



The JUST-GUARD Tech Group

Group Project – CMPG311

Group Members:

[D. De Jager] – (Group Leader)

[G. Gloss]

[R.Koekemoer]

[J. Blom]

[S. Paetzold]

Submission date: 09/05/2025

TABLE OF CONTENTS

Table of Contents.....	0
Project Phase 1 – Database Initial Study	4
Members of the group	4
Daniël De Jager (team leader).....	4
Jurie Blom	4
Gaby Gloss	4
Steve Paetzold	4
Ruan Koekemoer	4
Analyse company situation	4
2.1. Company objectives	4
2.2. Company operations.....	5
2.3. Company organizational structure	5
Define problems and constraints	7
Database system specification.....	8
4.1. Objectives to solve identified problems	8
4.2. Information required from database.....	8
4.3. Scope of the Database	8
4.4. Boundaries of the System.....	9
Project Phase 2 - Database Design	10
1. Conceptual design	10
1.1 Business Rules.....	10
1.1.1. Business rules on clients.....	10
1.1.2. Business rules on marketing and sales.....	10
1.1.3. Business rules on operations and software development	11
1.1.4. Business rules on support.....	11
1.1.5 Business rules on human resources	11
1.1.6 Information kept on clients	11
1.1.7 Information kept on employees.....	11
1.2 ER Diagram	13
.....	14
1.3 Notes on the ER diagram	15
Composite keys.....	15
Weak Relationships.....	15
Strong relationships.....	15
Weak Entities	15

Composite/Bridge Entities	16
Super entities	16
Sub-type entities.....	16
2. Logical Design	17
Project Phase 3 – Physical Design.....	19
1 Database Objects	19
1.1. Dropping Tables	19
1.2. Table Creation	20
1.2.1. Business Table	20
1.2.2. Partner Table	20
1.2.3. Partner on campaign Table	21
1.2.4. Campaign Table.....	21
1.2.5. Platform Table.....	21
1.2.6. Platform_In_Campaign Table.....	22
1.2.7. Client Table.....	22
1.2.8. Client_Contract Table.....	22
1.2.9. Invoice Table.....	23
1.2.10. Project Table	23
1.2.11. Team Table	24
1.2.12. Article Table	24
1.2.13. Category Table.....	24
1.2.14. FAQ Table.....	25
1.2.15. Ticket Table	25
1.2.16. Position Table	25
1.2.17. Employee Table	26
1.2.18. Employee_On_Team Table.....	26
1.2.19. Employee_Payroll Table	27
1.2.20. Employee_Deductions Table.....	27
1.2.21. Employee_Leave Table.....	28
1.2.22. Employee_Banking Table	28
1.2. Indexes.....	29
1.3. Views.....	30
1.3.1. Active Clients View.....	30
1.3.2. Client Contract View.....	30
1.3.3. Invoice Payment Status View	31
1.3.4. Project View	32
1.3.4. Position view	32

1.3.5. Leave view	33
1.3.6 Campaign view	33
2. Data	34
2.1. Loading	34
2.1.1. Loading the Business Table	34
2.1.2. Loading the Client Table	34
2.1.3. Loading the Client_Contract Table	35
2.1.4. Loading the Invoice Table	35
2.1.6. Loading the Project Table	36
2.1.6. Loading the Team Table	37
2.1.6. Loading the Position Table	37
2.1.7. Loading the Employee Table	37
2.1.8. Loading the Employee_deductions Table	38
2.1.9. Loading the Employee_leave Table	38
2.1.10. Loading the Employee_payroll Table	39
2.1.10. Loading the Employee_Banking Table	39
2.1.11. Loading the Article Table	39
2.1.12. Loading the Category Table	39
2.1.13. Loading the FAQ Table	40
2.1.14. Loading the Ticket Table	40
2.1.15. Loading the Partner Table	40
2.1.16. Loading the Campaign Table	40
2.1.17. Loading the Partner_on_Campaign Table	41
2.1.18. Loading the Employee_on_Team Table	41
2.1.19. Loading the Platform Table	41
2.1.20. Loading the Platform_in_Campaign Table	41
3. Queries	42
3.1. Limitation of Rows and Columns	42
3.2. Sorting	42
3.3. LIKE, AND and OR	44
3.4. Variables and Character Functions	44
3.5. Round and Trunc	44
3.6. Date Functions	45
3.7. Aggregate Functions	46
3.8. Group By and Having	47
3.9. Joins	47
3.10. Sub-Queries	48

3.11. SELECT Queries	49
3.12. Average Queries.....	50
3.13. Other Queries	51

Project Phase 1 – Database Initial Study

Members of the group

<p>Daniël De Jager (team leader) Role: Clients</p> 	<p>J. Blom Role: Support</p>
<p>G. Gloss Role: Operations</p>	<p>S. Paetzold Role: Marketing</p>
<p>R. Koekemoer Role: Human Resources</p>	

Analyse company situation

2.1. Company objectives

JUST-GUARD's mission is to design and deliver scalable, user-friendly, and cutting-edge Software-as-a-Service (SaaS) solutions to security-mindful industries. The solutions are aimed at empowering companies to automate processes, enhance productivity, and promote digital transformation in the current era. JUST-GUARD is committed to developing secure, data-backed, and cost-effective platforms that address evolving needs of current businesses, regardless of size or geography.

By leveraging cutting-edge technologies such as cloud computing, artificial intelligence, and automation, they seek to provide smooth integrations, user-friendly experiences, and real-time insights that enable their customers to simplify workflows and make confident decisions.

Their purpose is more than simply the delivery of software. JUST-GUARD plans to develop a long-term association with their users based on exceptional support, continual innovation, and a success pledge. With relentless research, innovation, and user feedback, they regularly modify their solutions in order to continue pushing the limits in technological breakthroughs, providing their customers with tools to help them compete in a constantly changing online world.

2.2. Company operations

2.2.1. Clients (clients, contracts, invoices)

JUST-GUARD Tech specialises in delivering security software products to clients that demand the best possible security to protect their high-risk assets. Such clients exist in the financial, cybersecurity, political, and technology industries. JUST-GUARD Tech acquires such clients and forms strategic relationships to achieve the long-term goals of both companies. In pursuit of these goals, contracts are signed and projects are initiated under these contracts.

2.2.2. Marketing & Sales (campaigns, marketing platforms, promotional partners)

The company's Marketing and Sales departments work hand in hand to grow business reach and impact through campaigns, lead generation and personal communications. Sales focus on lead generation, customer engagement, and revenue growth. Their marketing team handles campaigns, branding, and brand partnerships.

2.2.3. Operations & Software Development (sprints, projects, teams)

JUST-GUARD ensures efficient software delivery and infrastructure operations. This involves task tracking, setting sprint goals, and providing regular updates, while also overseeing internal software builds and managing infrastructure upgrades. They implement DevOps practices such as CI/CD automation, server management, and cloud resource optimization to ensure reliable, secure and scalable deployments.

2.2.4. Support (tickets, articles in documentation, FAQ entries)

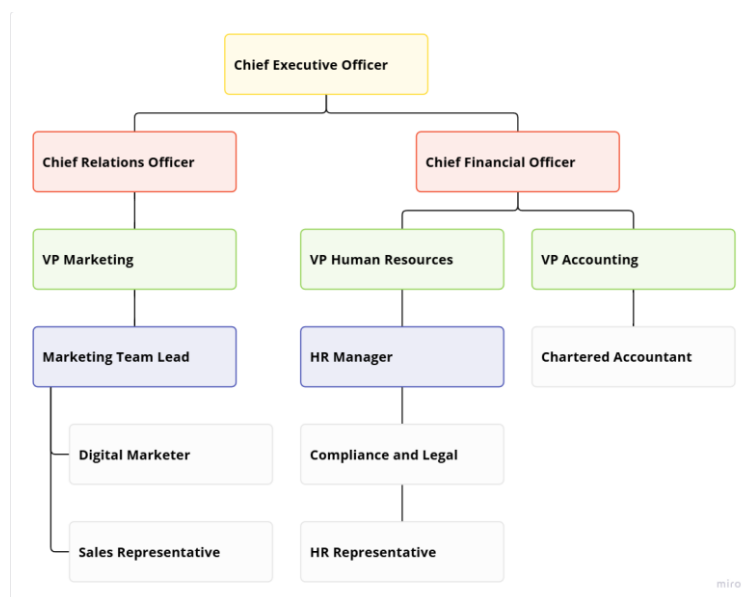
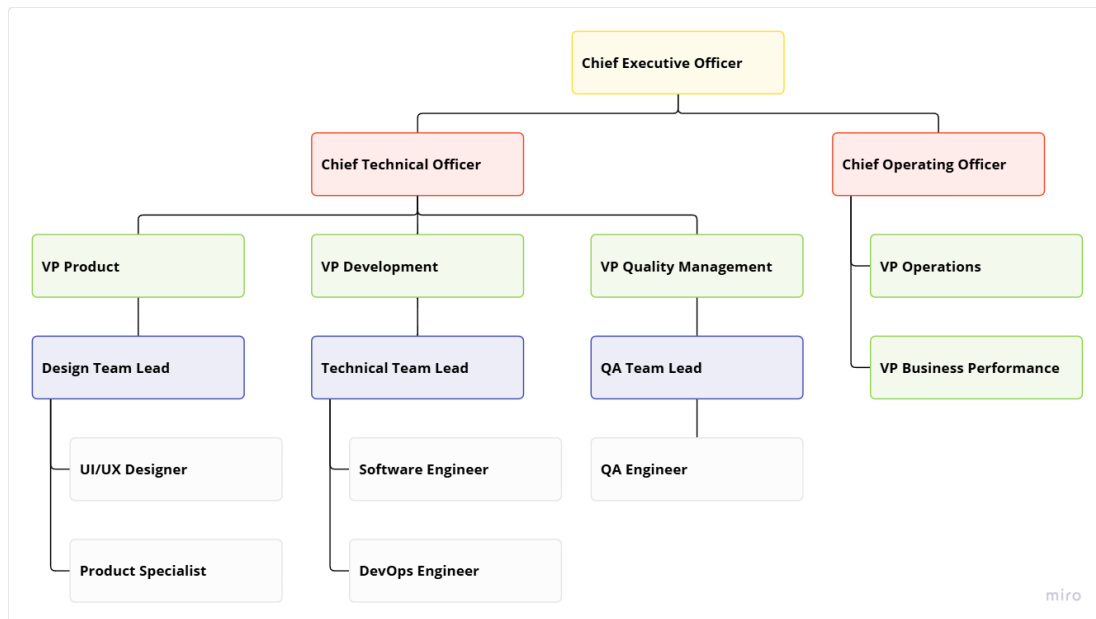
JUST-GUARD focuses on their commitment to customer success and extends beyond software development. They offer full support to make sure their clients get the most out of their SaaS solutions. Their support system is made to be proactive, responsive, and flexible enough to meet the specific requirements of companies across various industries.

2.2.5 HR (employees, contract/payroll, positions/job specs)

JUST-GUARD is a medium enterprise that has numerous employees, positions and availabilities. Downtime due to staff turnover has previously resulted in irreparable financial and reputational damage. A temporary employee managing structure has been implemented to better utilise data acquired by the human resource officers to ensure job satisfaction and enforce company culture.

2.3. Company organizational structure

The company has grown significantly since a client burst in the past few years, resulting in a more efficient organisational structure.



Define problems and constraints

The current system presented the following problems that necessitated this project:

- *Sudden growth in the number of clients and subsequent support tickets*
- *Invoices necessitated a database digitized to replace the existing paper-based system.*
- *A growing team of employees resulted in poor recordkeeping of salaries, employment contracts, and disciplinary actions.*
- *Communication needs to be improved between clients and support staff to ensure accurate exchange of information.*
- *The company is not getting enough exposure due to the lack of appropriate marketing.*
- *Support staff are jumping between projects and failing to produce satisfactory results due to disorganized instructions and a lack of recordkeeping.*
- *Employees are going out of the working bounds of their contracts due to the lack of appropriate task management.*
- *New employees have difficulty onboarding due to the lack of documentation.*
- *The company struggles to effectively onboard clients due to outdated documentation and a lack of information on certain functions of their applications.*

The new system will be subject to the following constraints:

- *The database must store information regarding clients (information, invoices, contracts) securely in 3NF.*
- *The database must store information regarding projects (information, sprints, teams) securely in 3NF.*
- *The database must store information regarding support (articles, FAQs, tickets) securely in 3NF.*
- *The database must store information regarding employees (personal information, positions, contracts) securely in 3NF.*
- *The database must store information regarding marketing (campaigns, partners, platforms) securely in 3NF.*
- *Each project must have at least one team assigned to it.*
- *One employee can be assigned to multiple teams as determined by their manager.*
- *Each employee can only hold one position at a time.*
- *Each employee must have an employee contract.*
- *Each project must have many FAQs and articles.*
- *Each project must have multiple sprints.*
- *Each client must have a project to which invoices are linked.*
- *Each client can have one or more projects, but each project only has one client.*
- *A client can open one or more support tickets.*
- *The company runs many marketing campaigns targeted at prospective clients. A campaign can use multiple platforms but can only use one promotional partner.*
- *There must be different access levels for various departments and positions.*
- *The system needs to be completed within 6 months.*
- *The system response time must not exceed 300ms.*
- *The system will have support for 10 users at once.*

Database system specification

4.1. Objectives to solve identified problems

- A new database that organises and efficiently handles the new influx of clients and tickets will be implemented.
- Within the new system's database there will be tables that handle the digitised version of the existing paper-based system.
- With the new systems digital database will help manage the recordkeeping of the increasing number of employees.
- An organised digital ticketing system will improve communication between clients and support staff.
- Marketing campaigns and other brand partnerships will be part of the database's design to increase business exposure.
- The new system will have defined access controls to keep employees within the constraints of their contracts work delegation.
- Documentation can be stored on the new system digitally to allow for organisation-wide access.
- Digitising documentation will allow for updating documentation in general and functions of applications.

4.2. Information required from database

JUST-GUARD will require information on:

- Current and previous clients, contract templates and billing information.
- Perceived public image, marketing campaigns, platforms where marketing runs and public image partners.
- Current and previous operations, projects and teams
- Support requests from clients, links to self-help documentation and frequently asked questions.
- Employee personal information, payroll and position specifications.

4.3. Scope of the Database

4.3.1. Clients (clients, contracts, invoices)

The database must store all relevant information on clients, such as their business names, contact person, phone numbers, addresses, and more. Other information related to clients must be stored in a contract database where information related to signed contracts, such as signed date, end date, financial information, and more. Also, invoicing related to these contracts and clients must be stored to keep track of payments.

4.3.2. Marketing & Sales (campaigns, marketing platforms, promotional partners)

The database must store brand contact details and other partnership information, while also recording campaign data, budgets, timelines, and impact/reach. Historical data on previous campaigns must be stored for trend analysis and projections. Included in the database are the promotional partners' data, past and present.

4.3.3. Operations & Software Development (sprints, projects, teams)

The database must store adequate information to assist SPRINT managers in managing sprint planning goals, establishing priorities and tasks, tracking progress throughout sprints, and storing historical data for sprint performance analysis. Also, the database must store relevant information on projects, including timelines, resources, budgets, and deliverables. Managers must be able to manage team structures, roles, responsibilities, capacity and workload based on information stored in the database.

4.3.4. Support (tickets, articles in documentation, FAQ entries)

Support tickets will be logged by clients and then stored in the database with relevant information, providing access to the support team and/or developers. Articles and documentation will be written by teams of developers about projects and utilizing new technologies, and relevant meta information on these FAQs and documentation must be stored in the database.

4.3.5 HR (employees, payroll/salaries, positions/job specs)

The Human Resources department portion of the database needs to store personal information regarding all job positions, employees, employee emergency contacts, and payroll to ensure a detailed record is easily accessible during emergencies and recruiting.

4.4. Boundaries of the System

- The database must be self-hosted on an on-premises.
- The budget for the development of the database is R11,300,000.
- The company only employs one database administrator to manage the database environment.
- The database must be implemented in a free DBMS to limit operational costs.
- The system must be primarily developed by the team of 5 system designers and developers.
- The project must be completed within six months of its inception.
- A team of two experts from JUST-GUARD are available for consultation 2 hours per day, while other employees are available on an appointment-only basis.

Project Phase 2 - Database Design

1. Conceptual design

1.1 Business Rules

The following business rules regarding JUST-GUARD are derived from the company operations specified in Project Phase 1.

JUST-GUARD's primary five areas to be expanded on regarding business rules are:

1. Clients
2. Marketing and sales
3. Operations and software development
4. Support
5. Human resources
6. Information kept on clients
7. Information kept on employees

1.1.1. Business rules on clients

- A business is approached by, or approaches, JUST-GUARD and must sign a contract to become a client of JUST-GUARD. These clients can have one or more contracts.
- Every month invoices are issued to clients based on contracts, with the number of installments dictating how many months the client is billed. This calculation is made by dividing the total budget by the number of installments. If a contract goes over budget, the budget must be re-negotiated and related figures must be re-calculated.
- One contract initiates one or more projects that fulfill the requirements written out in the contract.
- A contract must be paid within two weeks of the date of issue at the set due date.

1.1.2. Business rules on marketing and sales

- The second type of business in relation to JUST-GUARD is a marketing partner. Such partners are approached to work with the company on one or more marketing campaigns. More than one partner can be onboarded to a specific campaign.
- A marketing campaign can run across multiple platforms and multiple campaigns can reuse platforms previously used.
- The company wants to keep track of when a partner was first onboarded.

- A marketing campaign must begin at a specific date and end at a specific date.
- A campaign conversion is when a new client is successfully onboarded due to the efforts of that marketing campaign.

1.1.3. Business rules on operations and software development

- A project is established based on a contract signed with a client.
- Only one team is assigned to a project. Many teams can be created with varying combinations of employees specific to a project.

1.1.4. Business rules on support

- A project can have none to many tickets opened by request of the client. Each ticket can either be open or closed.
- A project has one or more FAQs written by the team to guide the clients who have questions. This can prevent the opening of unnecessary tickets.
- A project can have one or more articles describing the project's function or providing any other information that the client may need. Each article must fall into a specific category for easy organizing.

1.1.5 Business rules on human resources

- An employee is assigned to one or more teams that then work on a project.
- An employee fills a position at a company and multiple employees can hold one position if necessary.
- The business records various information on their employees such as personal information and payroll (including leave, banking, and deductions).
- An employee is added to the payroll and paid monthly. Payroll keeps track of employee deductions, leave, and banking information.

1.1.6 Information kept on clients

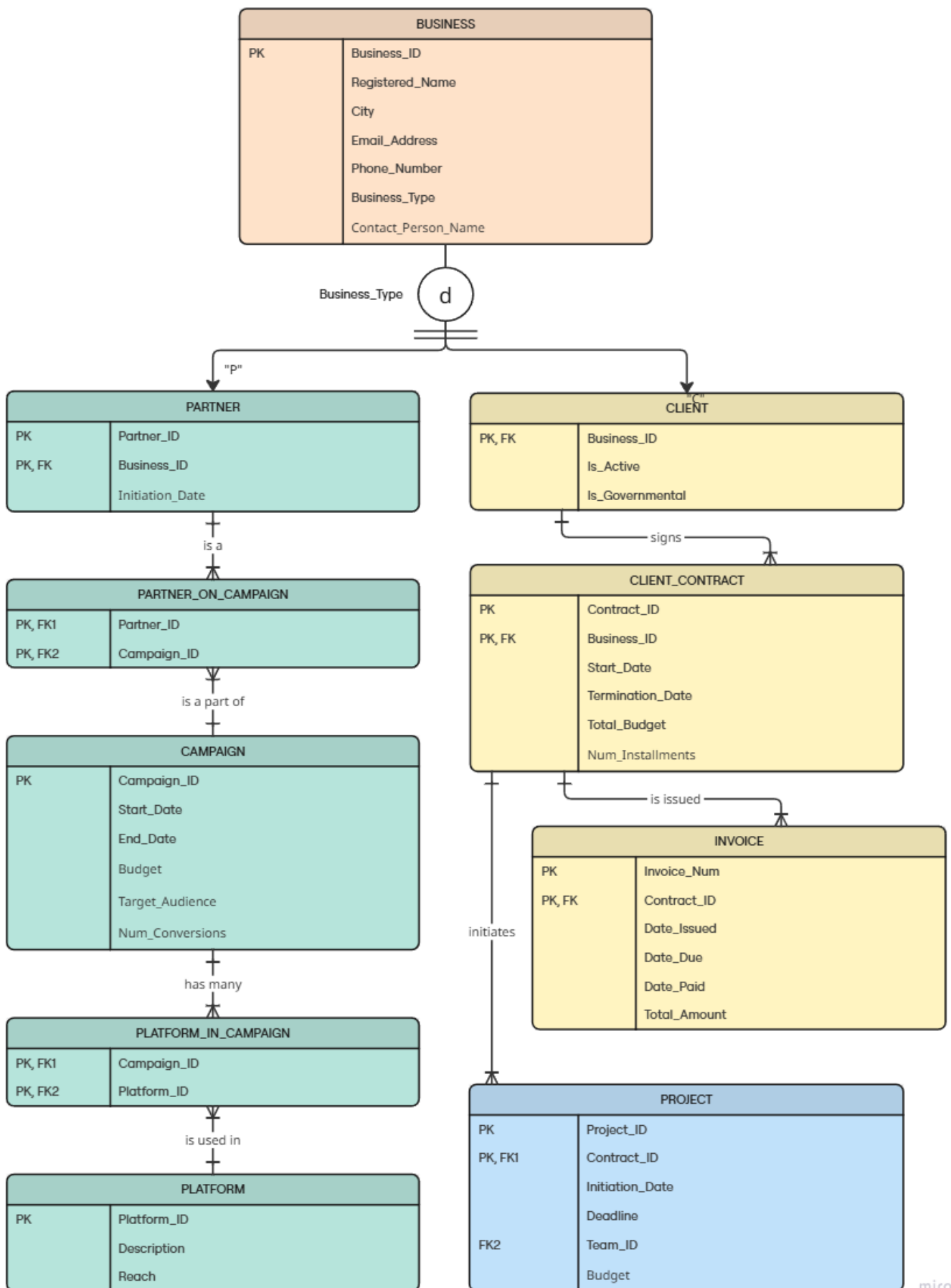
- Business identification number
- Whether they are active clients
- Whether they are governmental, state owned enterprise

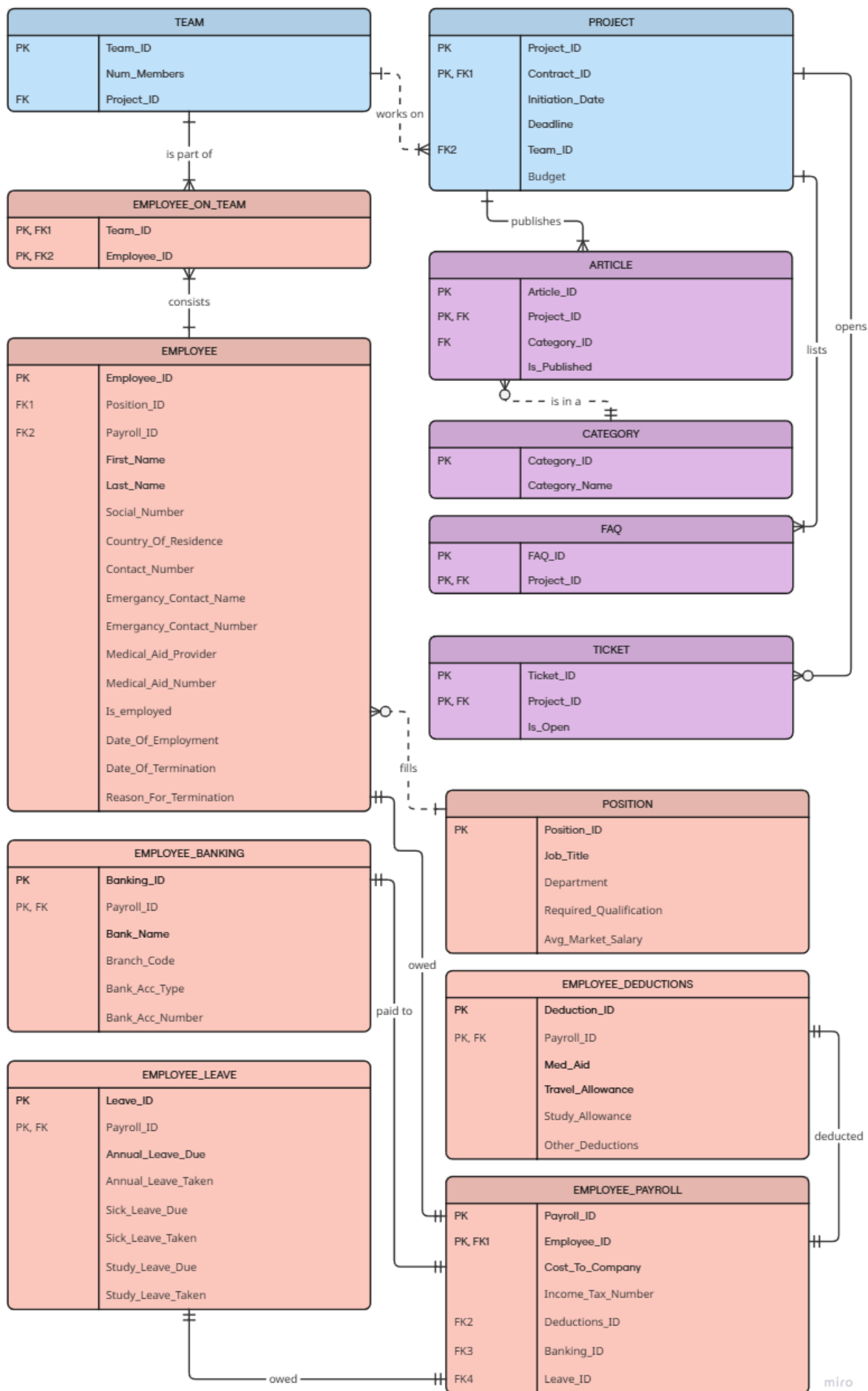
1.1.7 Information kept on employees

- First name
- Last name
- Social number (ID number or Passport number)
- Country of residence
- Contact number
- Emergency contact details
 - Contact name
 - Contact number
- Medical aid details
 - Provider

- Membership number
- Current company details
 - Current employment status
 - Date of employment
 - Date of termination
 - Reason for termination
- Income tax number
- Banking details
 - Bank name
 - Bank account type
 - Bank account number
 - Cost to company

1.2 ER Diagram





1.3 Notes on the ER diagram

Composite keys

- CONTRACT_ID and BUSINESS_ID in CLIENT_CONTRACT
- INVOICE_NUM and CONTRACT_ID in INVOICE
- DEDUCTION_ID and PAYROLL_ID in EMPLOYEE_DEDUCTIONS
- PAYROLL_ID and EMPLOYEE_ID in EMPLOYEE_PAYROLL
- BANKING_ID and PAYROLL_ID in EMPLOYEE_BANKING
- LEAVE_ID and PAYROLL_ID in EMPLOYEE_LEAVE
- TEAM_ID and EMPLOYEE_ID in EMPLOYEE_ON_TEAM
- PROJECT_ID and CONTRACT_ID in PROJECT
- ARTICLE_ID and PROJECT_ID in ARTICLE
- FAQ_ID and PROJECT_ID in FAQ
- TICKET_ID and PROJECT_ID in TICKET

Weak Relationships

- PROJECT to TEAM
- EMPLOYEE to POSITION
- ARTICLE to CATEGORY

Strong relationships

- BUSINESS supertype to its subtypes, CLIENT and PARTNER.
- CLIENT to CLIENT_CONTRACT
- CLIENT_CONTRACT to INVOICE
- CLIENT_CONTRACT to PROJECT
- PROJECT to TICKET
- PROJECT to ARTICLE
- PROJECT to FAQ
- PARTNER to PARTNER_ON_CAMPAIGN
- CAMPAIGN to PARTNER_ON_CAMPAIGN
- CAMPAIGN to PLATFORM_IN_CAMPAIGN
- PLATFORM to PLATFORM_IN_CAMPAIGN
- TEAM to EMPLOYEE_ON_TEAM
- EMPLOYEE to EMPLOYEE_ON_TEAM
- EMPLOYEE to EMPLOYEE_PAYROLL
- EMPLOYEE_PAYROLL to EMPLOYEE_BANKING
- EMPLOYEE_PAYROLL to EMPLOYEE_LEAVE
- EMPLOYEE_PAYROLL to EMPLOYEE_DEDUCTIONS

Weak Entities

- CLIENT
- CLIENT_CONTRACT
- INVOICE
- PROJECT
- PLATFORM_IN_CAMPAIGN
- PARTNER_ON_CAMPAIGN
- PARTNER

- ARTICLE
- FAQ
- TICKET
- EMPLOYEE_BANKING
- EMPLOYEE_LEAVE
- EMPLOYEE_PAYROLL
- EMPLOYEE_DEDUCTIONS
- EMPLOYEE_ON_TEAM

Composite/Bridge Entities

- EMPLOYEE_ON_TEAM
- PARTNER_ON_CAMPAIGN
- PLATFORM_IN_CAMPAIGN

Super entities

- BUSINESS

Sub-type entities

- CLIENT
- PARTNER

2. Logical Design

BUSINESS (**BUSINESS_ID(PK)**, REGISTERED_NAME, CITY, EMAIL_ADDRESS, PHONE_NUMBER, BUSINESS_TYPE, CONTACT_PERSON_NAME)

CLIENT (**BUSINESS_ID(PK)(FK)**, IS_ACTIVE, IS_GOVERNMENTAL)

CLIENT_CONTRACT (**CONTRACT_ID(PK)**, **BUSINESS_ID(PK)(FK)**, START_DATE, TERMINATION_DATE, TOTAL_BUDGET, NUM_INSTALLMENTS)

INVOICE (**INVOICE_NUM(PK)**, **CONTRACT_ID(PK)(FK)**, DATE_ISSUED, DATE_DUE, DATE_PAID, TOTAL_AMOUNT)

PROJECT (**PROJECT_ID(PK)**, **CONTRACT_ID(PK)(FK1)**, INITIATION_DATE, DEADLINE, **TEAM_ID(FK2)**, BUDGET)

TEAM (**TEAM_ID(PK)**, NUM_MEMBERS, PROJECT_ID)

ARTICLE(**ARTICLE_ID(PK)**, **PROJECT_ID(PK)(FK)**, IS_PUBLISHED, CATEGORY_ID)

CATEGORY (**CATEGORY_ID(PK)**, CATEGORY_NAME)

FAQ (**FAQ_ID(PK)**, **PROJECT_ID(PK)(FK)**)

TICKET (**TICKET_ID(PK)**, **PROJECT_ID(PK)(FK)**, IS_OPEN)

EMPLOYEE (**EMPLOYEE_ID(PK)**, **POSITION_ID(FK1)**, **PAYROLL_ID(FK2)**, FIRST_NAME, LAST_NAME, SOCIAL_NUMBER, COUNTRY_OF_RESIDENCE, CONTACT_NUMBER, EMERGENCY_CONTACT_NAME, EMERGENCY_CONTACT_NUMBER, MEDICAL_AID_PROVIDER, MEDICAL_AID_NUMBER, IS_EMPLOYED, DATE_OF_EMPLOYMENT, DATE_OF_TERMINATION, REASON_FOR_TERMINATION)

POSITION (**POSITION_ID(PK)**, JOB_TITLE, DEPARTMENT, REQUIRED_QUALIFICATION, AVG_MARKET_SALARY)

EMPLOYEE_PAYROLL (**PAYROLL_ID(PK)**, **EMPLOYEE_ID(PK,FK)**, COST_TO_COMPANY, TAX_INCOME_NUMBER, **DEDUCTIONS_ID(FK2)**, **BAKING_ID(FK3)**, **LEAVE_ID(FK4)**)

EMPLOYEE_DEDUCTIONS (**DEDUCTION_ID(PK)**, **PAYROLL_ID(PK,FK)**, MED_AID, TRAVEL_ALLOWANCE, STUDY_ALLOWANCE, OTHER_DEDUCTIONS)

EMPLOYEE_BANKING (**BANKING_ID(PK)**, **PAYROLL_ID(PK,FK)**, BANK_NAME, BRANCH_CODE, BANK_ACC_TYPE, BANK_ACC_NUMBER)

EMPLOYEE_LEAVE (LEAVE_ID(PK), PAYROLL_ID(PK,FK),
ANNUAL_LEAVE_DUE, ANNUAL_LEAVE_TAKEN, SICK_LEAVE_DUE,
SICK_LEAVE_TAKEN, STUDY_LEAVE_DUE, STUDY_LEAVE_TAKEN)

EMPLOYEE_ON_TEAM (TEAM_ID(PK,FK1), EMPLOYEE_ID(PK,FK2))

PARTNER (PARTNER_ID(PK), BUSINESS_ID(PK,FK), INITIATION_DATE)

PARTNER_ON_CAMPAIGN (PARTNER_ID(PK,FK1), CAMPAIGN_ID(PK,FK2))

CAMPAIGN (CAMPAIGN_ID(PK), START_DATE, END_DATE, BUDGET,
TARGET_AUDIENCE, NUM_CONVERSIONS)

PLATFORM_IN_CAMPAIGN (CAMPAIGN_ID(PK,FK1), PLATFORM_ID(PK,FK2))

PLATFORM (PLATFORM_ID(PK), DESCRIPTION, REACH)

Project Phase 3 – Physical Design

1 Database Objects

1.1. Dropping Tables

First, we drop tables to ensure there are no duplicates. Child tables who don't have any foreign keys stored elsewhere are dropped first to ensure no errors occur.

1. DROP TABLE INVOICE CASCADE CONSTRAINTS;
2. DROP TABLE CLIENT CASCADE CONSTRAINTS;
3. DROP TABLE PLATFORM_IN_CAMPAIGN CASCADE CONSTRAINTS;
4. DROP TABLE PARTNER_ON_CAMPAIGN CASCADE CONSTRAINTS;
5. DROP TABLE PLATFORM CASCADE CONSTRAINTS;
6. DROP TABLE CAMPAIGN CASCADE CONSTRAINTS;
7. DROP TABLE PARTNER CASCADE CONSTRAINTS;
8. DROP TABLE CLIENT_CONTRACT CASCADE CONSTRAINTS;
9. DROP TABLE PARTNER CASCADE CONSTRAINTS;
10. DROP TABLE BUSINESS CASCADE CONSTRAINTS;
11. DROP TABLE TEAM CASCADE CONSTRAINTS;
12. DROP TABLE CATEGORY CASCADE CONSTRAINTS;
13. DROP TABLE ARTICLE CASCADE CONSTRAINTS;
14. DROP TABLE FAQ CASCADE CONSTRAINTS;
15. DROP TABLE TICKET CASCADE CONSTRAINTS;
16. DROP TABLE PROJECT CASCADE CONSTRAINTS;
17. DROP TABLE EMPLOYEE CASCADE CONSTRAINTS;
18. DROP TABLE POSITION CASCADE CONSTRAINTS;
19. DROP TABLE EMPLOYEE_DEDUCTIONS CASCADE CONSTRAINTS;
20. DROP TABLE EMPLOYEE_PAYROLL CASCADE CONSTRAINTS;
21. DROP TABLE EMPLOYEE_BANKING CASCADE CONSTRAINTS;
22. DROP TABLE EMPLOYEE_LEAVE CASCADE CONSTRAINTS;
23. DROP TABLE EMPLOYEE_EDUCATION
24. DROP SEQUENCE seq_business_id;
25. DROP SEQUENCE seq_contract_id;
26. DROP SEQUENCE seq_invoice_num;
27. DROP SEQUENCE seq_project_id;
28. DROP SEQUENCE seq_team_id;
29. DROP SEQUENCE seq_position_id;
30. DROP SEQUENCE seq_article_id;
31. DROP SEQUENCE seq_category_id;
32. DROP SEQUENCE seq_faq_id;
33. DROP SEQUENCE seq_ticket_id;

34.	DROP SEQUENCE	seq_banking_id;
35.	DROP SEQUENCE	seq_position_id;
36.	DROP SEQUENCE	seq_employee_id;
37.	DROP SEQUENCE	seq_payroll_id;
38.	DROP SEQUENCE	seq_deduction_id;
39.	DROP SEQUENCE	seq_leave_id;
40.	DROP SEQUENCE	seq_banking_id;

1.2. Table Creation

1.2.1. Business Table

Business is the supertype for Client and Partner subtypes and establishes some basic information the company wants to collect and store. Some information such as City, Email Address, Phone_Number, and Contact_Person_Name can be left NULL in case it must be collected later, but the Registered_Name Business_Type must always be defined (along with the primary key, Business_ID).

CREATE TABLE BUSINESS
(
Business_ID NUMBER PRIMARY KEY,
Registered_Name VARCHAR2(100) NOT NULL,
City VARCHAR2(50),
Email_Address VARCHAR2(100),
Phone_Number VARCHAR2(20),
Business_Type CHAR(1) CHECK (Business_Type IN ('C', 'P')) NOT NULL,
Contact_Person_Name VARCHAR2(100)
);

The following sequence is created to define the business ID primary key, Business_ID:

CREATE SEQUENCE seq_business_id START WITH 1001 INCREMENT BY 1;

1.2.2. Partner Table

The PARTNER table is a subtype of BUSINESS and a super-type to the PARTNER_ON_CAMPAIGN. There can only be one partner on a campaign.

CREATE TABLE PARTNER
(
Business_ID NUMBER PRIMARY KEY,
Initiation_Date DATE,
FOREIGN KEY (Business_ID) REFERENCES BUSINESS(Business_ID)
);

The following sequence is created to define the partner ID primary key, Partner_ID:

CREATE SEQUENCE seq_partner_id START WITH 300 INCREMENT BY 1;

1.2.3. Partner on campaign Table

The PARTNER_ON_CAMPAIGN table is a subtype of PARTNER. There is only one partner on a campaign, a partner on campaign can only be part of one campaign.

CREATE TABLE PARTNER_ON_CAMPAIGN
(
Business_ID NUMBER,
Campaign_ID NUMBER,
CONSTRAINT pk_partners_on_campaign PRIMARY KEY (Business_ID,
Campaign_ID),
CONSTRAINT fk_poc_partner FOREIGN KEY (Business) REFERENCES
Partner (Business_ID),
CONSTRAINT fk_poc_campaign FOREIGN KEY (Campaign_ID)
REFERENCES Campaign (Campaign_ID)
);

No sequences will be created for PARTNER_ON_CAMPAIGN because it uses Partner_ID and Campaign_ID as primary keys.

1.2.4. Campaign Table

The CAMPAIGN table is the super-type to PLATFORM_IN_CAMPAIGN. There is one Partner per Campaign and each Campaign may have multiple Platforms.

CREATE TABLE CAMPAIGN
(
Campaign_ID NUMBER PRIMARY KEY,
Start_Date DATE NOT NULL,
End_Date DATE NOT NULL,
Budget DECIMAL NOT NULL,
Target_Audience VARCHAR(32),
Num_Conversions INT
);

The following sequence is created to define the campaign ID primary key, Campaign_ID: Campaign_ID

CREATE SEQUENCE seq_campaign_id START WITH 300 INCREMENT BY 1;

1.2.5. Platform Table

Create the Platform table that houses the reach and description that each platform has.

CREATE TABLE PLATFORM
(
Platform_ID NUMBER PRIMARY KEY,

Description VARCHAR(256),
Reach DECIMAL
);

The following sequence is created to define the platform ID primary key, Platform_ID:
Platform_ID

CREATE SEQUENCE seq_platform_id START WITH 300 INCREMENT BY 1;

1.2.6. Platform_In_Campaign Table

The Platform in campaign table links the campaign and platform tables with its constraints.

CREATE TABLE PLATFORM_IN_CAMPAIGN
(
Campaign_ID NUMBER,
Platform_ID NUMBER,
CONSTRAINT pk_platform_in_campaign PRIMARY KEY (Campaign_ID,
Platform_ID),
CONSTRAINT fk_pic_campaign FOREIGN KEY (Campaign_ID)
REFERENCES Campaign (Campaign_ID),
CONSTRAINT fk_pic_platform FOREIGN KEY (Platform_ID) REFERENCES
Platform (Platform_ID)
);

No sequences will be created for PLATFORM_IN_CAMPAIGN because it uses Campaign_ID and Platform_ID as primary keys.

1.2.7. Client Table

A Client is a subtype of the Business supertype. It stores further information specific to a Client of JUST-GUARD, specifically whether they are an active client (one with a contract that has not expired) and whether they are governmental or not (due to specific regulatory requirements for government contracts).

CREATE TABLE CLIENT
(
Business_ID NUMBER PRIMARY KEY,
Is_Active CHAR(1) CHECK (Is_Active IN ('Y', 'N')) NOT NULL,
Is_Governmental CHAR(1) CHECK (Is_Governmental IN ('Y', 'N')) NOT NULL,
FOREIGN KEY (Business_ID) REFERENCES BUSINESS(Business_ID)
);

No sequences are created for Client since it uses the Business_ID as its primary key.

1.2.8. Client_Contract Table

The Client_Contract tables stores information regarding contracts signed by a Client and JUST-GUARD. Contracts contain a plethora of information which is why the database only stores key information such as the client's ID, various dates, and financial information.

CREATE TABLE CLIENT_CONTRACT
(

Contract_ID NUMBER PRIMARY KEY,
Business_ID NUMBER NOT NULL,
Start_Date DATE NOT NULL,
Termination_Date DATE NOT NULL,
Total_Budget NUMBER(12, 2) NOT NULL,
Num_Installments NUMBER NOT NULL,
FOREIGN KEY (Business_ID) REFERENCES BUSINESS(Business_ID)
);

The following sequence is created to define the contract ID primary key, Contract_ID:

CREATE SEQUENCE seq_contract_id START WITH 2001 INCREMENT BY 1;

1.2.9. Invoice Table

The Invoice table stores invoices issued to Clients based on the budget, instalments, and dates set out in the Client_Contract. It stores information on various dates and the amount of the invoice calculated from the budget and instalments information.

CREATE TABLE INVOICE
(
Invoice_Num NUMBER PRIMARY KEY,
Contract_ID NUMBER NOT NULL,
Date_Issued DATE NOT NULL,
Date_Due DATE NOT NULL,
Date_Paid DATE,
Total_Amount NUMBER(12, 2) NOT NULL,
FOREIGN KEY (Contract_ID) REFERENCES CLIENT_CONTRACT(Contract_ID)
);

The following sequence is created to define the invoice number primary key, Invoice_Num:

CREATE SEQUENCE seq_invoice_num START WITH 3001 INCREMENT BY 1;

1.2.10. Project Table

The project table stores information on projects that are initiated by a Client_Contract. It keeps track of the budget, the deadline and the initiation date. A Team works on a Project.

CREATE TABLE PROJECT
(
Project_ID NUMBER PRIMARY KEY,
Contract_ID NUMBER NOT NULL,
Initiation_Date DATE NOT NULL,
Deadline DATE NOT NULL,
Team_ID NUMBER NOT NULL,
Budget NUMBER(12,2) NOT NULL,
FOREIGN KEY (Contract_ID) REFERENCES CLIENT_CONTRACT(Contract_ID),
FOREIGN KEY (Team_ID) REFERENCES TEAM(Team_ID)
);

The following sequence is created to define the project ID primary key, Project_ID:

CREATE SEQUENCE seq_project_id START WITH 100 INCREMENT BY 1;

1.2.11. Team Table

The team table keeps track of the amount of team members. A Team works on a Project.

CREATE TABLE TEAM
(
Team_ID NUMBER PRIMARY KEY,
Project_ID NUMBER NOT NULL,
Num_Members NUMBER NOT NULL,
FOREIGN KEY (Project_ID) REFERENCES PROJECT(Project_ID)
);

The following sequence is created to define the team ID primary key, Team_ID:

CREATE SEQUENCE seq_team_id START WITH 100 INCREMENT BY 1;

1.2.12. Article Table

The Article table keeps track of information regarding the project. A Project publishes an Article

CREATE TABLE ARTICLE
(
Article_ID NUMBER PRIMARY KEY,
Project_ID NUMBER NOT NULL,
Category_ID NUMBER NOT NULL,
Title VARCHAR2(100),
Publish_Date DATE,
FOREIGN KEY (Category_ID) REFERENCES Category (Category_ID),
FOREIGN KEY (Project_ID) REFERENCES Project (Project_ID)
);

The following sequence is created to define the article ID primary key, Article_ID:

CREATE SEQUENCE seq_article_id START WITH 100 INCREMENT BY 1;

1.2.13. Category Table

The Category table is responsible for categorizing the different articles with a specific category name. Articles are in Categories

CREATE TABLE CATEGORY
(
Category_ID NUMBER PRIMARY KEY,
Category_Name VARCHAR2(100)
);

The following sequence is created to define the category ID primary key, Category_ID:

CREATE SEQUENCE seq_category_ID START WITH 100 INCREMENT BY 1;

1.2.14. FAQ Table

The FAQ table keeps track of questions to certain projects. The Project lists a FAQ.

```
CREATE TABLE FAQ
```

```
(
```

```
FAQ_ID NUMBER PRIMARY KEY,
```

```
Project_ID NUMBER NOT NULL,
```

```
Question VARCHAR2(255)
```

```
);
```

The following sequence is created to define the FAQ ID primary key, FAQ_ID:

```
CREATE SEQUENCE seq_faq_id START WITH 100 INCREMENT BY 1;
```

1.2.15. Ticket Table

The ticket table keeps track of user issues. A Project opens tickets.

```
CREATE TABLE TICKET
```

```
(
```

```
Ticket_ID NUMBER PRIMARY KEY,
```

```
Project_ID NUMBER NOT NULL,
```

```
Is_Open CHAR(1),
```

```
Create_Date DATE,
```

```
Description VARCHAR2(255)
```

```
);
```

The following sequence is created to define the ticket ID primary key, Ticket_ID:

```
CREATE SEQUENCE seq_ticket_id START WITH 100 INCREMENT BY 1;
```

1.2.16. Position Table

The Position table keeps track of the Positions that Employees fill. It keeps track of the Job Title, Department, Required Qualification and Average Market Salary. A Position can be filled by none to many Employees.

```
CREATE TABLE POSITION
```

```
(
```

```
Position_ID NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
```

```
Job_Title VARCHAR2(100) NOT NULL,
```

```
Department VARCHAR2(100) NOT NULL,
```

```
Required_Qualification VARCHAR2(100) NOT NULL,
```

```
Avg_Market_Salary NUMBER(12,2) NOT NULL
```

```
);
```

The following sequence is created to define the position ID primary key, Position_ID:

```
CREATE SEQUENCE seq_position_id START WITH 1 INCREMENT BY 1;
```

1.2.17. Employee Table

The Employee table keeps track of employee related information. It keeps track of their full name, ID or Passport number, The country they are a resident of, an emergency contact, medical aid information (if provided) and their employment status, including date started, date terminated and reason for termination. One employee is a member of one or more teams.

CREATE TABLE EMPLOYEE
(
Employee_ID NUMBER PRIMARY KEY,
First_Name VARCHAR2(100) NOT NULL,
Last_Name VARCHAR2(100) NOT NULL,
Social_Number NUMBER NOT NULL,
Country_Of_Residence VARCHAR2(100) NOT NULL,
Contact_Number VARCHAR2(20) NOT NULL,
Emergency_Contact_Name VARCHAR2(100) NOT NULL,
Emergency_Contact_Number VARCHAR2(20) NOT NULL,
Medical_Aid_Provider VARCHAR2(100),
Medical_Aid_Number NUMBER,
Is_Employed CHAR(1) NOT NULL,
Date_Of_Employment DATE NOT NULL,
Date_Of_Termination DATE,
Reason_For_Termination VARCHAR2(255),
Position_ID NUMBER,
Payroll_ID NUMBER
);

The following sequence is created to define the employee ID primary key, Employee_ID:

CREATE SEQUENCE seq_employee_id START WITH 1 INCREMENT BY 1;

1.2.18. Employee_On_Team Table

The Employee_On_Team table is a junction table between Employee and Team. An Employee can be a part of multiple Teams and a Team can have multiple employees.

CREATE TABLE EMPLOYEE_ON_TEAM
(
Team_ID NUMBER,
Employee_ID NUMBER,
CONSTRAINT pk_employee_on_team PRIMARY KEY (Team_ID,
Employee_ID),
CONSTRAINT fk_team FOREIGN KEY (Team_ID) REFERENCES
TEAM(Team_ID),
CONSTRAINT fk_employee FOREIGN KEY (Employee_ID) REFERENCES
EMPLOYEE(Employee_ID)
);

No sequences will be created for EMPLOYEE_ON_TEAM because it uses Team_ID and Employee_ID as primary keys.

1.2.19. Employee_Payroll Table

The Employee_Payroll table keeps track of the financial side of employees information, its the parent entity of the Employee_Deductions, Employee_Banking and Employee_Leave tables. It stores information about an Employee's Cost to Company and their Tax Income number. Each employee is owed restitution from the employee payroll.

```
CREATE TABLE EMPLOYEE_PAYROLL
(
    Payroll_ID NUMBER PRIMARY KEY,
    Employee_ID NUMBER NOT NULL,
    Cost_To_Company NUMBER(6, 2) NOT NULL,
    Tax_Income_Number NUMBER NOT NULL,
    Deductions_ID NUMBER NOT NULL,
    Banking_ID NUMBER NOT NULL,
    Leave_ID NUMBER NOT NULL
);
```

The following sequence is created to define the payroll ID primary key, Payroll_ID:

```
CREATE SEQUENCE seq_payroll_id START WITH 1 INCREMENT BY 1;
```

1.2.20. Employee_Deductions Table

The Employee Deductions table stores information on finances that the company withholds from the employee. This is deductions such as company supplied medical aid, Travel allowance, Study allowance and other deductions (Unforseen deductions such as advances in salary or garnishing's, etc etc). A deduction can be set to 0, but each employee on the payroll has deductions.

```
CREATE TABLE EMPLOYEE_DEDUCTIONS
(
    Deduction_ID NUMBER PRIMARY KEY,
    Payroll_ID NUMBER NOT NULL,
    Med_Aid NUMBER NOT NULL,
    Travel_Allowance NUMBER NOT NULL,
    Study_Allowance NUMBER(6, 2) NOT NULL,
    Other_Deductions NUMBER(6, 2) NOT NULL,
    FOREIGN KEY (Payroll_ID) REFERENCES
    EMPLOYEE_PAYROLL(Payroll_ID)
);
```

The following sequence is created to define the deduction ID primary key, Deduction_ID:

```
CREATE SEQUENCE seq_deduction_id START WITH 1 INCREMENT BY 1;
```

1.2.21. Employee_Leave Table

The employee leave table stores information about paid time off owed to the employee. This can be annual leave, Sick leave and Study leave. Each Employee on the payroll is owed leave.

```
CREATE TABLE EMPLOYEE_LEAVE
(
  Leave_ID NUMBER PRIMARY KEY,
  Payroll_ID NUMBER NOT NULL,
  Study_Leave_Taken NUMBER,
  Annual_Leave_Due NUMBER NOT NULL,
  Annual_Leave_Taken NUMBER,
  Sick_Leave_Due NUMBER NOT NULL,
  Sick_Leave_Taken NUMBER,
  Study_Leave_Due NUMBER,
  FOREIGN KEY (Payroll_ID) REFERENCES
  EMPLOYEE_PAYROLL(Payroll_ID)
);
```

The following sequence is created to define the leave ID primary key, Leave_ID:

```
CREATE SEQUENCE seq_leave_id START WITH 1 INCREMENT BY 1;
```

1.2.22. Employee_Banking Table

The employee banking table keeps track of an employee's bank details for payment. It holds information such as Bank name, Branch Code, Bank account type and Bank account number. Each employee on the payroll must have one bank account.

```
CREATE TABLE EMPLOYEE_BANKING
(
  Banking_ID NUMBER PRIMARY KEY,
  Bank_Name VARCHAR2(100) NOT NULL,
  Payroll_ID NUMBER NOT NULL,
  Branch_Code NUMBER NOT NULL,
  Bank_Acc_Type VARCHAR2(20) NOT NULL,
  Bank_Acc_Number NUMBER NOT NULL,
  FOREIGN KEY (Payroll_ID) REFERENCES
  EMPLOYEE_PAYROLL(Payroll_ID)
);
```

The following sequence is created to define the banking ID primary key, Banking_ID:

```
CREATE SEQUENCE seq_banking_id START WITH 1 INCREMENT BY 1;
```

To avoid collisions or other errors, we alter the above table SQL as such:

```
ALTER TABLE PROJECT ADD CONSTRAINT fk_proj_contract FOREIGN KEY
(Contract_ID) REFERENCES CLIENT_CONTRACT(Contract_ID);
```

ALTER TABLE PROJECT ADD CONSTRAINT fk_proj_team FOREIGN KEY (Team_ID) REFERENCES TEAM(Team_ID);
ALTER TABLE TEAM ADD CONSTRAINT fk_team_project FOREIGN KEY (Project_ID) REFERENCES PROJECT(Project_ID);
ALTER TABLE EMPLOYEE ADD CONSTRAINT fk_employee_position FOREIGN KEY (Position_ID) REFERENCES POSITION(Position_ID);
ALTER TABLE EMPLOYEE ADD CONSTRAINT fk_employee_payroll FOREIGN KEY (Payroll_ID) REFERENCES EMPLOYEE_PAYROLL(Payroll_ID);
ALTER TABLE EMPLOYEE_PAYROLL ADD CONSTRAINT fk_payroll_employee FOREIGN KEY (Employee_ID) REFERENCES EMPLOYEE(Employee_ID);
ALTER TABLE EMPLOYEE_PAYROLL ADD CONSTRAINT fk_payroll_deductions FOREIGN KEY (Deductions_ID) REFERENCES EMPLOYEE_DEDUCTIONS(Deductions_ID);
ALTER TABLE EMPLOYEE_PAYROLL ADD CONSTRAINT fk_payroll_banking FOREIGN KEY (Banking_ID) REFERENCES EMPLOYEE_BANKING(Banking_ID);
ALTER TABLE EMPLOYEE_PAYROLL ADD CONSTRAINT fk_payroll_leave FOREIGN KEY (Leave_ID) REFERENCES EMPLOYEE_LEAVE(Leave_ID);

And add the following triggers:

```
CREATE OR REPLACE TRIGGER trg_partner_check BEFORE INSERT OR UPDATE ON
Partner FOR EACH ROW DECLARE v_type Business.Business_Type%TYPE; BEGIN
SELECT Business_Type INTO v_type FROM Business WHERE Business_ID
= :NEW.Business_ID;
IF v_type <> 'P' THEN RAISE_APPLICATION_ERROR( -20001, 'Cannot add Partner row:
Business_Type must be "P".'); END IF; END;

CREATE OR REPLACE TRIGGER trg_client_check BEFORE INSERT OR UPDATE ON
Client FOR EACH ROW DECLARE v_type Business.Business_Type%TYPE; BEGIN
SELECT Business_Type INTO v_type FROM Business WHERE Business_ID
= :NEW.Business_ID;
IF v_type <> 'C' THEN RAISE_APPLICATION_ERROR( -20002, 'Cannot add Client row:
Business_Type must be "C".'); END IF; END;
```

1.2. Indexes

Because primary keys are automatically indexed, we will only add indexes on other fields that we determine will be used often. These indexes will help the database user to find key information that is accessed often quickly without consuming a lot of database resources or time.

-- Index on BUSINESS Registered_Name (often searched)

```
CREATE INDEX idx_business_name ON BUSINESS (Registered_Name);
```

-- Index on CLIENT Is_Active (to speed up filtering by active/inactive)

```
CREATE INDEX idx_client_active ON CLIENT (Is_Active);
```

-- Index on CLIENT_CONTRACT Start_Date (for date range filtering)

```
CREATE INDEX idx_contract_start_date ON CLIENT_CONTRACT (Start_Date);
```

```
-- Index on INVOICE Date_Paid (to check payment status quickly)
CREATE INDEX idx_invoice_paid ON INVOICE (Date_Paid);

-- Index on PROJECT Deadline (to quickly check the projects deadline date)
CREATE INDEX idx_project_deadline ON PROJECT (Deadline);

-- Index on POSITION Job_Title (To quickly see positions in company)
CREATE INDEX idx_company_positions ON Position (Job_Title);

-- Index on EMPLOYEE Social_Number (to quickly find an employee based on
government identification)
CREATE INDEX idx_employee_social ON EMPLOYEE (Social_Number);

-- Index on CAMPAIGN Budget (regularly searched)
CREATE INDEX idx_campaign_budget ON CAMPAIGN (Budget);

-- Index on CAMPAIGN Num_Conversions (for fast performance analysis)
CREATE INDEX idx_campaign_conversions ON CAMPAIGN (Num_Conversions);
```

1.3. Views

1.3.1. Active Clients View

This view shows active clients along with their business details for easy overview of JUST-GUARD's current clients.

CREATE OR REPLACE VIEW vw_Active_Clients AS
SELECT
b.Business_ID,
b.Registered_Name,
b.City,
b.Email_Address,
c.Is_Active,
c.Is_Governmental
FROM
BUSINESS b
JOIN CLIENT c ON b.Business_ID = c.Business_ID;

Output:

	BUSINESS_ID	REGISTERED_NAME	CITY	EMAIL_ADDRESS	IS_ACTIVE	IS_GOVERNMENTAL
1	1001	Starlight Solutions	Johannesburg	contact@starlight.com	Y	N
2	1002	Ministry of Technology	Cape Town	hello@mintech.gov	Y	Y
3	1003	BrightSky Analytics	Bloemfontein	team@brightsky.io	Y	N

1.3.2. Client Contract View

This view shows active clients' name along with all their contracts for easy overview.

CREATE OR REPLACE VIEW vw_Client_Contracts AS

```

SELECT
  b.Registered_Name,
  cc.Contract_ID,
  cc.Start_Date,
  cc.Termination_Date,
  cc.Total_Budget,
  cc.Num_Installments
FROM
  CLIENT_CONTRACT cc
JOIN BUSINESS b ON cc.Business_ID = b.Business_ID;

```

Output:

	REGISTERED_NAME	CONTRACT_ID	START_DATE	TERMINATION_DATE	TOTAL_BUDGET	NUM_INSTALLMENTS
1	Starlight Solutions	2001	01/FEB/25	31/DEC/25	120000	10
2	Ministry of Technology	2003	01/JUN/25	31/OCT/25	50000	5
3	Ministry of Technology	2002	01/MAR/25	31/MAY/25	51000	3
4	BrightSky Analytics	2005	01/FEB/25	30/JUN/26	60000	5
5	BrightSky Analytics	2004	01/JAN/25	28/FEB/25	18000	2

1.3.3. Invoice Payment Status View

This view shows all invoices and their payment statuses for easy overview of which are paid and which are outstanding.

```

CREATE OR REPLACE VIEW vw_Invoice_Status AS
SELECT
  i.Invoice_Num,
  i.Contract_ID,
  i.Date_Issued,
  i.Date_Due,
  i.Date_Paid,
  CASE
    WHEN i.Date_Paid IS NULL THEN 'Unpaid'
    ELSE 'Paid'
  END AS Payment_Status,
  i.Total_Amount
FROM
  INVOICE i;

```

Output:

	INVOICE_NUM	CONTRACT_ID	DATE_ISSUED	DATE_DUE	DATE_PAID	PAYMENT_STATUS	TOTAL_AMOUNT
1	3011	2001	01/FEB/25	28/FEB/25	25/FEB/25	Paid	12000
2	3012	2001	01/MAR/25	31/MAR/25	29/MAR/25	Paid	12000
3	3013	2001	01/APR/25	30/APR/25	(null)	Unpaid	12000
4	3014	2002	01/MAR/25	28/FEB/25	30/MAR/25	Paid	17000
5	3015	2002	01/APR/25	30/APR/25	(null)	Unpaid	17000
6	3016	2004	01/JAN/25	31/JAN/25	30/JAN/25	Paid	9000
7	3017	2004	01/FEB/25	28/FEB/25	28/FEB/25	Paid	9000

1.3.4. Project View

The view shows the information on a project as well as the Team that is currently working on it, the Articles published on it, the FAQ and the Tickets currently open on said project.

CREATE OR REPLACE VIEW vw_Project_Deadlines AS
SELECT
p.Project_ID,
p.Contract_ID,
p.Initiation_Date,
p.Deadline,
p.Team_ID,
p.Budget,
a.Article_ID,
a.Category_ID as Article_Category_ID,
a.Is_Published,
f.FAQ_ID,
t.Ticket_ID,
t.Is_Open
FROM
PROJECT p
LEFT JOIN ARTICLE a ON p.Project_ID = a.Project_ID
LEFT JOIN FAQ f ON p.Project_ID = f.Project_ID
LEFT JOIN TICKET t ON p.Project_ID = t.Project_ID
LEFT JOIN TEAM tm ON p.Team_ID = tm.Team_ID
WHERE
p.Deadline > SYSDATE;

Output:

	PROJECT_ID	CONTRACT_ID	INITIATION_DATE	DEADLINE	TEAM_ID	BUDGET	ARTICLE_ID	ARTICLE_CATEGORY_ID	PUBLISH_DATE	FAQ_ID	TICKET_ID	IS_OPEN
1	100		2001 01/JAN/25	30/JUN/25	100	80000	101		100 15/JAN/24	100	100 Y	
2	101		2002 01/MAR/24	31/DEC/25	101	150000	102		101 20/FEB/25	101	101 Y	
3	102		2004 01/JAN/25	10/MAY/25	102	40000	100		102 30/APR/25	102	102 Y	
4	102		2004 01/JAN/25	10/MAY/25	102	40000	103		102 15/MAR/25	102	102 Y	
5	103		2005 01/FEB/25	31/OCT/25	103	100000	(null)	(null)	(null)	(null)	(null)	(null)
6	104		2005 01/MAY/25	31/OCT/25	103	50000	(null)	(null)	(null)	(null)	(null)	(null)

1.3.4. Position view

This view shows information about a the “Senior .net developer” position within the company

CREATE OR REPLACE VIEW vw_Senior_C_Developers AS
SELECT
Job_Title,
Department,
Required_Qualification,
Avg_Market_Salary
FROM

POSITION
WHERE
Job_Title = 'Senior .Net Developer';

Output:

JOB_TITLE	DEPARTMENT	REQUIRED_QUALIFICATION	AVG_MARKET_SALARY
1 Senior .Net Developer	Software Development	BSC IT	30000

1.3.5. Leave view

This view shows information about employees that have more than 15 days leave due.

CREATE OR REPLACE VIEW vw_Overdue_Leave AS
SELECT
el.Leave_ID,
el.Annual_Leave_Due,
ep.Payroll_ID,
ep.Employee_ID,
e.First_Name,
e.Last_Name,
e.Social_Number
FROM
EMPLOYEE_PAYROLL ep
RIGHT JOIN EMPLOYEE_LEAVE el ON el.Payroll_ID = ep.Payroll_ID
RIGHT JOIN EMPLOYEE e ON ep.Payroll_ID = e.Employee_ID
WHERE
el.Annual_Leave_Due > 14;

Output:

	LEAVE_ID	ANNUAL_LEAVE_DUE	PAYROLL_ID	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SOCIAL_NUMBER
1	1	15	1	1	Daniel	De Jager	1122334444551
2	2	20	2	2	Jurie	Blom	12341111123
3	3	18	3	3	Gaby	Gloss	987654321

1.3.6 Campaign view

The Campaign Platform view provides details that are relevant and often requested by people inside and outside the organisation.

CREATE OR REPLACE VIEW vw_Campaign_Platform AS
SELECT
c.Campaign_ID,
c.End_Date,
c.Budget,
c.Num_Conversions,
c.Target_Audience,
p.Platform_ID,
p.Description AS Platform_Description,

```

p.Reach AS Platform_Reach
FROM
CAMPAIGN c
JOIN PLATFORM_IN_CAMPAIGN pic ON c.Campaign_ID = pic.Campaign_ID
JOIN PLATFORM p ON pic.Platform_ID = p.Platform_ID;

```

Output:

	CAMPAIGN_ID	END_DATE	BUDGET	NUM_CONVERSIONS	TARGET_AUDIENCE	PLATFORM_ID	PLATFORM_DESCRIPTION	PLATFORM_REACH
1	300	30/JAN/25	50000	1250	Young Adults	302	Search Engine	100000
2	301	30/MAR/25	60000	1500	Families	300	Social Media	20000
3	302	15/AUG/25	75000	1800	Professionals	301	Email Marketing	30000

2. Data

2.1. Loading

2.1.1. Loading the Business Table

```

INSERT INTO BUSINESS (Business_ID, Registered_Name, City, Email_Address,
Phone_Number, Business_Type, Contact_Person_Name)
VALUES (SEQ_BUSINESS_ID.NEXTVAL, 'Starlight Solutions', 'Johannesburg',
'contact@starlight.com', '0234539694', 'P', 'Alice Johnson');
INSERT INTO BUSINESS (Business_ID, Registered_Name, City, Email_Address,
Phone_Number, Business_Type, Contact_Person_Name)
VALUES (SEQ_BUSINESS_ID.NEXTVAL, 'Ministry of Technology', 'Cape Town',
'hello@mintech.gov', '0233454203', 'P', 'Sir David Carter');
INSERT INTO BUSINESS (Business_ID, Registered_Name, City, Email_Address,
Phone_Number, Business_Type, Contact_Person_Name)
VALUES (SEQ_BUSINESS_ID.NEXTVAL, 'BrightSky Analytics', 'Bloemfontein',
'team@brightsky.io', '0866555908', 'P', 'Emily Zhang');
INSERT INTO BUSINESS (Business_ID, Registered_Name, City, Email_Address,
Phone_Number, Business_Type, Contact_Person_Name)
VALUES (SEQ_BUSINESS_ID.NEXTVAL, 'GreenTech Innovations', 'Durban',
'info@greentech.com', '0728193018', 'C', 'Brian Lee');
INSERT INTO BUSINESS (Business_ID, Registered_Name, City, Email_Address,
Phone_Number, Business_Type, Contact_Person_Name)
VALUES (SEQ_BUSINESS_ID.NEXTVAL, 'BlueWave Industries', 'Pretoria',
'support@bluewave.com', '0602949204', 'C', 'Carla Smith');
INSERT INTO BUSINESS (Business_ID, Registered_Name, City, Email_Address,
Phone_Number, Business_Type, Contact_Person_Name)
VALUES (SEQ_BUSINESS_ID.NEXTVAL, 'NovaMed Health', 'Port Elizabeth',
'contact@novamed.org', '0839348127', 'C', 'Dr. Thomas White');

```

2.1.2. Loading the Client Table

```

INSERT INTO CLIENT (Business_ID, Is_Active, Is_Governmental)
VALUES (1001, 'Y', 'N');
INSERT INTO CLIENT (Business_ID, Is_Active, Is_Governmental)

```

VALUES (1002, 'Y', 'Y');
INSERT INTO CLIENT (Business_ID, Is_Active, Is_Governmental)
VALUES (1003, 'Y', 'N');

2.1.3. Loading the Client_Contract Table

INSERT INTO CLIENT_CONTRACT (Contract_ID, Business_ID, Start_Date, Termination_Date, Total_Budget, Num_Installments)
VALUES (SEQ_CONTRACT_ID.NEXTVAL, 1001, DATE '2025-02-01', DATE '2025-12-31', 120000.00, 10);
INSERT INTO CLIENT_CONTRACT (Contract_ID, Business_ID, Start_Date, Termination_Date, Total_Budget, Num_Installments)
VALUES (SEQ_CONTRACT_ID.NEXTVAL, 1002, DATE '2025-03-01', DATE '2025-05-31', 51000.00, 3);
INSERT INTO CLIENT_CONTRACT (Contract_ID, Business_ID, Start_Date, Termination_Date, Total_Budget, Num_Installments)
VALUES (SEQ_CONTRACT_ID.NEXTVAL, 1002, DATE '2025-06-01', DATE '2025-10-31', 50000.00, 5);
INSERT INTO CLIENT_CONTRACT (Contract_ID, Business_ID, Start_Date, Termination_Date, Total_Budget, Num_Installments)
VALUES (SEQ_CONTRACT_ID.NEXTVAL, 1003, DATE '2025-01-01', DATE '2025-02-28', 18000.00, 2);
INSERT INTO CLIENT_CONTRACT (Contract_ID, Business_ID, Start_Date, Termination_Date, Total_Budget, Num_Installments)
VALUES (SEQ_CONTRACT_ID.NEXTVAL, 1003, DATE '2025-02-01', DATE '2026-06-30', 60000.00, 5);

2.1.4. Loading the Invoice Table

When loading the invoice table, we create one invoice for each month past that was part of a contract (including the current month if it's part of an ongoing contract) and divide the total budget by the number of instalments to get the invoice amount.

Contract 2001:

INSERT INTO INVOICE (Invoice_Num, Contract_ID, Date_Issued, Date_Due, Date_Paid, Total_Amount)
VALUES (SEQ_INVOICE_NUM.NEXTVAL, 2001, DATE '2025-02-01', DATE '2025-02-28', DATE '2025-02-25', 12000.00);
INSERT INTO INVOICE (Invoice_Num, Contract_ID, Date_Issued, Date_Due, Date_Paid, Total_Amount)
VALUES (SEQ_INVOICE_NUM.NEXTVAL, 2001, DATE '2025-03-01', DATE '2025-03-31', DATE '2025-03-29', 12000.00);
INSERT INTO INVOICE (Invoice_Num, Contract_ID, Date_Issued, Date_Due, Date_Paid, Total_Amount);
VALUES (SEQ_INVOICE_NUM.NEXTVAL, 2001, DATE '2025-04-01', DATE '2025-04-30', NULL, 12000.00);

Contract 2002:

INSERT INTO INVOICE (Invoice_Num, Contract_ID, Date_Issued, Date_Due, Date_Paid, Total_Amount)
VALUES (SEQ_INVOICE_NUM.NEXTVAL, 2002, DATE '2025-03-01', DATE '2025-02-28', DATE '2025-03-30', 17000.00);
INSERT INTO INVOICE (Invoice_Num, Contract_ID, Date_Issued, Date_Due, Date_Paid, Total_Amount)
VALUES (SEQ_INVOICE_NUM.NEXTVAL, 2002, DATE '2025-04-01', DATE '2025-04-30', NULL, 17000.00);

Contract 2003: Has not yet started, so no invoices have been issued yet.

Contract 2004:

INSERT INTO INVOICE (Invoice_Num, Contract_ID, Date_Issued, Date_Due, Date_Paid, Total_Amount)
VALUES (SEQ_INVOICE_NUM.NEXTVAL, 2004, DATE '2025-01-01', DATE '2025-01-31', DATE '2025-01-30', 9000.00);
INSERT INTO INVOICE (Invoice_Num, Contract_ID, Date_Issued, Date_Due, Date_Paid, Total_Amount)
VALUES (SEQ_INVOICE_NUM.NEXTVAL, 2004, DATE '2025-02-01', DATE '2025-02-28', DATE '2025-02-28', 9000.00);

Contract 2005:

INSERT INTO INVOICE (Invoice_Num, Contract_ID, Date_Issued, Date_Due, Date_Paid, Total_Amount)
VALUES (SEQ_INVOICE_NUM.NEXTVAL, 2005, DATE '2025-02-01', DATE '2025-02-28', DATE '2025-02-22', 12000.00);
INSERT INTO INVOICE (Invoice_Num, Contract_ID, Date_Issued, Date_Due, Date_Paid, Total_Amount)
VALUES (SEQ_INVOICE_NUM.NEXTVAL, 2005, DATE '2025-03-01', DATE '2025-03-31', DATE '2025-03-21', 12000.00);
INSERT INTO INVOICE (Invoice_Num, Contract_ID, Date_Issued, Date_Due, Date_Paid, Total_Amount)
VALUES (SEQ_INVOICE_NUM.NEXTVAL, 2005, DATE '2025-04-01', DATE '2025-04-30', NULL, 12000.00);

2.1.6. Loading the Project Table

INSERT INTO PROJECT(Project_ID, Contract_ID, Initiation_Date, Deadline, Team_ID, Budget)
VALUES (SEQ_PROJECT_ID.NEXTVAL, 2001, DATE('2025-01-01'), TO_DATE('2025-06-31'), 100, 80000.00);
INSERT INTO PROJECT(Project_ID, Contract_ID, Initiation_Date, Deadline, Team_ID, Budget)
VALUES (SEQ_PROJECT_ID.NEXTVAL, 2002, DATE('2024-03-01'), TO_DATE('2025-12-31'), 101, 150000.00);
INSERT INTO PROJECT(Project_ID, Contract_ID, Initiation_Date, Deadline, Team_ID, Budget)

<code>VALUES (SEQ_PROJECT_ID.NEXTVAL, 2004, DATE('2025-01-01'), TO_DATE('2025-05-10'), 102, 40000.00);</code>
<code>INSERT INTO PROJECT(Project_ID, Contract_ID, Initiation_Date, Deadline, Team_ID, Budget)</code>
<code>VALUES (SEQ_PROJECT_ID.NEXTVAL, 2005, DATE('2025-02-01'), TO_DATE('2025-10-31'), 103, 100000.00);</code>
<code>INSERT INTO PROJECT(Project_ID, Contract_ID, Initiation_Date, Deadline, Team_ID, Budget)</code>
<code>VALUES (SEQ_PROJECT_ID.NEXTVAL, 2005, DATE('2025-05-01'), TO_DATE('2025-10-31'), 103, 50000.00);</code>

2.1.6. Loading the Team Table

<code>INSERT INTO TEAM(Team_ID, Project_ID, Num_Members)</code>
<code>VALUES (SEQ_TEAM_ID.NEXTVAL, 100, 12);</code>
<code>INSERT INTO TEAM(Team_ID, Project_ID, Num_Members)</code>
<code>VALUES (SEQ_TEAM_ID.NEXTVAL, 101, 5);</code>
<code>INSERT INTO TEAM(Team_ID, Project_ID, Num_Members)</code>
<code>VALUES (SEQ_TEAM_ID.NEXTVAL, 102, 7);</code>
<code>INSERT INTO TEAM(Team_ID, Project_ID, Num_Members)</code>
<code>VALUES (SEQ_TEAM_ID.NEXTVAL, 103, 11);</code>

2.1.6. Loading the Position Table

<code>INSERT INTO POSITION(Position_ID, Job_Title, Department, Required_Qualification, Avg_Market_Salary)</code>
<code>VALUES (SEQ_POSITION_ID.NEXTVAL, "Senior .Net Developer", "Software Development", "BSC, BEng, BIT", 30000.00);</code>
<code>INSERT INTO POSITION(Position_ID, Job_Title, Department, Required_Qualification, Avg_Market_Salary)</code>
<code>VALUES (SEQ_POSITION_ID.NEXTVAL, "HR Manager", "Human Resources", "BCOM, BSC, BIT", 25000.00);</code>
<code>INSERT INTO POSITION(Position_ID, Job_Title, Department, Required_Qualification, Avg_Market_Salary)</code>
<code>VALUES (SEQ_POSITION_ID.NEXTVAL, "Marketing Director", "Marketing", "BA, BCOM", 20000.00);</code>

2.1.7. Loading the Employee Table

<code>INSERT INTO EMPLOYEE (Employee_ID, Position_ID, Payroll_ID, First_Name, Last_Name, Social_Number, Country_Of_Residence, Contact_Number, Emergency_Contact_Name, Emergency_Contact_Number, Medical_Aid_Provider, Medical_Aid_Number, Is_Employed, Date_Of_Employment, Date_Of_Termination, Reason_For_Termination)</code>
<code>VALUES (SEQ_POSITION_ID.NEXTVAL, 1, 1, 'Daniel', 'De Jager', 1122334444551,'South Africa', '+27234567890', 'Mrs De Jager', '+11239876543', 'Discovery', 234751,'Y', DATE '2025-01-10', NULL, NULL);</code>

INSERT INTO EMPLOYEE (Employee_ID, Position_ID, Payroll_ID, First_Name, Last_Name, Social_Number, Country_Of_Residence, Contact_Number, Emergency_Contact_Name, Emergency_Contact_Number, Medical_Aid_Provider, Medical_Aid_Number, Is_Employed, Date_Of_Employment, Date_Of_Termination, Reason_For_Termination)
VALUES (SEQ_POSITION_ID.NEXTVAL, 2, 2, 'Gaby', 'Gloss', 987654321, 'United Kingdom '+271234567890', 'Mr Gloss', '+441234567891', NULL, NULL, 'N', DATE '2015-05-12', 'YYYY-MM-DD', DATE '2023-06-01', 'Resigned');
INSERT INTO EMPLOYEE (Employee_ID, Position_ID, Payroll_ID, First_Name, Last_Name, Social_Number, Country_Of_Residence, Contact_Number, Emergency_Contact_Name, Emergency_Contact_Number, Medical_Aid_Provider, Medical_Aid_Number, Is_Employed, Date_Of_Employment, Date_Of_Termination, Reason_For_Termination)
VALUES (SEQ_POSITION_ID.NEXTVAL, 3, 3, 'Jurie', 'Blom', 12341111123,'South Africa', '+27234767190', 'Mrs Blom','+2733561122', 'Discovery', 111852,'Y', DATE '2015-12-01', DATE '2025-06-01', 'Retired');

2.1.8. Loading the Employee_deductions Table

INSERT INTO EMPLOYEE_DEDUCTIONS (Deduction_ID, Payroll_ID, Med_Aid, Travel_Allowance, Study_Allowance, Other_Deductions)
VALUES (SEQ_DEDUCTION_ID.NEXTVAL, 1, 400.00, 2000, 3000, 100.00);
INSERT INTO EMPLOYEE_DEDUCTIONS (Deduction_ID, Payroll_ID, Med_Aid, Travel_Allowance, Study_Allowance, Other_Deductions)
VALUES (SEQ_DEDUCTION_ID.NEXTVAL ,2, 550.00, 1800, 2500, 150.00);
INSERT INTO EMPLOYEE_DEDUCTIONS (Deduction_ID, Payroll_ID, Med_Aid, Travel_Allowance, Study_Allowance, Other_Deductions)
VALUES (SEQ_DEDUCTION_ID.NEXTVAL , 3, 600.00, 2200, 3200, 200.00);

2.1.9. Loading the Employee_leave Table

INSERT INTO EMPLOYEE_LEAVE (Leave_ID, Payroll_ID, Annual_Leave_Due, Annual_Leave_Taken, Sick_Leave_Due, Sick_Leave_Taken, Study_Leave_Due, Study_Leave_Taken)
VALUES (SEQ_LEAVE_ID.NEXTVAL , 1, 15, 5, 10, 2, 5, 1);
INSERT INTO EMPLOYEE_LEAVE (LEAVE_ID, Payroll_ID, Annual_Leave_Due, Annual_Leave_Taken, Sick_Leave_Due, Sick_Leave_Taken, Study_Leave_Due, Study_Leave_Taken)
VALUES (SEQ_LEAVE_ID.NEXTVAL , 2, 20, 3, 12, 1, 4, 0);
INSERT INTO EMPLOYEE_LEAVE (Leave_ID, Payroll_ID, Annual_Leave_Due, Annual_Leave_Taken, Sick_Leave_Due, Sick_Leave_Taken, Study_Leave_Due, Study_Leave_Taken)
VALUES (SEQ_LEAVE_ID.NEXTVAL , 3, 18, 7, 9, 3, 3, 1);

2.1.10. Loading the Employee_payroll Table

<code>INSERT INTO EMPLOYEE_PAYROLL (Payroll_ID, Employee_ID, Cost_To_Company, Tax_Income_Number, Deductions_ID, Banking_ID, Leave_ID)</code>
<code>VALUES (SEQ_PAYROLL_ID.NEXTVAL , 1, 50000.00, 9001123456, 1, 1, 1);</code>
<code>INSERT INTO EMPLOYEE_PAYROLL (Payroll_ID, Employee_ID, Cost_To_Company, Tax_Income_Number, Deductions_ID, Banking_ID, Leave_ID)</code>
<code>VALUES (SEQ_PAYROLL_ID.NEXTVAL , 2, 52000.00, 9001123457, 2, 2, 2);</code>
<code>INSERT INTO EMPLOYEE_PAYROLL (Payroll_ID, Employee_ID, Cost_To_Company, Tax_Income_Number, Deductions_ID, Banking_ID, Leave_ID)</code>
<code>VALUES (SEQ_PAYROLL_ID.NEXTVAL , 3, 55000.00, 9001123458, 3, 3, 3);</code>

2.1.10. Loading the Employee_Banking Table

<code>INSERT INTO EMPLOYEE_BANKING (Banking_ID, Payroll_ID, Bank_Name, Branch_Code, Bank_Acc_Type, Bank_Acc_Number)</code>
<code>VALUES (SEQ_BANKING_ID.NEXTVAL, 1, 'First National Bank', 250655, 'Savings', 12345678901);</code>
<code>INSERT INTO EMPLOYEE_BANKING (Banking_ID, Payroll_ID, Bank_Name, Branch_Code, Bank_Acc_Type, Bank_Acc_Number)</code>
<code>VALUES (SEQ_BANKING_ID.NEXTVAL, 2, 'ABSA Bank', 632005, 'Cheque', 23456789012);</code>
<code>INSERT INTO EMPLOYEE_BANKING (Banking_ID, Payroll_ID, Bank_Name, Branch_Code, Bank_Acc_Type, Bank_Acc_Number)</code>
<code>VALUES (SEQ_BANKING_ID.NEXTVAL, 3, 'Standard Bank', 051001, 'Savings', 34567890123);</code>

2.1.11. Loading the Article Table

<code>INSERT INTO ARTICLE (Article_ID, Project_ID, Category_ID, Title, Publish_Date)</code>
<code>VALUES (SEQ_ARTICLE_ID.NEXTVAL,100, 100, 'Fixing bugs', DATE '2024-01-15');</code>
<code>INSERT INTO ARTICLE (Article_ID, Project_ID, Category_ID, Title, Publish_Date)</code>
<code>VALUES (SEQ_ARTICLE_ID.NEXTVAL,101, 101, 'Reporting Bugs', DATE '2025-02-20', True);</code>
<code>INSERT INTO ARTICLE (Article_ID, Project_ID, Category_ID, Title, Publish_Date)</code>
<code>INSERT INTO ARTICLE (Article_ID, Project_ID, Category_ID, Title, Publish_Date)</code>
<code>VALUES (SEQ_ARTICLE_ID.NEXTVAL,102, 102, 'Fixing the code', DATE '2025-04-30');</code>

2.1.12. Loading the Category Table

<code>INSERT INTO CATEGORY (Category_ID, Category_Name)</code>
<code>VALUES (SEQ_CATEGORY_ID.NEXTVAL, 'Bug reports');</code>
<code>INSERT INTO CATEGORY (Category_ID, Category_Name)</code>

VALUES (SEQ_CATEGORY_ID.NEXTVAL, 'Known issues');
INSERT INTO CATEGORY (Category_ID, Category_Name)
VALUES (SEQ_CATEGORY_ID.NEXTVAL, 'Bug fixes');

2.1.13. Loading the FAQ Table

INSERT INTO FAQ (FAQ_ID, Project_ID, Question)
VALUES (SEQ_FAQ_ID.NEXTVAL, 100, 'How do I reset my password?');
INSERT INTO FAQ (FAQ_ID, Project_ID, Question)
VALUES (SEQ_FAQ_ID.NEXTVAL, 101, 'Why can I not see all the columns?');
INSERT INTO FAQ (FAQ_ID, Project_ID, Question)
VALUES (SEQ_FAQ_ID.NEXTVAL, 102, 'Is there a way to export all the data?');

2.1.14. Loading the Ticket Table

INSERT INTO TICKET (Ticket_ID, Project_ID, Is_Open, Create_Date, Description)
VALUES (SEQ_TICKET_ID.NEXTVAL, 100, 'Y', DATE '2025-01-15', 'Why is my application not loading in 0.00001 seconds?');
INSERT INTO TICKET (Ticket_ID, Project_ID, Is_Open, Create_Date, Description)
VALUES (SEQ_TICKET_ID.NEXTVAL, 101, 'Y', DATE '2025-01-15', 'Is there an option in the application that I did not ask for?');
INSERT INTO TICKET (Ticket_ID, Project_ID, Is_Open, Create_Date, Description)
VALUES (SEQ_TICKET_ID.NEXTVAL, 102, 'Y', DATE '2025-01-15', 'Why is this thing not in the application? I expect you to read my mind');

2.1.15. Loading the Partner Table

INSERT INTO PARTNER (Business_ID, Initiation_Date)
VALUES (1004, DATE '2023-01-15');
INSERT INTO PARTNER (Business_ID, Initiation_Date)
VALUES (1005, DATE '2024-02-20');
INSERT INTO PARTNER (Business_ID, Initiation_Date)
VALUES (1006, DATE '2025-03-10');

2.1.16. Loading the Campaign Table

INSERT INTO CAMPAIGN (Campaign_ID, Start_Date, End_Date, Budget, Target_Audience, Num_Conversions)
VALUES (SEQ_CAMPAIGN_ID.NEXTVAL, DATE '2025-04-01', DATE '2023-06-30', 50000, 'Young Adults', 1250);
INSERT INTO CAMPAIGN (Campaign_ID, Start_Date, End_Date, Budget, Target_Audience, Num_Conversions)
VALUES (SEQ_CAMPAIGN_ID.NEXTVAL, DATE '2025-05-15', DATE '2024-08-15', 75000, 'Professionals', 1800);

INSERT INTO CAMPAIGN (Campaign_ID, Start_Date, End_Date, Budget, Target_Audience, Num_Conversions)
VALUES (SEQ_CAMPAIGN_ID.NEXTVAL, DATE '2025-07-01', DATE '2025-09-30', 60000, 'Families', 1500);

2.1.17. Loading the Partner_on_Campaign Table

INSERT INTO PARTNER_ON_CAMPAIGN (Business_ID, Campaign_ID)
VALUES (1004, 300);
INSERT INTO PARTNER_ON_CAMPAIGN (Business_ID, Campaign_ID)
VALUES (1005, 301);
INSERT INTO PARTNER_ON_CAMPAIGN (Business_ID, Campaign_ID)
VALUES (1006, 302);

2.1.18. Loading the Employee_on_Team Table

INSERT INTO EMPLOYEE_ON_TEAM (Team_ID, Employee_ID)
VALUES (101, 1);
INSERT INTO EMPLOYEE_ON_TEAM (Team_ID, Employee_ID)
VALUES (102, 2);
INSERT INTO EMPLOYEE_ON_TEAM (Team_ID, Employee_ID)
VALUES (103, 3);

2.1.19. Loading the Platform Table

INSERT INTO PLATFORM (Platform_ID, Description, Reach)
VALUES (SEQ_PLATFORM_ID.NEXTVAL, 'Social Media', 20000);
INSERT INTO PLATFORM (Platform_ID, Description, Reach)
VALUES (SEQ_PLATFORM_ID.NEXTVAL, 'Email Marketing', 30000);
INSERT INTO PLATFORM (Platform_ID, Description, Reach)
VALUES (SEQ_PLATFORM_ID.NEXTVAL, 'Search Engine', 100000);

2.1.20. Loading the Platform_in_Campaign Table

INSERT INTO PLATFORM_IN_CAMPAIGN (Campaign_ID, Platform_ID)
VALUES (1, 1);
INSERT INTO PLATFORM_IN_CAMPAIGN (Campaign_ID, Platform_ID)
VALUES (1, 2);
INSERT INTO PLATFORM_IN_CAMPAIGN (Campaign_ID, Platform_ID)
VALUES (2, 1);

3. Queries

3.1. Limitation of Rows and Columns

Selects the top two clients order by total amount invoiced.

```
SELECT *  
FROM (SELECT B.Business_ID, B.Registered_Name, SUM(I.Total_Amount) AS  
Total_Invoiced  
FROM BUSINESS B  
JOIN CLIENT C ON B.Business_ID = C.Business_ID  
JOIN CLIENT_CONTRACT CC ON B.Business_ID = CC.Business_ID  
JOIN INVOICE I ON CC.Contract_ID = I.Contract_ID  
GROUP BY B.Business_ID, B.Registered_Name  
ORDER BY Total_Invoiced DESC)  
WHERE ROWNUM <= 2;
```

Output:

	BUSINESS_ID	REGISTERED_NAME	TOTAL_INVOICED
1	1001	Starlight Solutions	36000
2	1002	Ministry of Technology	34000

3.2. Sorting

Lists all the invoices in a table from highest to lowest sorted by Total_Amount.

```
SELECT Invoice_Num, Contract_ID, Total_Amount  
FROM Invoice  
ORDER BY Total_Amount DESC;
```

Output:

	INVOICE_NUM	CONTRACT_ID	TOTAL_AMOUNT
1	3014	2002	17000
2	3015	2002	17000
3	3013	2001	12000
4	3012	2001	12000
5	3011	2001	12000
6	3017	2004	9000
7	3016	2004	9000

Select teams with more than 2 team members and order from most to least members.

```
SELECT tm.Team_ID, tm.Num_Members, tm.Project_ID  
FROM TEAM tm  
JOIN PROJECT pr ON tm.Project_ID = pr.Project_ID  
JOIN CLIENT_CONTRACT cc ON pr.Contract_ID = cc.Contract_ID  
WHERE tm.Num_Members > 2
```

```
ORDER BY tm.Num_Members DESC
```

```
GROUP BY Team_ID;
```

Output:

	TEAM_ID	NUM_MEMBERS	PROJECT_ID
1	100	12	100
2	112	12	100
3	103	11	103
4	115	11	103
5	114	7	102
6	102	7	102
7	101	5	101
8	113	5	101

Order all the active campaigns by number of conversions greatest to least.

```
SELECT Campaign_ID, Num_Conversions, Budget
```

```
FROM CAMPAIGN
```

```
ORDER BY Num_Conversions DESC;
```

Output:

	CAMPAIGN_ID	NUM_CONVERSIONS	BUDGET
1	302	1800	75000
2	301	1500	60000
3	300	1250	50000

Select most recent open tickets

```
SELECT *
```

```
FROM TICKET
```

```
WHERE Is_Open = 'Y'
```

```
ORDER BY Create_Date DESC;
```

Output:

	TICKET_ID	PROJECT_ID	IS_OPEN	CREATE_DATE	DESCRIPTION
1	100	100	Y	01/MAY/25	Why is my application not loading in 0.00001 seconds?
2	102	102	Y	28/APR/25	Why is this thing not in the application? I expect you to read my mind
3	101	101	Y	15/APR/25	Is there an option in the application that I did not ask for?

Select all Categories and order by their name.

```
SELECT *
```

```
FROM CATEGORY
```

```
ORDER BY Category_Name;
```

Output:

	CATEGORY_ID	CATEGORY_NAME
1	102	Bug fixes
2	100	Bug reports
3	101	Known issues

3.3. LIKE, AND and OR

Finds all active clients who have contact persons with a prefix of “Sir” or “Dr” (for scheduling important networking events with VIPs).

```
SELECT B.Business_ID, B.Registered_Name, B.Contact_Person_Name
FROM BUSINESS B
JOIN CLIENT C ON B.Business_ID = C.Business_ID
WHERE C.Is_Active = 'Y'
AND (B.Contact_Person_Name LIKE 'Sir%'
OR B.Contact_Person_Name LIKE 'Dr%');
```

Output:

	BUSINESS_ID	REGISTERED_NAME	CONTACT_PERSON_NAME
1		1002 Ministry of Technology	Sir David Carter

3.4. Variables and Character Functions

Shows the business’s name as well as their contact email domain, which will likely be their website.

```
SELECT B.Registered_Name as “Business Name”,
CASE WHEN INSTR(B.email_address, '@') > 0 THEN SUBSTR(B.Email_Address,
INSTR(B.Email_Address, '@') + 1) ELSE ‘Invalid Email’
END AS “Likely Website”
FROM BUSINESS B
JOIN CLIENT C ON B.Business_ID = C.Business_ID
ORDER BY B.Registered_Name ASC;
```

Output:

	Business Name	Likely Website
1	BrightSky Analytics	brightsky.io
2	Ministry of Technology	mintech.gov
3	Starlight Solutions	starlight.com

3.5. Round and Trunc

Shows each invoice with its total truncated to the nearest hundred to make comparison visually easier.

```
SELECT Invoice_Num, TRUNC(Total_Amount, 2) AS Truncated_Total
FROM INVOICE
ORDER BY Total_Amount DESC;
```

Output:

	INVOICE_NUM	TRUNCATED_TOTAL
1	3014	17000
2	3015	17000
3	3013	12000
4	3012	12000
5	3011	12000
6	3017	9000
7	3016	9000

A summary of budgets across multiple campaigns.

```
SELECT TRUNC(AVG(Budget), 2) AS Average_Budget,
MAX(Budget) AS Maximum_Budget,
MIN(Budget) AS Minimum_Budget,
COUNT(*) AS Total_Campaigns
FROM CAMPAIGN;
```

Output:

	AVERAGE_BUDGET	MAXIMUM_BUDGET	MINIMUM_BUDGET	TOTAL_CAMPAIGNS
1	61666.66	75000	50000	3

3.6. Date Functions

Reveals the number of days before or past the due date an invoice was paid. A positive number indicates days before the due date, while negative numbers indicate how many days late it was paid.

```
SELECT Invoice_Num, Date_Issued, Date_Due, Date_Paid, Date_Due -
Date_Paid AS Days_Early_Or_Late
FROM INVOICE
WHERE Date_Paid IS NOT NULL;
```

Output:

	INVOICE_NUM	DATE_ISSUED	DATE_DUE	DATE_PAID	DAYS_EARLY_OR_LATE
1	3016	01/JAN/25	31/JAN/25	30/JAN/25	1
2	3011	01/FEB/25	28/FEB/25	25/FEB/25	3
3	3017	01/FEB/25	28/FEB/25	28/FEB/25	0
4	3012	01/MAR/25	31/MAR/25	29/MAR/25	2
5	3014	01/MAR/25	28/FEB/25	30/MAR/25	-30

Select all the projects which deadlines are within the next month.

```
SELECT pr.Project_ID, pr.Initiation_Date, pr.Deadline, pr.Budget, pr.Contract_ID,
cc.Business_ID
FROM PROJECT pr
JOIN CLIENT_CONTRACT cc ON pr.Contract_ID = cc.Contract_ID
WHERE pr.Deadline > SYSDATE
AND pr.Deadline <= ADD_MONTHS(SYSDATE,1);
```

Output:

	PROJECT_ID	INITIATION_DATE	DEADLINE	BUDGET	CONTRACT_ID	BUSINESS_ID
1	102	01/JAN/25	10/MAY/25	40000	2004	1003

Select all Campaigns that are past their end date.

```
SELECT Campaign_ID, Start_Date, End_Date
FROM CAMPAIGN
WHERE End_Date <= SYSDATE;
```

Output:

	CAMPAIGN_ID	START_DATE	END_DATE
1	300	01/APR/23	30/JAN/25
2	301	01/JUL/23	30/MAR/25

Select all Articles that were published this year.

```
SELECT Article_ID, Title
FROM ARTICLE
WHERE Publish_Date > DATE '2025-01-01'
AND Publish_Date < DATE '2026-01-01';
```

Output:

	ARTICLE_ID	TITLE
1	100	Addressing response time concerns
2	102	Reporting Bugs
3	103	The Fixing of Code

Select all projects that have a duration longer than a year;

```
SELECT *
FROM PROJECT
WHERE MONTHS_BETWEEN(Deadline, Initiation_Date > 12);
```

Output:

	PROJECT_ID	CONTRACT_ID	INITIATION_DATE	DEADLINE	TEAM_ID	BUDGET
1	101	2002	01/MAR/24	31/DEC/25	101	150000

3.7. Aggregate Functions

Lists the total invoiced amount grouped according to the contract.

```
SELECT Contract_ID, SUM(Total_Amount) AS Total_Invoiced
FROM INVOICE
GROUP BY Contract_ID;
```

Output:

	CONTRACT_ID	TOTAL_INVOICED
1	2001	36000
2	2004	18000
3	2002	34000

List how many projects each team is currently working on.

```
SELECT Team_ID, Count(Project_ID) AS Projects_On_Team
FROM PROJECT
GROUP BY Team_ID
ORDER BY Team_ID;
```

Output:

	TEAM_ID	PROJECTS_ON_TEAM
1	100	1
2	101	1
3	102	1
4	103	2

3.8. Group By and Having

Shows contracts with a length of more than 3 months, indicating a contract that extends across at least two quarters.

```
SELECT Contract_ID, START_DATE, TERMINATION_DATE,
TRUNC(MONTHS_BETWEEN(TERMINATION_DATE, START_DATE)) AS
Contract_Length_Months
FROM CLIENT_CONTRACT
WHERE MONTHS_BETWEEN(TERMINATION_DATE, START_DATE) > 3;
```

Output:

	CONTRACT_ID	START_DATE	TERMINATION_DATE	CONTRACT_LENGTH_MONTHS
1	2001	01/FEB/25	31/DEC/25	10
2	2003	01/JUN/25	31/OCT/25	4
3	2005	01/FEB/25	30/JUN/26	16

Indicates all campaigns grouped by their target audience.

```
SELECT Target_Audience, SUM(Num_Conversions) AS Total_Conversions
FROM CAMPAIGN
GROUP BY Target_Audience;
```

Output:

	TARGET_AUDIENCE	TOTAL_CONVERSIONS
1	Young Adults	1250
2	Families	1500
3	Professionals	1800

3.9. Joins

Provides some more details for invoice documents such as the business name, the contract start date, and the amount.

```
SELECT I.Invoice_Num, B.Registered_Name, CC.Start_Date, I.Total_Amount
```

```
FROM INVOICE I
JOIN Client_Contract CC ON I.Contract_ID = CC.Contract_ID
JOIN BUSINESS B ON CC.Business_ID = B.Business_ID;
```

Output:

	INVOICE_NUM	REGISTERED_NAME	START_DATE	TOTAL_AMOUNT
1	3017	BrightSky Analytics	01/JAN/25	9000
2	3016	BrightSky Analytics	01/JAN/25	9000
3	3015	Ministry of Technology	01/MAR/25	17000
4	3014	Ministry of Technology	01/MAR/25	17000
5	3013	Starlight Solutions	01/FEB/25	12000
6	3012	Starlight Solutions	01/FEB/25	12000
7	3011	Starlight Solutions	01/FEB/25	12000

Shows campaigns, partners on the campaigns, and the partner information where the budget is greater than fifty thousand.

```
SELECT c.Campaign_ID, c.Budget, p.Business_ID
FROM CAMPAIGN c
JOIN PARTNER_ON_CAMPAIGN poc ON c.Campaign_ID = poc.Campaign_ID
JOIN PARTNER p ON poc.Business_ID = p.Business_ID
WHERE c.Budget > 50000;
```

Output:

	CAMPAIGN_ID	BUDGET	BUSINESS_ID
1	301	60000	1005
2	302	75000	1006

Provides some more information on the FAQ.

```
SELECT f.FAQ_ID, f.Project_ID, p.Initiation_Date, Team_ID
FROM FAQ f
JOIN PROJECT p ON f.FAQ_ID = p.Project_ID;
```

Output:

	FAQ_ID	PROJECT_ID	INITIATION_DATE	TEAM_ID
1	100	100	01/JAN/25	100
2	101	101	01/MAR/24	101
3	102	102	01/JAN/25	102

3.10. Sub-Queries

Gets invoices from the contract with the highest total budget.

```
SELECT *
FROM INVOICE
WHERE Contract_ID = (
SELECT Contract_ID
FROM CLIENT_CONTRACT
```

```
WHERE Total_Budget = (SELECT MAX(Total_Budget) FROM
CLIENT_CONTRACT));
```

Output:

	INVOICE_NUM	CONTRACT_ID	DATE_ISSUED	DATE_DUE	DATE_PAID	TOTAL_AMOUNT
1	3011	2001	01/FEB/25	28/FEB/25	25/FEB/25	12000
2	3012	2001	01/MAR/25	31/MAR/25	29/MAR/25	12000
3	3013	2001	01/APR/25	30/APR/25	(null)	12000

Find campaigns that have a reach of over 20,000.

```
SELECT Campaign_ID, Budget, Target_Audience
FROM CAMPAIGN
WHERE Campaign_ID IN (
    SELECT pic.Campaign_ID
    FROM PLATFORM_IN_CAMPAIGN pic
    JOIN PLATFORM p ON pic.Platform_ID = p.Platform_ID
    WHERE p.Reach > 20000);
```

Output:

	CAMPAIGN_ID	BUDGET	TARGET_AUDIENCE
1	300	50000	Young Adults
2	302	75000	Professionals

3.11. SELECT Queries

Indicates where employees live in the world.

```
SELECT DISTINCT Country_of_Residence
FROM EMPLOYEE;
```

Output:

	COUNTRY_OF_RESIDENCE
1	South Africa
2	United Kingdom

Indicates the average employee salary by country.

```
SELECT e.Country_of_Residence, AVG(p.Cost_To_Company) AS Avg_Salary
FROM EMPLOYEE e
JOIN EMPLOYEE_PAYROLL p ON e.Employee_ID = p.Employee_ID
GROUP BY e.Country_Of_Residence;
```

Output:

	COUNTRY_OF_RESIDENCE	AVG_SALARY
1	South Africa	51000
2	United Kingdom	55000

Select total deductions per employee.

```
SELECT Payroll_ID, (Med_Aid + Travel_Allowance + Study_Allowance +
Other_Deductions) AS Total_Deductions
FROM EMPLOYEE_DEDUCTIONS
ORDER BY (Med_Aid + Travel_Allowance + Study_Allowance +
Other_Deductions) DESC;
```

Output:

	PAYROLL_ID	TOTAL_DEDUCTIONS
1	3	6200
2	1	5500
3	2	5000

Select all teams that have more than 5 members

```
SELECT *
FROM TEAM
WHERE Num_Members > 5;
```

Output:

	TEAM_ID	PROJECT_ID	NUM_MEMBERS
1	100	100	12
2	102	102	7
3	103	103	11
4	112	100	12
5	114	102	7
6	115	103	11

3.12. Average Queries

Shows average employee salary of all employees on payroll.

```
SELECT TRUNC(AVG(Cost_To_Company)) AS Average_Salary
FROM EMPLOYEE_PAYROLL ;
```

Output:

	AVERAGE_SALARY
1	52333

Calculates the average Reach of all active campaigns

```
SELECT AVG(p.Reach) AS Average_Campaign_Reach
FROM CAMPAIGN c
JOIN PLATFORM_IN_CAMPAIGN pic ON c.Campaign_ID = pic.Campaign_ID
JOIN PLATFORM p ON pic.Platform_ID = p.Platform_ID
WHERE c.End_Date > SYSDATE;
```

Output:

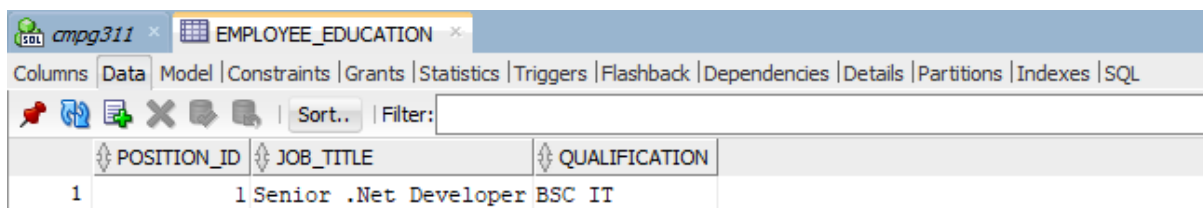
AVERAGE_CAMPAIGN_REACH
1 30000

3.13. Other Queries

Creates a new table of all employees with university degrees in Information Technology

```
CREATE TABLE EMPLOYEE_EDUCATION AS
SELECT EMPLOYEE_EDUCATION (Job_ID, Position_Title,
Required_Qualification AS Qualification)
FROM POSITION
WHERE Required_Qualification LIKE '%IT';
```

Output:



POSITION_ID	JOB_TITLE	QUALIFICATION
1	1 Senior .Net Developer	BSC IT

Indicate the number of articles in each category.

```
SELECT C.Category_Name, COUNT(*) AS Article_Count
FROM ARTICLE A
JOIN CATEGORY C ON A.Category_ID = C.Category_ID
GROUP BY C.Category_Name;
```

Output:

CATEGORY_NAME	ARTICLE_COUNT
1 Bug fixes	2
2 Known issues	1
3 Bug reports	1