Group 21

Database Principles Coursework

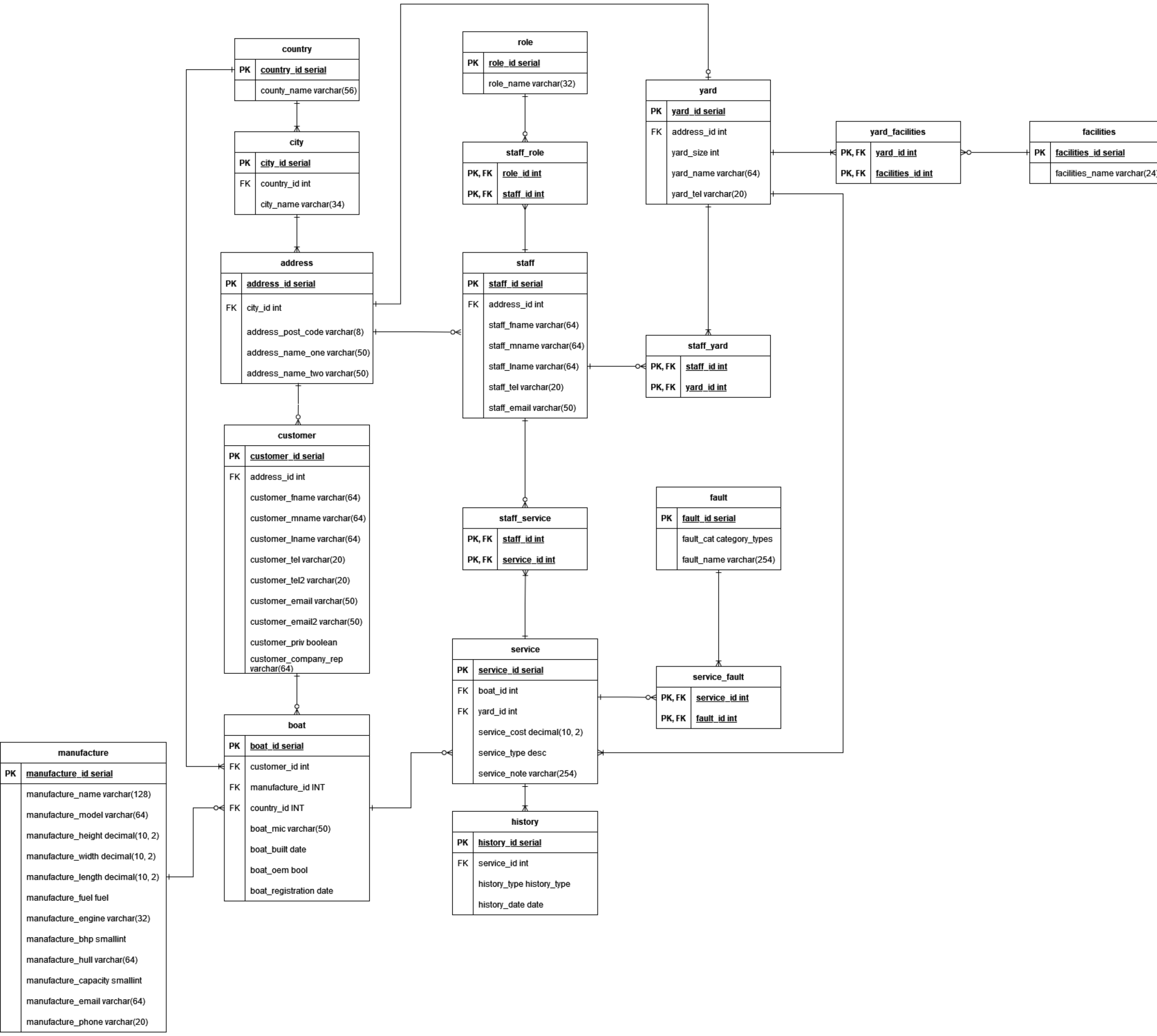
UP2122177

UP2122796

UP2166428

Contents

Entity Relationship Diagram (ERD)



Assumptions

- A boat can only be owned by one person.
- A boat can only be registered to one country.
- A boat can only be within one dockyard.
- A boat can be modified from a manufacturer, therefore it might have something different compared to the original OEM state.
- A customer/staff can share an address.
- A customer/staff may not have a middle name.
- A yard can only have one address.
- A staff member can work at several different yards, to fill shortages etc.
- A staff/yard email address will be the yard/staff name and @solent, therefore an extra attribute for the email is not required.
- A service cost may not be finalised until the end of a service marked as ‘COMPLETE’.
- A customer will most likely not be a private client, thus customer_priv will be set to ‘false’ unless stated otherwise.
- A customer may have alternative contact methods i.e. daytime/nighttime telephone number.
- A service can have many history statuses, for tracking purposes.
- A service may need notes about what was done, which can be updated at a later time.
- A boat identification (boat_mic) is used for parts, along with the service identification.

Data Dictionary

ADDRESS						
Attribute Name	**KEY**	**INDEX**	**Data Type & Size**	**Domains & Constraints**	**FK Reference**	**Description**
address_id	**PK**		**SERIAL**			
city_id	**FK**		**INT**	**NOT NULL**	city.city_id	
address_postcode		**Y**	**VARCHAR(8)**	**NOT NULL**		
address_one			**VARCHAR(50)**	**NOT NULL**		
address_two			**VARCHAR(50)**			

BOAT

Attribute Name	KEY	INDEX	Data Type & Size	Domains & Constraints	FK Reference	Description
boat_id	**PK**		**SERIAL**			
customer_id	**FK**	**Y**	**INT**	**NOT NULL**	customer.customer_id	
manufacture_id	**FK**		**INT**	**NOT NULL**	manufacture.manufacture_id	
country_id	**FK**		**INT**	**NOT NULL**	country.country_id	
boat_mic		**Y**	**VARCHAR(50)**	**UNIQUE, NOT NULL**		The boat(s) unique identifier, also used to cross reference parts used in a service
boat_built			**DATE**	**NOT NULL**		Date of boat built
boat_oem			**BOOLEAN**	**NOT NULL**		Check whether the boat is modified or an

SERVICE

Attribute Name	KEY	INDEX	Data Type & Size	Domains & Constraints	FK Reference	Description
service_id	**PK**		**SERIAL**			
boat_id	**FK**	**Y**	**INT**	**NOT NULL**	boat.boat_id	
yard_id	**FK**		**INT**	**NOT NULL**	yard.yard_id	
service_cost			**DECIMAL(10, 2)**	**NOT NULL**		
service_type		**Y**	**ENUM**	**NOT NULL**		SERVICE, CHECKUP, REPAIR, OTHERX
service_note			**VARCHAR(254)**			A note to explain the service description in a bit more depth, etc.

YARD

Attribute Name	KEY	INDEX	Data Type & Size	Domains & Constraints	FK Reference	Description
yard_id	**PK**		**SERIAL**			
address_id	**FK**		**INT**	**NOT NULL**	address.address_id	

yard_size			INT	NOT NULL		Square foot of the yard
yard_name		Y	VARCHAR(64)	NOT NULL		
yard_tel			VARCHAR(20)	NOT NULL		

YARD_FACILITIES						
Attribute Name	KEY	INDEX	Data Type & Size	Domains & Constraints	FK Reference	Description
yard_id	PK/FK		INT	NOT NULL	yard.yard_id	
facilities_id	PK/FK		INT	NOT NULL	facilities.facilities_id	

CITY						
Attribute Name	KEY	INDEX	Data Type & Size	Domains & Constraints	FK Reference	Description
city_id	PK		SERIAL			
country_id	FK		INT	NOT NULL	country.country_id	
city_name		Y	VARCHAR(34)	NOT NULL		

CUSTOMER						
Attribute Name	KEY	INDEX	Data Type & Size	Domains & Constraints	FK Reference	Description
customer_id	PK		SERIAL			
address_id	FK		INT	NOT NULL	address.address_id	
customer_fname			VARCHAR(64)	NOT NULL		
customer_mname			VARCHAR(64)			
customer_lname			VARCHAR(64)	NOT NULL		
customer_tel1		Y	VARCHAR(20)	UNIQUE, NOT NULL		
customer_tel2			VARCHAR(20)			Alternative Telephone
customer_email1		Y	VARCHAR(50)	UNIQUE, NOT NULL		
customer_email2			VARCHAR(50)			Alternative Email

customer_priv			**BOOLEAN**	**DEFAULT 'F'**		Used for private clients, specifically for businesses
customer_represent_company		**Y**	**VARCHAR(64)**			Business Name

COUNTRY						
Attribute Name	**KEY**	**INDEX**	**Data Type & Size**	**Domains & Constraints**	**FK Reference**	**Description**
country_id	**PK**		**SERIAL**			
country_name		**Y**	**VARCHAR(54)**	**UNIQUE, NOT NULL**		

FACILITIES						
Attribute Name	**KEY**	**INDEX**	**Data Type & Size**	**Domains & Constraints**	**FK Reference**	**Description**
facilities_id	**PK**		**SERIAL**			
facilities_name			**VARCHAR(24)**	**NOT NULL**		The facilities of a yard

ROLE						
Attribute Name	**KEY**	**INDEX**	**Data Type & Size**	**Domains & Constraints**	**FK Reference**	**Description**
role_id	**PK**		**SERIAL**			
role_name			**VARCHAR(32)**	**NOT NULL**		

STAFF						
Attribute Name	**KEY**	**INDEX**	**Data Type & Size**	**Domains & Constraints**	**FK Reference**	**Description**
staff_id	**PK**		**SERIAL**			
address_id	**FK**		**INT**	**NOT NULL**	address.address_id	
staff_fname			**VARCHAR(64)**	**NOT NULL**		
staff_mname			**VARCHAR(64)**			
staff_lname			**VARCHAR(64)**	**NOT NULL**		
staff_tel		**Y**	**VARCHAR(20)**	**UNIQUE, NOT NULL**		

staff_email		Y	**VARCHAR(50)**	**UNIQUE, NOT NULL**		

STAFF_ROLE						
Attribute Name	**KEY**	**INDEX**	**Data Type & Size**	**Domains & Constraints**	**FK Reference**	**Description**
role_id	**PK/FK**		**INT**	**NOT NULL**	role.role_id	
staff_id	**PK/FK**		**INT**	**NOT NULL**	staff.staff_id	

HISTORY						
Attribute Name	**KEY**	**INDEX**	**Data Type & Size**	**Domains & Constraints**	**FK Reference**	**Description**
history_id	**PK**		**SERIAL**			
service_id	**FK**		**INT**	**NOT NULL**	service.service_id	
history_type		**Y**	**ENUM**	**NOT NULL**		BOOKED, ONGOING, COMPLETE
history_date			**DATE**	**NOT NULL**		

MANUFACTURE						
Attribute Name	**KEY**	**INDEX**	**Data Type & Size**	**Domains & Constraints**	**FK Reference**	**Description**
manufacture_id	**PK**		**SERIAL**			
manufacture_name		**Y**	**VARCHAR(128)**	**UNIQUE, NOT NULL**		
manufacture_model			**VARCHAR(64)**	**NOT NULL**		Model Name
manufacture_height			**DECIMAL(10, 2)**	**NOT NULL**		
manufacture_length			**DECIMAL(10, 2)**	**NOT NULL**		
manufacture_width			**DECIMAL(10, 2)**	**NOT NULL**		
manufacture_fuel			**ENUM**	**NOT NULL**		FUEL, PETROL, HYBRID
manufacture_engine			**VARCHAR(32)**	**NOT NULL**		Engine of the Boat

manufacture_bhp			**SMALLINT**	**NOT NULL**		Horse Power of the Engine
manufacture_hull			**VARCHAR(64)**	**NOT NULL**		Hull of the Ship i.e. V-Shaped
manufacture_capacity			**SMALLINT**	**NOT NULL**		How many people can fit on a boat
manufacture_email		**Y**	**VARCHAR(64)**	**UNIQUE, NOT NULL**		
manufacture_phone		**Y**	**VARCHAR(20)**	**UNIQUE, NOT NULL**		

STAFF_YARD

Attribute Name	**KEY**	**INDEX**	**Data Type & Size**	**Domains & Constraints**	**FK Reference**	**Description**
staff_id	**PK/ FK**		**INT**	**NOT NULL**	staff.staff_id	
yard_id	**PK/ FK**		**INT**	**NOT NULL**	yard.yard_id	

STAFF_SERVICE

Attribute Name	**KEY**	**INDEX**	**Data Type & Size**	**Domains & Constraints**	**FK Reference**	**Description**
staff_id	**PK/ FK**		**INT**	**NOT NULL**	staff.staff_id	
service_id	**PK/ FK**		**INT**	**NOT NULL**	service.service_id	

SERVICE_FAULT

Attribute Name	**KEY**	**INDEX**	**Data Type & Size**	**Domains & Constraints**	**FK Reference**	**Description**
service_id	**PK/ FK**		**INT**	**NOT NULL**	service.service_id	
fault_id	**PK/ FK**		**INT**	**NOT NULL**	fault.fault_id	

FAULT

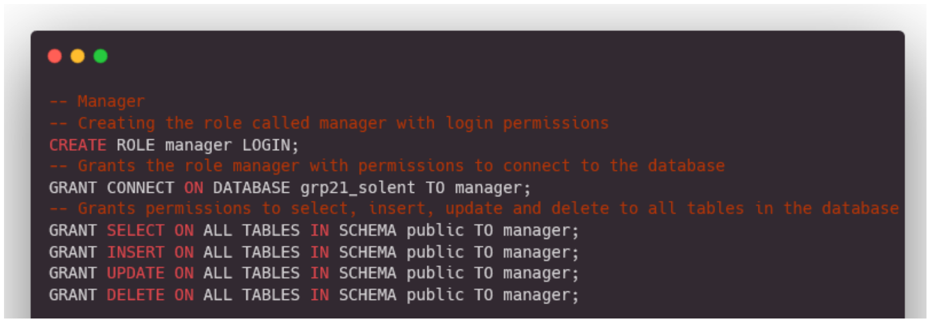
Attribute Name	KEY	INDEX	Data Type & Size	Domains & Constraints	FK Reference	Description
fault_id	**PK/ FK**		**SERIAL**			
fault_cat			**ENUM**	**NOT NULL**		MINOR, SERIOUS, DANGEROUS
fault_name			**VARCHAR(64)**	**NOT NULL**		

Security

Security in a database is critical for protecting sensitive information and ensuring the integrity and confidentiality of the data that is stored on it. Below are the privileges and roles that are integrated into the database system.

Manager

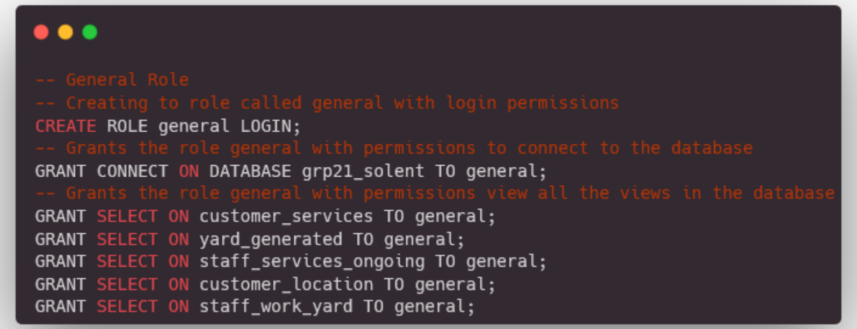
The managerial role holds the second-highest level of authority within the database hierarchy, superseded only by the Database Administrator (DBA). The manager possesses comprehensive privileges, including the ability to query, insert, update, and delete data across all tables within the database system. However, the manager is expressly restricted from creating and dropping tables, as these tasks are exclusively within the purview of the Database Administrator, who retains sole authority over such operations.



```
-- Manager
-- Creating the role called manager with login permissions
CREATE ROLE manager LOGIN;
-- Grants the role manager with permissions to connect to the database
GRANT CONNECT ON DATABASE grp21_solent TO manager;
-- Grants permissions to select, insert, update and delete to all tables in the database
GRANT SELECT ON ALL TABLES IN SCHEMA public TO manager;
GRANT INSERT ON ALL TABLES IN SCHEMA public TO manager;
GRANT UPDATE ON ALL TABLES IN SCHEMA public TO manager;
GRANT DELETE ON ALL TABLES IN SCHEMA public TO manager;
```

General

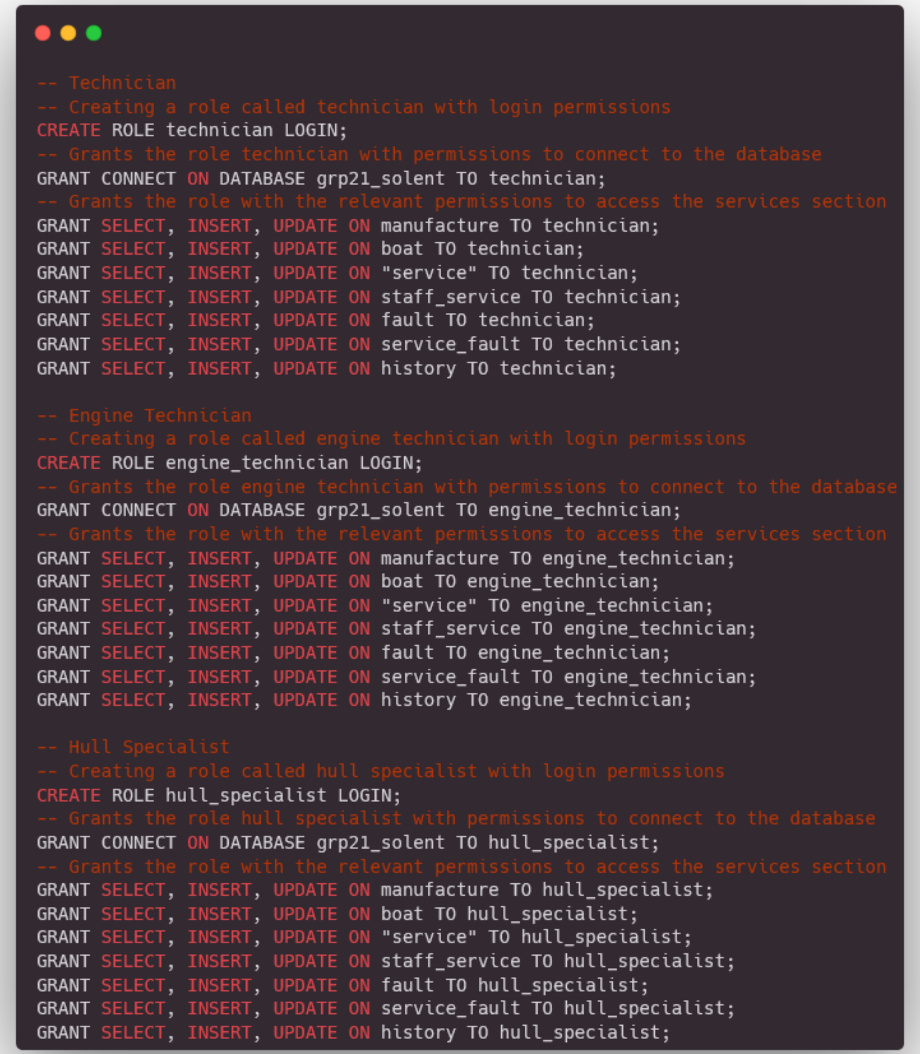
The general role is assigned the most restricted level of permissions within the database system. This role is limited to the privilege of selecting views created in the database, with explicit restrictions imposed on adding, modifying, or deleting any data. This stringent access control is implemented to safeguard the integrity of the data by preventing unauthorised modifications or deletions.



```
-- General Role
-- Creating to role called general with login permissions
CREATE ROLE general LOGIN;
-- Grants the role general with permissions to connect to the database
GRANT CONNECT ON DATABASE grp21_solent TO general;
-- Grants the role general with permissions view all the views in the database
GRANT SELECT ON customer_services TO general;
GRANT SELECT ON yard_generated TO general;
GRANT SELECT ON staff_services_ongoing TO general;
GRANT SELECT ON customer_location TO general;
GRANT SELECT ON staff_work_yard TO general;
```

Technician, Engine Technician and Hull Specialists

All engineering roles are granted permissions to access a comprehensive set of tables pertaining to engineering and service domains, encompassing tables such as manufacture, boat, and services, among others. These roles are endowed with the privileges to select, insert, and update data within the designated tables. Notably, the sole restriction within the scope of these accessible tables is the absence of the right to delete data, ensuring data integrity is preserved within this context.



```
-- Technician
-- Creating a role called technician with login permissions
CREATE ROLE technician LOGIN;
-- Grants the role technician with permissions to connect to the database
GRANT CONNECT ON DATABASE grp21_solent TO technician;
-- Grants the role with the relevant permissions to access the services section
GRANT SELECT, INSERT, UPDATE ON manufacture TO technician;
GRANT SELECT, INSERT, UPDATE ON boat TO technician;
GRANT SELECT, INSERT, UPDATE ON "service" TO technician;
GRANT SELECT, INSERT, UPDATE ON staff_service TO technician;
GRANT SELECT, INSERT, UPDATE ON fault TO technician;
GRANT SELECT, INSERT, UPDATE ON service_fault TO technician;
GRANT SELECT, INSERT, UPDATE ON history TO technician;

-- Engine Technician
-- Creating a role called engine technician with login permissions
CREATE ROLE engine_technician LOGIN;
-- Grants the role engine technician with permissions to connect to the database
GRANT CONNECT ON DATABASE grp21_solent TO engine_technician;
-- Grants the role with the relevant permissions to access the services section
GRANT SELECT, INSERT, UPDATE ON manufacture TO engine_technician;
GRANT SELECT, INSERT, UPDATE ON boat TO engine_technician;
GRANT SELECT, INSERT, UPDATE ON "service" TO engine_technician;
GRANT SELECT, INSERT, UPDATE ON staff_service TO engine_technician;
GRANT SELECT, INSERT, UPDATE ON fault TO engine_technician;
GRANT SELECT, INSERT, UPDATE ON service_fault TO engine_technician;
GRANT SELECT, INSERT, UPDATE ON history TO engine_technician;

-- Hull Specialist
-- Creating a role called hull specialist with login permissions
CREATE ROLE hull_specialist LOGIN;
-- Grants the role hull specialist with permissions to connect to the database
GRANT CONNECT ON DATABASE grp21_solent TO hull_specialist;
-- Grants the role with the relevant permissions to access the services section
GRANT SELECT, INSERT, UPDATE ON manufacture TO hull_specialist;
GRANT SELECT, INSERT, UPDATE ON boat TO hull_specialist;
GRANT SELECT, INSERT, UPDATE ON "service" TO hull_specialist;
GRANT SELECT, INSERT, UPDATE ON staff_service TO hull_specialist;
GRANT SELECT, INSERT, UPDATE ON fault TO hull_specialist;
GRANT SELECT, INSERT, UPDATE ON service_fault TO hull_specialist;
GRANT SELECT, INSERT, UPDATE ON history TO hull_specialist;
```

Optimisation

TRANSACTION TABLE	BOAT				SERVICE				HISTORY				STAFF_SERVICE				STAFF			
	C	R	U	D	C	R	U	D	C	R	U	D	C	R	U	D	C	R	U	D
Finding the staff assigned to the services that are ongoing		X				X				X				X				X		

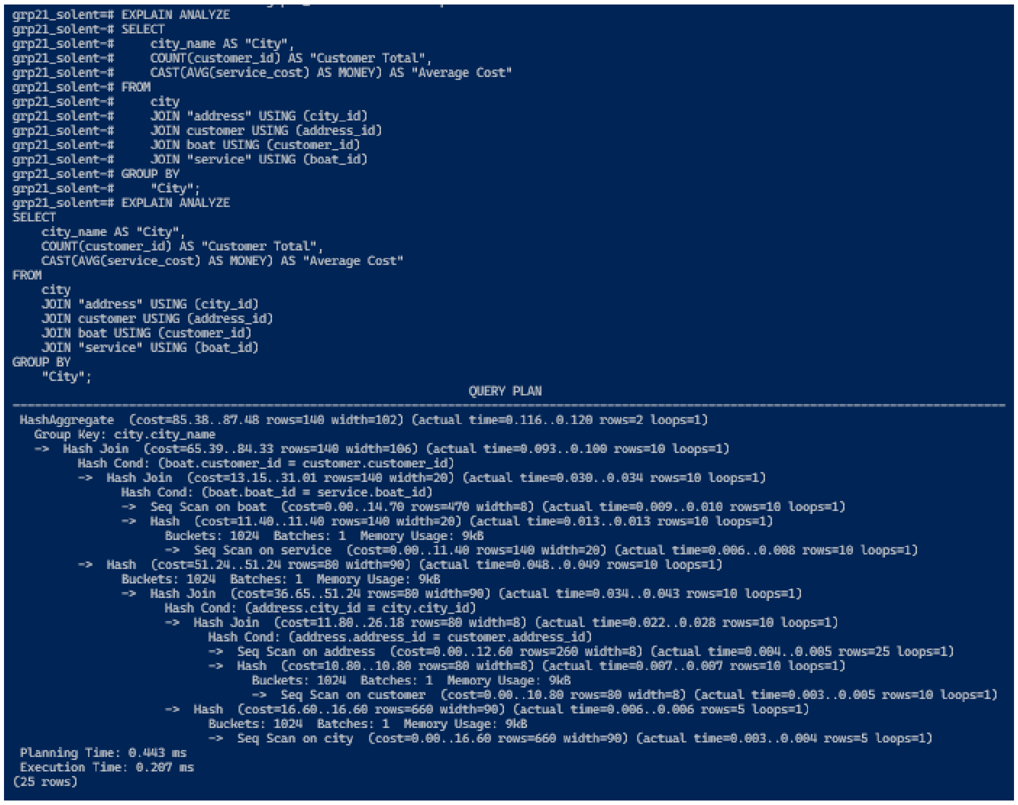
TRANSACTION TABLE	ADDRESS	CUSTOMER	BOAT	MANUFACTU	SERVICE	HISTORY	CITY

													RER															
	C	R	U	D	C	R	U	D	C	R	U	D	C	R	U	D	C	R	U	D	C	R	U	D	C	R	U	D
Finding all the customer details with completed services		X				X				X				X				X				X				X		

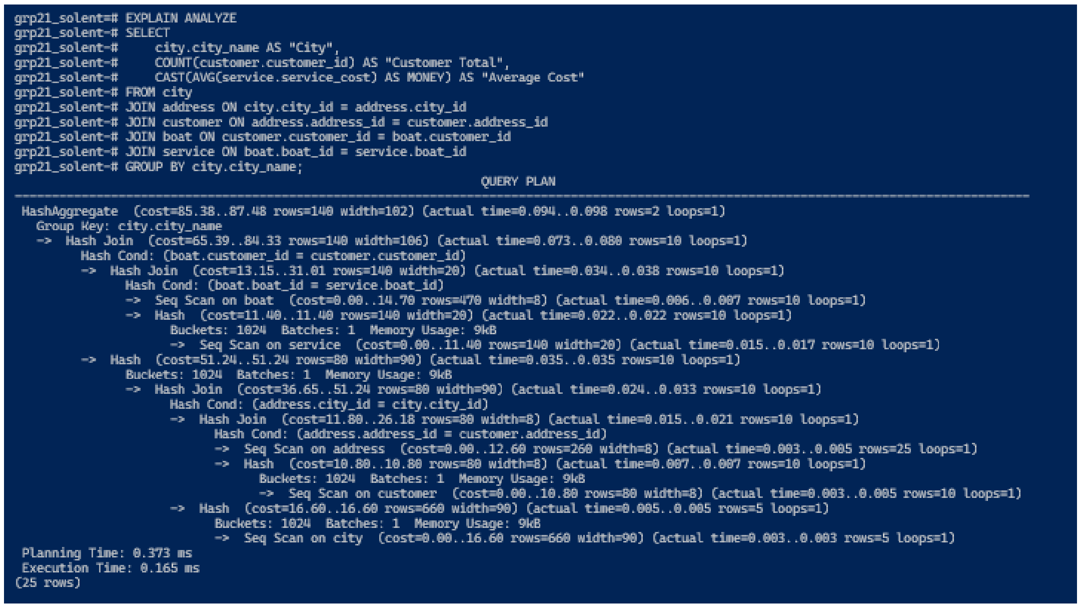
TRANSACTION TABLE	STAFF_YARD				STAFF				STAFF_ROLE				ROLE				YARD			
	C	R	U	D	C	R	U	D	C	R	U	D	C	R	U	D	C	R	U	D
Generating a boatyard report		X				X				X				X				X		
	ADDRESS				CITY				YARD_FACILITIES				STAFF_SERVICE							
	C	R	U	D	C	R	U	D	C	R	U	D	C	R	U	D				
		X				X				X				X						

Queries that are regularly executed on a database system need to be optimised. This is to ensure that queries can be executed more quickly and efficiently meaning the users experience reduced wait times and receive query results in a more timely and efficient manner. Another reason why queries need to be optimised is to reduce cost, reducing the load on hardware resources can translate to cost savings.

A way a query can be optimised is shown in the following query below



```
grp21_solent=# EXPLAIN ANALYZE
grp21_solent-# SELECT
grp21_solent-#     city_name AS "City",
grp21_solent-#     COUNT(customer_id) AS "Customer Total",
grp21_solent-#     CAST(AVG(service_cost) AS MONEY) AS "Average Cost"
grp21_solent-# FROM
grp21_solent-#     city
grp21_solent-#     JOIN "address" USING (city_id)
grp21_solent-#     JOIN customer USING (address_id)
grp21_solent-#     JOIN boat USING (customer_id)
grp21_solent-#     JOIN "service" USING (boat_id)
grp21_solent-# GROUP BY
grp21_solent-#     "City";
grp21_solent=# EXPLAIN ANALYZE
SELECT
    city_name AS "City",
    COUNT(customer_id) AS "Customer Total",
    CAST(AVG(service_cost) AS MONEY) AS "Average Cost"
FROM
    city
    JOIN "address" USING (city_id)
    JOIN customer USING (address_id)
    JOIN boat USING (customer_id)
    JOIN "service" USING (boat_id)
GROUP BY
    "City";
                                                              QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------------------
 HashAggregate  (cost=85.38..87.48 rows=140 width=102) (actual time=0.116..0.120 rows=2 loops=1)
   Group Key: city.city_name
   ->  Hash Join  (cost=65.39..84.33 rows=140 width=106) (actual time=0.093..0.100 rows=10 loops=1)
         Hash Cond: (boat.customer_id = customer.customer_id)
         ->  Hash Join  (cost=13.15..31.01 rows=140 width=20) (actual time=0.030..0.034 rows=10 loops=1)
               Hash Cond: (boat.boat_id = service.boat_id)
               ->  Seq Scan on boat  (cost=0.00..14.70 rows=470 width=8) (actual time=0.009..0.010 rows=10 loops=1)
               ->  Hash  (cost=11.40..11.40 rows=140 width=20) (actual time=0.013..0.013 rows=10 loops=1)
                     Buckets: 1024  Batches: 1  Memory Usage: 9kB
                     ->  Seq Scan on service  (cost=0.00..11.40 rows=140 width=20) (actual time=0.006..0.008 rows=10 loops=1)
         ->  Hash  (cost=51.24..51.24 rows=80 width=90) (actual time=0.048..0.049 rows=10 loops=1)
               Buckets: 1024  Batches: 1  Memory Usage: 9kB
               ->  Hash Join  (cost=36.65..51.24 rows=80 width=90) (actual time=0.034..0.043 rows=10 loops=1)
                     Hash Cond: (address.city_id = city.city_id)
                     ->  Hash Join  (cost=11.80..26.18 rows=80 width=8) (actual time=0.022..0.028 rows=10 loops=1)
                           Hash Cond: (address.address_id = customer.address_id)
                           ->  Seq Scan on address  (cost=0.00..12.60 rows=260 width=8) (actual time=0.004..0.005 rows=25 loops=1)
                           ->  Hash  (cost=10.80..10.80 rows=80 width=8) (actual time=0.007..0.007 rows=10 loops=1)
                                 Buckets: 1024  Batches: 1  Memory Usage: 9kB
                                 ->  Seq Scan on customer  (cost=0.00..10.80 rows=80 width=8) (actual time=0.003..0.005 rows=10 loops=1)
                     ->  Hash  (cost=16.60..16.60 rows=660 width=90) (actual time=0.006..0.006 rows=5 loops=1)
                           Buckets: 1024  Batches: 1  Memory Usage: 9kB
                           ->  Seq Scan on city  (cost=0.00..16.60 rows=660 width=90) (actual time=0.003..0.004 rows=5 loops=1)
 Planning Time: 0.443 ms
 Execution Time: 0.207 ms
(25 rows)
```



```
grp21_solent=# EXPLAIN ANALYZE
grp21_solent-# SELECT
grp21_solent-#     city.city_name AS "City",
grp21_solent-#     COUNT(customer.customer_id) AS "Customer Total",
grp21_solent-#     CAST(AVG(service.service_cost) AS MONEY) AS "Average Cost"
grp21_solent-# FROM city
grp21_solent-# JOIN address ON city.city_id = address.city_id
grp21_solent-# JOIN customer ON address.address_id = customer.address_id
grp21_solent-# JOIN boat ON customer.customer_id = boat.customer_id
grp21_solent-# JOIN service ON boat.boat_id = service.boat_id
grp21_solent-# GROUP BY city.city_name;
                                                              QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------------
 HashAggregate  (cost=85.38..87.48 rows=140 width=102) (actual time=0.094..0.098 rows=2 loops=1)
   Group Key: city.city_name
   ->  Hash Join  (cost=65.39..84.33 rows=140 width=106) (actual time=0.073..0.080 rows=10 loops=1)
         Hash Cond: (boat.customer_id = customer.customer_id)
         ->  Hash Join  (cost=13.15..31.01 rows=140 width=20) (actual time=0.034..0.038 rows=10 loops=1)
               Hash Cond: (boat.boat_id = service.boat_id)
               ->  Seq Scan on boat  (cost=0.00..14.70 rows=470 width=8) (actual time=0.006..0.007 rows=10 loops=1)
               ->  Hash  (cost=11.40..11.40 rows=140 width=20) (actual time=0.022..0.022 rows=10 loops=1)
                     Buckets: 1024  Batches: 1  Memory Usage: 9kB
                     ->  Seq Scan on service  (cost=0.00..11.40 rows=140 width=20) (actual time=0.015..0.017 rows=10 loops=1)
         ->  Hash  (cost=51.24..51.24 rows=80 width=90) (actual time=0.035..0.035 rows=10 loops=1)
               Buckets: 1024  Batches: 1  Memory Usage: 9kB
               ->  Hash Join  (cost=36.65..51.24 rows=80 width=90) (actual time=0.024..0.033 rows=10 loops=1)
                     Hash Cond: (address.city_id = city.city_id)
                     ->  Hash Join  (cost=11.80..26.18 rows=80 width=8) (actual time=0.015..0.021 rows=10 loops=1)
                           Hash Cond: (address.address_id = customer.address_id)
                           ->  Seq Scan on address  (cost=0.00..12.60 rows=260 width=8) (actual time=0.003..0.005 rows=25 loops=1)
                           ->  Hash  (cost=10.80..10.80 rows=80 width=8) (actual time=0.007..0.007 rows=10 loops=1)
                                 Buckets: 1024  Batches: 1  Memory Usage: 9kB
                                 ->  Seq Scan on customer  (cost=0.00..10.80 rows=80 width=8) (actual time=0.003..0.005 rows=10 loops=1)
                     ->  Hash  (cost=16.60..16.60 rows=660 width=90) (actual time=0.005..0.005 rows=5 loops=1)
                           Buckets: 1024  Batches: 1  Memory Usage: 9kB
                           ->  Seq Scan on city  (cost=0.00..16.60 rows=660 width=90) (actual time=0.003..0.003 rows=5 loops=1)
 Planning Time: 0.373 ms
 Execution Time: 0.165 ms
(25 rows)
```

The query has been optimised by joining the different tables with the “ON” syntax instead of the “USING” syntax. Even though the execution time in processing this query is small with a difference of 0.042 milliseconds over time, that small time difference will decrease the cost of running the query through the life cycle of the database system.

Professional, Legal and Ethical Issues

Developing a database system for a real-world client involves several considerations relating to the professional, legal and ethical aspects of that system. Below are the key issues that will need to be taken into account if this system is implemented into real-world use.

Professional Issues

The professional issue when designing and implementing a database system for a client is to make sure the design of the database adheres to industry standards and best practices to ensure compatibility, interoperability and scalability of the system with other systems a client would use in conjunction with the database while following coding standards and naming conventions for consistency and maintainability of the database system for future improvements and maintenance with other Database Administrators that were not directly involved in designing and implementing the first iteration of the system.

Legal Issues

This issue would be one of the most important ones, complying with data protection regulations in the countries in which the company operates. In the United Kingdom under the Data Protection Act 2018, the company would be required to follow the strict rules called “data protection principles” when using the personal data of staff and customers. The company must make sure the data is:

- Used fairly, lawfully and transparently
- Used for specified, explicit purposes
- Used in a way that is adequate, relevant and limited to only what is necessary
- Accurate and where necessary kept up to date
- Kept for no longer than is necessary
- Handled in a way that ensures appropriate security, including protection against unlawful or unauthorised processing, access, loss destruction or damage

The company would have to abide by the rights of the staff and customers under the Act for the information that is stored ensuring the right to:

- Be informed about how that data is being used
- Access personal data
- Have incorrect data updated
- Have data erased
- stop or restrict the processing of the data
- Data portability (allowing the right to get and reuse your data for different services)
- Object to how the data is processed in certain circumstances

These principles are also in place in the European Union under the General Data Protection Regulation (GDPR) meaning the company would have to abide by these regulations if their operations are within countries that are members of the European Union.

Sources

GDPR. (2018). General Data Protection Regulation (GDPR). General Data Protection Regulation (GDPR); Intersoft Consulting. https://gdpr-info.eu/

GOV.UK. (2018). Data Protection Act. Gov.uk; Gov.uk. https://www.gov.uk/data-protection

Ethical Issues

The ethics within creating and managing a database for a client are making sure that the data you collect and store are only the necessary data for the intended purpose and ensuring transparent communications about the way the data within the database is collected. Another ethical implementation that is needed is to make sure the database system is accessible to individuals with diverse needs and abilities and consider the needs of all potential users to avoid excluding specific groups of users.

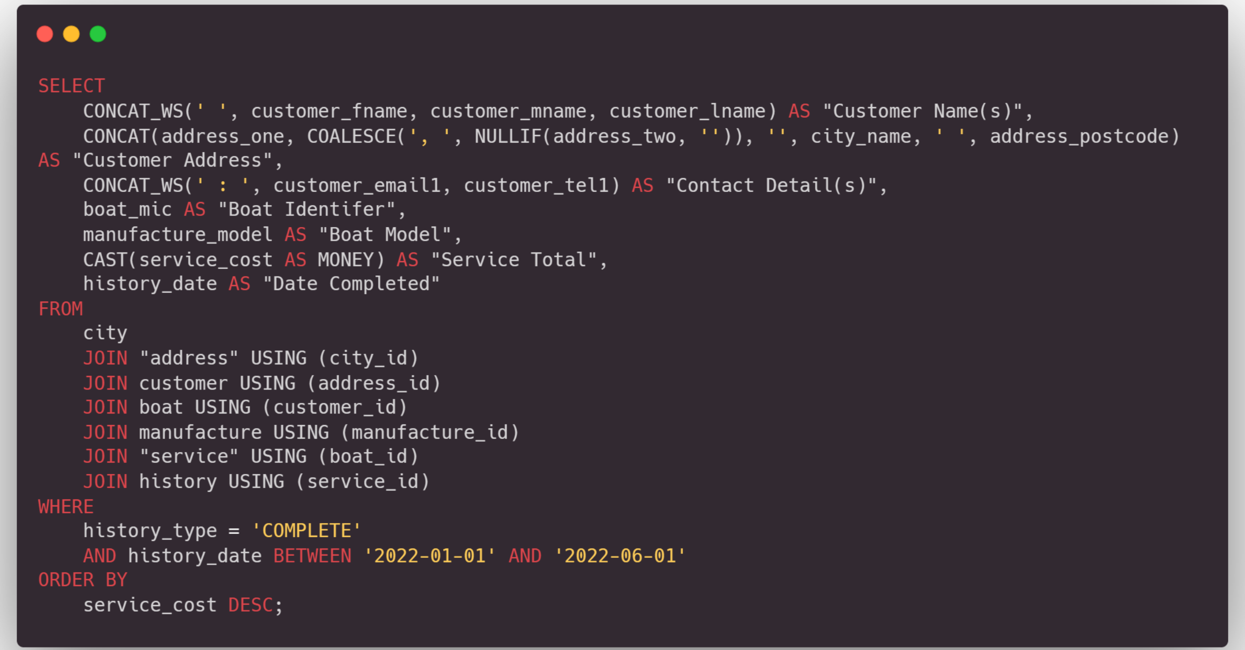
Queries

The queries have been designed to meet the business requirements, specifically to drive revenue in the appropriate direction for Solent, thus the majority of the queries focus on expansion, clients, and cash flow.

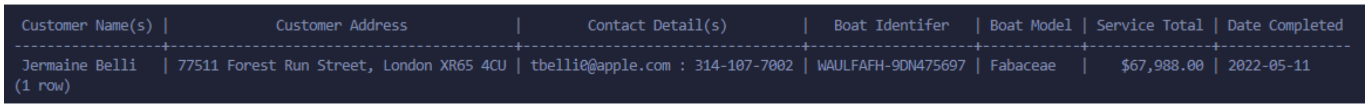
Query One

This query is used to check all customers who have had a service marked as ‘**COMPLETE**’, showing all the detail(s) of the customer i.e. name, and contact information. Whilst also showing the boat information and the total cost of the service.

This query is specifically helpful to see which customers have had the most services/repairs to their boat, however, this query can also be extended to get a report of all the services/repairs that took place during a certain date range period i.e. from the ‘2023-11-21’ to ‘2023-12-08’ for data-analysis, this could be achieved by adding **‘AND history_status BETWEEN xxxx-xx-xx AND xxxx-xx-xx**’.



```sql
SELECT
    CONCAT_WS(' ', customer_fname, customer_mname, customer_lname) AS "Customer Name(s)",
    CONCAT(address_one, COALESCE(', ', NULLIF(address_two, '')), '', city_name, ' ', address_postcode)
AS "Customer Address",
    CONCAT_WS(' : ', customer_email1, customer_tel1) AS "Contact Detail(s)",
    boat_mic AS "Boat Identifer",
    manufacture_model AS "Boat Model",
    CAST(service_cost AS MONEY) AS "Service Total",
    history_date AS "Date Completed"
FROM
    city
    JOIN "address" USING (city_id)
    JOIN customer USING (address_id)
    JOIN boat USING (customer_id)
    JOIN manufacture USING (manufacture_id)
    JOIN "service" USING (boat_id)
    JOIN history USING (service_id)
WHERE
    history_type = 'COMPLETE'
    AND history_date BETWEEN '2022-01-01' AND '2022-06-01'
ORDER BY
    service_cost DESC;
```

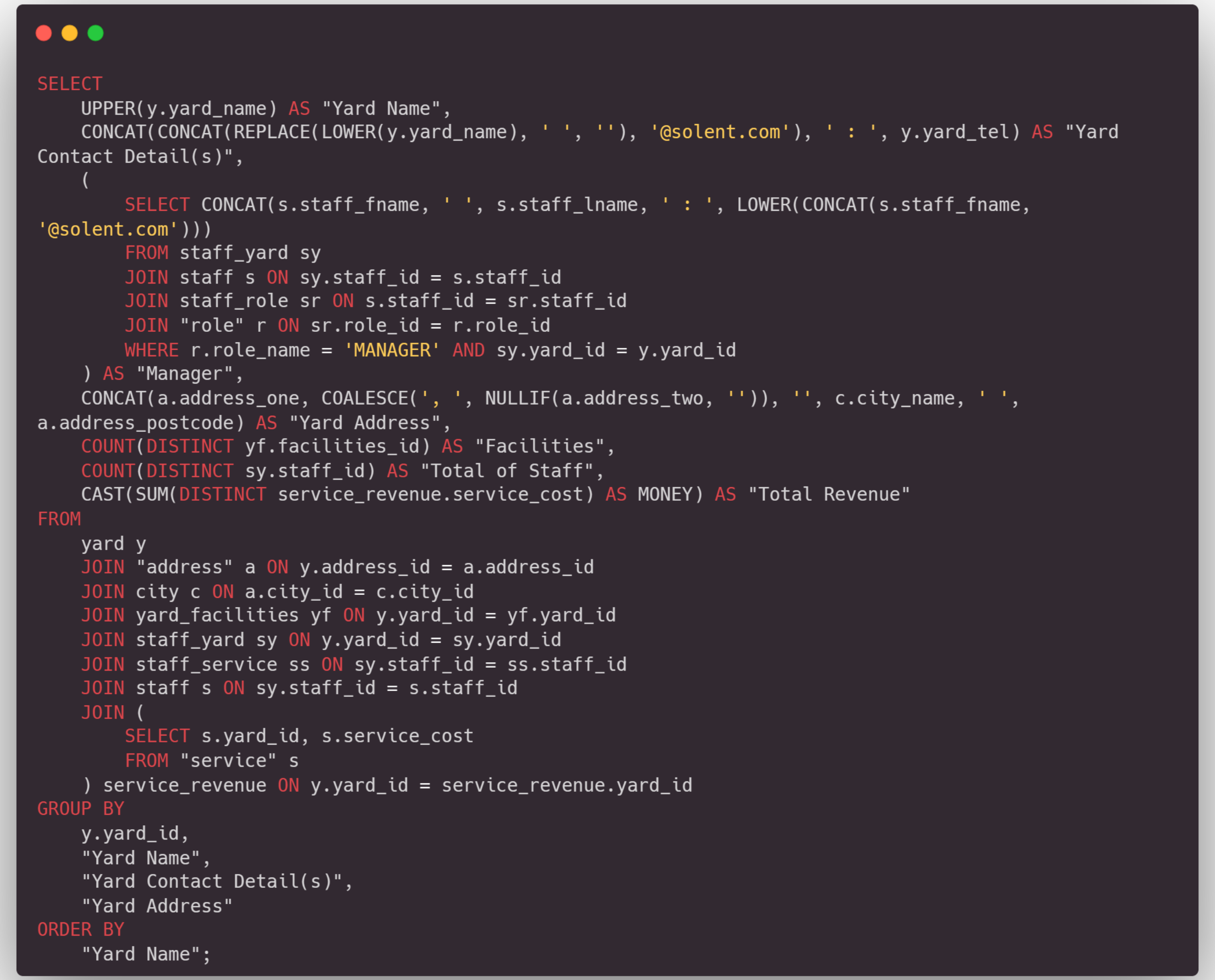


```
 Customer Name(s) |             Customer Address             |          Contact Detail(s)           |   Boat Identifer   | Boat Model | Service Total | Date Completed
------------------+------------------------------------------+--------------------------------------+--------------------+------------+---------------+----------------
 Jermaine Belli   | 77511 Forest Run Street, London XR65 4CU | tbelli0@apple.com : 314-107-7002 | WAULFAFH-9DN475697 | Fabaceae   |    $67,988.00 | 2022-05-11
(1 row)
```

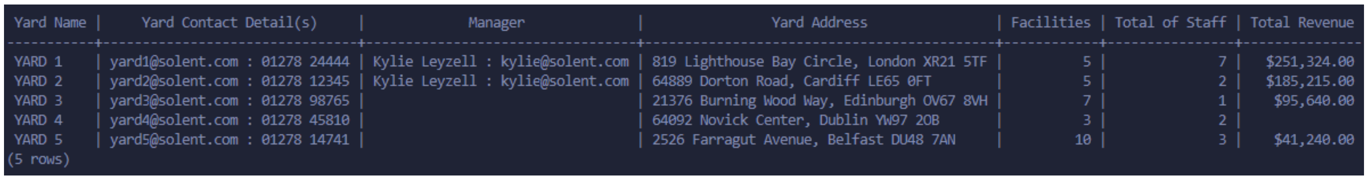
Query Two

This query is used for an overall report of a yard, including the yard name, yard contact information, the manager, address, the number of facilities, the total of staff working at a yard, and finally the total revenue the yard has generated.

This query is especially useful for managing each individual yard, for staff shortages, expansion, etc.



```sql
SELECT
    UPPER(y.yard_name) AS "Yard Name",
    CONCAT(CONCAT(REPLACE(LOWER(y.yard_name), ' ', ''), '@solent.com'), ' : ', y.yard_tel) AS "Yard
Contact Detail(s)",
    (
        SELECT CONCAT(s.staff_fname, ' ', s.staff_lname, ' : ', LOWER(CONCAT(s.staff_fname,
'@solent.com')))
        FROM staff_yard sy
        JOIN staff s ON sy.staff_id = s.staff_id
        JOIN staff_role sr ON s.staff_id = sr.staff_id
        JOIN "role" r ON sr.role_id = r.role_id
        WHERE r.role_name = 'MANAGER' AND sy.yard_id = y.yard_id
    ) AS "Manager",
    CONCAT(a.address_one, COALESCE(', ', NULLIF(a.address_two, '')), '', c.city_name, ' ',
a.address_postcode) AS "Yard Address",
    COUNT(DISTINCT yf.facilities_id) AS "Facilities",
    COUNT(DISTINCT sy.staff_id) AS "Total of Staff",
    CAST(SUM(DISTINCT service_revenue.service_cost) AS MONEY) AS "Total Revenue"
FROM
    yard y
    JOIN "address" a ON y.address_id = a.address_id
    JOIN city c ON a.city_id = c.city_id
    JOIN yard_facilities yf ON y.yard_id = yf.yard_id
    JOIN staff_yard sy ON y.yard_id = sy.yard_id
    JOIN staff_service ss ON sy.staff_id = ss.staff_id
    JOIN staff s ON sy.staff_id = s.staff_id
    JOIN (
        SELECT s.yard_id, s.service_cost
        FROM "service" s
    ) service_revenue ON y.yard_id = service_revenue.yard_id
GROUP BY
    y.yard_id,
    "Yard Name",
    "Yard Contact Detail(s)",
    "Yard Address"
ORDER BY
    "Yard Name";
```

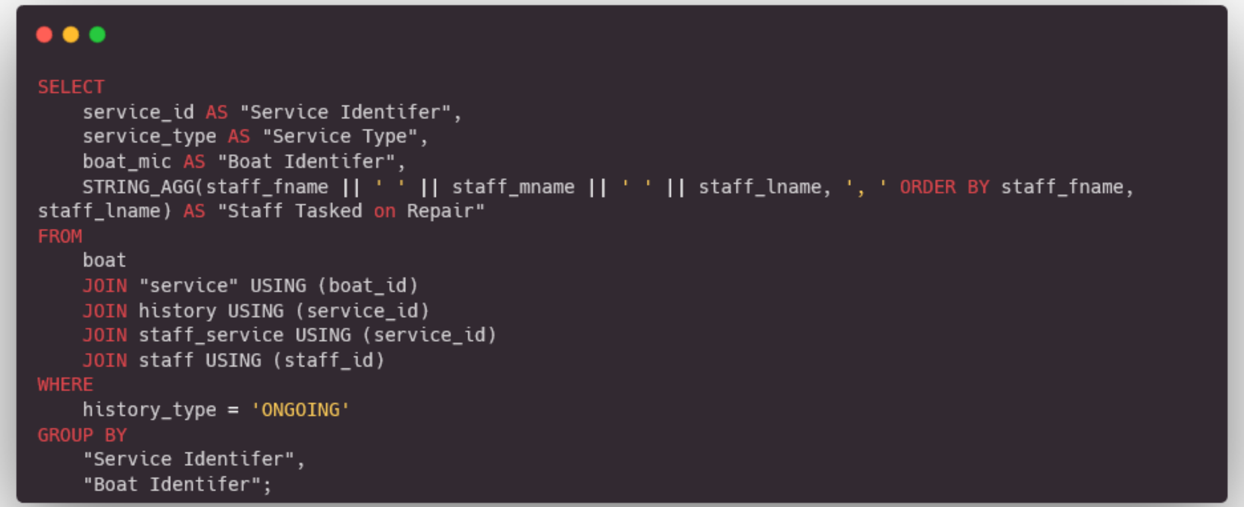


```
 Yard Name |    Yard Contact Detail(s)     |            Manager              |               Yard Address                | Facilities | Total of Staff | Total Revenue
-----------+-------------------------------+---------------------------------+-------------------------------------------+------------+----------------+---------------
 YARD 1    | yard1@solent.com : 01278 24444 | Kylie Leyzell : kylie@solent.com | 819 Lighthouse Bay Circle, London XR21 5TF |          5 |              7 |   $251,324.00
 YARD 2    | yard2@solent.com : 01278 12345 | Kylie Leyzell : kylie@solent.com | 64889 Dorton Road, Cardiff LE65 0FT        |          5 |              2 |   $185,215.00
 YARD 3    | yard3@solent.com : 01278 98765 |                                 | 21376 Burning Wood Way, Edinburgh OV67 8VH |          7 |              1 |    $95,640.00
 YARD 4    | yard4@solent.com : 01278 45810 |                                 | 64092 Novick Center, Dublin YW97 2OB       |          3 |              2 |
 YARD 5    | yard5@solent.com : 01278 14741 |                                 | 2526 Farragut Avenue, Belfast DU48 7AN     |         10 |              3 |    $41,240.00
(5 rows)
```

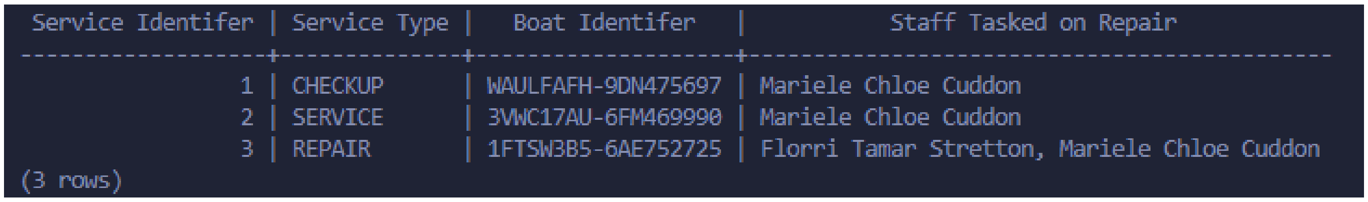
Query Three

This query is used to see all the staff that have been assigned to a service/repair, however, this query could also be extended to see the service within each yard by staff, by selecting the 'yard_id'.

This is especially great for seeing if a service requires more staff.



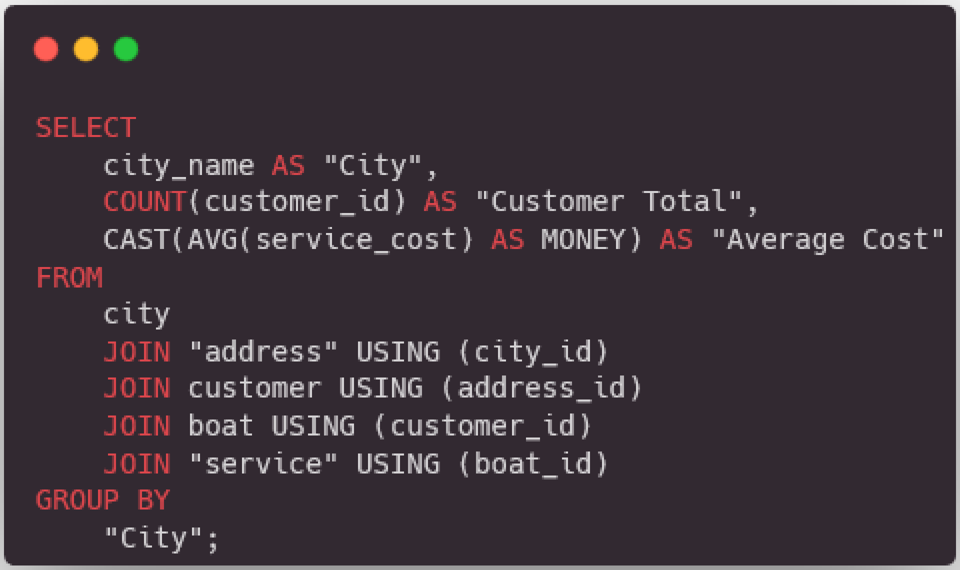
```
SELECT
    service_id AS "Service Identifer",
    service_type AS "Service Type",
    boat_mic AS "Boat Identifer",
    STRING_AGG(staff_fname || ' ' || staff_mname || ' ' || staff_lname, ', ' ORDER BY staff_fname,
staff_lname) AS "Staff Tasked on Repair"
FROM
    boat
    JOIN "service" USING (boat_id)
    JOIN history USING (service_id)
    JOIN staff_service USING (service_id)
    JOIN staff USING (staff_id)
WHERE
    history_type = 'ONGOING'
GROUP BY
    "Service Identifer",
    "Boat Identifer";
```



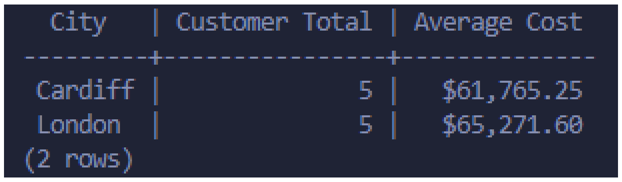
```
 Service Identifer | Service Type |   Boat Identifer   |            Staff Tasked on Repair
-------------------+--------------+--------------------+---------------------------------------------
                 1 | CHECKUP      | WAULFAFH-9DN475697 | Mariele Chloe Cuddon
                 2 | SERVICE      | 3VWC17AU-6FM469990 | Mariele Chloe Cuddon
                 3 | REPAIR       | 1FTSW3B5-6AE752725 | Florri Tamar Stretton, Mariele Chloe Cuddon
(3 rows)
```

Query Four

This query is used to see the customers registered within the database and the city they live in, this is then used to find the average cost of customers from these cities, which can be used for expansion purposes for solent boats.



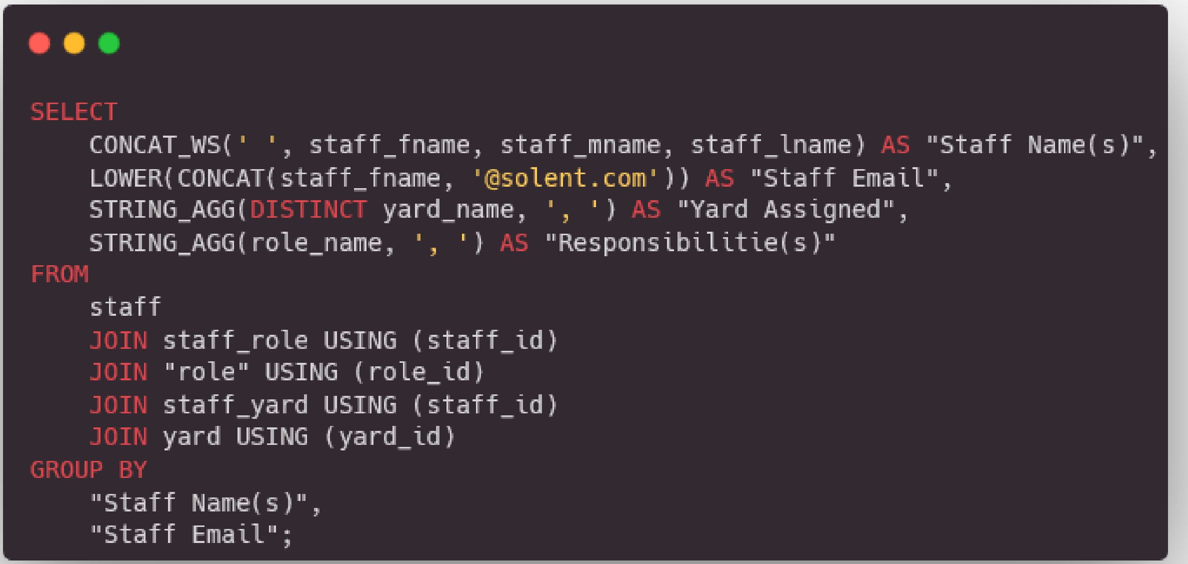
```
SELECT
    city_name AS "City",
    COUNT(customer_id) AS "Customer Total",
    CAST(AVG(service_cost) AS MONEY) AS "Average Cost"
FROM
    city
    JOIN "address" USING (city_id)
    JOIN customer USING (address_id)
    JOIN boat USING (customer_id)
    JOIN "service" USING (boat_id)
GROUP BY
    "City";
```



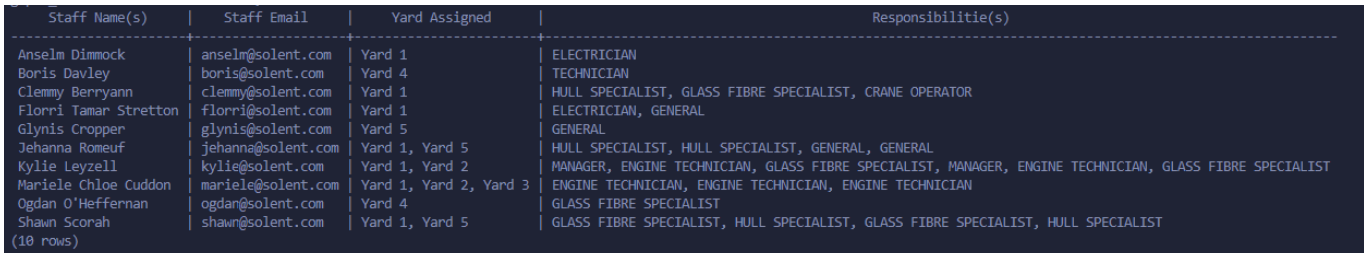
```
  City   | Customer Total | Average Cost
---------+----------------+--------------
 Cardiff |              5 |   $61,765.25
 London  |              5 |   $65,271.60
(2 rows)
```

Query Five

This query is used to see the staff responsibilities and the yard they are assigned to, this is essentially a more in-depth view of each yard from query two, like query two an administrator can move staff to another yard for shortages, etc.



```sql
SELECT
    CONCAT_WS(' ', staff_fname, staff_mname, staff_lname) AS "Staff Name(s)",
    LOWER(CONCAT(staff_fname, '@solent.com')) AS "Staff Email",
    STRING_AGG(DISTINCT yard_name, ', ') AS "Yard Assigned",
    STRING_AGG(role_name, ', ') AS "Responsibilitie(s)"
FROM
    staff
    JOIN staff_role USING (staff_id)
    JOIN "role" USING (role_id)
    JOIN staff_yard USING (staff_id)
    JOIN yard USING (yard_id)
GROUP BY
    "Staff Name(s)",
    "Staff Email";
```



```
     Staff Name(s)     |    Staff Email     |     Yard Assigned      |                                  Responsibilitie(s)
-----------------------+--------------------+------------------------+--------------------------------------------------------------------------------------------------
 Anselm Dimmock        | anselm@solent.com  | Yard 1                 | ELECTRICIAN
 Boris Davley          | boris@solent.com   | Yard 4                 | TECHNICIAN
 Clemmy Berryann       | clemmy@solent.com  | Yard 1                 | HULL SPECIALIST, GLASS FIBRE SPECIALIST, CRANE OPERATOR
 Florri Tamar Stretton | florri@solent.com  | Yard 1                 | ELECTRICIAN, GENERAL
 Glynis Cropper        | glynis@solent.com  | Yard 5                 | GENERAL
 Jehanna Romeuf        | jehanna@solent.com | Yard 1, Yard 5         | HULL SPECIALIST, HULL SPECIALIST, GENERAL, GENERAL
 Kylie Leyzell         | kylie@solent.com   | Yard 1, Yard 2         | MANAGER, ENGINE TECHNICIAN, GLASS FIBRE SPECIALIST, MANAGER, ENGINE TECHNICIAN, GLASS FIBRE SPECIALIST
 Mariele Chloe Cuddon  | mariele@solent.com | Yard 1, Yard 2, Yard 3 | ENGINE TECHNICIAN, ENGINE TECHNICIAN, ENGINE TECHNICIAN
 Ogdan O'Heffernan     | ogdan@solent.com   | Yard 4                 | GLASS FIBRE SPECIALIST
 Shawn Scorah          | shawn@solent.com   | Yard 1, Yard 5         | GLASS FIBRE SPECIALIST, HULL SPECIALIST, GLASS FIBRE SPECIALIST, HULL SPECIALIST
(10 rows)
```