "event_type": "10 Km Run",
                   "entry_no": 1,
                   "starttime": "08:30:57",
                   "finishtime": "09:38:08",
                   "elapsedtime": "01:07:11"
               },
               {
                   "competitor_name": "Rob De Costella",
                   "competitor_phone": "0422888999",
                   "event_type": "10 Km Run",
                   "entry_no": 2,
                   "starttime": "08:32:05",
                   "finishtime": "09:27:06",
```

```
                "elapsedtime": "00:55:01"
            },
            … <<only some team members shown>>
        ]
    },
    … <<only one team's data shown>>
]
```

[8 marks]

Write the MongoDB commands for the following questions, 6(b) - 6(d), in the supplied MongoDB script file named **T6-rm-mongo.mongodb.js**.

You must not add any further comments to the supplied MongoDB script file nor remove/rename any comments indicated by //
Only use the solution space to enter your answer, ENSURE every statement ends with a ;

(b) Create a new collection and insert all documents generated in 6(a) above into MongoDB. Provide a drop collection statement right above the create collection statement. You may pick any collection name. After the documents have been inserted, use an appropriate db.find command to list all the documents you added.

[1 mark]

(c) List the carnival date, carnival name, team name and team leader's name for all teams with one or more competitors who competed in either the *5 Km Run* or the *10 Km Run* events in that carnival. You must ensure that the query returns some output. If you need to add further data for this task, you must return and add it as part of task 2 and then rerun rm-schema-insert.sql and all tasks (tasks 1, 2, 3 and 4) before running task 6.

[2 marks]

(d) A competitor named *Jackson Bull* (phone number: *0422412524*) decided to form a team named *The Great Runners* for the *RM WINTER SERIES CAULFIELD 2025* carnival that will be held on *29-Jun-2025*. At this point, Jackson is the sole team member and is recorded as the team leader, registering for the *5 Km Run* event. You may manually decide the _id, email and entry_no.

(i) Write the necessary MongoDB commands to add this new team to the collection. Use an appropriate db.find command which shows only the details of *The Great Runners* team after making the change so that you show/confirm the change which was made.

Another competitor named *Steve Bull* (phone number: *0422251427*) decided to join Jackson's team which was created in (i). Steve is registering for the *10 Km Run* event.

(ii) Write the necessary MongoDB commands to add this new team member into the existing team within the collection. Use an appropriate db.find command which shows only the details of *The Great Runners* team after making the change so that you show/confirm the change which was made.

[4 marks]