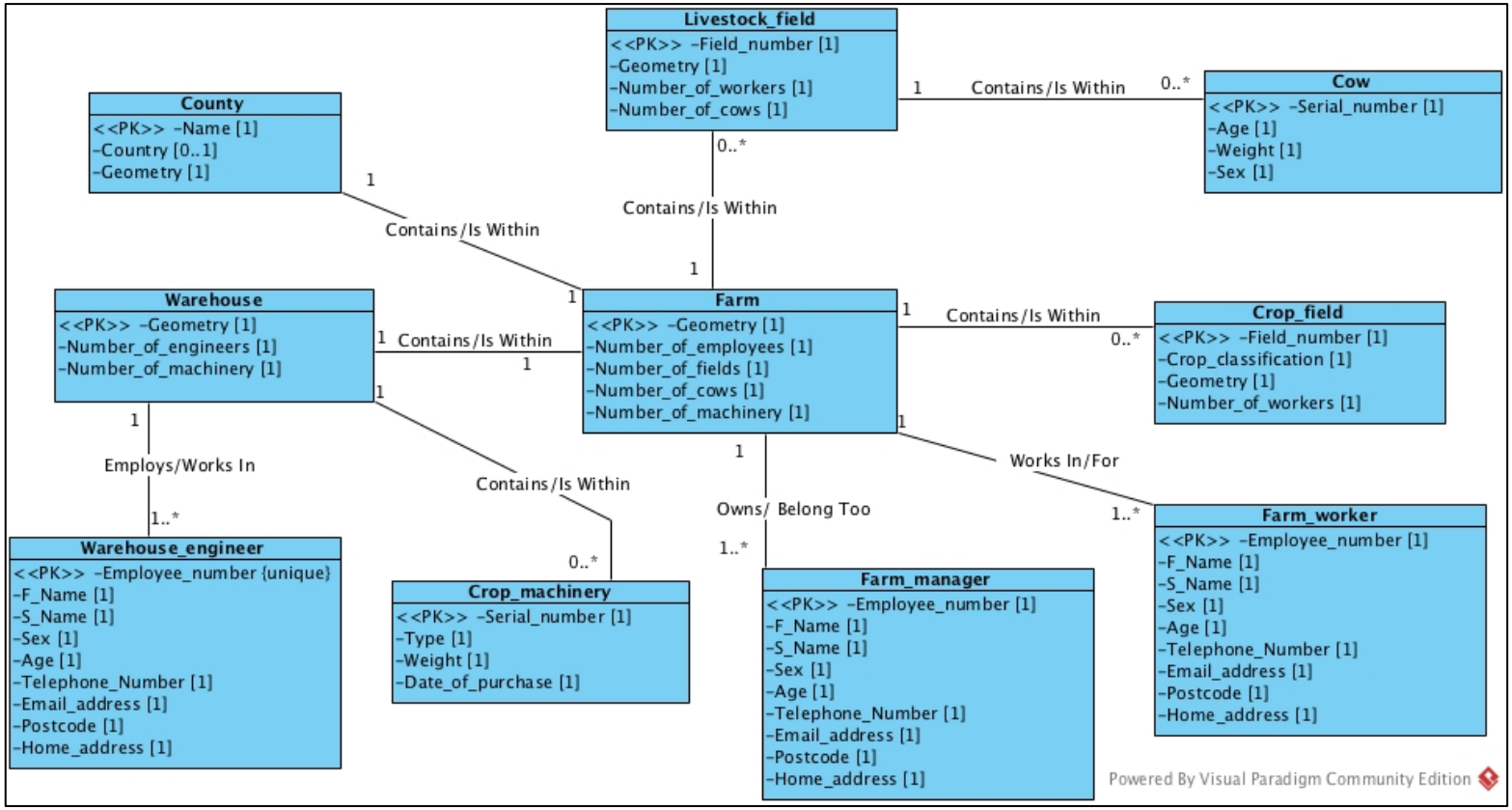
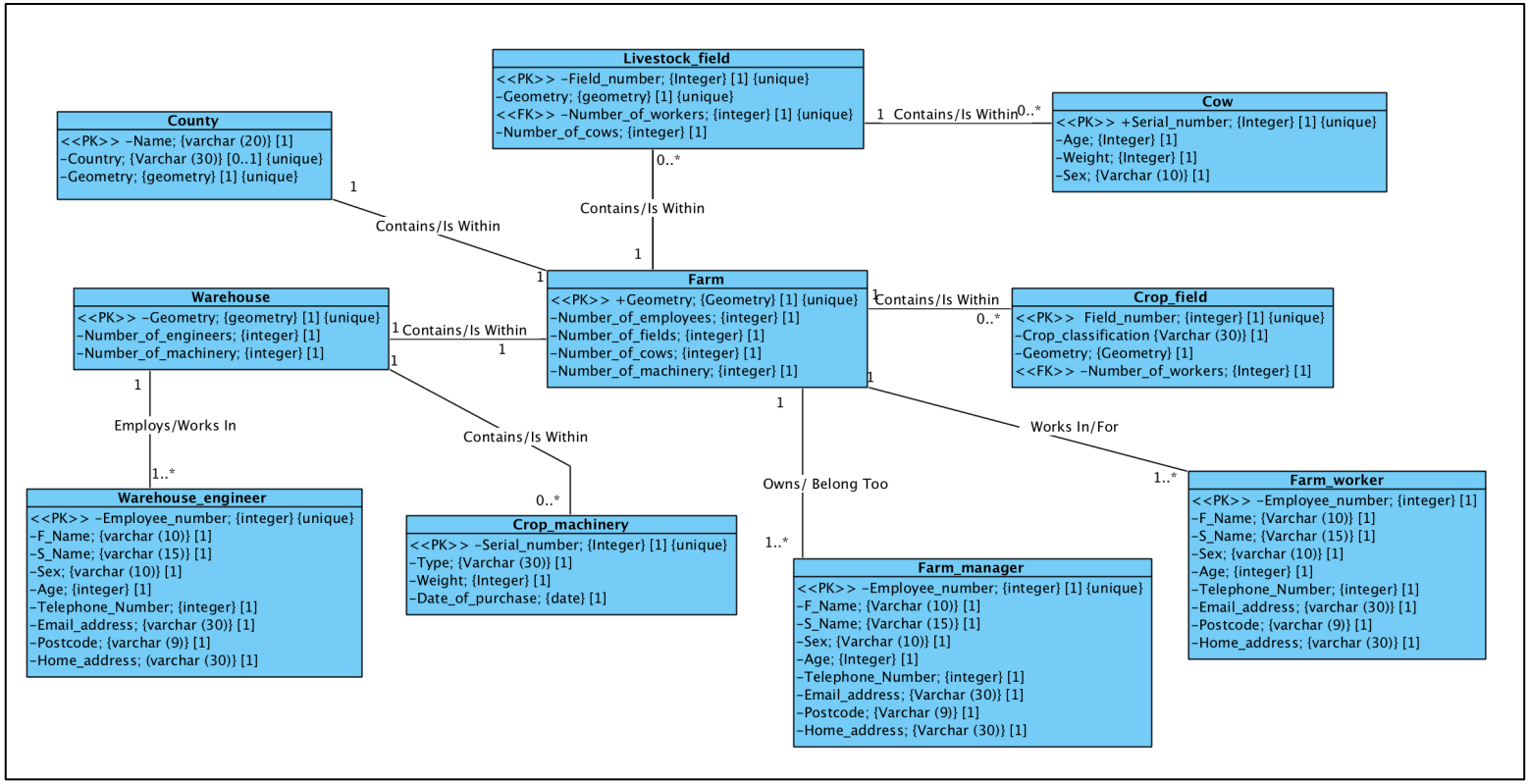
Spatial Data Management - Assignment 2

Part A - Database Creation

1. Conceptual UML diagram from Assignment 1.



2. The logical UML diagram derived from the conceptual diagram. Make sure this diagram is derived directly from the conceptual diagram you presented.



3. The SQL scripts you used to create and populate the tables (the tables and data should also be created inside your PostgreSQL work area).

Entity Name	CREATE TABLE SCRIPT	PRIMARY KEY CONSTRAINT	FOREIGN KEY CONSTRAINTS	UNIQUE CONSTRAINTS	CHECK CONSTRAINTS
County	CREATE TABLE public."County" ("Name" character varying(20) NOT NULL, "Country" character varying(30) NOT NULL, "Geometry" geometry NOT NULL, CONSTRAINT "County_pkey" PRIMARY KEY ("Name") USING INDEX TABLESPACE teaching2018tablespace, CONSTRAINT "County_Geometry_key" UNIQUE ("Geometry") USING INDEX TABLESPACE teaching2018tablespace, CONSTRAINT "County_Name_key" UNIQUE ("Name") USING INDEX TABLESPACE teaching2018tablespace) WITH (OIDS = FALSE) TABLESPACE teaching2018tablespace; ALTER TABLE public."County" OWNER to user73;	CONSTRAINT "County_pkey" PRIMARY KEY ("Name") USING INDEX TABLESPACE teaching2018tablespace,		CONSTRAINT "County_Geometry_key" UNIQUE ("Geometry") USING INDEX TABLESPACE teaching2018tablespace, CONSTRAINT "County_Name_key" UNIQUE ("Name") USING INDEX TABLESPACE teaching2018tablespace	
Cow	CREATE TABLE public."Cow" ("Serial_number" integer NOT NULL, "Age" integer NOT NULL, "Weight" integer NOT NULL, "Sex" character varying(10) NOT NULL, CONSTRAINT "Cow_pkey" PRIMARY KEY ("Serial_number") USING INDEX TABLESPACE teaching2018tablespace,	CONSTRAINT "Cow_pkey" PRIMARY KEY ("Serial_number") USING INDEX TABLESPACE teaching2018tablespace,		CONSTRAINT "Cow_Serial_number_key" UNIQUE ("Serial_number") USING INDEX TABLESPACE teaching2018tablespace,	CONSTRAINT "Cow_Sex_check" CHECK ("Sex"::text = 'Male'::text OR "Sex"::text = 'Female'::text), CONSTRAINT "Cow_Age_check" CHECK ("Age" > 0),

	CONSTRAINT "Cow_Serial_number_key" UNIQUE ("Serial_number") USING INDEX TABLESPACE teaching2018tablespace, CONSTRAINT "Cow_Sex_check" CHECK ("Sex"::text = 'Male'::text OR "Sex"::text = 'Female'::text), CONSTRAINT "Cow_Age_check" CHECK ("Age" > 0), CONSTRAINT "Cow_Serial_number_check" CHECK ("Serial_number" >= 30000 AND "Serial_number" <= 40000), CONSTRAINT "Cow_Weight_check" CHECK ("Weight" > 0)) WITH (OIDS = FALSE) TABLESPACE teaching2018tablespace; ALTER TABLE public."Cow" OWNER to user73;				CONSTRAINT "Cow_Serial_number_check" CHECK ("Serial_number" >= 30000 AND "Serial_number" <= 40000), CONSTRAINT "Cow_Weight_check" CHECK ("Weight" > 0)
Crop Field	CREATE TABLE public."Crop_field" ("Field_number" integer NOT NULL, "Crop_classification" character varying(30) NOT NULL, "Number_of_workers" integer NOT NULL, "Geometry" geometry NOT NULL, CONSTRAINT "Crop_field_pkey" PRIMARY KEY ("Field_number") USING INDEX TABLESPACE teaching2018tablespace, CONSTRAINT "Crop_field_Field_number_key" UNIQUE ("Field_number") USING INDEX TABLESPACE teaching2018tablespace, CONSTRAINT "Crop_field_Field_number_check" CHECK ("Field_number" >= 10000 AND "Field_number" <= 20000), CONSTRAINT "Crop_field_Crop_classification_check" CHECK ("Crop_classification"::text = 'Cereals'::text OR	CONSTRAINT "Crop_field_pkey" PRIMARY KEY ("Field_number") USING INDEX TABLESPACE teaching2018tablespace,	ALTER TABLE public."Crop_field" ADD CONSTRAINT "Crop_field_Number_of_workers_fkey" FOREIGN KEY ("Number_of_workers") REFERENCES public."Farm_worker" ("Employee_number") MATCH SIMPLE ON UPDATE NO ACTION	CONSTRAINT "Crop_field_Field_number_key" UNIQUE ("Field_number") USING INDEX TABLESPACE teaching2018tablespace,	CONSTRAINT "Crop_field_Field_number_check" CHECK ("Field_number" >= 10000 AND "Field_number" <= 20000), CONSTRAINT "Crop_field_Crop_classification_check" CHECK ("Crop_classification"::text = 'Cereals'::text OR "Crop_classification"::text =

	"Crop_classification"::text = 'Vegetables'::text OR "Crop_classification"::text = 'Oilseed'::text OR "Crop_classification"::text = 'Root'::text OR "Crop_classification"::text = 'Spice'::text OR "Crop_classification"::text = 'Fruit and Nuts'::text OR "Crop_classification"::text = 'Leguminous'::text OR "Crop_classification"::text = 'Sugar'::text OR "Crop_classification"::text = 'Other'::text)) WITH (OIDS = FALSE) TABLESPACE teaching2018tablespace; ALTER TABLE public."Crop_field" OWNER to user73;		ON DELETE NO ACTION;		'Vegetables'::text OR "Crop_classification"::text = 'Oilseed'::text OR "Crop_classification"::text = 'Root'::text OR "Crop_classification"::text = 'Spice'::text OR "Crop_classification"::text = 'Fruit and Nuts'::text OR "Crop_classification"::text = 'Leguminous'::text OR "Crop_classification"::text = 'Sugar'::text OR "Crop_classification"::text = 'Other'::text)
Crop machinery	CREATE TABLE public."Crop_machinery" ("Serial_number" integer NOT NULL, "Type" character varying(30) NOT NULL, "Weight" integer NOT NULL, "Date_of_purchase" date NOT NULL, CONSTRAINT "Crop_machinery_pkey" PRIMARY KEY ("Serial_number") USING INDEX TABLESPACE teaching2018tablespace, CONSTRAINT "Unique ID" UNIQUE ("Serial_number") USING INDEX TABLESPACE teaching2018tablespace, CONSTRAINT "Crop_machinery_Serial_number_check" CHECK ("Serial_number" >= 10000 AND "Serial_number" <= 20000),	CONSTRAINT "Crop_machinery_pkey" PRIMARY KEY ("Serial_number") USING INDEX TABLESPACE teaching2018tablespace,		CONSTRAINT "Unique ID" UNIQUE ("Serial_number") USING INDEX TABLESPACE teaching2018tablespace,	CONSTRAINT "Crop_machinery_Serial_number_check" CHECK ("Serial_number" >= 10000 AND "Serial_number" <= 20000), CONSTRAINT "Crop_machinery_Type_check" CHECK ("Type"::text = 'Seed planter'::text OR "Type"::text = 'Plough'::text OR

	CONSTRAINT "Crop_machinery_Type_check" CHECK ("Type"::text = 'Seed planter'::text OR "Type"::text = 'Plough'::text OR "Type"::text = 'Fertiliser'::text OR "Type"::text = 'Tractor'::text), CONSTRAINT "Crop_machinery_Weight_check" CHECK ("Weight" > 0)) WITH (OIDS = FALSE) TABLESPACE teaching2018tablespace; ALTER TABLE public."Crop_machinery" OWNER to user73;				"Type"::text = 'Fertiliser'::text OR "Type"::text = 'Tractor'::text), CONSTRAINT "Crop_machinery_Weight_check" CHECK ("Weight" > 0)
Farm	CREATE TABLE public."Farm" ("Number_of_employees" integer NOT NULL, "Number_of_fields" integer NOT NULL, "Number_of_cows" integer NOT NULL, "Number_of_machinery" integer NOT NULL, "Geometry" geometry NOT NULL, CONSTRAINT "Farm_pkey" PRIMARY KEY ("Geometry") USING INDEX TABLESPACE teaching2018tablespace, CONSTRAINT "Farm_check" CHECK ("Number_of_employees" > 0 AND "Number_of_fields" > 0 AND "Number_of_cows" > 0 AND "Number_of_machinery" > 0)) WITH (OIDS = FALSE) TABLESPACE teaching2018tablespace; ALTER TABLE public."Farm" OWNER to user73;	CONSTRAINT "Farm_pkey" PRIMARY KEY ("Geometry") USING INDEX TABLESPACE teaching2018tablespace,			CONSTRAINT "Farm_check" CHECK ("Number_of_employees" > 0 AND "Number_of_fields" > 0 AND "Number_of_cows" > 0 AND "Number_of_machinery" > 0)

Farm manager	CREATE TABLE public."Farm_manager" ("Employee_number" integer NOT NULL, "F_Name" character varying(10) NOT NULL, "S_Name" character varying(15) NOT NULL, "Sex" character(10) NOT NULL, "Age" integer NOT NULL, "Telephone_Number" integer NOT NULL, "Email_address" character varying(30) NOT NULL, "Postcode" character varying(9) NOT NULL, "Home_address" character varying(30) NOT NULL, CONSTRAINT "Farm_manager_pkey" PRIMARY KEY ("Employee_number") USING INDEX TABLESPACE teaching2018tablespace, CONSTRAINT "Farm_manager_Employee_number_key" UNIQUE ("Employee_number") USING INDEX TABLESPACE teaching2018tablespace, CONSTRAINT "Farm_manager_Age_check" CHECK ("Age" > 0), CONSTRAINT "Farm_manager_Sex_check" CHECK ("Sex" = 'Male'::bpchar OR "Sex" = 'Female'::bpchar), CONSTRAINT "Farm_manager_Employee_number_check" CHECK ("Employee_number" >= 0 AND "Employee_number" <= 10000)) WITH (OIDS = FALSE) TABLESPACE teaching2018tablespace; ALTER TABLE public."Farm_manager" OWNER to user73;	CONSTRAINT "Farm_manager_pkey" PRIMARY KEY ("Employee_number") USING INDEX TABLESPACE teaching2018tablespace,		CONSTRAINT "Farm_manager_Employee_number_key" UNIQUE ("Employee_number") USING INDEX TABLESPACE teaching2018tablespace,	CONSTRAINT "Farm_manager_Age_check" CHECK ("Age" > 0), CONSTRAINT "Farm_manager_Sex_check" CHECK ("Sex" = 'Male'::bpchar OR "Sex" = 'Female'::bpchar), CONSTRAINT "Farm_manager_Employee_number_check" CHECK ("Employee_number" >= 0 AND "Employee_number" <= 10000)
Farm Worker	CREATE TABLE public."Farm_worker" (	CONSTRAINT "Farm_worker_pkey"		CONSTRAINT "Farm_worker_Employ	CONSTRAINT "Farm_worker_Empl

	"Employee_number" integer NOT NULL, "F_Name" character varying(10) NOT NULL, "S_Name" character varying(15) NOT NULL, "Sex" character(10) NOT NULL, "Age" integer NOT NULL, "Telephone_Number" integer NOT NULL, "Email_address" character varying(30) NOT NULL, "Postcode" character varying(9) NOT NULL, "Home_address" character varying(30) NOT NULL, CONSTRAINT "Farm_worker_pkey" PRIMARY KEY ("Employee_number") USING INDEX TABLESPACE teaching2018tablespace, CONSTRAINT "Farm_worker_Employee_number_key" UNIQUE ("Employee_number") USING INDEX TABLESPACE teaching2018tablespace, CONSTRAINT "Farm_worker_Employee_number_check" CHECK ("Employee_number" >= 20000 AND "Employee_number" <= 30000), CONSTRAINT "Farm_worker_Sex_check" CHECK ("Sex" = 'Male'::bpchar OR "Sex" = 'Female'::bpchar), CONSTRAINT "Farm_worker_Age_check" CHECK ("Age" >= 18)) WITH (OIDS = FALSE) TABLESPACE teaching2018tablespace; ALTER TABLE public."Farm_worker" OWNER to user73;	PRIMARY KEY ("Employee_number") USING INDEX TABLESPACE teaching2018tablespace,		ee_number_key" UNIQUE ("Employee_number") USING INDEX TABLESPACE teaching2018tablespace,	oyee_number_check" CHECK ("Employee_number" >= 20000 AND "Employee_number" <= 30000), CONSTRAINT "Farm_worker_Sex_check" CHECK ("Sex" = 'Male'::bpchar OR "Sex" = 'Female'::bpchar), CONSTRAINT "Farm_worker_Age_check" CHECK ("Age" >= 18)
Livestock Field	CREATE TABLE public."Livestock_field" ("Field_number" integer NOT NULL,	CONSTRAINT "Livestock_field_pkey"	ALTER TABLE public."Livestock_field"	CONSTRAINT "Livestock_field_Field_	CONSTRAINT "Livestock_field_Field_number_check"

	"Number_of_workers" integer NOT NULL, "Number_of_cows" integer NOT NULL, "Geometry" geometry NOT NULL, CONSTRAINT "Livestock_field_pkey" PRIMARY KEY ("Field_number") USING INDEX TABLESPACE teaching2018tablespace, CONSTRAINT "Livestock_field_Field_number_key" UNIQUE ("Field_number") USING INDEX TABLESPACE teaching2018tablespace, CONSTRAINT "Livestock_field_Field_number_check" CHECK ("Field_number" >= 10000 AND "Field_number" <= 20000), CONSTRAINT "Livestock_field_Number_of_cows_check" CHECK ("Number_of_cows" >= 0 AND "Number_of_cows" <= 10)) WITH (OIDS = FALSE) TABLESPACE teaching2018tablespace; ALTER TABLE public."Livestock_field" OWNER to user73;	PRIMARY KEY ("Field_number") USING INDEX TABLESPACE teaching2018tablespace,	ADD CONSTRAINT "Livestock_field_Number_of_workers_fkey" FOREIGN KEY ("Number_of_workers") REFERENCES public."Farm_worker" ("Employee_number") MATCH SIMPLE ON UPDATE NO ACTION ON DELETE NO ACTION; ALTER TABLE public."Livestock_field" ADD CONSTRAINT "Livestock_field_Number_of_workers_fkey1" FOREIGN KEY ("Number_of_workers") REFERENCES public."Farm_worker" ("Employee_number") MATCH SIMPLE ON UPDATE NO ACTION	number_key" UNIQUE ("Field_number") USING INDEX TABLESPACE teaching2018tablespace,	CHECK ("Field_number" >= 10000 AND "Field_number" <= 20000), CONSTRAINT "Livestock_field_Number_of_cows_check" CHECK ("Number_of_cows" >= 0 AND "Number_of_cows" <= 10)

			ON DELETE NO ACTION;		
Warehouse	CREATE TABLE public."Warehouse" ("Number_of_engineers" integer NOT NULL, "Number_of_machinery" integer NOT NULL, "Geometry" geometry NOT NULL, CONSTRAINT "Warehouse_pkey" PRIMARY KEY ("Geometry") USING INDEX TABLESPACE teaching2018tablespace) WITH (OIDS = FALSE) TABLESPACE teaching2018tablespace; ALTER TABLE public."Warehouse" OWNER to user73; UPDATE public."Warehouse" SET "Geometry"= ST_SetSRID("Geometry",27700);	CONSTRAINT "Warehouse_pkey" PRIMARY KEY ("Geometry") USING INDEX TABLESPACE teaching2018tablespace			
Warehouse engineer	CREATE TABLE public."Warehouse_engineer" ("Employee_number" integer NOT NULL, "F_Name" character varying(10) NOT NULL, "S_Name" character varying(15) NOT NULL, "Sex" character(10) NOT NULL, "Age" integer NOT NULL, "Telephone_Number" integer NOT NULL, "Email_address" character varying(30) NOT NULL, "Postcode" character varying(9) NOT NULL, "Home_address" character varying(30) NOT NULL, CONSTRAINT "Warehouse_engineer_pkey" PRIMARY KEY ("Employee_number")	CONSTRAINT "Warehouse_engineer_pkey" PRIMARY KEY ("Employee_number") USING INDEX TABLESPACE teaching2018tablespace,		CONSTRAINT "Warehouse_engineer_Employee_number_key" UNIQUE ("Employee_number") USING INDEX TABLESPACE teaching2018tablespace,	CONSTRAINT "Warehouse_engineer_Employee_number_check" CHECK ("Employee_number" >= 10000 AND "Employee_number" <= 20000), CONSTRAINT "Warehouse_engineer_Sex_check" CHECK ("Sex" = 'Male'::bpchar OR

	USING INDEX TABLESPACE teaching2018tablespace, CONSTRAINT "Warehouse_engineer_Employee_number_key" UNIQUE ("Employee_number") USING INDEX TABLESPACE teaching2018tablespace, CONSTRAINT "Warehouse_engineer_Employee_number_check" CHECK ("Employee_number" >= 10000 AND "Employee_number" <= 20000), CONSTRAINT "Warehouse_engineer_Sex_check" CHECK ("Sex" = 'Male'::bpchar OR "Sex" = 'Female'::bpchar), CONSTRAINT "Warehouse_engineer_Age_check" CHECK ("Age" >= 18)) WITH (OIDS = FALSE) TABLESPACE teaching2018tablespace; ALTER TABLE public."Warehouse_engineer" OWNER to user73;				"Sex" = 'Female'::bpchar), CONSTRAINT "Warehouse_engineer_Age_check" CHECK ("Age" >= 18)

4. The SQL showing the INSERT statements for each table, presented as a table as shown here:

Table Name	SQL Insert Statements
County	INSERT INTO public."County"("Name", "Country", "Geometry") VALUES ('Bath', 'United Kingdom', ST_GeomFromText('POLYGON((372150 168436, 390099 166655, 387296 153496, 366626 156072, 372150 168436))',27700));
Cow	INSERT INTO public."Cow"("Serial_number", "Age", "Weight", "Sex") VALUES (34042, 5, 330, 'Male'), (30292, 7, 301, 'Male'), (38010, 8, 346, 'Female'), (35551, 5, 307, 'Male'), (30172, 6, 366, 'Male'), (39299, 7, 381, 'Female'), (38801, 3, 288, 'Female'), (33982, 7, 369, 'Female'), (37721, 2, 225, 'Male'), (31933, 9, 309, 'Male'), (34998, 4, 331, 'Female'), (32020, 8, 374, 'Female');
Crop_field	INSERT INTO public."Crop_field" ("Field_number", "Crop_classification", "Number_of_workers", "Geometry") VALUES (10001, 'Cereals', 22990, ST_GeomFromText('POLYGON((373494 157243,

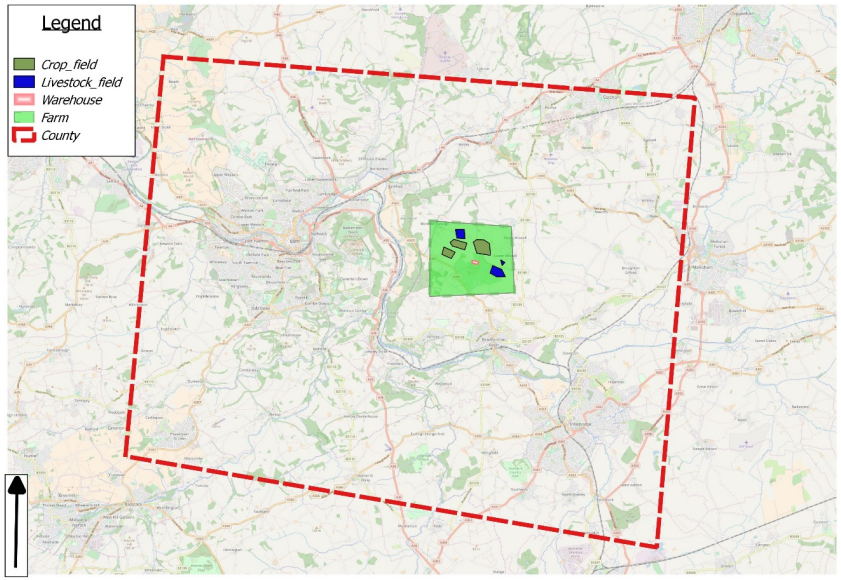
	373707 157254, 373661 157159, 373491 157181, 373494 157243))',27700)), (10002, 'Vegetables', 24723, ST_GeomFromText('POLYGON((373558 157511, 373723 157445, 373662 157398, 373567 157392, 373459 157476, 373558 157511))',27700)), (10003, 'Cereals', 28171, ST_GeomFromText('POLYGON((373725 157326, 373928 157343, 373949 157256, 373835 157206, 373665 157272, 373725 157326))',27700));
Crop_machinery	INSERT INTO public."Crop_machinery"("Serial_number", "Type", "Weight", "Date_of_purchase") VALUES (11882, 'Seed planter', 2017, '2010-05-17'), (12713, 'Fertiliser', 1989, '2014-02-06'), (13449, 'Tractor', 2399, '2011-11-03'), (13929, 'Seed planter', 2088, '2008-07-06'), (15643, 'Tractor', 1968, '2013-02-19'), (15991, 'Fertiliser', 2088, '2010-12-29'), (17372, 'Plough', 2001, '2013-11-27'), (18393, 'Fertiliser', 1989, '2015-04-29'), (19341, 'Plough', 1788, '2011-09-09');

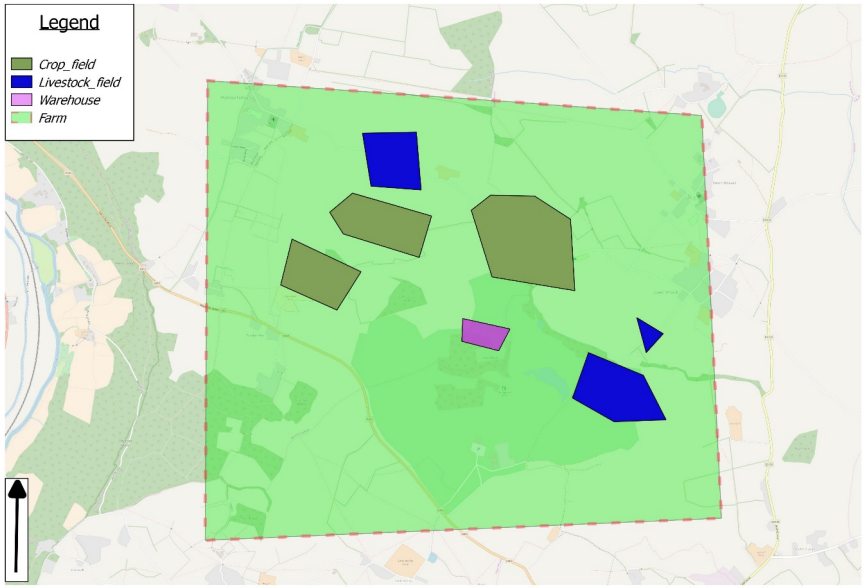
Farm	INSERT INTO public."Farm"("Number_of_employees", "Number_of_fields", "Number_of_cows", "Number_of_machinery", "Geometry") VALUES (12, 6, 12, 9, ST_GeomFromText('POLYGON((372438 157661, 376505 157604, 375585 155473, 372177 155896, 372438 157661))',27700));
Farm_manager	INSERT INTO public."Farm_manager"("Employee_number", "F_Name", "S_Name", "Sex", "Age", "Telephone_Number", "Email_address", "Postcode", "Home_address") VALUES (07728, 'Peter', 'Kingsman', 'Male', 53, 01225-475775, 'P.Kingsman@yahoo.net', 'BA1 1JG', '7 Cotswold Close');
Farm_worker	INSERT INTO public."Farm_worker"("Employee_number", "F_Name", "S_Name", "Sex", "Age", "Telephone_Number", "Email_address", "Postcode", "Home_address") VALUES (24723, 'Peter', 'Henley', 'Male', 27, 01225-929911, 'P.Henley@yahoo.net', 'BA2 1YG', '3 Convent Square'), (22990, 'Helen', 'Kingsman', 'Female', 48, 01225-383802, 'H.Kingsman@yahoo.net', 'BA1 1GG', '5 Cotswold Close'), (28171, 'Elizabeth', 'Phillips', 'Female', 19, 01225-668191, 'E.Phillips@hotmail.com', 'BA19 4II', '91A Hillsbury Avenue'), (28282, 'Bethany', 'Lammond', 'Female', 28, 01225-921812, 'B.Lammond@yahoo.net', 'BA15 90E', '16 Hudson Close'), (23027, 'James', 'Huntley', 'Male', 32, 01225-289202, 'J.Huntley@yahoo.net', 'BA4 6EE', '91 Avenue'), (21095, 'Oscar', 'Huntley', 'Male', 22, 01225-777221, 'O.Huntley@hotmail.com', 'BA4 6EE', '91 Avenue');
Livestock_field	INSERT INTO public."Livestock_field"("Field_number", "Number_of_workers", "Number_of_cows", "Geometry") VALUES (10004, 21095, 3, ST_GeomFromText('POLYGON((373036 156548, 373015 156557, 372983 156757, 373061 156804, 373036 156548))',27700)), (10005, 23027, 4, ST_GeomFromText('POLYGON((373687 156773,

	373695 156777, 373686 156714, 373554 156673, 373489 156799, 373687 156773))',27700)), (10006, 28282, 5, ST_GeomFromText('POLYGON((373655 156544, 373698 156441, 373675 156389, 373535 156435, 373453 156578, 373655 156544))',27700));
Warehouse	INSERT INTO public."Warehouse"("Number_of_engineers", "Number_of_machinery","Geometry") VALUES (6, 9, ST_GeomFromText('POLYHEDRALSURFACE(((374467 156494 0, 374405 156308 0, 374315 156326 0, 374417 156494 0, 374467 156494 0)), ((374467 156494 0, 374405 156308 0, 374405 156308 100, 374467 156494 100, 374467 156494 0)), ((374405 156308 0, 374315 156326 0, 374315 156326 100, 374405 156308 100, 374405 156308 0)), ((374315 156326 0,

	374417 156494 0, 374417 156494 100, 374315 156326 100, 374315 156326 0)), ((374417 156494 0, 374467 156494 0, 374467 156494 100, 374417 156494 100, 374417 156494 0)), ((374467 156494 100, 374405 156308 100, 374315 156326 100, 374417 156494 100, 374467 156494 100)))',27700));
Warehouse_engineer	INSERT INTO public."Warehouse_engineer"("Employee_number", "F_Name", "S_Name", "Sex", "Age", "Telephone_Number", "Email_address", "Postcode", "Home_address") VALUES (12202, 'Janet', 'Rossman', 'Female', 34, 01225-737829, 'R.Janet@outlook.com', 'BA38PJ', '1 Hudson Close'), (14537, 'Andrew', 'Garfield', 'Male', 18, 01225-619102, 'A.Garfield@outlook.com', 'BA98PJ', '8 Hudson Close'), (19299, 'Bruce', 'Rossman', 'Male', 39, 01225-737829, 'B.Rossman@outlook.com', 'BA38PJ', '1 Hudson Close'), (13738, 'Herman', 'Hibbert', 'Male', 29, 01225-918918, 'HH2@outlook.com', 'BA198GJ', '17 Mark Place'), (19292, 'Tresta', 'Mansfield', 'Female', 22, 01225-120205, 'T.Mansfield@bing.com', 'BA75HF', '481 Hudson Close'), (13040, 'Jennifer', 'Phillips', 'Female', 48, 01225-340403, 'Jenny.Phil@outlook.com', 'BA91KK', '4 Cotswold Place');

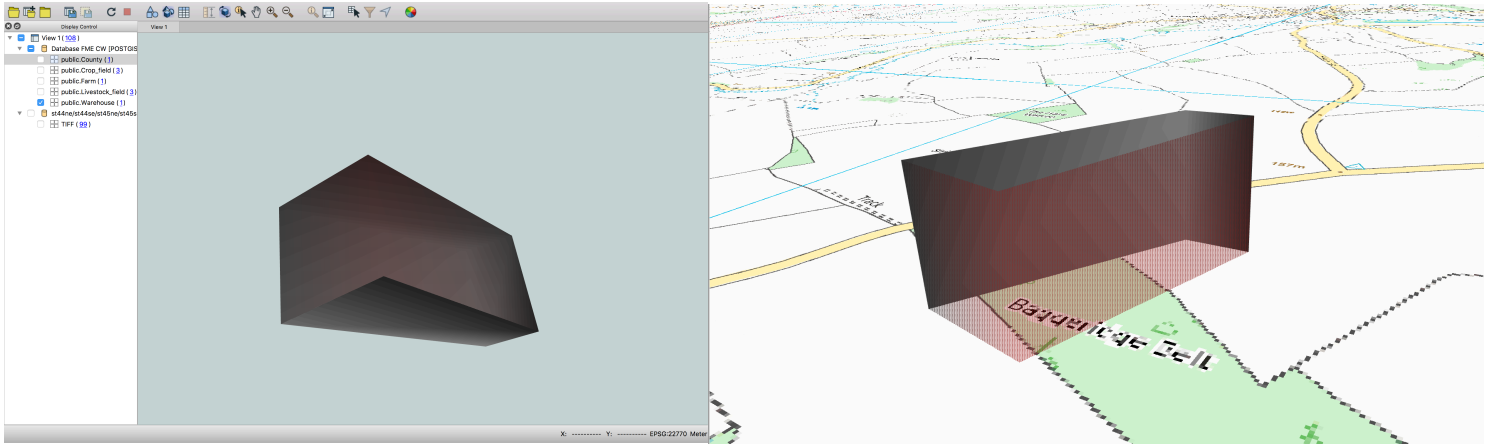
5. A map (created in QGIS) of any spatial data you have as part of this exercise.

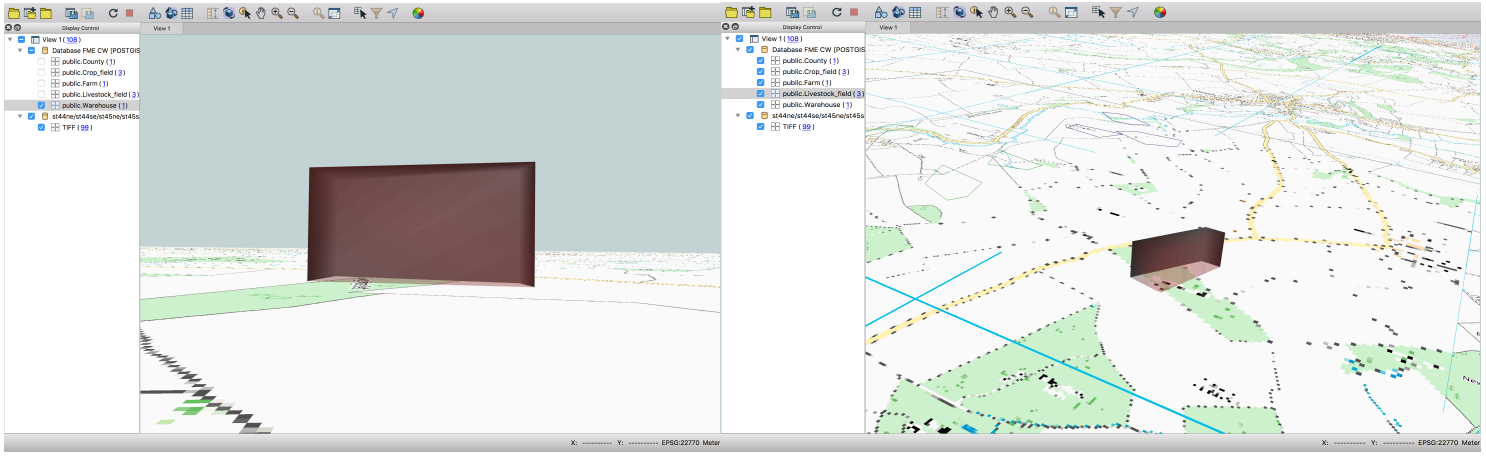




Images displaying polygon spatial data with basemap from OpenStreetMap via the Tile Server function on QGIS using link: https://tile.openstreetmap.org/{z}/{x}/{y}.

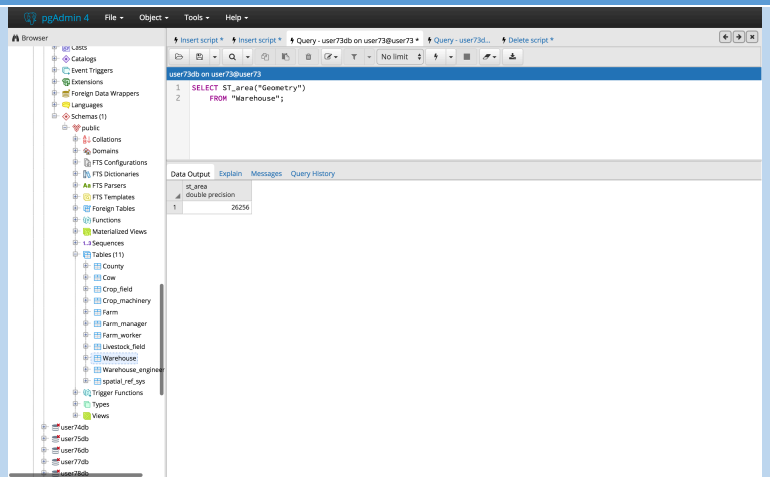
6. A 3D screen shot (from FME) of your 3D data, with appropriate background mapping.

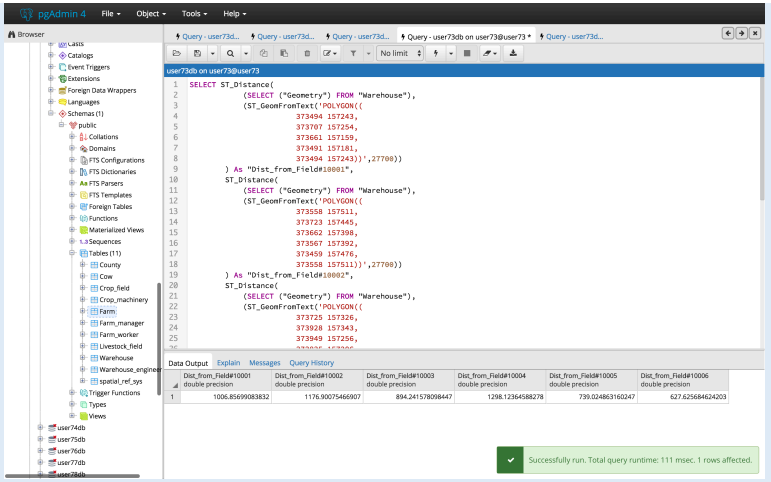




Images displaying the 3D (Warehouse) presented in FME Inspector. Background map OS VectorMap® Local Backdrop Colour Raster.

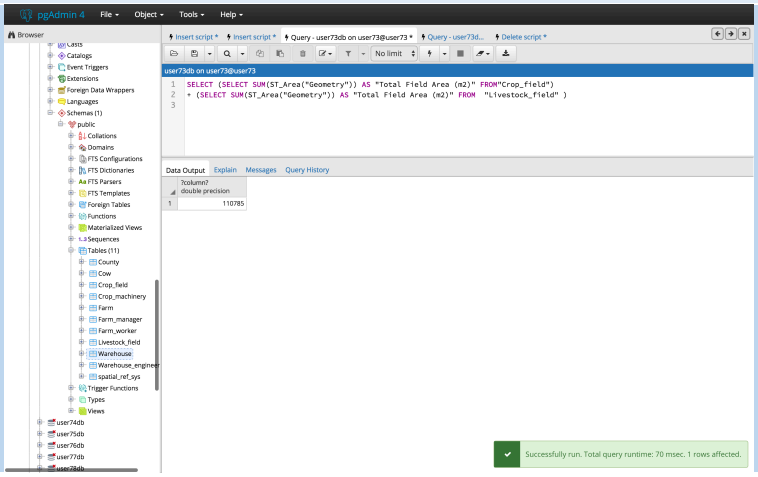
7. A table showing the list of Functional Requirements from Assignment 1 (unchanged) and the SQL you used to answer each requirement, structured as follows:

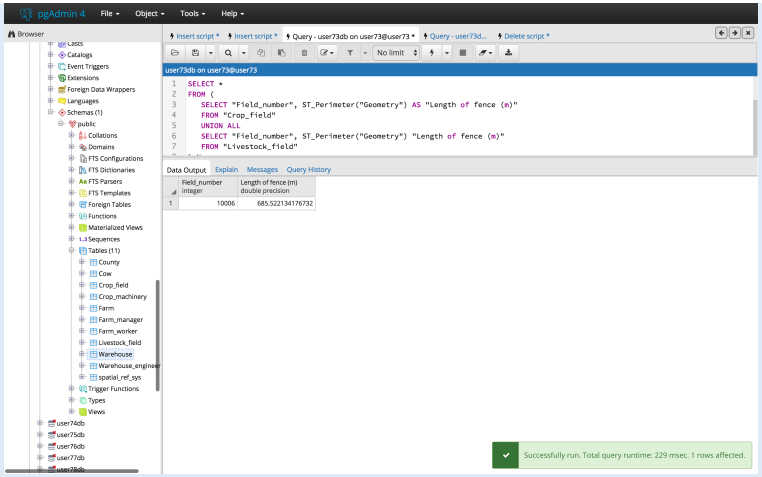
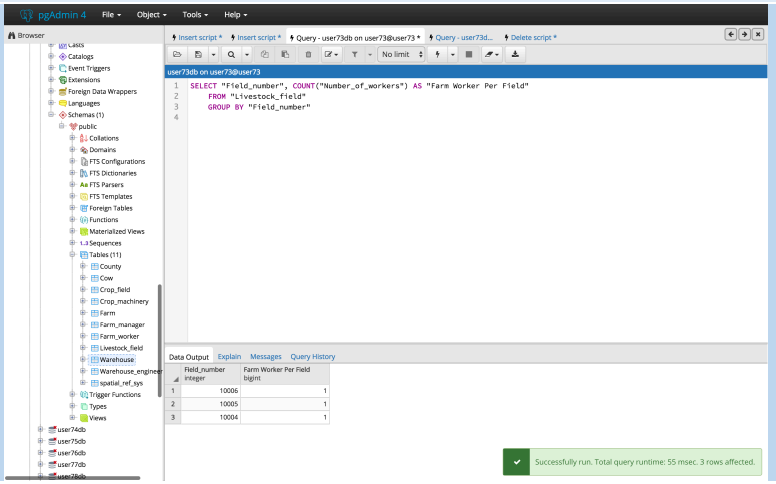
Requirement #	Requirement Description	List of Table(s) Involved	Spatial Query	Join	SQL Query	Screenshot of results from PG Admin
1	Determine the total volume of the Warehouse.	Warehouse	Yes	No	SELECT ST_area("Geometry") FROM "Warehouse";	

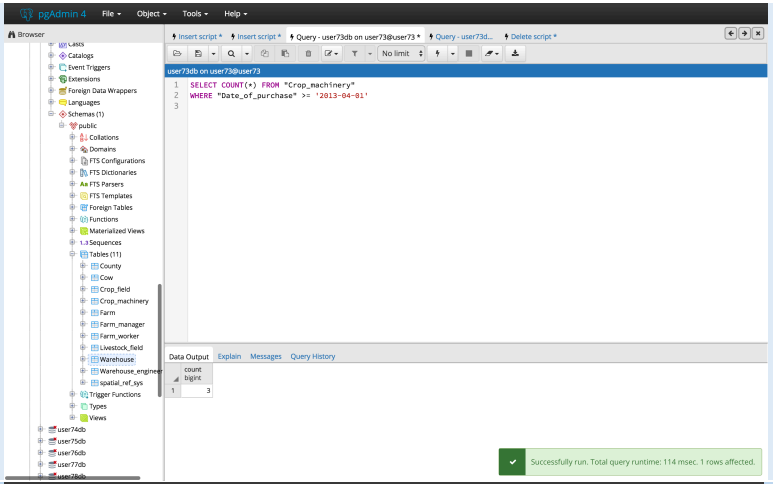
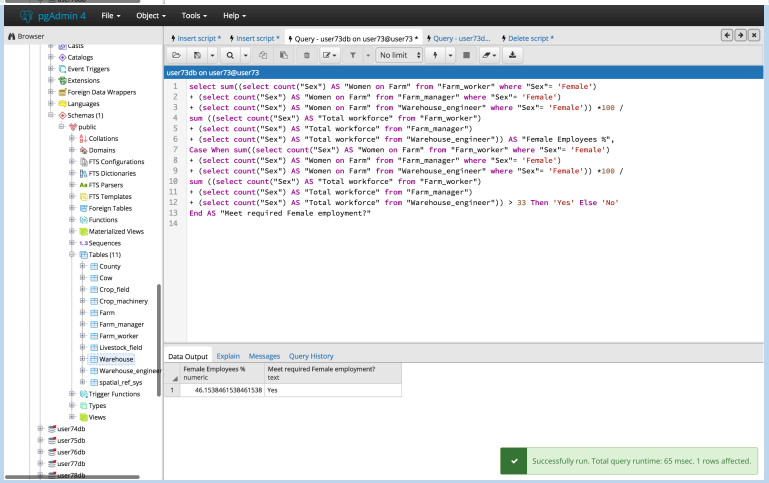
2	Find out the maximum distance from a field to the Warehouse.	Warehouse Livestock_field Crop_field	Yes	Yes	SELECT ST_Distance((SELECT ("Geometry") FROM "Warehouse"), (ST_GeomFromText('POLYGON((373494 157243, 373707 157254, 373661 157159, 373491 157181, 373494 157243))',27700))) As "Dist_from_Field#10001", ST_Distance((SELECT ("Geometry") FROM "Warehouse"), (ST_GeomFromText('POLYGON((373558 157511, 373723 157445, 373662 157398, 373567 157392, 373459 157476,	

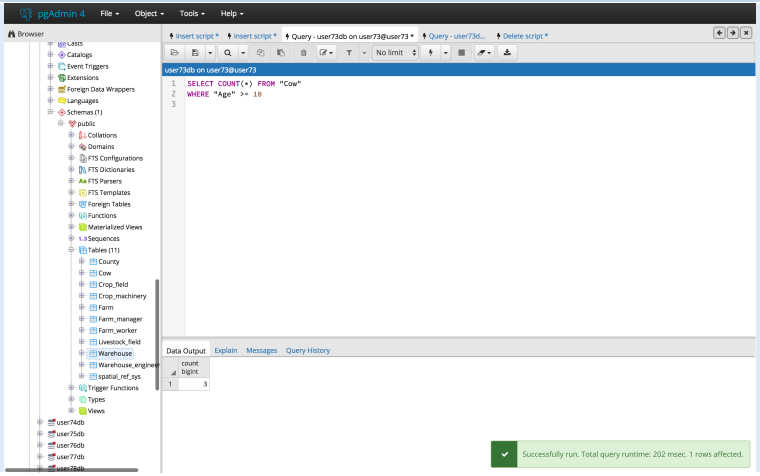
					373558 157511))',27700))) As "Dist_from_Field#10002", ST_Distance((SELECT ("Geometry") FROM "Warehouse"), (ST_GeomFromText('POLYG ON((373725 157326, 373928 157343, 373949 157256, 373835 157206, 373665 157272, 373725 157326))',27700))) As "Dist_from_Field#10003", ST_Distance((SELECT ("Geometry") FROM "Warehouse"), (ST_GeomFromText('POLYG ON((373036 156548, 373015 156557,	

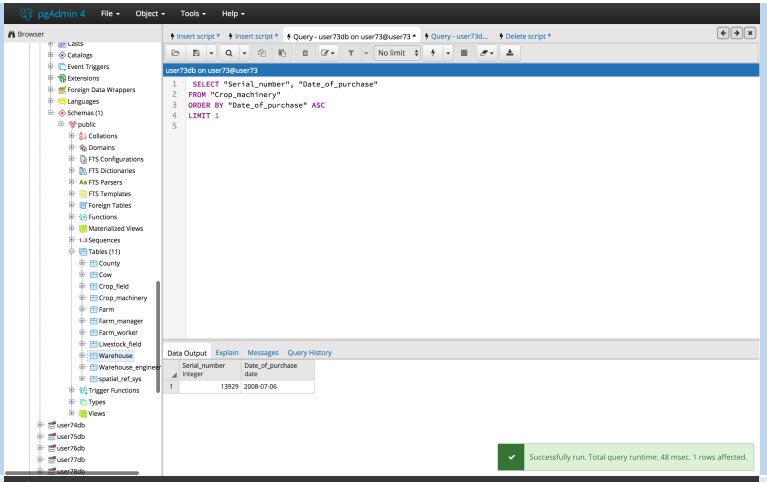
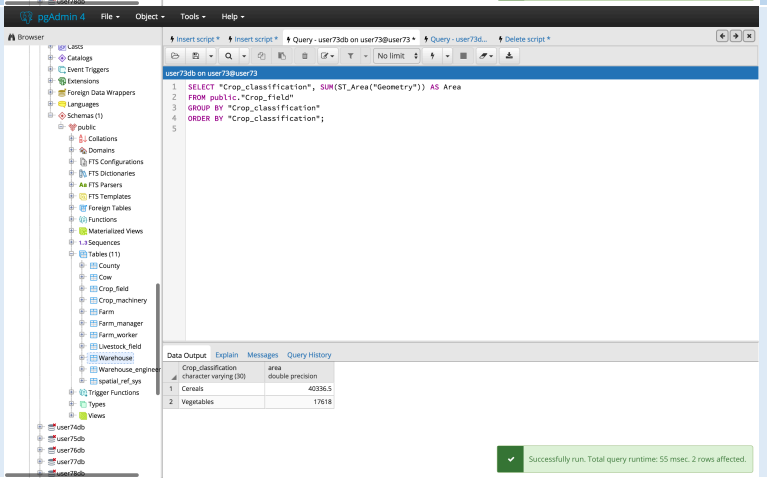
					372983 156757, 373061 156804, 373036 156548))',27700))) As "Dist_from_Field#10004", ST_Distance((SELECT ("Geometry") FROM "Warehouse"), (ST_GeomFromText('POLYG ON((373687 156773, 373695 156777, 373686 156714, 373554 156673, 373489 156799, 373687 156773))',27700))) As "Dist_from_Field#10005", ST_Distance((SELECT ("Geometry") FROM "Warehouse"),	

					(ST_GeomFromText('POLYGON((373655 156544, 373698 156441, 373675 156389, 373535 156435, 373453 156578, 373655 156544))',27700))) As "Dist_from_Field#10006";	
3	Determine the combined total surface area of all the fields in the farm.	Livestock_field Crop_field	Yes	Yes	SELECT (SELECT SUM(ST_Area("Geometry")) AS "Total Field Area (m2)" FROM"Crop_field") + (SELECT SUM(ST_Area("Geometry")) AS "Total Field Area (m2)" FROM "Livestock_field")	

4	Determine the total length of fence around the largest field.	Livestock_field Crop_field	Yes	Yes	SELECT * FROM (SELECT "Field_number", ST_Perimeter("Geometry") AS "Length of fence (m)" FROM "Crop_field" UNION ALL SELECT "Field_number", ST_Perimeter("Geometry") "Length of fence (m)" FROM "Livestock_field") N ORDER BY "Length of fence (m)" DESC LIMIT 1	
5	Determine how many Farm workers work on a Livestock field.	Farm_Worker Livestock_field	Yes	Yes	SELECT "Field_number", COUNT("Number_of_workers") AS "Farm Worker Per Field" FROM "Livestock_field" GROUP BY "Field_number"	

6	Find out which crop machinery has the oldest date of purchase.	Crop_machinery	No	No	SELECT "Serial_number", "Date_of_purchase" FROM "Crop_machinery" ORDER BY "Date_of_purchase" ASC LIMIT 1	
7	Determine if the female employees contribute to more than 33% of the total farm workforce.	Warehouse_engineer Farm_worker Farm_manager	No	Yes	select sum((select count("Sex") AS "Women on Farm" from "Farm_worker" where "Sex"= 'Female') + (select count("Sex") AS "Women on Farm" from "Farm_manager" where "Sex"= 'Female') + (select count("Sex") AS "Women on Farm" from "Warehouse_engineer" where "Sex"= 'Female')) *100 / sum ((select count("Sex") AS "Total workforce" from "Farm_worker") + (select count("Sex") AS "Total workforce" from "Farm_manager") + (select count("Sex") AS "Total workforce" from "Warehouse_engineer")) AS "Female Employees %", Case When sum((select count("Sex") AS "Women on Farm" from "Farm_worker" where "Sex"= 'Female')	

					+ (select count("Sex") AS "Women on Farm" from "Farm_manager" where "Sex"= 'Female') + (select count("Sex") AS "Women on Farm" from "Warehouse_engineer" where "Sex"= 'Female')) *100 / sum ((select count("Sex") AS "Total workforce" from "Farm_worker") + (select count("Sex") AS "Total workforce" from "Farm_manager") + (select count("Sex") AS "Total workforce" from "Warehouse_engineer")) > 33 Then 'Yes' Else 'No' End AS "Meet required Female employment?"	
8	Count the number of cows are over 10 years of age.	Cow	No	No	SELECT COUNT(*) FROM "Cow" WHERE "Age" >= 10	

9	Determine how many crop machinery items are over 5 years old.	Crop_machinery	No	No	SELECT COUNT(*) FROM "Crop_machinery" WHERE "Date_of_purchase" >= '2013-04-01'	
10	Determine which crop type has the largest potential yield based on surface area.	Crop_field	Yes	No	SELECT "Crop_classification", SUM(ST_Area("Geometry")) AS Area FROM public."Crop_field" GROUP BY "Crop_classification" ORDER BY "Crop_classification";	

Part B – Research Project.

Support for Spatial Functionality in NoSQL Databases.

NoSQL is most commonly used to describe a non-relational Database Management System (DBMS) that doesn't include SQL (Agawal and Rajan, 2017). NoSQL databases are horizontal scalable databases which can be divided in to four classes: Document, Column, Graph, and Key-Value (Lourenco et al., 2015). Varying classes within NoSQL databases is due to their specific build purpose, each class offers specific solutions depending on user needs. This is contrast to traditional relational databases, where one system is applied to all needs (Khan, Ahmed, Shahzad, 2017; Lourenco et al., 2015; Buerli and Obispo, 2012). Recent development of NoSQL has resulted in many of companies preferring the services of non-relational over relational databases (Strauch, n.d.). This can be predominantly attributed to NoSQL's ability to perform more efficiently when handling particularly large datasets, which has become increasing important due to the expansion of data volume and frequency, most notably in geospatial datasets (Venkatraman et al., 2016; Kaur and Rani, 2013; Li and Manoharan, 2013).

An increase in production of geospatial information has resulted in increasing need for databases to have appropriate spatial functions. Spatial functions are used to construct multiple spatial-features based on the spatial properties of specified objects. Only document and graph stored NoSQL databases are commonly spatially enabled, in comparison all relational databases contain spatial functions. (Schmid, Galicz, and Reinhardt, 2015; Zee and Scholten, 2013)

Neo4j is an embedded graph-orientated open source DBMS, written in Java script, that uses graph format data storage rather than tables. The graph database has many powerful functionalities including fast querying and traversing, model flexibility, and efficiently storing billions of nodes and relationships (Grolinger et al., 2013; Kaur and Rani, 2013). Neo4j contains a spatial extension that allows spatial querying for feature geometry attributes stored within nodes. (Oussous et al., 2015; Santos, Moro, Davis, 2015)

Neo4j Spatial is a library of utilities that enables spatial operations on spatial data, principally enabling topology operations by adding spatial indexes to predetermined located data to search within a defined region (Oussous et al., 2015). Neo4j Spatial supports data that form all

common geometry types, as well as, supporting imports of ESRI and OSM Shapefiles, spatial querying using R-Tree index, and GeoTools applications like GeoServer and uDig. (Neo4j-contrib.github.io., 2017)

One of Neo4j's most commonly used spatial features is that of the shortest path algorithm and connectivity queries (Gao et al., 2014; Holzschuher and Peinl, 2013; Vicknair et al., 2010). This has been observed giving much greater levels of performance when compared to SQL databases. Santos, Moro and Davis (2015) found Neo4j processed approximately double the number of vertices per second than that of PostGIS' pgRouting. This is due to it being a graph orientated database and having built-in route classes based on Dijkstra (1959), and therefore doesn't require the spatial extension or an R-Tree-over-graph index structure to perform shortest path calculations. (Santos, Moro, and Davis, 2015)

Baas (2012) drew a comprehensive comparison of Neo4j and PostgreSQL ability to store and querying spatial vector data. The subjective testing showed Neo4j operates many spatial functionalities competently, regardless of the size of data, additionally providing numerous methods of query execution. Conversely, objective testing found Neo4j to contain major limitations, most notably with bounding box queries.

Although Neo4j and NoSQL being advantageous for many spatial applications, many studies prefer relational databases, often attributed to NoSQL's relative immaturity as functional DBMS and lack of extensive spatial functions (Baas, 2012). Santos, Moro and Davis tested Neo4j in many spatial applications and overall found that it performed weaker than PostgreSQL in multiple scenarios, particularly out-performing the NoSQL database system when querying multi-table and nearby POI radius'. They attributed this to Neo4j poor use of R-Tree index, as well as an underperforming file management interface. Simon (2015) expands stating NoSQL databases are better suited for simple spatial functionalities, not providing greater spatial analysis support than SQL models.

Exemplified by Neo4j, NoSQL databases may provide many suitable applications and are often observed out-performing their SQL counterparts when performed adhering to each specified strengths (Khan, Ahmed, and Shahzad, 2017; Lourenco et al., 2015). Neo4j can be considered a suitable alternative for performing specific operations, for example shortest path analysis and connectivity queries over local regions of a graph (Baas, 2012). Comparatively, relational databases perform more efficiently when performing complex queries on structured data (Oussous et al., 2015). To conclude, both relational and non-relational databases have respective advantages and limitations, currently neither provide complete services that suit all operations and therefore each model must be selected depending on the spatial functionality required.

References

Agarwal, S. and Rajan, K.S., 2017. Analyzing the performance of NoSQL vs. SQL databases for Spatial and Aggregate queries. In *Free and Open Source Software for Geospatial (FOSS4G) Conference Proceedings* (Vol. 17, No. 1, p. 4).

Agoub, A., Kunde, F. and KADA, M., Potential of Graph Databases in Representing and Enriching Standardized Geodata.

Baas, B. (2012). *Neo4j versus PostGIS*. Masters Thesis. Geographical Information Management and Applications (GIMA).

Buerli, M. and Obispo, C.P.S.L., 2012. The current state of graph databases. *Department of Computer Science, Cal Poly San Luis Obispo, mbuerli@ calpoly. edu*, *32*(3), pp.67-83.

Dijkstra, E.W., 1959. A note on two problems in connexion with graphs. *Numerische mathematik*, *1*(1), pp.269-271.

Gao, J., Zhou, J., Yu, J.X. and Wang, T., 2014. Shortest path computing in relational dbmss. *IEEE transactions on knowledge and data engineering*, *26*(4), pp.997-1011.

Grolinger, K., Higashino, W.A., Tiwari, A. and Capretz, M.A., 2013. Data management in cloud environments: NoSQL and NewSQL data stores. *Journal of Cloud Computing: advances, systems and applications*, *2*(1), p.22.

Highscalability.com. (2009). *Neo4j - a Graph Database that Kicks Buttox - High Scalability*. [online] Available at: http://highscalability.com/blog/2009/6/13/neo4j-a-graph-database-that-kicks-buttox.html [Accessed 9 Apr. 2018].

Holzschuher, F. and Peinl, R., 2013, March. Performance of graph query languages: comparison of cypher, gremlin and native access in Neo4j. In *Proceedings of the Joint EDBT/ICDT 2013 Workshops* (pp. 195-204). ACM.

Kaur, K. and Rani, R., 2013, October. Modeling and querying data in NoSQL databases. In *Big Data, 2013 IEEE International Conference on* (pp. 1-7). IEEE.

Khan, W., ahmed, E. and Shahzad, W. (2017). Predictive Performance Comparison Analysis of Relational & NoSQL Graph Databases. *International Journal of Advanced Computer Science and Applications*, 8(5).

Koitzsch, K., 2017. Relational, NoSQL, and Graph Databases. In *Pro Hadoop Data Analytics* (pp. 63-76). Apress, Berkeley, CA.

Li, Y. and Manoharan, S., 2013, August. A performance comparison of SQL and NoSQL databases. In *Communications, computers and signal processing (PACRIM), 2013 IEEE pacific rim conference on* (pp. 15-19). IEEE.

SWQW1 – Spatial Data Management

Lourenço, J.R., Cabral, B., Carreiro, P., Vieira, M. and Bernardino, J., 2015. Choosing the right NoSQL database for the job: a quality attribute evaluation. *Journal of Big Data*, *2*(1), p.18.

Medhi, S. and Baruah, H.K., 2017. Relational database and graph database: A comparative analysis. *Journal of Process Management. New Technologies*, *5*(2), pp.1-9.

Neo4j-contrib.github.io. (2017). *Neo4j Spatial*. [online] Available at: https://neo4j-contrib.github.io/spatial/ [Accessed 9 Apr. 2018].

Oussous, A., Benjelloun, F.Z., Lahcen, A.A. and Belfkih, S., 2015. Comparison and classification of nosql databases for big data. In *Proceedings of International Conference on Big Data, Cloud and Applications*.

Santos P.O., Moro M.M., Davis C.A. (2015) Comparative Performance Evaluation of Relational and NoSQL Databases for Spatial and Mobile Applications. In: Chen Q., Hameurlain A., Toumani F., Wagner R., Decker H. (eds) Database and Expert Systems Applications. DEXA 2015. Lecture Notes in Computer Science, vol 9261.

Schmid, S., Galicz, E. and Reinhardt, W., 2015. Performance investigation of selected SQL and NoSQL databases. *AGILE 2015–Lisbon*, pp.9-12.

Simion, B., 2015. *Analyzing and improving the performance of spatial database processing* (Doctoral dissertation, University of Toronto (Canada)).

Sitalakshmi Venkatraman, Kiran Fahd, Samuel Kaspi, Ramanathan Venkatraman,"SQL Versus NoSQL Movement with Big Data Analytics", International Journal of Information Technology and Computer Science(IJITCS), Vol.8, No.12, pp.59-66, 2016. DOI: 10.5815/ijitcs.2016.12.07

Strauch, C. (n.d.). *NoSQL Databases*. [ebook] Selected Topics on Software-Technology: Computer Science and Media, Stuttgart Media University. Available at: http://www.christof-strauch.de/nosqldbs.pdf [Accessed 9 Apr. 2018].

Vicknair, C., Macias, M., Zhao, Z., Nan, X., Chen, Y. and Wilkins, D., 2010, April. A comparison of a graph database and a relational database: a data provenance perspective. In *Proceedings of the 48th annual Southeast regional conference*(p. 42). ACM.

Zee, E. and Scholten, H.J., 2013. Application of geographical concepts and spatial technology to the Internet of Things.