INFO20003 Semester 1, 2025

Assignment 2: SQL
Due: 11:59pm Friday, 2 May, 2025
Weighting: 10% of your total assessment

EV-XYZ: Electric vehicle and charger database

Description

EV-XYZ is a platform you're creating to help keep track of its electric vehicles, charging stations, and charging activities.

An electric vehicle (EV) charging station provides charging facilities with different charging rates and costs to the electric vehicles. The charging stations can also be associated with other facilities like cafés and restaurants.

Charging station

For each charging station, the system records its details, that are – the address of the charging station (as street address, suburb, state, postcode), and the establishment date. Each charging station is also associated with at least one 'company' that owns that charging station. A charging station can be jointly owned by multiple companies.

Each charging station has at least one charging 'outlet' where electric vehicles can plug-in for charging. An outlet of a charging station can be uniquely identified with the charging station's ID and the outlet's ID, as 'charging station ID X, outlet ID Y'. Each outlet has a charging rate in kW (e.g. 120), and the charging cost per kwh is also recorded (in $/kWh, e.g. 0.25 $/kWh). Different outlets of the same charging station can have different charging costs.

The system also stores information about 'facilities' (e.g., a café or restaurant), if they are associated with a charging station. A facility can provide discount coupons, which can be used for discounted rates of a 'charging event'. For each coupon, the system stores some values of the coupon, which are – the unique coupon ID, and discount value. A coupon can only be issued by one facility and used in at most one charging event.

Electric vehicle (EV) + People

Each electric vehicle is associated with a unique vehicle identification number (VIN), manufacturer company, model name, year, capacity of the battery (in kWh, e.g. 60kWh). For each manufacturer company - the name of the company, a unique ABN number, and the current CEO's name are stored. Sometimes an EV company is owned by a parent EV company, which the model also stores.

Each electric vehicle is registered to one person. For each person, the system stores that person's (unique) driving license number, and their name. One person can have multiple electric vehicles registered with them.

Charging event

The system maintains the information of all charging events – that is, which electric vehicle is charged at which outlet of a charging station. When a person wants to charge a car, they request to charge at a particular charging station. The person who charges the car may not necessarily be the car's registered owner, so we record the license number of the person who is charging. Once an outlet is available, the system will assign an outlet to the person, and they may use it to start charging. The kWh a charge event consumed is also recorded after charging is completed.

A charging event may or may not use a discount coupon, where the coupon can only be from one of the facilities. A discount coupon represents a 'percentage discount' (e.g. a value of 0.5 indicates a 50% discount).

Data Model

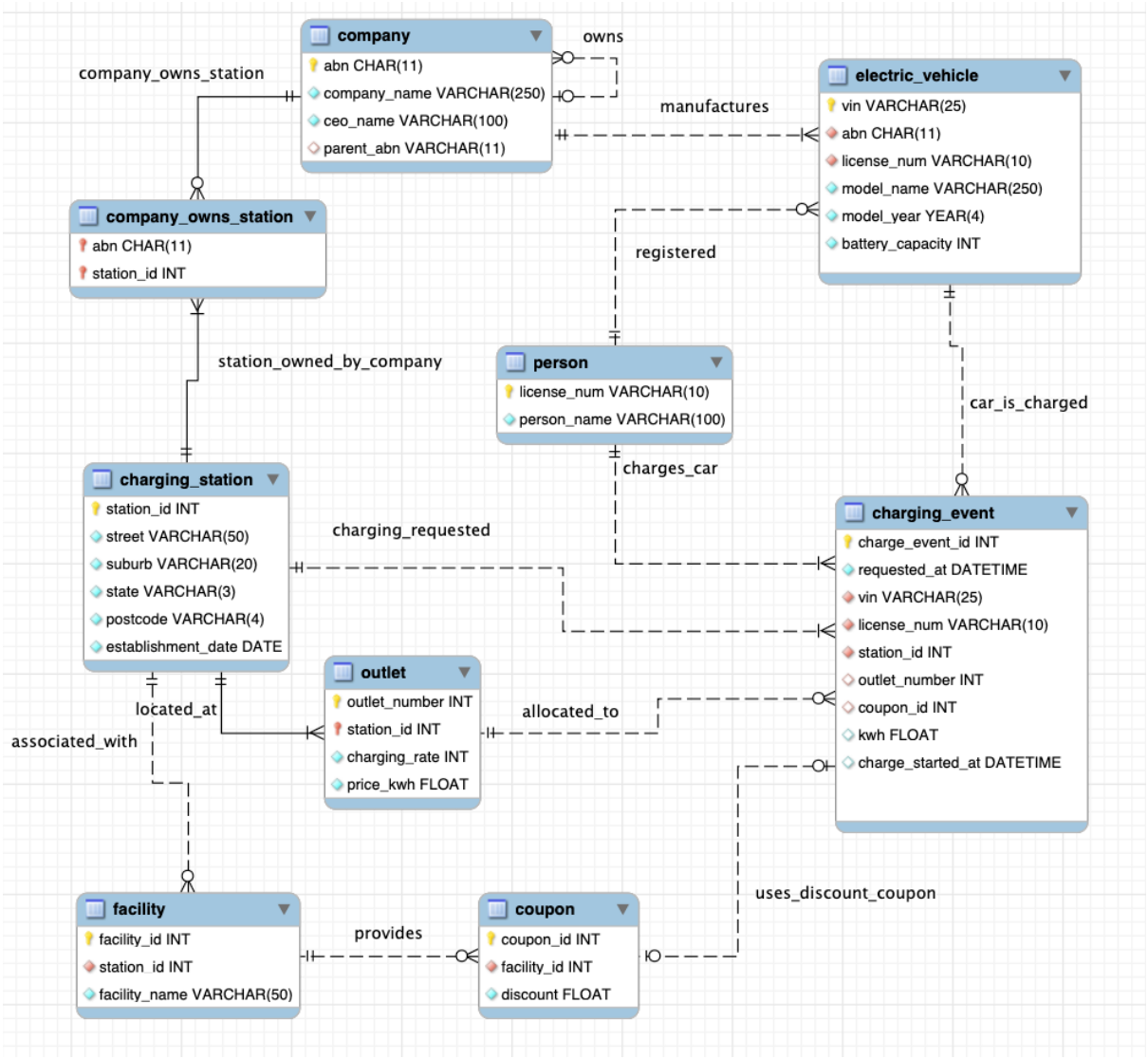


Figure 1: The physical ER model of EV-XYZ

Assignment 2 Setup

A dataset is provided which you can use when developing your solutions. To set up the dataset, download the file **ev_2025.sql** from the Assignment link on Canvas and run it in Workbench. This script creates the database tables and populates them with data. Note that this dataset is provided for you to ***experiment*** with, but it is ***not*** the same dataset as what your queries will be tested against (the schema will stay the same, but the data itself may be different). *This means when designing your queries you must consider edge cases even if they are not represented in this particular data set, and should not hardcode information like IDs into your queries.*

The script is designed to run against your account on the Engineering IT server (info20003db.eng.unimelb.edu.au). *If you want to install the schema on your own MySQL Server installation, uncomment the lines at the beginning of the script.*

⚠ **Do NOT disable only_full_group_by mode when completing this assignment**. This mode is the default, and is turned on in all default installs of MySQL workbench. *You can check whether it is turned on by running the query `SELECT @@sql_mode;`. The command should return a string containing "ONLY_FULL_GROUP_BY" or "ANSI". When testing, our test server WILL have this mode turned on, and if your query fails due to this, you will lose marks.*

The SQL tasks

In this section are listed 10 questions for you to answer. Write one (single) SQL statement per question. Subqueries and nesting are allowed within a *single* SQL statement

⚠ In general, we care more about correctness than constructing the 'most efficient' query (computationally, or in terms of number of characters/lines). However, you may be penalized for writing *overly* complicated SQL statements (e.g the query is 2-3x longer than required, using superfluous joins, etc), using very poor formatting, using very poor alias naming, or other decisions that make it hard for us to read/understand what you're trying to do when marking!

⚠ **DO NOT USE VIEWS (or 'WITH' statements/common table expressions) to answer questions.**

1. Find the model name and model year of the vehicle with the highest battery capacity. If there are ties, return a row for each of those model name and year with equal highest capacity. Your query should return results of the form (model_name, model_year, battery_capacity). (1 mark)
2. Find all the charging stations with at least one outlet of 100 or higher charging rate. Do not repeat the same station multiple times in the result if it has multiple outlets which meet the criteria. Your query should return results of the form (station_id, state, postcode). (1 mark)
3. Find all the charging stations that do not have any facility associated with them. Your query should return results of the form (station_id). (1 mark)
4. Find all the people who have electric vehicles registered in their name, where that vehicle has no charging event in the database. Only include people with at least one car registered to them that meets this criteria. Your query should return (license_number, name, total_num_of_cars_with_no_charge_event_registered_to_person), ordered by name in increasing order. (2 marks)
5. Find all facilities that have ever issued a coupon, but had no coupons redeemed on "2025-01-01" (i.e., no charging event *requested* charging using that coupon on that day). Your query should return all such facilities in the form (facility_id). (2 marks)
6. Find all vehicle models and model years that, on average, charge more than 50kWh when they charge at outlets with a charging rate > 68 kW. If a charging event has NULL for kWh value, it should not be considered in the average. The average_kwh must be **rounded to two decimal**

places (hint: use the `Round` function). Return results as (model_name, model_year, company_name, rounded_average_kwh). (2 marks).

7. Find the total number of vehicles manufactured by the company with an ABN of '1', or any of that company's child or grandchild companies. Your query should return a single value of the form (total_number_manufactured) (2 marks).

Further clarification for Q7:

If a company X is owned by company Y, then X is the child company of Y. If company Y is owned by company Z, then X is the grandchild company of Z. You may assume there are no 'great-grandchild' companies (see example below). You may also assume that there are no circular relationships, e.g., if X is a child or grandchild of Y, then Y cannot be a child or grandchild of X.

For example, suppose that the `Company` table looked like the following:

abn	**company_name**	**parent_abn**	***Note****
"1"	"General Motors LLC"	NULL	
"2"	"GMC"	"1"	*'child' company of "1"*
"3"	"Hummer, Inc"	"2"	*'grandchild' company of "1"*

Since the company with abn "2" is a child of (owned by) company "1", and company "3" is a child of company "2", answering this question would involve finding the total number of cars manufactured by companies "1", "2" and "3". There will never be a company which has a parent_abn of "3", since that would then be a "great-grandchild company".

8. Find all vehicles that have only ever been charged by people who are NOT the registered owner of the vehicle. Only include vehicles in the result that have been in at least one charging event. Return results as (VIN). Charging events with NULL kWh should still be considered. (3 marks)

9. Find all (person, car) pairings where the person has charged that car at every outlet of every station that is both located in a postcode between 3000 and 4000 (including 3000 but not 4000) and owned by the manufacturer of the car. Return results as (license_number, VIN). Only consider stations owned by the company directly, not by child companies. Charging events with NULL kWh should still be considered. (3 marks)

Further clarification for Q9:

- If a carY has been charged at all outlets matching the criteria by personX, and additionally has been charged at all outlets matching the criteria by personW, the results would include rows (license_number_personX, vin_carY) *and* (license_number_personW, vin_carY).
- A row in the output of the query indicates that the *same* person charged the *same* car at all outlets that match the criteria for *that* car. Say there exists a carY, and station1 and station2 are the only two stations that fulfil the criteria for carY (have a postcode of 3xxx, and are owned by the manufacturer of carY). Say there exists a personA who has charged carY at every outlet of station1 but never charged at any outlet of station2. A different personB also exists, who has charged the same carY at every outlet of station2 but never at any outlet of station1. In this instance, *no rows should be returned as result*, because no *single* person charged carY at every outlet matching the given criteria (even though the car was charged at every outlet by *somebody*).

10. What was the total income of outlet `2` of the charging station located at street address `125 Collins Street` in postcode `3000` in January 2025? Use the `requested_at` date to determine whether a charging event was on that date. Your query should return a single value of the form (total_income), **rounded to two decimal places** (hint: use the `Round` function, and round *after* performing any aggregations). Note that you should consider the income after applying any discounts (see hint below). (3 marks)

Hint: The income generated from a single charging event E at an outlet O which used coupon C for a discount can be calculated as:

E.kwh x *O.price_kwh* x *C.discount*

SQL Response Formatting Requirements

To help us mark your assignment queries as quickly/accurately as possible, please ensure that:

1. Your query returns the projected attributes in the same order as given in the question, and does not include additional columns.

E.g., if the question asks 'return as (userId, name)', then:

- DO: **"SELECT userId, name …"**
- ⚠ DON'T: **"SELECT name, userId…"**

You can, however, rename/name the columns to whatever you'd like using `**AS**`, only the **order** matters.

2. Do NOT use "databaseName.tableName" format.

E.g.:

- DO: **"SELECT userId FROM users…"**
- ⚠ DON'T: **"SELECT userId FROM coltonc.users …".**

Note that you *can* use tableName.columnName format, like researchers.email.

3. Ensure that you are using single quotes(') for strings

Double quotes should only be used for table names (but you shouldn't need to do this since we don't have spaces in our table names)

E.g.:

- DO: **…WHERE name = 'bob'**
- ⚠DON'T: **…WHERE name = "bob"…**

4. Do NOT delete the special comment markers in the SQL template file.

These include (where X is the question number):

-- BEGIN QX

-- END QX

-- END OF ASSIGNMENT

These help us mark your assignment!

5. Comments are optional, but will help tutors to understand your code!

Submission Instructions

Your submission will be in the form of an SQL script. There is a template file on the LMS, into which you will paste your solutions and fill in your student details (more information below).

This .sql file should be submitted on Canvas by **6pm** on the due date of **Friday 2nd May**. Name your submission as 987654.sql, where 987654 corresponds to **YOUR** student id.

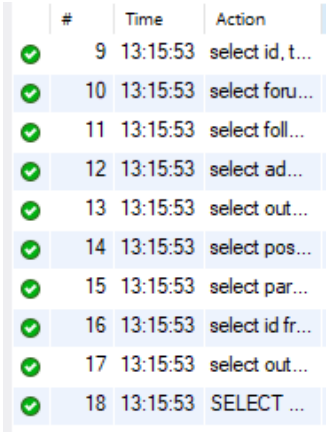
Filling in the template file:

The template file on the LMS has spaces for you to fill in your student details and your answers to the questions. There is also an example prefilled script available on the LMS as well. Below are screenshots from those two documents explaining the steps you need to take to submit your solutions:

Step	Example

1. At the top of the template, you'll need to replace "XXXXXXXX" with your student number and name	*Template*
	`-- Your Name: XXXXXXXX`
	`-- Your Student Number: XXXXXXX`
	Example Filled in
	`-- Your Name: Colton Carner`
	`-- Your Student Number: 693281`

2. For each question 1-10, place your SQL solution in between the "BEGIN QX" and "END QX" markers. **Ensure each query is terminated with a semicolon ";"**	*Template*
	`-- BEGIN Q1`
	`-- END Q1`
	Example Filled in
	`-- BEGIN Q1`
	`-- **This is an example of a comment which will not be executed`
	`-- **(comments are not NEEDED, but if your query is complex you can leave some)`
	`-- **Below is an example of how you would enter your answer:`
	`SELECT *`
	`FROM delivery`
	`NATURAL JOIN deliveryitem`
	`WHERE supplierid = 101;`
	`-- **It's OK to add more space in between the 'BEGIN QX' and 'END QX' markers`
	`-- **for each question, just don't DELETE the markers!`
	`-- **Make sure you fill out your name + student num at the top of the document`
	`-- **After reading / understanding these comments in the Q1 section`
	`-- **(comments starting with '**'), feel free to delete them`
	`-- **(don't delete lines without **)`
	`-- END Q1`

3. Test that your script is valid SQL by running it from MySQL Workbench. Run the entire script by copy-pasting this entire file into a new workbench tab, placing your cursor at the start of the file (without selecting anything), and pressing the lightning bolt to run the entire file.  All queries should run successfully one after another. If not, check to make sure you added semicolons ';' after each query.	*All 10 queries ran sequentially and were successful.* 

Late submission

Unless you have an approved extension (see below), you will be penalised -10% of the total number of marks in the assignment per day day (including weekdays and weekends) that your submission is late. For instance, if you received a 78% raw score, but submitted 2 days late, you'd receive a 58% score for the assignment.

Requesting a Submission Deadline Extension

If you need an extension due to a valid (medical) reason, you need to follow the procedure described in FEIT Extensions and Special consideration page: https://canvas.lms.unimelb.edu.au/courses/210122/pages/feit-extensions-and-special-consideration?module_item_id=6469145.

Reminder: INFO20003 Hurdle Requirements

To pass INFO20003, you must pass two hurdles:

- **Hurdle 1:** Obtain at least 50% (15/30) for the three assignments (each worth 10%)
- **Hurdle 2:** Obtain at least 50% (35/70) for the combination of the quizzes and final exam

Therefore, it is our recommendation that you attempt every assignment and question in the exam.

GOOD LUCK!