**Cache Crusaders**

**Group 1**

Project Phase 3

Physical Design

(CMPG311)

**Group Members:**

[40897745 – J. Smit] – (Group Leader)

[42638844 – F. van Vuuren]

[24904635 – F. Burger]

[43151906 – R. van Rooyen]

Submission date: 08/04/2024

Contents

[**Project Phase 1 – Database Initial Study** 3](file:///C:\Users\Fanie\Documents\GitHub\Cache-Crusaders\CMPG%20311%20Assignment%20Phase%203.docx#_Toc165042512)

[1. Members of the group 3](file:///C:\Users\Fanie\Documents\GitHub\Cache-Crusaders\CMPG%20311%20Assignment%20Phase%203.docx#_Toc165042513)

[2. Analyse company situation 4](file:///C:\Users\Fanie\Documents\GitHub\Cache-Crusaders\CMPG%20311%20Assignment%20Phase%203.docx#_Toc165042514)

[Company objectives 4](file:///C:\Users\Fanie\Documents\GitHub\Cache-Crusaders\CMPG%20311%20Assignment%20Phase%203.docx#_Toc165042515)

[Company operations 4](file:///C:\Users\Fanie\Documents\GitHub\Cache-Crusaders\CMPG%20311%20Assignment%20Phase%203.docx#_Toc165042516)

[Company Organisational Structure 5](file:///C:\Users\Fanie\Documents\GitHub\Cache-Crusaders\CMPG%20311%20Assignment%20Phase%203.docx#_Toc165042517)

[3. Define problems and constraints 6](file:///C:\Users\Fanie\Documents\GitHub\Cache-Crusaders\CMPG%20311%20Assignment%20Phase%203.docx#_Toc165042518)

[Problems 6](file:///C:\Users\Fanie\Documents\GitHub\Cache-Crusaders\CMPG%20311%20Assignment%20Phase%203.docx#_Toc165042519)

[Constraints 6](file:///C:\Users\Fanie\Documents\GitHub\Cache-Crusaders\CMPG%20311%20Assignment%20Phase%203.docx#_Toc165042520)

[4. Database system specification 7](file:///C:\Users\Fanie\Documents\GitHub\Cache-Crusaders\CMPG%20311%20Assignment%20Phase%203.docx#_Toc165042521)

[Objectives To Solve Problems Identified 7](file:///C:\Users\Fanie\Documents\GitHub\Cache-Crusaders\CMPG%20311%20Assignment%20Phase%203.docx#_Toc165042522)

[Information Required From Database 9](file:///C:\Users\Fanie\Documents\GitHub\Cache-Crusaders\CMPG%20311%20Assignment%20Phase%203.docx#_Toc165042523)

[Scope 9](file:///C:\Users\Fanie\Documents\GitHub\Cache-Crusaders\CMPG%20311%20Assignment%20Phase%203.docx#_Toc165042524)

[Boundaries 9](file:///C:\Users\Fanie\Documents\GitHub\Cache-Crusaders\CMPG%20311%20Assignment%20Phase%203.docx#_Toc165042525)

[**Project Phase 2 – Database Design** 10](file:///C:\Users\Fanie\Documents\GitHub\Cache-Crusaders\CMPG%20311%20Assignment%20Phase%203.docx#_Toc165042526)

[1. Conceptual Design 10](file:///C:\Users\Fanie\Documents\GitHub\Cache-Crusaders\CMPG%20311%20Assignment%20Phase%203.docx#_Toc165042527)

[Business Rules 10](file:///C:\Users\Fanie\Documents\GitHub\Cache-Crusaders\CMPG%20311%20Assignment%20Phase%203.docx#_Toc165042528)

[Entity Relationship Diagram 11](file:///C:\Users\Fanie\Documents\GitHub\Cache-Crusaders\CMPG%20311%20Assignment%20Phase%203.docx#_Toc165042529)

[2. Logical Design 12](file:///C:\Users\Fanie\Documents\GitHub\Cache-Crusaders\CMPG%20311%20Assignment%20Phase%203.docx#_Toc165042530)

# **Project Phase 1 – Database Initial Study**

## 1. Members of the group



Renier van Rooyen (43151906)



Franco Burger (24904635)



Jaundre Smit (40897745)



Fanie van Vuuren (42638844)

## 2. Analyse company situation

### Company objectives

The main objective and mission of the company is to provide a computer building and repair service that is customisable, reliable and uses high quality components. This means that the database must be able to handle complex queries related to the building and repair of custom computers. It should be able to store and manage things such as orders, part availability, user data, repair requests and technician data.

The objectives can be broken down further into the following key points:

* **Customer satisfaction**: The most important thing is always customer satisfaction. This is achieved through the use of high-quality parts but also through a database system that operates smoothly.
* **Technological innovation**: To provide the best quality service the company needs to stay on top of any new advancements in technology. Being up to date on the newest technology leads to smoother systems operation and an overall better customer experience.
* **Scalability**: The database in particular needs to be able to handle the increased load as the company grows. The last thing the company wants is to have to redesign the entire system as it gets bigger.
* **Adaptability**: The database also needs to be able to adapt to changing business requirements and needs to be able to incorporate new technologies as they arise.

### Company operations

The company operates in an online environment where customers place orders or request repairs. They need to be able to see available parts when choosing the specifications for their new computer. The company’s operations would involve several different database operations. These might include the following:

* **Order management**: For daily operation, the database needs to store details pertaining to orders placed by customers. In order for the operation to run smoothly, information on the status and specifics of orders needs to be readily available. The database needs to be able to handle new orders, update existing ones and retrieve order details.
* **Inventory management**: The system used by the company would also need to provide details on the different parts they have available. This would be used to guide technicians on what parts they can use to repair a computer, but it is also crucial for users looking to choose new parts for their computer.
* **Repair management**: The system needs to store data on the status of repairs and needs to give feedback on things such as parts required and the total cost of the repair. This is critical to ensure the repair operation can function as intended.

### Company Organisational Structure

The organisational structure of the company determines which employees have access to what information. It reflects the user roles and permissions in the database.

* **Customer Service Representatives**: They would have access to data pertaining to orders and repairs tables to manage customer issues and questions. They would need to perform operations such as viewing order and repair details, updating order status, and responding to customer complaints. They would also need to coordinate with other departments such as salespeople to resolve customer issues.
* **Technicians**: They would have access to inventory as well as certain elements of data from orders and repair tables to build and repair computers. For them to sign off on a repair or report that an order is finished, they would need to be able to interface with the database and update some of the data in these two tables.
* **Sales and Marketing Personnel**: These personnel would need access to data from both the sales and repairs data tables. They would need this data to carry out their task which includes creating sales reports, assessing sales patterns, and organising market campaigns. For them to function optimally, they would need to work together with people from the customer service team.
* **IT personnel**: The IT team would need full access to all parts of the database. They need to ensure the database is functioning as intended and that there are no security issues. They are also the ones that need to install upgraded and make sure the database is optimised.

## 3. Define problems and constraints

### Problems

* Customers might want to change parts of their PC build while waiting for their order to be completed.
* Customers may want a PC build that’s more powerful than their budget allows for.
* During the repair process more problems might come to light than the customer, or the initial inspection of the PC, found and was quoted to resolve.
* Any kind of delay in parts ordering or update in already pre-booked parts could cause a shortage of specific parts quoted for a build or repair.
* Human error is always a concern as something might get damaged while being worked on.
* Customer data could be lost due to faulty code, and it causes a delay in service for repeat customers.
* Data security is always a problem as the customers’ information is private to them and the business.
* Double orders can arise if the system update takes too long or doesn’t update properly and it can cause the company to accidentally order inventory twice or charge a customer twice.

### Constraints

* We can only build PCs with parts we have available.
* A PC part can only be so damaged before it needs to be replaced and repair is impossible.
* Each build and repair will take time to complete and there are only so many work hours in a day.
* Customers that have a specific budget can only afford that budget and we need to remain within the given budget.
* The prices of parts are set so maintaining a budget will overrule meeting build specifications. We can only match the build desired as much as the budget allows.
* A repair needs to remain within the quota even if additional unseen repairs arise during the repair process. It’s our responsibility to find all issues with the PC during the inspection phase of repairs.

## 4. Database system specification

### Objectives To Solve Problems Identified

1. **Database Management Objective:**
   * Develop a robust database system capable of handling complex queries efficiently.
   * Implement mechanisms for seamless storage and management of orders, part availability, user data, repair requests, and technician data.
   * Ensure data integrity and reliability to support the company's mission of providing customisable and reliable computer building and repair services.
2. **Customer Satisfaction Objective:**
   * Enhance customer satisfaction by improving the responsiveness and usability of the database system.
   * Incorporate customer feedback mechanisms within the system to continuously improve service quality.
   * Implement measures to ensure timely order processing, accurate order tracking, and effective communication with customers regarding their orders and repairs.
3. **Technological Innovation Objective:**
   * Establish a process for regularly assessing and adopting new technological advancements relevant to the company's operations.
   * Integrate emerging technologies into the database system to enhance performance, security, and user experience.
   * Foster a culture of innovation within the organization to encourage proactive exploration and implementation of cutting-edge solutions.
4. **Scalability Objective:**
   * Design the database architecture with scalability in mind, allowing it to accommodate increasing data volumes and user loads.
   * Implement scalable infrastructure and database optimization techniques to support the company's growth trajectory.
   * Conduct periodic scalability assessments and perform necessary upgrades or adjustments to maintain system performance as the company expands.
5. **Adaptability Objective:**
   * Build flexibility into the database system to easily adapt to evolving business requirements and technological advancements.
   * Establish a framework for regular updates and enhancements to ensure the system remains aligned with changing needs.
   * Foster collaboration between IT and business stakeholders to identify and implement necessary adaptations proactively.
6. **Access Control and Security Objective:**
   * Implement robust access control mechanisms to restrict database access based on employee roles and responsibilities.
   * Enhance data security measures to protect sensitive information from unauthorised access or breaches.
   * Conduct regular security audits and implement updates or patches to address potential vulnerabilities and ensure compliance with data protection regulations.
7. **Interdepartmental Coordination Objective:**
   * Facilitate seamless communication and collaboration between different departments through the database system.
   * Integrate functionalities that enable cross-departmental sharing of relevant data and insights.
   * Implement workflows and notifications to streamline interdepartmental processes and resolve issues promptly.
8. **IT Management Objective:**
   * Establish clear responsibilities and procedures for IT personnel to monitor, maintain, and optimise the database system.
   * Provide ongoing training and support to IT staff to ensure proficiency in database management and troubleshooting.
   * Implement robust backup and recovery protocols to safeguard data integrity and minimise downtime in case of system failures or disruptions.

### Information Required From Database

* Customer information
* Order information
* Inventory information
* Repair information
* Employee information
* Sales data
* Technical information

### Scope

* **Order management**: Add new build orders and update the status of orders as they change.
* **Inventory management**: Add restocked items, monitor stock levels, and reduce the stock of an item if it is ordered.
* **Repair management**: Add new repair orders and update the progress of repairs in progress.
* **Staff management**: Add, remove, and update the details of staff members and assign relevant staff to specific orders or repairs.
* **Customer Management**

### Boundaries

* **Storage**: Limited amount of each part that can be in stock at a time and limited storage space for repairs.
* **Operational Capacity**: A set limited number of orders and repairs that each personnel member can work on at a time.
* **Budget**: Small-scale company with a limited budget.
* **Device Compatibility**: The database needs to be accessible to all relevant staff members on all main operating systems such as Windows 11, macOS, Android and IOS.

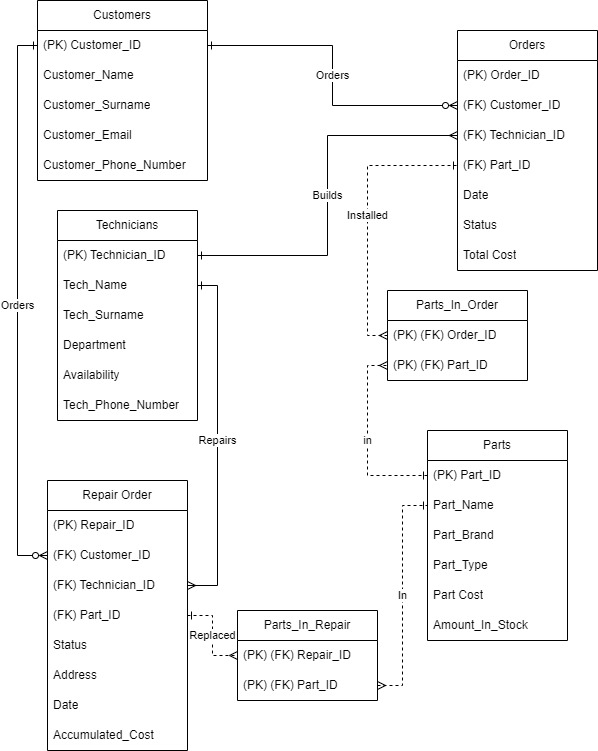
# **Project Phase 2 – Database Design**

## Conceptual Design

### Business Rules

* A customer can request multiple repairs.
* A customer can request multiple builds.
* A customer can request many parts.
* A repair can only be requested by one customer.
* A build can only be requested by one customer.
* Parts can only be ordered by one customer.
* A build can contain many parts from the inventory(denoted by stock).
* A part can only go to one build from the inventory.
* A repair can contain none-to-many parts from the inventory.
* One staff can work on one repair at a time.
* One staff can work on one build at a time.
* Many builds can be ordered.
* Many parts can be ordered.
* Many repairs can be ordered.
* One order contains one build.
* One order contains one list of ordered parts.
* One order contains one repair.

### Entity Relationship Diagram



## Logical Design

* **Customers(Customer\_ID**(PK), Customer\_Name, Customer\_Surname, Customer\_Email, Customer\_Phone\_Number**)**
* **Technicians(Technician\_ID(**PK**)**, Tech\_Name, Tech\_Surname, Department, Availibility**)**
* **Parts(Part\_ID(**PK**)**, Part\_Name, Part\_Brand, Part\_Type, Part\_cost, Amount\_In\_Stock**)**
* **Orders(Order\_ID(**PK**)**, Customer\_ID(FK), Technician\_ID(FK), Parts\_ID(FK), Date, Status, Total\_Cost**)**
* A diagram of a data flow

  Description automatically generated with medium confidence**Repairs(Repair\_ID(**PK**)**, Customer\_ID(FK), Technician\_ID(FK), Parts\_ID(FK), Status, Adress, Date, Accumulated\_Cost**)**

# **Project Phase 3 – Physical Design**

## 1. DATABASE OBJECTS

Create the database objects in Oracle SQL developer for the scenario and ERD provided in Project phase 2. and provide the SQL statements used to create the database objects as well as a brief description thereof.

* 1. **TABLES**

Creates all the tables and sequences needed to make the database work

--------------------------------------------------------

-- Drop all Tables

--------------------------------------------------------

DROP TABLE ORDERS CASCADE CONSTRAINTS;

DROP TABLE ADDRESS CASCADE CONSTRAINTS;

DROP TABLE CUSTOMERS CASCADE CONSTRAINTS;

DROP TABLE PARTS CASCADE CONSTRAINTS;

DROP TABLE PARTS\_IN\_ORDER CASCADE CONSTRAINTS;

DROP TABLE PARTS\_IN\_REPAIR CASCADE CONSTRAINTS;

DROP TABLE PROVINCES CASCADE CONSTRAINTS;

DROP TABLE REPAIRS CASCADE CONSTRAINTS;

DROP TABLE STATUS CASCADE CONSTRAINTS;

DROP TABLE TECHNICIANS CASCADE CONSTRAINTS;

DROP TABLE DEPARTMENTS CASCADE CONSTRAINTS;

---------------------------------------------------------

--Drop all Sequences

---------------------------------------------------------

DROP SEQUENCE ORDER\_ID\_value;

DROP SEQUENCE ADDRESS\_ID\_value;

DROP SEQUENCE CUSTOMER\_ID\_value;

DROP SEQUENCE PART\_ID\_value;

DROP SEQUENCE PIO\_ID\_value;

DROP SEQUENCE PIR\_ID\_value;

DROP SEQUENCE PROVINCE\_ID\_value;

DROP SEQUENCE REPAIR\_ID\_value;

DROP SEQUENCE STATUS\_ID\_value;

DROP SEQUENCE TECHNICIAN\_ID\_value;

DROP SEQUENCE DEPARTMENT\_ID\_value;

--------------------------------------------------------

-- Create all Tables

--------------------------------------------------------

--------------------------------------------------------

-- DDL for Table ORDERS

--------------------------------------------------------

CREATE TABLE ORDERS

(

ORDER\_ID NUMBER NOT NULL,

CUSTOMER\_ID NUMBER,

TECHNICIAN\_ID NUMBER,

ORDER\_DATE DATE,

ORDER\_STATUS NUMBER,

TOTAL\_COST NUMBER(10, 2)

);

--------------------------------------------------------

-- DDL for Table ADDRESS

--------------------------------------------------------

CREATE TABLE ADDRESS

(

ADDRESS\_ID NUMBER NOT NULL PRIMARY KEY,

STREET\_NAME VARCHAR2(40 BYTE),

STREET\_NUMBER NUMBER,

CITY VARCHAR2(40 BYTE),

PROVINCE\_ID NUMBER

);

COMMENT ON COLUMN ADDRESS.PROVINCE\_ID IS 'Links to Provinces table';

--------------------------------------------------------

-- DDL for Table CUSTOMERS

--------------------------------------------------------

CREATE TABLE CUSTOMERS

(

CUSTOMER\_ID NUMBER NOT NULL PRIMARY KEY

, CUSTOMER\_EMAIL VARCHAR2(50 BYTE)

, CUSTOMER\_PHONE\_NUMBER VARCHAR2(15 BYTE)

, CUSTOMER\_NAME VARCHAR2(50 BYTE)

, CUSTOMER\_SURNAME VARCHAR2(50 BYTE)

, ADDRESS\_ID NUMBER NOT NULL

);

COMMENT ON COLUMN CUSTOMERS.ADDRESS\_ID IS 'Links to Address Table';

--------------------------------------------------------

-- DDL for Table PARTS

--------------------------------------------------------

CREATE TABLE PARTS

(

PART\_ID NUMBER NOT NULL PRIMARY KEY,

PART\_NAME VARCHAR2(35 BYTE),

PART\_BRAND VARCHAR2(35 BYTE),

PART\_TYPE VARCHAR2(35 BYTE),

PART\_COST FLOAT(2),

PARTS\_IN\_STOCK NUMBER

);

--------------------------------------------------------

-- DDL for Table PARTS\_IN\_ORDER

--------------------------------------------------------

CREATE TABLE PARTS\_IN\_ORDER

(

PIO\_ID NUMBER PRIMARY KEY,

ORDER\_ID NUMBER,

PARTS\_ID NUMBER

);

--------------------------------------------------------

-- DDL for Table PARTS\_IN\_REPAIR

--------------------------------------------------------

CREATE TABLE PARTS\_IN\_REPAIR

(

PIR\_ID NUMBER PRIMARY KEY,

REPAIR\_ID NUMBER,

PART\_ID NUMBER

);

--------------------------------------------------------

-- DDL for Table PROVINCES

--------------------------------------------------------

CREATE TABLE PROVINCES

(

PROVINCE\_ID NUMBER NOT NULL PRIMARY KEY,

PROVINCE\_NAME VARCHAR2(30 BYTE)

);

--------------------------------------------------------

-- DDL for Table REPAIR\_ORDER

--------------------------------------------------------

CREATE TABLE REPAIRS

(

REPAIR\_ID NUMBER NOT NULL PRIMARY KEY,

CUSTOMER\_ID NUMBER NOT NULL,

TECHNICIAN\_ID NUMBER NOT NULL,

STATUS\_ID NUMBER NOT NULL,

ADDRESS\_ID NUMBER,

REPAIR\_DATE DATE,

ACCUMULATED\_COST FLOAT(126)

);

--------------------------------------------------------

-- DDL for Table STATUS

--------------------------------------------------------

CREATE TABLE STATUS

(

STATUS\_ID NUMBER NOT NULL PRIMARY KEY,

STATUS\_NAME VARCHAR2(35 BYTE)

);

COMMENT ON COLUMN STATUS.STATUS\_NAME IS 'In Progress/Todo/In Transit/Completed/ETC';

--------------------------------------------------------

-- DDL for Table TECHNICIANS

--------------------------------------------------------

CREATE TABLE TECHNICIANS

(

TECHNICIAN\_ID NUMBER NOT NULL PRIMARY KEY,

AVAILABILITY NUMBER,

TECH\_SURNAME VARCHAR2(35 BYTE),

TECH\_PHONE\_NUMBER VARCHAR2(10 BYTE),

TECH\_NAME VARCHAR2(35 BYTE),

DEPARTMENT\_ID NUMBER

);

COMMENT ON COLUMN TECHNICIANS.AVAILABILITY IS 'Each department role has a set max jobs, this is each user''s current jobs count';

COMMENT ON COLUMN TECHNICIANS.DEPARTMENT\_ID IS 'The technicians department';

--------------------------------------------------------

-- DDL for Table DEPARTMENTS

--------------------------------------------------------

CREATE TABLE DEPARTMENTS

(

DEPARTMENT\_ID INT NOT NULL PRIMARY KEY

, DEPARTMENT\_NAME VARCHAR2(40)

, DEPARTMENT\_WORK\_CAPACITY INTEGER

);

--

--------------------------------------------------------

-- Add contraints

--------------------------------------------------------

-- Orders

ALTER TABLE ORDERS

ADD CONSTRAINT PK\_ORDERS PRIMARY KEY (ORDER\_ID);

ALTER TABLE ORDERS

ADD CONSTRAINT ORDERS\_CUST\_FK1 FOREIGN KEY (CUSTOMER\_ID) REFERENCES CUSTOMERS(CUSTOMER\_ID) ON DELETE CASCADE;

ALTER TABLE ORDERS

ADD CONSTRAINT ORDERS\_TECHN\_FK2 FOREIGN KEY (TECHNICIAN\_ID) REFERENCES TECHNICIANS(TECHNICIAN\_ID) ON DELETE CASCADE;

--

-- Customers

ALTER TABLE CUSTOMERS

ADD CONSTRAINT CUSTOMER\_ADDRESS\_FK FOREIGN KEY (ADDRESS\_ID) REFERENCES ADDRESS(ADDRESS\_ID);

--

-- Technicians

ALTER TABLE TECHNICIANS

ADD CONSTRAINT TECH\_DEPT\_FK FOREIGN KEY (DEPARTMENT\_ID) REFERENCES DEPARTMENTS(DEPARTMENT\_ID);

--

-- Repairs

ALTER TABLE REPAIRS

ADD CONSTRAINT REP\_CUST\_FK1 FOREIGN KEY (CUSTOMER\_ID) REFERENCES CUSTOMERS(CUSTOMER\_ID) ON DELETE CASCADE;

ALTER TABLE REPAIRS

ADD CONSTRAINT REP\_TECHN\_FK2 FOREIGN KEY (TECHNICIAN\_ID) REFERENCES TECHNICIANS(TECHNICIAN\_ID) ON DELETE CASCADE;

ALTER TABLE REPAIRS

ADD CONSTRAINT REP\_STATUS\_FK4 FOREIGN KEY (STATUS\_ID) REFERENCES STATUS(STATUS\_ID) ON DELETE CASCADE;

--

--PARTS\_IN\_REPAIRS

ALTER TABLE PARTS\_IN\_REPAIR

ADD CONSTRAINT PIR\_PARTS\_FK2 FOREIGN KEY (PART\_ID)REFERENCES PARTS(PART\_ID);

ALTER TABLE PARTS\_IN\_REPAIR

ADD CONSTRAINT PIR\_REP\_FK1 FOREIGN KEY (REPAIR\_ID) REFERENCES ORDERS(ORDER\_ID);

--

--PARTS\_IN\_ORDER

ALTER TABLE PARTS\_IN\_ORDER

ADD CONSTRAINT PIO\_ORDERS\_FK1 FOREIGN KEY(ORDER\_ID)REFERENCES ORDERS(ORDER\_ID);

ALTER TABLE PARTS\_IN\_ORDER

ADD CONSTRAINT PIO\_ORDERS\_FK2 FOREIGN KEY (PARTS\_ID)REFERENCES PARTS(PART\_ID);

--

* 1. **INDEXES**

**Creates all the necessary indexes**

CREATE INDEX TECH\_NAMESURNAME\_IDX ON TECHNICIANS (TECH\_NAME, TECH\_SURNAME);

CREATE INDEX PARTS\_COSTTYPE\_IDX ON PARTS (PART\_COST, PART\_TYPE);

CREATE INDEX CUSTOMERS\_FULLNAME\_IDX ON CUSTOMERS (CUSTOMER\_NAME, CUSTOMER\_SURNAME);

CREATE INDEX ADDRESS\_STREET\_IDX ON ADDRESS (STREET\_NAME, STREET\_NUMBER);

CREATE INDEX DEPART\_CAPACITY\_IDX ON DEPARTMENTS (DEPARTMENT\_WORK\_CAPACITY DESC);

CREATE INDEX ORDER\_STATUSDATE\_IDX ON ORDERS (ORDER\_STATUS, ORDER\_DATE);

CREATE INDEX PROVINCE\_NAME\_IDX ON PROVINCES (PROVINCE\_NAME);

CREATE INDEX REPAIR\_STATUSDATE\_IDX ON REPAIRS (REPAIR\_DATE, STATUS\_ID);

* 1. **VIEWS**

**Creates all the useful views**

CREATE OR REPLACE VIEW LOCATION\_VIEW

AS SELECT PROVINCES.PROVINCE\_NAME, ADDRESS.CITY

FROM PROVINCES, ADDRESS WHERE PROVINCES.PROVINCE\_ID = ADDRESS.PROVINCE\_ID;

CREATE OR REPLACE VIEW CUSTOMER\_VIEW

AS SELECT CUSTOMERS.CUSTOMER\_NAME, CUSTOMERS.CUSTOMER\_SURNAME, ADDRESS.STREET\_NAME, ADDRESS.STREET\_NUMBER

FROM CUSTOMERS, ADDRESS WHERE CUSTOMERS.ADDRESS\_ID = ADDRESS.ADDRESS\_ID;

CREATE OR REPLACE VIEW REPAIRSTATUS\_VIEW

AS SELECT REPAIRS.REPAIR\_ID, STATUS.STATUS\_NAME

FROM REPAIRS, STATUS WHERE STATUS.STATUS\_ID = REPAIRS.STATUS\_ID;

CREATE OR REPLACE VIEW TECHNICIANS\_VIEW

AS SELECT REPAIRS.REPAIR\_ID, TECHNICIANS.TECH\_NAME, TECHNICIANS.TECH\_SURNAME

FROM REPAIRS, TECHNICIANS WHERE REPAIRS.TECHNICIAN\_ID = TECHNICIANS.TECHNICIAN\_ID;

CREATE OR REPLACE VIEW ORDERPARTS\_VIEW

AS SELECT O.ORDER\_ID, P.PART\_NAME, P.PART\_TYPE, P.PART\_COST

FROM ORDERS O

JOIN PARTS\_IN\_ORDER PIO ON O.ORDER\_ID = PIO.ORDER\_ID

JOIN PARTS P ON PIO.PARTS\_ID = P.PART\_ID;

CREATE OR REPLACE VIEW REPAIRPARTS\_VIEW

AS SELECT R.REPAIR\_ID, P.PART\_NAME, P.PART\_TYPE, P.PART\_COST

FROM REPAIRS R

JOIN PARTS\_IN\_REPAIR PIR ON R.REPAIR\_ID = PIR.REPAIR\_ID

JOIN PARTS P ON PIR.PART\_ID = P.PART\_ID;

* 1. **DATA LOADING**

**Loads data into all the tables. With at least 3 entries each**

-----------------------------------------------------------

--Data Insert

-----------------------------------------------------------

--Provinces

INSERT INTO PROVINCES (PROVINCE\_ID, PROVINCE\_NAME) VALUES

(PROVINCE\_ID\_VALUE.nextval, 'Eastern Cape');

INSERT INTO PROVINCES (PROVINCE\_ID, PROVINCE\_NAME) VALUES

(PROVINCE\_ID\_VALUE.nextval, 'Free State');

INSERT INTO PROVINCES (PROVINCE\_ID, PROVINCE\_NAME) VALUES

(PROVINCE\_ID\_VALUE.nextval, 'Gauteng');

INSERT INTO PROVINCES (PROVINCE\_ID, PROVINCE\_NAME) VALUES

(PROVINCE\_ID\_VALUE.nextval, 'KwaZulu-Natal');

INSERT INTO PROVINCES (PROVINCE\_ID, PROVINCE\_NAME) VALUES

(PROVINCE\_ID\_VALUE.nextval, 'Limpopo');

INSERT INTO PROVINCES (PROVINCE\_ID, PROVINCE\_NAME) VALUES

(PROVINCE\_ID\_VALUE.nextval, 'Mpumalanga');

INSERT INTO PROVINCES (PROVINCE\_ID, PROVINCE\_NAME) VALUES

(PROVINCE\_ID\_VALUE.nextval, 'Northern Cape');

INSERT INTO PROVINCES (PROVINCE\_ID, PROVINCE\_NAME) VALUES

(PROVINCE\_ID\_VALUE.nextval, 'North West');

INSERT INTO PROVINCES (PROVINCE\_ID, PROVINCE\_NAME) VALUES

(PROVINCE\_ID\_VALUE.nextval, 'Western Cape');

--

-- Status

INSERT INTO STATUS (STATUS\_ID, STATUS\_NAME) VALUES

(STATUS\_ID\_VALUE.nextval, 'In Transit');

INSERT INTO STATUS (STATUS\_ID, STATUS\_NAME) VALUES

(STATUS\_ID\_VALUE.nextval, 'In Progress');

INSERT INTO STATUS (STATUS\_ID, STATUS\_NAME) VALUES

(STATUS\_ID\_VALUE.nextval, 'Completed');

INSERT INTO STATUS (STATUS\_ID, STATUS\_NAME) VALUES

(STATUS\_ID\_VALUE.nextval, 'To Do');

INSERT INTO STATUS (STATUS\_ID, STATUS\_NAME) VALUES

(STATUS\_ID\_VALUE.nextval, 'Standby');

INSERT INTO STATUS (STATUS\_ID, STATUS\_NAME) VALUES

(STATUS\_ID\_VALUE.nextval, 'Cancelled');

--

-- Departments

INSERT INTO DEPARTMENTS (DEPARTMENT\_ID, DEPARTMENT\_NAME) VALUES

(DEPARTMENT\_ID\_VALUE.nextval, 'Customer Service Representatives');

INSERT INTO DEPARTMENTS (DEPARTMENT\_ID, DEPARTMENT\_NAME) VALUES

(DEPARTMENT\_ID\_VALUE.nextval, 'Technicians');

INSERT INTO DEPARTMENTS (DEPARTMENT\_ID, DEPARTMENT\_NAME) VALUES

(DEPARTMENT\_ID\_VALUE.nextval, 'Sales and Marketing Personnel');

INSERT INTO DEPARTMENTS (DEPARTMENT\_ID, DEPARTMENT\_NAME) VALUES

(DEPARTMENT\_ID\_VALUE.nextval, 'IT Personnel');

--

--ADDRESS

INSERT INTO address

VALUES (ADDRESS\_ID\_VALUE.nextval,'8th street',20,'Polokwane',5);

INSERT INTO address

VALUES (ADDRESS\_ID\_VALUE.nextval,'5th street',8,'Stofberg',2);

INSERT INTO address

VALUES (ADDRESS\_ID\_VALUE.nextval,'7th street',99,'Pretoria',3);

--

--Customers

INSERT INTO customers

VALUES (CUSTOMER\_ID\_VALUE.nextval, 'customer@email.com','0661024086','Fanie','Van Vuuren',1);

INSERT INTO customers

VALUES (CUSTOMER\_ID\_VALUE.nextval, 'customer2@email.com','0715896321','Ben','Van Vuuren',2);

INSERT INTO customers

VALUES (CUSTOMER\_ID\_VALUE.nextval, 'customer3@email.com','0324928569','Veronica','Van Vuuren',3);

--

--Parts

INSERT INTO PARTS

VALUES (PART\_ID\_VALUE.nextval,'RTX 3060','Nvidea','Graphics Card', 40000,20);

INSERT INTO PARTS

VALUES (PART\_ID\_VALUE.nextval,'GTX 1020','Nvidea','Graphics Card', 4000,14);

INSERT INTO PARTS

VALUES (PART\_ID\_VALUE.nextval,'GTX 1060 TI','Nvidea','Graphics Card', 8000,8);

INSERT INTO PARTS

VALUES (PART\_ID\_VALUE.nextval,'I3 400Hz','Intel','CPU', 1000,3);

--

--Technicians

INSERT INTO TECHNICIANS

VALUES (TECHNICIAN\_ID\_VALUE.nextval,1,'Burger',0412589636,'Franco',2);

INSERT INTO TECHNICIANS

VALUES (TECHNICIAN\_ID\_VALUE.nextval,0,'Ricon',4589637845,'David',4);

INSERT INTO TECHNICIANS

VALUES (TECHNICIAN\_ID\_VALUE.nextval,1,'Smith',0152364150,'Steven',2);

--

--Order

INSERT INTO ORDERS

VALUES (ORDER\_ID\_VALUE.nextval,1,1,'01-Jan-23',5,0);

INSERT INTO PARTS\_IN\_ORDER

VALUES (PIO\_ID\_VALUE.nextval,1,1);

UPDATE ORDERS

SET total\_cost = (SELECT PART\_COST FROM PARTS WHERE PART\_ID = 1)\*1.1

WHERE ORDER\_ID = '1';

INSERT INTO ORDERS

VALUES (ORDER\_ID\_VALUE.nextval,2,3,'10-Jan-23',5,0);

INSERT INTO PARTS\_IN\_ORDER

VALUES (PIO\_ID\_VALUE.nextval,2,2);

UPDATE ORDERS

SET total\_cost = (SELECT PART\_COST FROM PARTS WHERE PART\_ID = 2)\*1.1

WHERE ORDER\_ID = '2';

INSERT INTO ORDERS

VALUES (ORDER\_ID\_VALUE.nextval,3,2,'15-Jan-23',5,0);

INSERT INTO PARTS\_IN\_ORDER

VALUES (PIO\_ID\_VALUE.nextval,3,1);

UPDATE ORDERS

SET total\_cost = (SELECT PART\_COST FROM PARTS WHERE PART\_ID = 1)\*1.1

WHERE ORDER\_ID = '3';

--Repair

INSERT INTO REPAIRS

VALUES (REPAIR\_ID\_VALUE.nextval,1,1,5,(SELECT ADDRESS\_ID FROM CUSTOMERS WHERE CUSTOMER\_ID = '1'),'01-Jan-23',0);

INSERT INTO PARTS\_IN\_REPAIR

VALUES (PIR\_ID\_VALUE.nextval,1,1);

UPDATE REPAIRS

SET ACCUMULATED\_COST = (SELECT PART\_COST FROM PARTS WHERE PART\_ID = '1')\*1.1

WHERE REPAIR\_ID = '1';

INSERT INTO REPAIRS

VALUES (REPAIR\_ID\_VALUE.nextval,3,2,5,(SELECT ADDRESS\_ID FROM CUSTOMERS WHERE CUSTOMER\_ID = '3'),'01-Jan-23',0);

INSERT INTO PARTS\_IN\_REPAIR

VALUES (PIR\_ID\_VALUE.nextval,2,3);

UPDATE REPAIRS

SET ACCUMULATED\_COST = (SELECT PART\_COST FROM PARTS WHERE PART\_ID = '3')\*1.1

WHERE REPAIR\_ID = '2';

INSERT INTO REPAIRS

VALUES (REPAIR\_ID\_VALUE.nextval,2,3,5,(SELECT ADDRESS\_ID FROM CUSTOMERS WHERE CUSTOMER\_ID = '1'),'01-Jan-23',0);

INSERT INTO PARTS\_IN\_REPAIR

VALUES (PIR\_ID\_VALUE.nextval,3,4);

UPDATE REPAIRS

SET ACCUMULATED\_COST = (SELECT PART\_COST FROM PARTS WHERE PART\_ID = '4')\*1.1

WHERE REPAIR\_ID = '3';

## 2. QUERIES

-----------

--Queries--

-----------

--Q1

--Used if you need to access all the data about a customer and their address

--EXAMPLE: You need to check where a customer is located

-----------

SELECT

C.CUSTOMER\_ID,

C.CUSTOMER\_EMAIL,

C.CUSTOMER\_PHONE\_NUMBER,

C.CUSTOMER\_NAME,

C.CUSTOMER\_SURNAME,

A.STREET\_NAME,

A.STREET\_NUMBER,

A.CITY,

A.PROVINCE\_ID

FROM

CUSTOMERS C

JOIN

ADDRESS A ON C.ADDRESS\_ID = A.ADDRESS\_ID

WHERE

C.CUSTOMER\_ID = '&CUSTOMER\_ID';

-----------

--Q2

--Used to track all the orders done by a specific customer

--EXAMPLE: You need to check what a customer ordered

-----------

SELECT

CUSTOMERS.CUSTOMER\_ID,

CUSTOMERS.CUSTOMER\_EMAIL,

CUSTOMERS.CUSTOMER\_PHONE\_NUMBER,

CUSTOMERS.CUSTOMER\_NAME,

CUSTOMERS.CUSTOMER\_SURNAME,

ORDERS.ORDER\_DATE,

ORDERS.ORDER\_STATUS,

ORDERS.ORDER\_ID,

ORDERS.TOTAL\_COST

FROM

CUSTOMERS

JOIN

ORDERS ON CUSTOMERS.CUSTOMER\_ID = ORDERS.CUSTOMER\_ID

WHERE

UPPER(CUSTOMER\_NAME) LIKE UPPER('%&CUSTOMER\_NAME%');

-----------

--Q3

--Provides a summary of the months income

--EXAMPLE: You need to check how much was made this month

-----------

SELECT ROUND(SUM(X.TOTAL\_COST), 2) AS TOTAL\_MONTH\_COST

FROM

(SELECT O.TOTAL\_COST,O.ORDER\_DATE,CUSTOMER\_ID FROM ORDERS O WHERE O.ORDER\_DATE >= '01-JAN-23')

UNION ALL

SELECT R.ACCUMULATED\_COST,R.REPAIR\_DATE,CUSTOMER\_ID FROM REPAIRS R WHERE r.repair\_date >= '01-JAN-23') X

ORDER BY TOTAL\_COST;

-----------

--Q4

--Provides the statistics of all the parts in Stock

--EXAMPLE: You need specific information about the stocks of the parts in store

-----------

SELECT

SUM(PARTS\_IN\_STOCK) AS "Total Parts",

SUM(PART\_COST \* PARTS\_IN\_STOCK) AS "Total Value (R)",

ROUND(SUM(PART\_COST \* PARTS\_IN\_STOCK) / SUM(PARTS\_IN\_STOCK), 2) AS "Avg Cost (R)",

ROUND(MAX(PART\_COST \* PARTS\_IN\_STOCK)) AS "Most Valuable (R)",

ROUND(MIN(PART\_COST \* PARTS\_IN\_STOCK)) AS "Least Valuable (R)",

MAX(PARTS\_IN\_STOCK) AS "Highest Stock",

MIN(PARTS\_IN\_STOCK) AS "Lowest Stock",

COUNT(DISTINCT PART\_NAME) AS "Unique Parts",

SUM(CASE WHEN PARTS\_IN\_STOCK < 10 THEN 1 ELSE 0 END) AS "Parts Below 10"

FROM PARTS;

-----------

--Q5

--Checks for available technicians

--EXAMPLE: You need to check who can take on another order right now

-----------

SELECT

(TECH\_NAME || ' ' || TECH\_SURNAME) AS "AVAILABLE TECHNICIANS",

AVAILABILITY

FROM

TECHNICIANS

WHERE

AVAILABILITY < 5

AND department\_id = 2

ORDER BY

TECH\_SURNAME DESC;

-----------

--Q6

--Adds an order

--EXAMPLE: A customer buys something

-----------

INSERT INTO ORDERS

VALUES (ORDER\_ID\_value.nextval, '&CUSTOMER\_ID', '&TECHNICIAN\_ID', '&ORDER\_DATE', '&ORDER\_STATUS',

ROUND((SELECT

SUM(p.PART\_COST) FROM PARTS p

JOIN PARTS\_IN\_ORDER i

ON p.PART\_ID = i.PARTS\_ID

JOIN ORDERS o ON o.ORDER\_ID = i.ORDER\_ID

WHERE o.ORDER\_ID = ORDER\_ID\_value

GROUP BY o.ORDER\_ID)\*1.10, 2));

-----------

--Q7

--Changes an orders status

--EXAMPLE: An order is done so it need to change from in progress to done

-----------

UPDATE ORDERS

SET ORDER\_STATUS = '&ORDER\_STATUS'

WHERE ORDER\_ID = '&ORDER\_ID';

-----------

--Q8

-- Adds a new repair order

--EXAMPLE: Customer orders a repair order

-----------

INSERT INTO REPAIRS

VALUES (REPAIR\_ID\_value.nextval, '&CUSTOMER\_ID', '&TECHNICIAN\_ID', '&STATUS\_ID', '&ADDRESS\_ID', '&REPAIR\_DATE',

ROUND((SELECT

SUM(p.PART\_COST)

FROM PARTS p

JOIN PARTS\_IN\_REPAIR i

ON p.PART\_ID = i.PART\_ID

JOIN REPAIRS r ON r.REPAIR\_ID = i.REPAIR\_ID

WHERE r.REPAIR\_ID = REPAIR\_ID\_value

GROUP BY r.REPAIR\_ID)\*1.10, 2));

-----------

--Q9

-- Changes a repair orders status

--EXAMPLE: You update when an repair order is done

-----------

UPDATE REPAIRS

SET STATUS\_ID = '&STATUS\_ID', REPAIR\_DATE = '&REPAIR\_DATE'

WHERE REPAIR\_ID = '&REPAIR\_ID';

-----------

--Q10

--Inserts a new type of part

--EXAMPLE: You got a new part in stock that wasn’t previously available

-----------

INSERT INTO PARTS

VALUES (PART\_ID\_value.nextval, '&PART\_NAME', '&PART\_BRAND', '&PART\_TYPE', '&PART\_COST', '&PARTS\_IN\_STOCK');

-----------

--Q11

-- Lets you update a parts cost or how many are in stock

--EXAMPLE: A new shipment of old parts has arrived

-----------

UPDATE PARTS

SET PART\_COST = '&PART\_COST', PARTS\_IN\_STOCK = '&PARTS\_IN\_STOCK'

WHERE PART\_ID = '&PART\_ID';

-----------

--Q12

--Lets you add a new technician

--EXAMPLE: You hire a new guy

-----------

INSERT INTO TECHNICIANS (TECHNICIAN\_ID, AVAILABILITY, TECH\_NAME, TECH\_SURNAME, TECH\_PHONE\_NUMBER, DEPARTMENT\_ID)

VALUES (TECHNICIAN\_ID\_value.NEXTVAL, 0, '&Name', '&Surname', '&PhoneNumber', '&DepartmentID');

-----------

--Q13

--Lets you change a Technicians Department

--EXAMPLE: A technician is switched between departments

-----------

UPDATE TECHNICIANS

SET DEPARTMENT\_ID = '&New\_department\_id'

WHERE TECHNICIAN\_ID = '&TechnicianID' ;

-----------

--Q14

--Lets you add a new customer

--EXAMPLE: A new customer wants to buy something so they register

-----------

INSERT INTO customers

VALUES (CUSTOMER\_ID\_VALUE.nextval, '&CUSTOMER\_EMAIL','&CUSTOMER\_CELL\_NUMBER','&CUSTOMER\_FIRST\_NAME','&CUSTOMER\_LAST\_NAME','&ADDRESS\_ID');

-----------

--Q15

--Lets you change any info regarding a Customer

--EXAMPLE: A customer needs to change their info or you system, ei they got married so their surname changed

-----------

UPDATE CUSTOMERS

SET CUSTOMER\_EMAIL = '&Customer\_Email',CUSTOMER\_PHONE\_NUMBER = '&CUSTOMER\_CELL\_NUMBER',CUSTOMER\_NAME = '&CUSTOMER\_NAME',CUSTOMER\_SURNAME = '&CUSTOMER\_SURNAME',ADDRESS\_ID = '&ADDRESS\_ID'

WHERE CUSTOMER\_ID = '&CUSTOMER\_ID';

-----------

--Q16

--Lets you remove a customer from the database

--EXAMPLE: A customer wants to remove their data from your system or they haven’t ordered in a long time

-----------

DELETE FROM CUSTOMERS

WHERE CUSTOMER\_ID = '&CUSTOMER\_ID';

-----------

--Q17

--Lets you know the total value of all parts combined

--EXAMPLE: You need to check how much stick is lying around in the store

-----------

SELECT SUM(PART\_COST \* PARTS\_IN\_STOCK)

AS Total\_Value

FROM PARTS;

-----------

--Q18

--Checks order status on all orders of customers sorted by date

--EXAMPLE: You need to check if an order is ready to be shipped

-----------

SELECT O.\*, C.CUSTOMER\_ID, C.CUSTOMER\_NAME, C.CUSTOMER\_SURNAME, C.CUSTOMER\_PHONE\_NUMBER, C.CUSTOMER\_EMAIL, S.STATUS\_NAME

FROM ORDERS O

JOIN CUSTOMERS C ON O.CUSTOMER\_ID = C.CUSTOMER\_ID

JOIN STATUS S ON O.ORDER\_STATUS = S.STATUS\_ID

WHERE O.ORDER\_STATUS = '&Status\_ID'

ORDER BY O.ORDER\_DATE ASC;

|  |  |
| --- | --- |
| **What need to be found** | **Where it can be found** |
| **2.1 LIMITATION OF ROWS AND COLUMNS** | **Q3,Q5** |
| **2.2 SORTING** | **Q5,Q18** |
| **2.3 LIKE, AND AND OR** | **Q2,Q5** |
| **2.4 VARIABLES AND CHARACTER FUNCTIONS** | **Q1,Q2,Q3** |
| **2.5 ROUND AND/OR TRUNC** | **Q3,Q4** |
| **2.6 DATE FUNCTIONS** | **Q3,Q18** |
| **2.7 AGGREGATE FUNCTIONS** | **Q4** |
| **2.8 GROUP BY AND HAVING** | **Q3,Q6** |
| **2.9 JOINS** | **Q1,Q2,Q3** |
| **2.10 SUB-QUERIES** | **Q6,Q8** |