Inserting Data Using Foreign Key Lookups

NB: remember you have to replace ucfscde with your own UCL username

Example 1

insert into ucfscde.bus_station_cleaner
(bus_station_id, cleaner_id, hours_per_week)
values
((select bus_station_id from ucfscde.bus_station where name = 'Main Bus Station'), (select cleaner_id from ucfscde.cleaners where cleaner_name = 'Sarah' and cleaner_surname = 'Smith' and date_of_birth = '1992-05-23'), 25);

Column Name	**Value**	**Explanation**
bus_station_cleaner_id	(No need to insert anything, not included in the list of columns)	Generated automatically as the data type is SERIAL so the system will just take the next available ID from the pile. This means that - If you delete a row the ID for that row doesn't get reused - If more than one person is inserting data you will just get the next available number SO – YOU DON'T HAVE A GUARANTEE THAT THE FIRST ROW OF DATA YOU INSERT WILL HAVE ID = 2, THE SECOND ROW WILL HAVE ID = 2 ETC
bus_station_id		We need to find the ID of the specific bus station, and we can't guarantee that it will have ID = 1 even though it is the first one we inserted (e.g. what if other people are also inserting data?) So we need to find out what ID the system has assigned to the Main Bus Station.

Firstly, what do we need ? We only need the ID number, which is in the column bus_station_id

So we use a SELECT statement (SELECT is how we get data from the database)

SELECT bus_station_id
We also need to tell the system which table we want to get this data FROM

FROM ucfscde.bus_stations

Now we need to filter the data – and find the bus station WHERE the name is *Main Bus Station*

Name information is stored in the *name* column in our table, so we want to have

- name = 'Main Bus Station'

NB: We need quotes as it is a string (text)
NB: Make sure to use straight single quotes!
NB: Note at we use = so the string must match perfectly

Linking it all together

SELECT bus_station_id
FROM ucfscde.bus_station
WHERE name = 'Main Bus Station'

You can run this as a separate statement to test if it works, before you include it in the INSERT statement.

cleaner_id	‘UCL Sensors’	We need to find the ID of the specific cleaner, and we can’t guarantee that they will have ID = 2 even though Sarah Smith is the second person on the list. So we need to find out what ID number the system has assigned to Sarah Smith First, as an answer, we only need the cleaner_id so that is what we SELECT. *SELECT cleaner_id* We also need to say which table we are selecting the data FROM: *FROM ucfscde.cleaners* We need to find the ID of the cleaner WHERE the following details are correct: - Name is Sarah - Surname is Smith - Date_of_birth is 1992-05-23 Looking at the names of the columns in our table this translates to: - cleaner_name = 'Sarah' - cleaner_surname = 'Smith' - date_of_birth = '1992-05-23' NB: Dates need quote marks too NB: Make sure to use straight single quotes! We need ALL of these criteria to be TRUE (so if there is Sarah Jones with the same date of birth, that would be the wrong Sarah, if there is James Smith that would also be the wrong person) So we need:

		cleaner_name = 'Sarah' AND cleaner_surname = 'Smith' AND date_of_birth = '1992-05-23'
		
		This will return a value only if all three criteria are met
		
		Putting this in a final SELECT statement:
		
		SELECT cleaner_id
		FROM ucfscde.cleaners
		WHERE
		cleaner_name = 'Sarah' AND cleaner_surname = 'Smith' AND date_of_birth = '1992-05-23'
		
		A good way to test this is to run that SELECT statement on its own first, and make sure you get an answer

hours_per_week	25	Just insert the number – no need for quotation marks as this is a number not text

Example 2

insert into ucfscde.air_quality_sensors
(location, sensor_make, sensor_installation_date, waiting_room_id)
values
(st_geomfromtext('POINT(500051 500051)',27700),'UCL Sensors','20190202', (select waiting_room_id from ucfscde.waiting_room b
where st_intersects(b.location, (st_geomfromtext('POINT(500051 500051)',27700))))) ;

Column Name	**Value**	**Explanation**
sensor_id	(No need to insert anything, not included in the list of columns)	Generated automatically as the data type is SERIAL so the system will just take the next available ID from the pile. This means that - If you delete a row the ID for that row doesn't get reused - If more than one person is inserting data you will just get the next available number SO – YOU DON'T HAVE A GUARANTEE THAT THE FIRST ROW OF DATA YOU INSERT WILL HAVE ID = 1, THE SECOND ROW WILL HAVE ID = 2 ETC
location	st_geomfromtext('POINT(500051 500051)',27700)	This creates the POINT geometry of the air quality sensor.
sensor_make	'UCL Sensors'	The make of the sensor – just a string/text NB: note the use of straight quotes
sensor_installation_date	20190202	The date – but we've used an integer as the column type is an integer
waiting_room_id	(select waiting_room_id from ucfscde.waiting_room b where st_intersects(b.location, (st_geomfromtext('POINT(500051 500051)',27700)))))	This is the foreign key relationship – in other words, we want to know which waiting room the sensor is actually in. We don't know the ID value of the waiting room when we insert the data (as ID values do not always just go 1, 2, 3, 4 etc especially when there are many people inserting data into the database). So we need to find out the ID value of the specific waiting room that the sensor is inside. To do this we use the information from the waiting_room table that actually makes each waiting_room different from all the other waiting rooms – i.e. its **location** (we can't have two waiting rooms at the same location in space) Firstly, what information do we need? We need the waiting_room_id, so: SELECT waiting_room_id

		Secondly, which table has that information? ucfscde.waiting_room. We'll call it *b* for short to save typing it out more than once FROM ucfscde.waiting_room b So then we need to see which specific waiting room (which row in the ucfscde.waiting_room table) has a location WHERE the location actually INTERSECTS the point that we are creating (INTERSECTS means that the two are somehow connected in space – in this case, the sensor is on the wall of the waiting room). So we have: ST_INTERSECTS (location of the waiting room, location of the air quality sensor) (ST_ refers to a 'spatial transform' – you'll see this on all the special location/spatial SQL that we use on this module) The waiting room already exists in the database, so we take the **location** of the waiting room (which is a polygon) from the ucfscde.waiting_room table – i.e. b.location (Remember, the letter **b** is just a short way of referring to the table ucfscde.waiting_room) We don't have the sensor in the table yet (as we are just inserting it now using this SQL statement), so we take the location of the sensor from ST_GEOMFROMTEXT *('POINT(500051 500051)',27700))* And then we intersect them to find out which waiting room the sensor is in. *ST_INTERSECTS (b.location, (ST_GEOMFROMTEXT('POINT(500051 500051)',27700)))* Linking this all together

		SELECT waiting_room_id
		FROM ucfscde.waiting_room b
		WHERE
		ST_INTERSECTS (b.location, (ST_GEOMFROMTEXT ('POINT(500051 500051)',27700)))
		
		NB: Make sure the number of opening brackets is the same as the number of closing brackets.
		
		You can run this SQL on its own to test that it works, before including it in your INSERT statement.