НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

(повна назва інституту/факультету)

КАФЕДРА інформатики та програмної інженерії

(повна назва кафедри)

**КУРСОВА РОБОТА**

з дисципліни «Бази даних»

(назва дисципліни)

на тему: ”База даних для підтримки діяльності рекламного агентства”**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Студента (ки) \_\_2\_\_\_ курсу \_\_ІП-11\_\_\_\_ групи спеціальності 121 «Інженерія програмного забезпечення»

Прищепа В.С.

(прізвище та ініціали)

Керівник \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Національна шкала \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_ Оцінка ECTS \_\_\_\_\_\_\_

Члени комісії \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (вчене звання, науковий ступінь, прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (вчене звання, науковий ступінь, прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (вчене звання, науковий ступінь, прізвище та ініціали)

Київ – 2022 рік

**Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»**

Факультет Інформатики та обчислювальної техніки

(повна назва)

Кафедра Інформатики та програмної інженерії

(повна назва)

Дисципліна Бази даних

Курс 2 Група ІП-11 Семестр 1

**З А В Д А Н Н Я**

**НА КУРСОВУ РОБОТУ СТУДЕНТУ**

Прищепа Владислав Станіславович

(прізвище, ім’я, по батькові)

1. Тема роботи База даних для підтримки діяльності рекламного агентства\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ керівник роботи \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

1. Строк подання студентом роботи \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
2. Вихідні дані до роботи \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Дата видачі завдання\_\_\_\_\_\_\_\_\_**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| №  з/п | Назва етапів виконання курсового проекту | Строк виконання етапів проекту | Примітка |
| 1. | Вступ | 06.01.2023 |  |
| 2. | Опис предметного середовища | 06.01.2023 |  |
| 3. | Постановка задачі | 06.01.2023 |  |
| 4. | Побудова ER-моделі | 06.01.2023 |  |
| 5. | Побудова реляційної схеми | 06.01.2023 |  |
| 6. | Створення бази даних з допомогою обраної СУБД | 07.01.2023 |  |
| 7. | Імпортування даних | 07.01.2023 |  |
| 8. | Створення користувачів та реалізація їх функціоналу | 07.01.2023 |  |
| 9. | Створення SQL запитів | 08.01.2023 |  |
| 10. | Висновок | 09.01.2023 |  |
| 11. | Перелік посилань | 09.01.2023 |  |

**Студент \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(підпис ) (прізвище та ініціали)

**Керівник роботи \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(підпис ) (прізвище та ініціали)

**АНОТАЦІЯ**

Пояснювальна записка до курсової роботи: 55 сторінок, 36 рисунків, 1 таблиця, 2 посилання.

Об’єкт дослідження: база даних для підтримки діяльності рекламного агентства.

Мета роботи: закріплення навичок з проектування, реалізації реляційних баз даних та їх використання для практичних задач.

Проведено аналіз предметного середовища, визначено сутності, атрибути та зв’язки між об’єктами. Побудована ER-модель предметного середовища, реляційну схему бази даних, виділено первинні та зовнішні ключі, визначено обмеження для підтримки цілісності даних. Розроблено скрипти для побудови спроектованої бази даних, імпортовано дані в неї, виконано різноманітні запити до неї.

Виконана програмна реалізація бази даних для підтримки діяльності рекламного агентства ( варіант 25 ).

ЗМІСТ

[Вступ 8](#_Toc124230658)

[1 Опис предметної області 9](#_Toc124230659)

[2 Постановка завдання 10](#_Toc124230660)

[3 ER-Діаграма 11](#_Toc124230661)

[3.1 Бізнес правила 11](#_Toc124230662)

[3.2 Вибір сутностей 11](#_Toc124230663)

[3.3 Набори атриутів сутностей 12](#_Toc124230664)

[4 Реляційна модель бази даних 16](#_Toc124230665)

[4.1 Побудова необхідних відношень та визначення первинних та зовнішніх ключів 16](#_Toc124230666)

[4.2 Визначення обмежень цілісності для спроектованих відношень 16](#_Toc124230667)

[5 Реалізація бази даних 17](#_Toc124230668)

[5.1 Створення бази даних у форматі системи упраління базою даних PostgreSQL 17](#_Toc124230669)

[5.2 Імпортування даних в таблицю 21](#_Toc124230670)

[6 Створення користувачів бази даних 23](#_Toc124230671)

[6.1 Створення кадрового менеджера 23](#_Toc124230672)

[6.2 Створення бухгалтера 23](#_Toc124230673)

[6.3 Створення агента рекламного агентства 23](#_Toc124230674)

[7 SQL Запити 24](#_Toc124230675)

[7.1 Тригери 24](#_Toc124230676)

[7.1.1 checkdate 24](#_Toc124230677)

[7.1.2 countmat 25](#_Toc124230678)

[7.1.3 checksum 26](#_Toc124230679)

[7.1.4 ifsmallcom 27](#_Toc124230680)

[7.1.5 checkamemp 29](#_Toc124230681)

[7.1.6 ispayed 31](#_Toc124230682)

[7.2 Функції та процедури 32](#_Toc124230683)

[7.2.1 outcome 33](#_Toc124230684)

[7.2.2 pop\_ord\_type 34](#_Toc124230685)

[7.2.3 pop\_dis 36](#_Toc124230686)

[7.2.4 pop\_distrib 37](#_Toc124230687)

[7.2.5 freedep 38](#_Toc124230688)

[7.2.6 unpayed 40](#_Toc124230689)

[7.3 Запити 41](#_Toc124230690)

[7.3.1 Список усіх замовлень 41](#_Toc124230691)

[7.3.2 Список усіх матеріалів, які були замовлені у дистриб’юторів для виконання замовлень 42](#_Toc124230692)

[7.3.3 Список усіх угод, які були укладені із замовниками на замовлення 44](#_Toc124230693)

[7.3.4 Список усіх матеріалів, які треба доставити у відповідні відділи на вказані замовлення 45](#_Toc124230694)

[7.3.5 Список усіх виданих зарплат робітникам 47](#_Toc124230695)

[7.3.6 Список відділів та загальної кількості виданих замовлень кожному відділу 49](#_Toc124230696)

[7.4 Представлення 50](#_Toc124230697)

[7.4.1 stuff 50](#_Toc124230698)

[7.4.2 transaction 51](#_Toc124230699)

[7.5 Індекси 52](#_Toc124230700)

[Висновок 54](#_Toc124230701)

[Перелік посилань 55](#_Toc124230702)

Вступ

Бази даних є важливою частиною будь-якого бізнесу, адже за допомогою них можна зберігати важливі дані про клієнтів, замовлення, робітників та прослідковувати залежності між явищами. На відміну від минулого, де все зберігалося на папері, бази даних дають можливість швидко отримувати інформацію, що нас цікавить.

Обрана мною тема є актуальною, тому що реклама є важливою частиною нашого життя, а рекламні агентства інформують пересічних громадян про акції та новинки споживчого ринку. Для створення великих обсягів реклами необхідно мати правильну організацію, з чим дуже допомагають бази даних.

Дана робота присвячена вивченню розробки програмного забезпечення з SQL, і стосується проектування бази даних для підтримки діяльності рекламного агентства, що допоможе організувати роботу та облік замовлень. Задача полягає в аналізі предметної області, побудови ER-діаграми та датологічної схеми реляційної бази даних, написанні SQL скриптів та запитів до неї. Виконання роботи проведено з допомогою PostgreSQL, оскільки вона є безкоштовним і легким в освоєнні засобом побудови баз даних.

# **1** **Опис предметної області**

Основним об’єктом дослідження предметної області є створення реклами та процеси пов’язані з ними. Реклама створюється за замовленням клієнта.

Зазвичай замовлення створюється за договором між агентством та клієнтом і оплачується на місті переведенням коштів з банківського рахунку. Кожне замовлення має свій вид реклами, як от розклеювання листівок, реклама на рекламних щитах тощо. Замовлення осягає певний набір районів у місті (чи міст, якщо агентство працює по всій області чи країні). Замовлення віддають певному відділу агентства, що буде займатись цим замовленням, йому визначають термін виконання роботи.

Для замовлення необхідно у поставників замовити необхідні матеріали для виготовлення реклами та її розміщення.

У кожному відділку працює колектив, кожний учасник якого має свою посаду. Працівникам необхідно нараховувати зарплатню.

# **2** **Постановка завдання**

Метою даної роботи є розробка бази даних для збереження результатів організації виконання замовлень на рекламу та організації робочого процесу. Тобто організація даних має бути такою в базі, щоб було максимально зручно вибудовувати план виконання замовлень. Виділимо основні задачі:

Агент рекламного агентства має мати змогу:

* Ввести облік замовлень, клієнтів, договорів.
* Ввести облік замовлень матеріалів у поставників
* Переглядати перелік усіх транзакцій з банків, якими були оплачені замовлення
* Знати про найпопулярніші види реклами, райони для їх розповсюдження та поставників матеріалів для надання рекомендацій клієнтам

Кадровий менеджер має мати змогу:

* Наймати та звільняти працівників та прив’язувати їх до відділів
* Вести облік діючих відділів агентства
* Змінювати дані про працівників

Бухгалтер має мати змогу:

* Переглядати перелік відділів та їх працівників
* Ввести облік нарахування та видачі заробітних плат працівникам
* Ввести облік усіх транзакцій з банків, якими були оплачені замовлення

# **3 ER-Діаграма**

Після аналізу було виділено такі сутності та зв’зки між ними:

# **3.1 Бізнес правила**

* 1. У відділі може працювати до 20 працівників, при кількості працівників менше 10 відділ вважається з малою кількістю працівників, при назначені такого відділку на замовлення повідомляти про його малу комплектацію.
  2. Зарплатня для працівника складає від 5 до 20 тисяч
  3. Один відділ може бути назначений на декілька замовлень одночасно
  4. Для кожного району міста формується окреме замовлення і може бути назначені різні відділи; кожне таке замовлення посилається на договір загального замовлення, до договору додається платіжка, що покриває усі замовлення, що стосуються договору; допускається можливість однією платіжкою сплачувати декілька замовлень (замовлення з різними договорами)
  5. Для запобігання марної витрати матеріалів, на кожне замовлення необхідно виділяти до 5 різних замовлень матеріалів у поставника
  6. На замовлення виділяти не менше тижня
  7. Строго вести облік нарахування та видачі зарплатні, не видавати зарплатню більше одного разу

# **3.2 Вибір сутностей**

* Замовлення
* Тип замовлення
* Договір
* Клієнт
* Чек
* Банк
* Матеріал
* Поставник
* Район
* Відділ
* Працівник
* Зарплатня

# **3.3 Набори атриутів сутностей**

Таблиця 3.1 — Сутності та їхні атрибути

|  |  |
| --- | --- |
| Сутність | Атрибути |
| orders | id  department\_id  order\_type\_id  deal\_id  district\_id  amount  date\_of\_beginning  date\_of\_end |
| order\_type | id  name |
| client | id  name  phone\_number |
| deal | id  number\_of\_document  date  client\_id  receipt\_id |
| receipt | id  bank\_id  sum  date |
| bank | id  name |
| district | id  name |
| material | id  order\_id  distributor\_id  name  amount |
| distributor | id  name |
| department | id  name |
| employee | id  name  phone\_number  passport\_id  department\_id  position |
| salary | id  employee\_id  sum  date  is\_payed |

Сутність **distributor** пов’язана ***один до багатьох*** із сутністю **material,** адже у одного поставника може бути замовлено декілька матеріалів.

Сутність **orders** пов’язана ***один до багатьох*** із сутністю **material,** адже на одне замовлення може знадобитися декілька матеріалів.

Сутність **order\_type** пов’язана ***один до багатьох*** із сутністю **order,** адже декілька замовлень можуть бути одного типу.

Сутність **district** пов’язана ***один до багатьох*** із сутністю **order,** адже в одному районі можуть бути виконані декілька замовлень.

Сутність **department** пов’язана ***один до багатьох*** із сутністю **order,** адже відділ може виконувати декілька замовлень.

Сутність **department** пов’язана ***один до багатьох*** із сутністю **employee,** адже у відділку працює декілька працівників.

Сутність **employee** пов’язана ***один до багатьох*** із сутністю **salary,** адже працівнику можуть нарахувати декілька заробітних плат.

Сутність **receipt** пов’язана ***один до багатьох*** із сутністю **deal,** адже одним платежем можна покрити декілька замовлень (мова йде про загальні замовлення, для яких складається договір, до якого прикріплюється чек).

Сутність **bank** пов’язана ***один до багатьох*** із сутністю **receipt,** адже одним банком можуть користуватися декілька клієнтів, через що вони можуть надіслати свої платіжки з одного банку.

Сутність **deal** пов’язана ***один до багатьох*** із сутністю **order,** адже договір може покривати декілька районів міста, через що на кожен район, для зручності організації роботи, створюється окреме замовлення.

Сутність **client** пов’язана ***один до багатьох*** із сутністю **deal,** адже клієнт може декілька раз звернутися до агентства й зробити декілька замовлень.

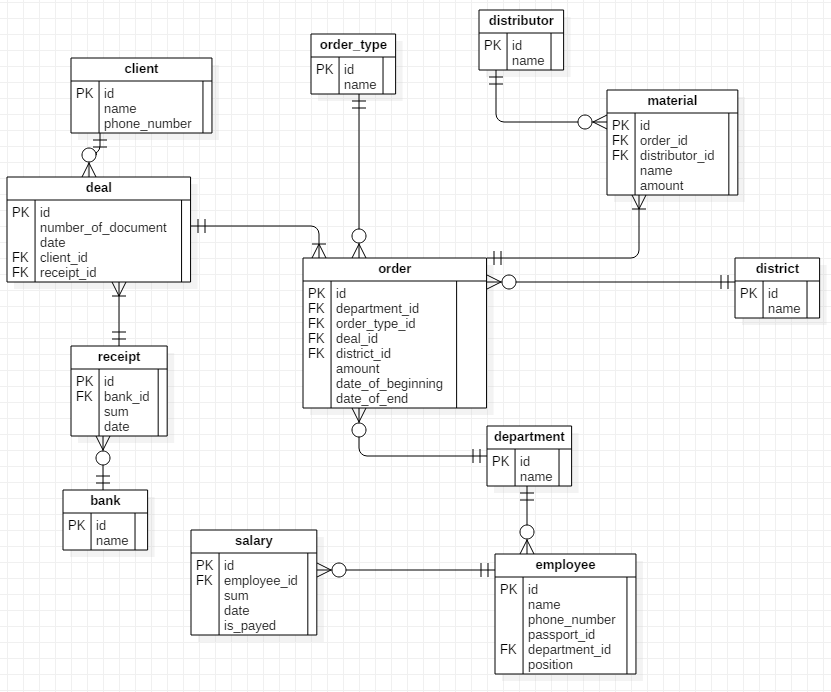


Рисунок 3.1 - ER-діаграма

# **4 Реляційна модель бази даних**

# **4.1 Побудова необхідних відношень та визначення первинних та зовнішніх ключів**

На рисунку можна побачити, що база даних знаходить у 3 нормальній формі, оскільки: поля таблиці декомпозовані, атрибути функціально повно залежать від первинного ключа, кожен неключовий атрибут не є транзитивно залежним від первинного ключа.

# **4.2 Визначення обмежень цілісності для спроектованих відношень**

1. Якщо не існує зовнішних ключів, що посилаються на значення головного ключа батьківського рядка, то тільки у цьому випадку його можна видалити. Для цього треба додати до таблиці обмеження з відповідними параметрами: ALTER TABLE child ADD CONSTRAINT fk\_parent\_child FOREIGN KEY (parent\_id) REFERENCES parent (id) ON UPDATE CASCADE ON DELETE RESTRICT
2. Обов’язкові артрибути таблиць мають обмеження NOT NULL для запобігання помилок при роботі з даними

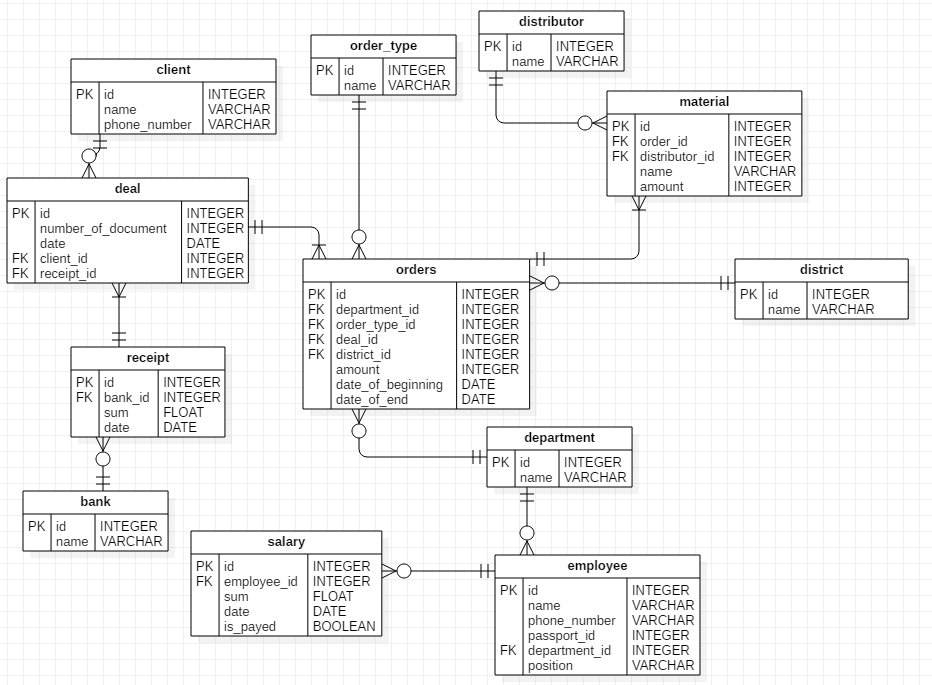


Рисунок 4.1 — Реляційна схема бази даних

# **5 Реалізація бази даних**

# **5.1 Створення бази даних у форматі системи упраління базою даних PostgreSQL**

CREATE TABLE IF NOT EXISTS client (

id INT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,

name VARCHAR(40) NOT NULL,

phone\_number VARCHAR(20) NOT NULL

);

CREATE TABLE IF NOT EXISTS bank (

id INT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,

name VARCHAR(40) NOT NULL

);

CREATE TABLE IF NOT EXISTS order\_type (

id INT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,

name VARCHAR(40) NOT NULL

);

CREATE TABLE IF NOT EXISTS distributor (

id INT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,

name VARCHAR(40) NOT NULL

);

CREATE TABLE IF NOT EXISTS district (

id INT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,

name VARCHAR(40) NOT NULL

);

CREATE TABLE IF NOT EXISTS department (

id INT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,

name VARCHAR(40) NOT NULL

);

CREATE TABLE IF NOT EXISTS receipt (

id INT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,

bank\_id INT NOT NULL,

sum NUMERIC(9, 2) NOT NULL,

date DATE NOT NULL

);

ALTER TABLE receipt ADD CONSTRAINT fk\_bank\_receipt

FOREIGN KEY(bank\_id) REFERENCES bank(id)

ON UPDATE CASCADE ON DELETE RESTRICT;

CREATE TABLE IF NOT EXISTS deal (

id INT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,

number\_of\_document INT NOT NULL,

date DATE NOT NULL,

client\_id INT NOT NULL,

receipt\_id INT NOT NULL

);

ALTER TABLE deal ADD CONSTRAINT fk\_client\_deal

FOREIGN KEY(client\_id) REFERENCES client(id)

ON UPDATE CASCADE ON DELETE RESTRICT;

ALTER TABLE deal ADD CONSTRAINT fk\_receipt\_deal

FOREIGN KEY(receipt\_id) REFERENCES receipt(id)

ON UPDATE CASCADE ON DELETE RESTRICT;

CREATE TABLE IF NOT EXISTS employee (

id INT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,

name VARCHAR(40) NOT NULL,

phone\_number VARCHAR(20) NOT NULL,

passport\_id INT NOT NULL,

department\_id INT NOT NULL,

position VARCHAR(20) NOT NULL

);

ALTER TABLE employee ADD CONSTRAINT fk\_department\_employee

FOREIGN KEY(department\_id) REFERENCES department(id)

ON UPDATE CASCADE ON DELETE RESTRICT;

CREATE TABLE IF NOT EXISTS salary (

id INT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,

employee\_id INT NOT NULL,

sum NUMERIC(7, 2) NOT NULL,

date DATE NOT NULL,

is\_payed boolean NOT NULL

);

ALTER TABLE salary ADD CONSTRAINT fk\_employee\_salary

FOREIGN KEY(employee\_id) REFERENCES employee(id)

ON UPDATE CASCADE ON DELETE RESTRICT;

CREATE TABLE IF NOT EXISTS orders (

id INT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,

department\_id INT NOT NULL,

order\_type\_id INT NOT NULL,

deal\_id INT NOT NULL,

district\_id INT NOT NULL,

amount INT NOT NULL,

date\_of\_beginning DATE NOT NULL,

date\_of\_end DATE NOT NULL

);

ALTER TABLE orders ADD CONSTRAINT fk\_department\_orders

FOREIGN KEY(department\_id) REFERENCES department(id)

ON UPDATE CASCADE ON DELETE RESTRICT;

ALTER TABLE orders ADD CONSTRAINT fk\_order\_type\_orders

FOREIGN KEY(order\_type\_id) REFERENCES order\_type(id)

ON UPDATE CASCADE ON DELETE RESTRICT;

ALTER TABLE orders ADD CONSTRAINT fk\_deal\_orders

FOREIGN KEY(deal\_id) REFERENCES deal(id)

ON UPDATE CASCADE ON DELETE RESTRICT;

ALTER TABLE orders ADD CONSTRAINT fk\_district\_orders

FOREIGN KEY(district\_id) REFERENCES district(id)

ON UPDATE CASCADE ON DELETE RESTRICT;

CREATE TABLE IF NOT EXISTS material (

id INT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,

order\_id INT NOT NULL,

distributor\_id INT NOT NULL,

name VARCHAR(40) NOT NULL,

amount INT NOT NULL

);

ALTER TABLE material ADD CONSTRAINT fk\_orders\_material

FOREIGN KEY(order\_id) REFERENCES orders(id)

ON UPDATE CASCADE ON DELETE RESTRICT;

ALTER TABLE material ADD CONSTRAINT fk\_distributor\_material

FOREIGN KEY(distributor\_id) REFERENCES distributor(id)

ON UPDATE CASCADE ON DELETE RESTRICT;

# **5.2 Імпортування даних в таблицю**

Імпортуємо CSV-файли в БД:

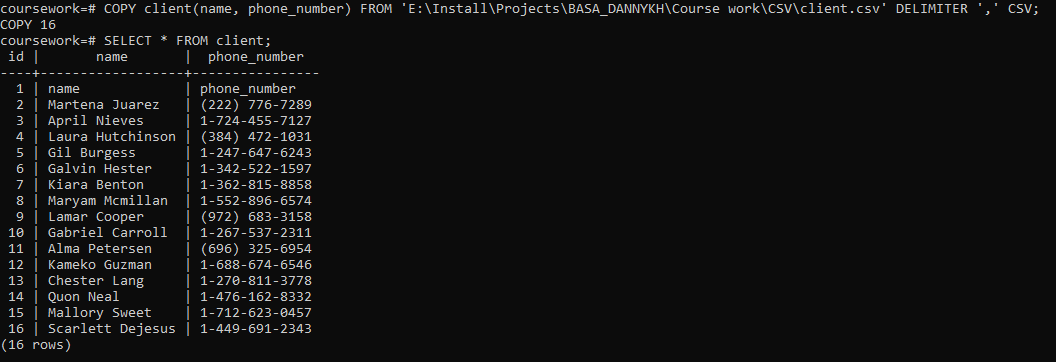


Рисунок 5.2.1 - Імпортування даних в таблиці client за допомогою CSV-файлу

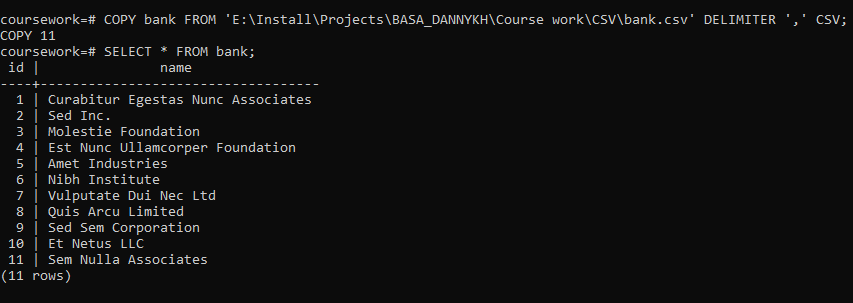


Рисунок 5.2.2 - Імпортування даних в таблиці bank за допомогою CSV-файлу

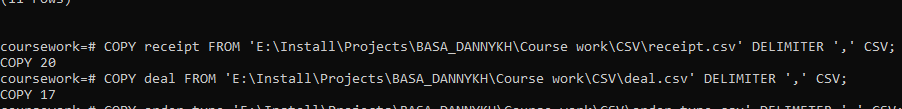


Рисунок 5.2.3 - Імпортування даних в таблиць receipt, deal за допомогою CSV-файлів

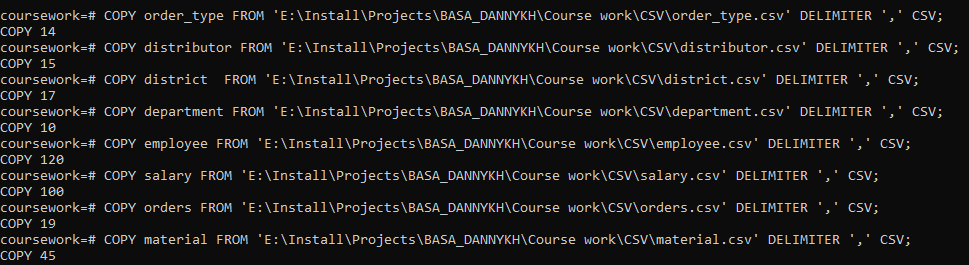


Рисунок 5.2.4 - Імпортування даних в таблиць order\_type, distributor, district, department, employee, salary, orders, material за допомогою CSV-файлів

# **6 Створення користувачів бази даних**

# **6.1 Створення кадрового менеджера**

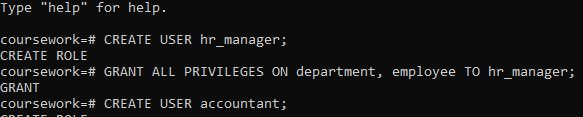


Рисунок 6.1 – Створення користувача кадрового менеджера та надання йому відповідних прав

# **6.2 Створення бухгалтера**

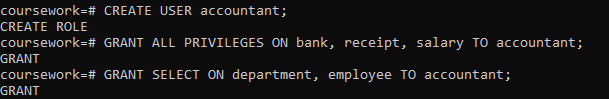


Рисунок 6.2 – Створення користувача бухгалтера та надання йому відповідних прав

# **6.3 Створення агента рекламного агентства**

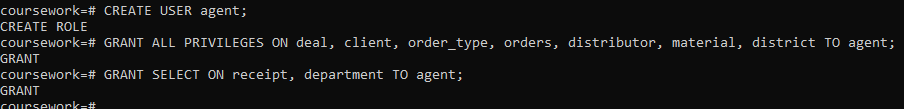


Рисунок 6.3 – Створення користувача агента рекламного агентства та надання йому відповідних прав

# **7 SQL Запити**

# **7.1 Тригери**

# 7.1.1 checkdate

Створення тригера для таблиці orders, щоб не допускати проміжок часу менше тижня на виконання замовлення при виконанні INSERT

CREATE OR REPLACE FUNCTION chkdtln()

RETURNS trigger

AS $$

DECLARE

a interval;

BEGIN

a = AGE(NEW.date\_of\_end, NEW.date\_of\_beginning);

IF a < '7 days'

THEN

RAISE EXCEPTION 'Not enough time for order!';

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER checkdate

BEFORE INSERT ON orders

FOR EACH ROW EXECUTE PROCEDURE chkdtln();

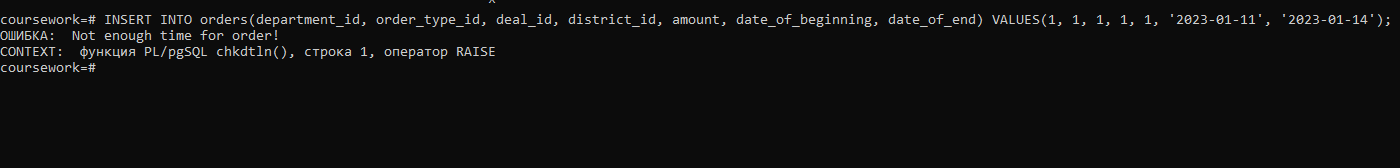


Рисунок 7.1 – Повідомлення про помилку, що надано замало часу на виконання замовлення

# 7.1.2 countmat

Створення тригера на таблицю material, щоб не допускати кількість матеріалів на одне замовлення більше 5

CREATE OR REPLACE FUNCTION cntmat()

RETURNS trigger

AS $$

DECLARE

a integer;

BEGIN

SELECT

COUNT(\*)

INTO

a

FROM

material

WHERE

order\_id = NEW.order\_id;

IF a > 4

THEN

RAISE EXCEPTION 'Limit of purchases on the order is reached!';

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER countmat

BEFORE INSERT ON material

FOR EACH ROW EXECUTE PROCEDURE cntmat();

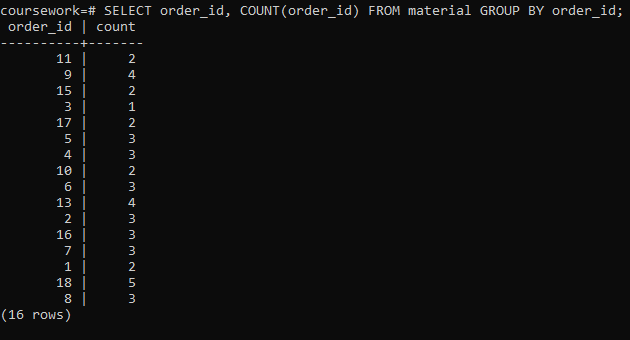


Рисунок 7.2 – Кількість замовлених позицій із матеріалами на кожне замовлення

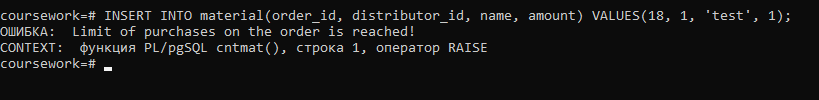


Рисунок 7.3 – Повідомлення про помилку, що на замовлення за вказаним id замовлено максимальну кількість позицій із матеріалами.

# 7.1.3 checksum

Створення тригера на таблицю salary, що не допускає нарахування зарплатні за межами допустимих значень (від 5 до 20 тисяч)

CREATE OR REPLACE FUNCTION chksl()

RETURNS trigger

AS $$

BEGIN

IF NEW.sum < 5000

THEN

RAISE EXCEPTION 'The sum is too small!';

END IF;

IF NEW.sum > 20000

THEN

RAISE EXCEPTION 'The sum is too big!';

END IF; RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER checksum

BEFORE INSERT ON salary

FOR EACH ROW EXECUTE PROCEDURE chksl();

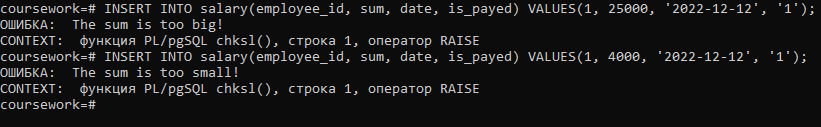


Рисунок 7.4 – Повідомлення про помилки, що в першому випадку сума завелика, в другому - замала

# 7.1.4 ifsmallcom

Створення тригера на таблицю orders, щоб повідомляти, коли на створене замовлення назначений відділ з малою кількістю працівників (менше 10)

CREATE OR REPLACE FUNCTION ifsmldp()

RETURNS trigger

AS $$

DECLARE

a integer;

n VARCHAR;

BEGIN

SELECT

COUNT(\*)

INTO

a

FROM

employee

WHERE

department\_id = NEW.department\_id;

IF a < 10

THEN

SELECT

name

INTO

n

FROM

department

WHERE

id = NEW.department\_id;

RAISE NOTICE '% has small amount of employees!', n;

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER ifsmallcom

AFTER INSERT ON orders

FOR EACH ROW EXECUTE PROCEDURE ifsmldp();

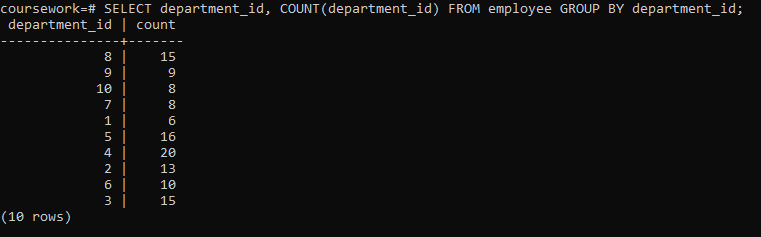


Рисунок 7.5 – Кількість працівників у кожному відділі

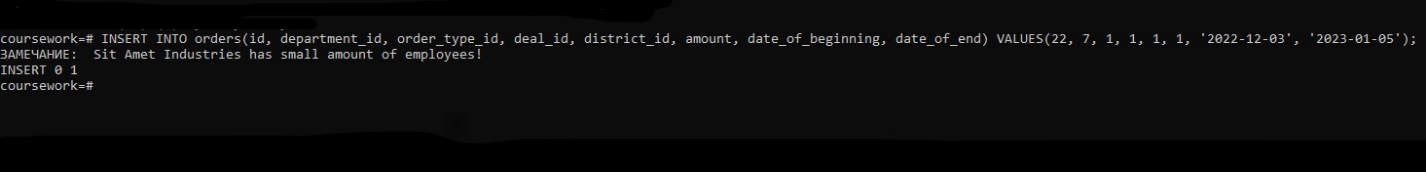


Рисунок 7.6 – Повідомлення, що у відділі, на який було назначено це замовлення, працівників менше 10

# 7.1.5 checkamemp

Створення тригера на таблицю employee, щоб не допускати кількість працівників у одному відділі більше 20

CREATE OR REPLACE FUNCTION chkszdp()

RETURNS trigger

AS $$

DECLARE

a integer;

n VARCHAR;

BEGIN

SELECT

COUNT(\*)

INTO

a

FROM

employee

WHERE

department\_id = NEW.department\_id;

IF a > 19

THEN

SELECT

name

INTO

n

FROM

department

WHERE

id = NEW.department\_id;

RAISE EXCEPTION '% is already full!', n;

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER checkamemp

BEFORE INSERT ON employee

FOR EACH ROW EXECUTE PROCEDURE chkszdp();

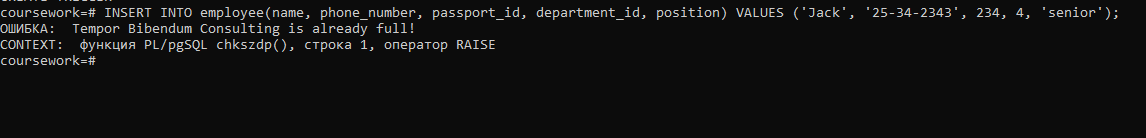


Рисунок 7.7 – Повідомлення про помилку, що у відділі вже нема місць для нових працівників

# 7.1.6 ispayed

Створення тригера на таблицю salary, щоб не видавати двічі зарплатню, що вже була видана

CREATE OR REPLACE FUNCTION ispyd()

RETURNS trigger

AS $$

DECLARE

a boolean;

b VARCHAR;

BEGIN

SELECT

is\_payed

INTO

a

FROM

salary

WHERE

id = NEW.id;

IF a = '1' AND NEW.is\_payed = '1'

THEN

SELECT

name

INTO

b

FROM

employee

WHERE

id = NEW.employee\_id;

RAISE EXCEPTION '% has received this salary!', b;

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER ispayed

BEFORE UPDATE ON salary

FOR EACH ROW EXECUTE PROCEDURE ispyd();

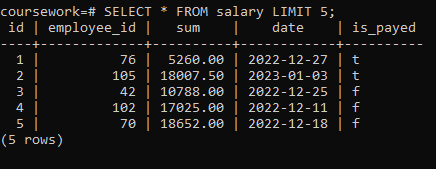


Рисунок 7.8 – Перелік нарахованих зарплат працівникам

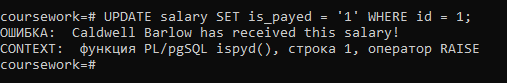


Рисунок 7.9 – Повідомлення про помилку, що нарахована зарплата, яку намагались видати, вже була видана працівнику

# **7.2 Функції та процедури**

# 7.2.1 outcome

Створення функції, що підрахує чистий прибуток агентства (з урахуванням усіх нарахованих зарплат) за вказаний проміжок часу

CREATE OR REPLACE FUNCTION outcome(dat1 DATE, dat2 DATE)

RETURNS NUMERIC

AS $$

DECLARE

a NUMERIC;

b NUMERIC;

BEGIN

SELECT

SUM(sum)

INTO

a

FROM

receipt

WHERE

date > dat1 AND date < dat2;

SELECT

SUM(sum)

INTO

b

FROM

salary

WHERE

date > dat1 AND date < dat2;

a = a - b;

RETURN a;

END;

$$ LANGUAGE plpgsql;

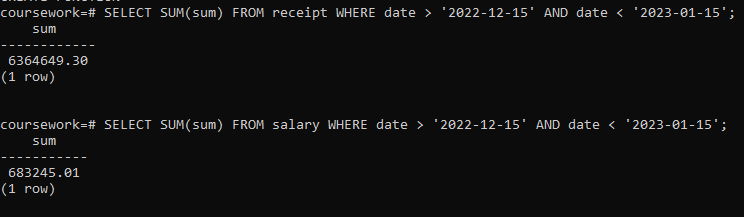


Рисунок 7.10 – Загальні суми грошей, які були зароблені та які були нараховані працівникам в якості зарплат в період з середини грудня 22 року до середини січня 23 року

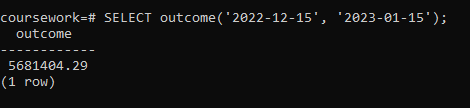


Рисунок 7.11 – Чистий заробіток агентства в період з середини грудня 22 року до середини січня 23 року

# 7.2.2 pop\_ord\_type

Створення процедури для швидкого визначення найпопулярнішого типу замовлення серед замовників

CREATE OR REPLACE PROCEDURE pop\_ord\_type()

AS $$

DECLARE

a INT;

b VARCHAR;

BEGIN

SELECT

order\_type\_id

INTO

a

FROM

orders

GROUP BY

order\_type\_id

ORDER BY

COUNT(order\_type\_id) DESC

LIMIT 1;

SELECT

name

INTO

b

FROM

order\_type

WHERE

id = a;

RAISE NOTICE 'ID:%, Name:%', a, b;

END;

$$ LANGUAGE plpgsql;

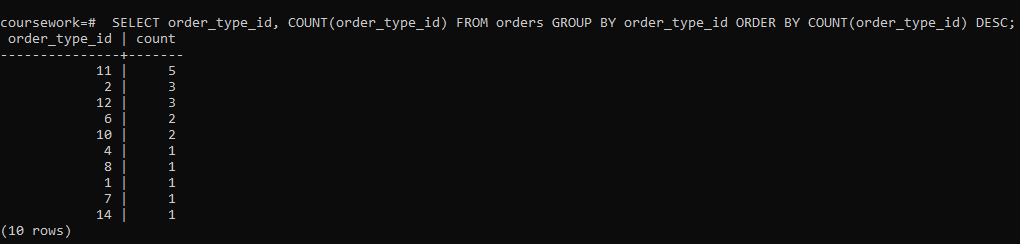


Рисунок 7.12 – Перелік кількості замовлень із кожним типом замовлення

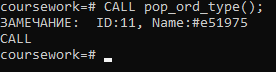


Рисунок 7.13 – Результат виконання процедури pop\_ord\_type для визначення найпопулярнішого типу замовлення

# 7.2.3 pop\_dis

Створення функції для швидкого визначення найпопулярнішого району для розміщення своєї реклами серед замовників

CREATE OR REPLACE FUNCTION pop\_dis()

RETURNS VARCHAR

AS $$

DECLARE

a INT;

b VARCHAR;

BEGIN

SELECT

district\_id

INTO

a

FROM

orders

GROUP BY

district\_id

ORDER BY

COUNT(district\_id) DESC

LIMIT 1;

SELECT

name

INTO

b

FROM

district

WHERE

id = a;

RETURN b;

END;

$$ LANGUAGE plpgsql;

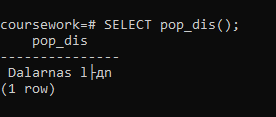


Рисунок 7.14 – Результат виконання функції pop\_dis

# 7.2.4 pop\_distrib

Створення процедури для швидкого визначення дистриб’ютора матеріалів, у якого агентство найчастіше робить замовлення

CREATE OR REPLACE PROCEDURE pop\_distrib(Distributor\_name INOUT VARCHAR)

AS $$

DECLARE

a INT;

BEGIN

SELECT

distributor\_id

INTO

a

FROM

material

GROUP BY

distributor\_id

ORDER BY

COUNT(distributor\_id) DESC

LIMIT 1;

SELECT

name

INTO

Distributor\_name

FROM

distributor

WHERE

id = a;

END;

$$ LANGUAGE plpgsql;

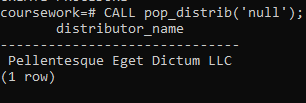


Рисунок 7.15 - Результат виконання процедури pop\_distrib

# 7.2.5 freedep

Створення функції для визначення переліку вільних відділів на вказану дату

CREATE OR REPLACE FUNCTION freedep(dat DATE)

RETURNS TABLE(department\_name VARCHAR)

AS $$

BEGIN

RETURN QUERY

SELECT

name

FROM

department

WHERE

id NOT IN (

SELECT

department\_id

FROM

orders

WHERE

date\_of\_beginning < dat AND date\_of\_end > dat

);

END;

$$ LANGUAGE plpgsql;

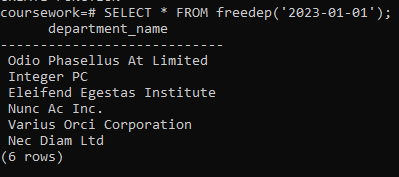


Рисунок 7.16 – Результат виконання функції freedep

# 7.2.6 unpayed

Створення функції для визначення усіх зарплат, що не були видані працівникам вказаного відділу

CREATE OR REPLACE FUNCTION unpayed(depid INT)

RETURNS TABLE(employee\_name VARCHAR, sum NUMERIC, date DATE)

AS $$

BEGIN

RETURN QUERY

SELECT

employee.name AS nam,

salary.sum AS summoney,

salary.date AS dat

FROM

salary,

employee

WHERE

employee.id = salary.employee\_id AND salary.is\_payed = '0' AND employee.department\_id = depid;

END;

$$ LANGUAGE plpgsql;

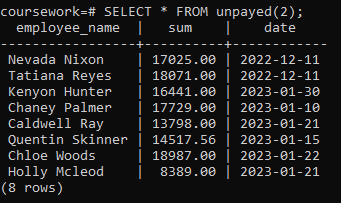


Рисунок 7.17 – Результат виконання функції unpayed

# **7.3 Запити**

# 7.3.1 Список усіх замовлень

Створення списку усіх замовлень із вказанням району, типу замовлення, розміру та термінів виконання

SELECT

id,

(

SELECT

name AS type

FROM

order\_type

WHERE

order\_type.id = orders.order\_type\_id

),

amount,

(

SELECT

name AS district

FROM

district

WHERE

district.id = orders.district\_id

),

date\_of\_beginning,

date\_of\_end

FROM

orders;

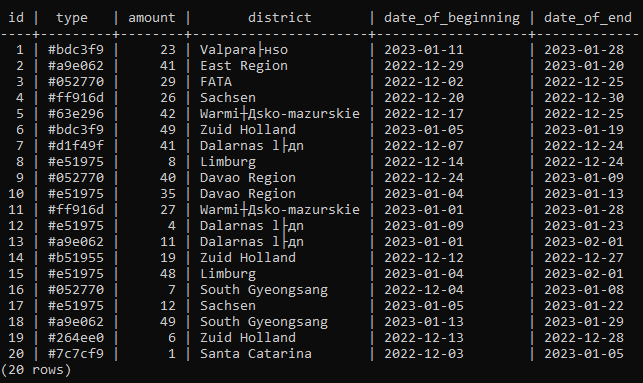


Рисунок 7.18 – Список усіх замовлень

# 7.3.2 Список усіх матеріалів, які були замовлені у дистриб’юторів для виконання замовлень

Створення списку усіх матеріалів, що були замовленні у дистриб’юторів для виконання замовлень

SELECT

id,

(

SELECT

id AS order\_id

FROM

orders

WHERE

orders.id = material.order\_id

),

name AS material,

amount,

(

SELECT

name AS distributor

FROM

distributor

WHERE

distributor.id = material.distributor\_id

)

FROM

material;

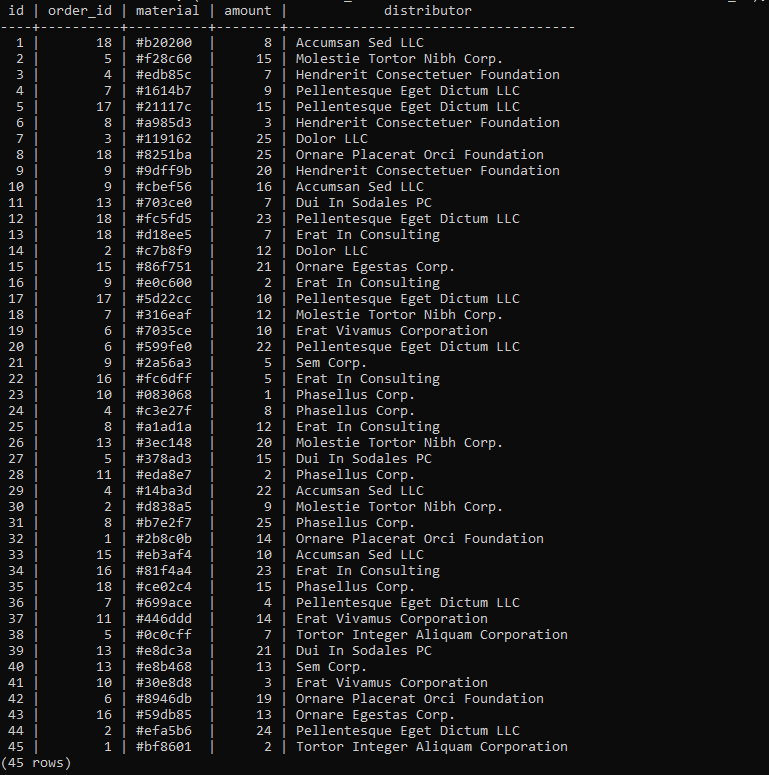


Рисунок 7.19 – Список усіх матеріалів, замовлених для замовлень

# 7.3.3 Список усіх угод, які були укладені із замовниками на замовлення

Створення списку усіх угод, що були укладені із замовниками, дат укладення та сум на оплату відповідних замовлень

SELECT

orders.id AS order\_id,

deal.number\_of\_document,

deal.date,

receipt.sum,

client.name AS name\_of\_client

FROM

orders

INNER JOIN

deal

ON

orders.deal\_id = deal.id

INNER JOIN

receipt

ON

deal.receipt\_id = receipt.id

INNER JOIN

client

ON

deal.client\_id = client.id;

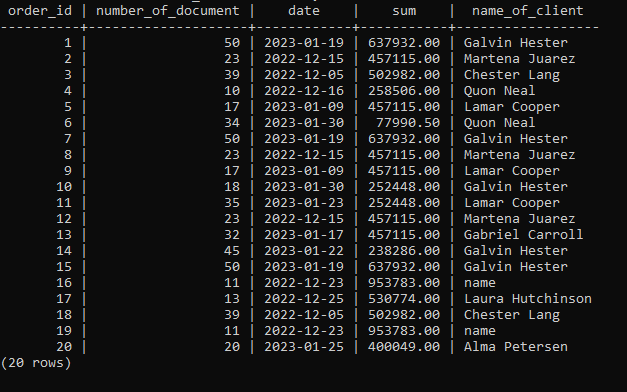


Рисунок 7.20 – Список всіх укладених із замовниками угод на замовлення

# 7.3.4 Список усіх матеріалів, які треба доставити у відповідні відділи на вказані замовлення

Створення списку усіх необхідних матеріалів, які треба надати відділам для виконання замовлень

SELECT

material.id AS material\_id,

material.name AS material\_name,

material.amount,

orders.id AS order\_id,

department.name AS department

FROM

material

INNER JOIN

orders

ON

material.order\_id = orders.id

INNER JOIN

department

ON

orders.department\_id = department.id;

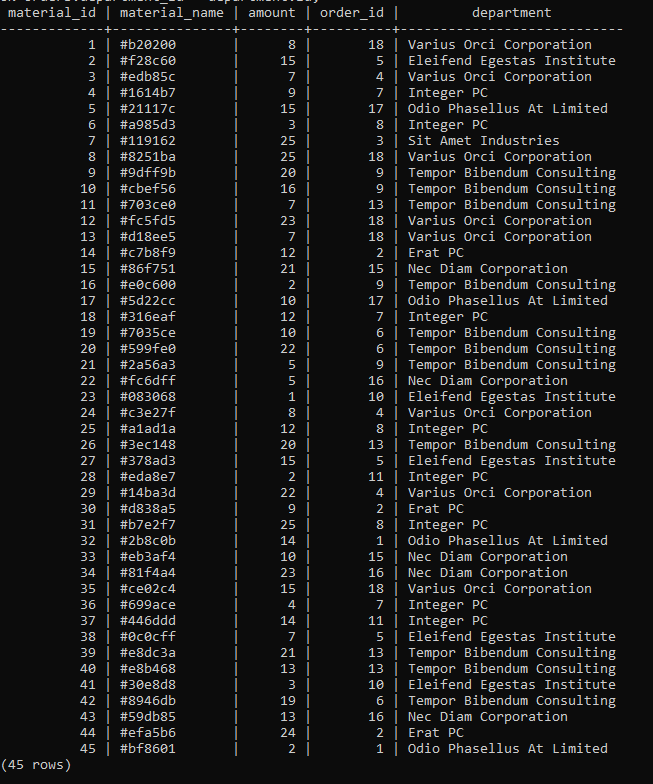


Рисунок 7.21 – Список матеріалів, які треба надати відділам для виконання відповідних замовлень

# 7.3.5 Список усіх виданих зарплат робітникам

Створення списку усіх зарплат, виданих робітникам відділів, по датам нарахування

SELECT

employee.name AS employee,

department.name AS department,

salary.sum, salary.date

FROM

employee

INNER JOIN

department

ON

employee.department\_id = department.id

INNER JOIN

salary

ON

salary.employee\_id = employee.id

WHERE

salary.is\_payed = '1';

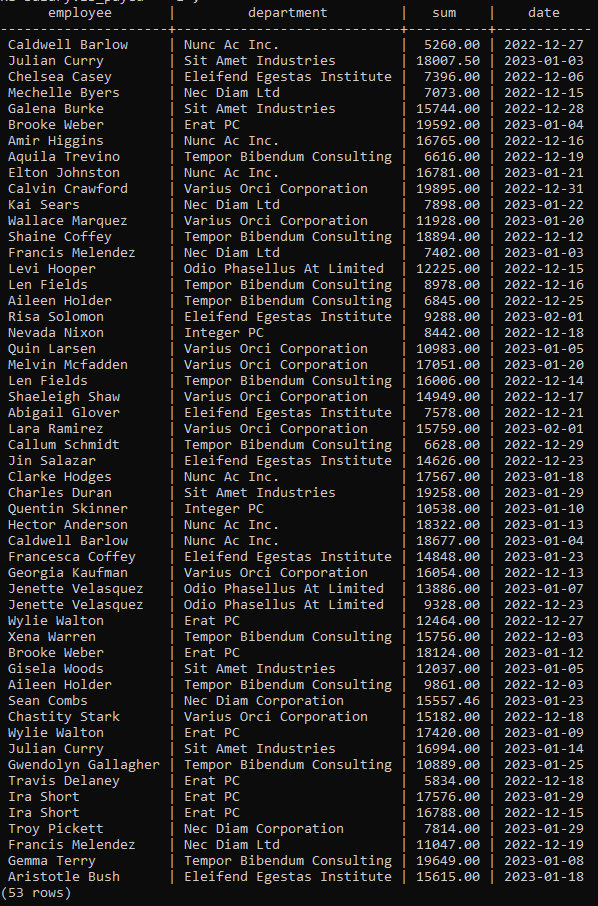


Рисунок 7.22 – Список усіх виданих зарплат робітникам

# 7.3.6 Список відділів та загальної кількості виданих замовлень кожному відділу

Створення списку відділів та загальної кількості назначених замовлень на кожний із них

SELECT

name AS department,

(

SELECT

COUNT(department\_id) AS amount

FROM

orders

WHERE

orders.department\_id = department.id

GROUP BY

department\_id

)

FROM

department

ORDER BY

amount DESC;

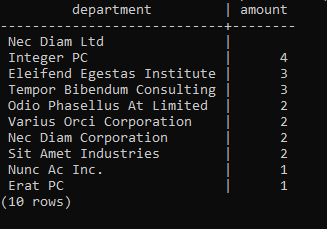


Рисунок 7.23 – Список відділів та кількості назначених замовлень на них

# **7.4 Представлення**

# 7.4.1 stuff

Створення представлення для відображення списку всіх працівників із вказанням відділу, у якому вони працюють

CREATE VIEW stuff AS

SELECT

name,

passport\_id,

phone\_number,

(

SELECT

name AS department

FROM

department

WHERE

department.id = employee.department\_id

),

position

FROM

employee;

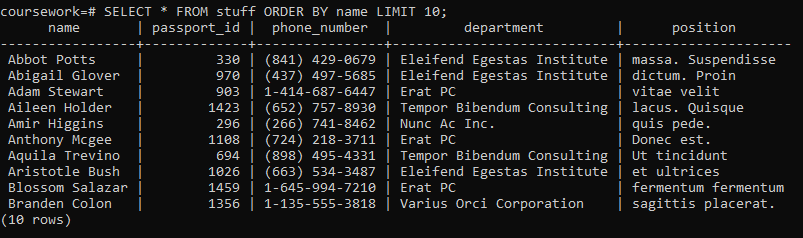


Рисунок 7.24 – Вибірка 10 робітників із представлення stuff

# 7.4.2 transaction

Створення представлення для відображення всіх транзакцій з банків на рахунок рекламного агентства

CREATE VIEW transaction AS

SELECT

id,

(

SELECT

name AS bank

FROM

bank

WHERE

bank.id = receipt.bank\_id

),

sum,

date

FROM

receipt;

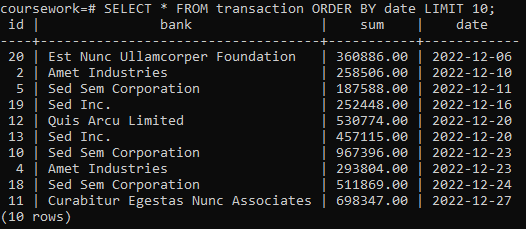


Рисунок 7.25 – Вибірка 10 транзакцій із представлення transaction

# **7.5 Індекси**

Виконаємо простий SQL-запит, де покажемо ім’я клієнта, номер телефону та номер документу угоди. Спочатку без використання індекса:

EXPLAIN ANALYZE

SELECT

client.name AS client,

client.phone\_number,

deal.number\_of\_document AS document

FROM

client

INNER JOIN

deal

ON

deal.client\_id = client.id

ORDER BY

client;

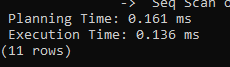


Рисунок 7.26 - Запит без використання індексів

Застосуємо індекси:

CREATE INDEX client\_index ON client (name, phone\_number);



Рисунок 7.27 – Запит із використанням індексів

Бачимо, що ефективність виконання запиту із застосуванням індексів збільшилася у 1.13 разів. Ця різниця стане ще помітнішею на великих базах даних. Отже, індекси є важливими у роботі з базами даних.

# **Висновок**

Бази даних — це важливий інструмент для ефективної роботи будь-якого сервісу. Вони дозволяють зручно та швидко зберігати інформацію та обробляти її, щоб виявити певні тренди споживчого ринку.

У цій роботі було спроектовано базу даних для підтримки діяльності рекламного агентства. У ході написання праці було вивчено предметне середовище, бізнес-правила, створено ER-діаграму, реляційну модель бази даних у третій нормальній формі. Також були написані SQL-скрипти, користувачі бази даних, тригери, представлення, функції та процедури.

# **Перелік посилань**

1. <https://www.postgresql.org/docs/>
2. https://www.postgresqltutorial.com/