КЫРГЫЗСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИИ КАФЕДРА «ИНФОРМАТИКА И ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА»

ОТЧЕТ

по производственной практике

студент: Керимбеков Нурбек.

группы: ИВТ-1-22

Руководитель практики доцент: Исраилова Н.А.

БИШКЕК – 2025

Отчёт о прохождении производственной практики в ОАО «Optima Bank»

Сроки практики: 20 января 2025 г. – 28 февраля 2025 г.
Место практики: Департамент развития мобильного банкинга

Введение

Производственная практика проходила в ОАО «Optima Bank», в Департаменте развития мобильного банкинга. Целью практики являлось получение практических навыков в области Android-разработки, закрепление теоретических знаний и участие в реальном проекте.

Цели и задачи практики

Основные задачи практики:

- Изучить язык программирования Kotlin;
- Освоить современные инструменты Android-разработки, включая Android Studio и Jetpack Compose;
- Разработать тестовое приложение (калькулятор) для закрепления базовых навыков;
- Принять участие в создании мобильного приложения Travel;
- Освоить систему контроля версий GitHub;
- Научиться работать с архитектурными паттернами (ViewModel, Repository, Singleton).

Ход практики

1-я неделя (20.01.25 – 26.01.25):

- Оформление на практику, прохождение инструктажа и техники безопасности.

2-я неделя (27.01.25 – 02.02.25):

- Ознакомление с правилами конфиденциальности информации.
- Получение первых заданий от ментора.
- Составление плана по изучению языка Kotlin и фреймворка Jetpack Compose.

3-я неделя (03.02.25 – 09.02.25):
- Изучение принципов ООП на Kotlin, а также основных принципов SOLID.
- Разработка учебного проекта «Калькулятор» с использованием XML-верстки.
- Ознакомление с Android SDK и базовыми компонентами Android-приложений.

4-я неделя (10.02.25 – 16.02.25):
- Начало работы над проектом Travel.
- Разработка интерфейсов приложения с помощью Jetpack Compose.
- Создание и доработка экранов, настройка элементов пользовательского интерфейса.

5-я неделя (17.02.25 – 23.02.25):
- Реализация части backend-логики приложения Travel.
- Настройка взаимодействия экранов, интеграция базовых функций.

6-я неделя (24.02.25 – 28.02.25):
- Оптимизация кода, изучение ViewModel и паттерна Repository.
- Реализация принципа Singleton.
- Финальная проверка и улучшение проекта Travel.
- Подведение итогов практики.

Освоенные технологии и инструменты

За время практики были изучены и применены следующие технологии и инструменты:
- Язык программирования Kotlin
- Освоены основы синтаксиса, работа с классами, объектами, функциями, коллекциями.

- Изучены лямбда-выражения, расширяющие функции, data-классы, sealed-классы и корутины для асинхронных операций.
- Особое внимание было уделено принципам SOLID, что позволило писать более структурированный и поддерживаемый код.

Android Studio

- Настройка проектов, работа с Gradle, отладка и профилирование приложений.
- Использование эмулятора и подключение реального устройства для тестирования.

Jetpack Compose

- Освоена декларативная разработка интерфейсов.
- Использованы элементы Column, Row, Box, Card, LazyColumn.
- Реализована работа с состоянием через State и remember.
- Настроена простая навигация между экранами.

Android SDK

- Изучены компоненты: Activity, Fragment, Intent, жизненный цикл приложений.
- Настроена работа с ресурсами (строки, цвета, стили) и обработка системных разрешений.

Система контроля версий GitHub

- Работа с Git: создание репозитория, коммиты, пуши, ветвление и слияние.
- Освоены основы командной разработки.

Архитектурные решения

- MVVM (Model-View-ViewModel) – разделение логики и интерфейса.
- Repository – для организации доступа к данным.
- Singleton – для управления глобальными объектами.

Дополнительно изучено:

- Работа с Kotlin Coroutines для асинхронных задач.
- Использование LiveData и StateFlow для реактивного обновления интерфейса.
- Оптимизация кода и базовое тестирование приложения.
- Настройка локализации и работа с ресурсами.

Выполненные проекты

1. Учебное приложение «Калькулятор»

Описание:

- Приложение разработано на Kotlin с использованием Jetpack Compose.
- Основная цель – закрепление знаний Kotlin, работы с Compose и базовых принципов Android-разработки.
- Реализованы все базовые арифметические операции: сложение, вычитание, умножение, деление, использование скобок и десятичных чисел.
- Приложение содержит удобный интерфейс с кнопками и отображением текущего выражения и результата.
- Первая практическая работа на Kotlin.
- Использована XML-разметка для интерфейса.
- Реализованы базовые арифметические операции.

Разбор кода

1. MainActivity.kt

Листинг кода:

```
class MainActivity : ComponentActivity() {
   override fun onCreate(savedInstanceState: Bundle?) {
      super.onCreate(savedInstanceState)
      setContent {
         CalculatorScreen()
      }
   }
}
```

- Класс наследуется от ComponentActivity.
- В onCreate вызывается setContent { CalculatorScreen() }, что позволяет использовать Jetpack Compose для построения интерфейса.

2. CalculatorScreen() – основной экран
Листинг кода:
```
@Composable
fun CalculatorScreen() {
    var expression by remember { mutableStateOf("0") }
    var result by remember { mutableStateOf("0") }
```

3. Интерфейс кнопок
Листинг кода:
```
val buttons = listOf(
    listOf("AC", "C", "(", ")"),
    listOf("7", "8", "9", "/"),
    listOf("4", "5", "6", "*"),
    listOf("1", "2", "3", "-"),
    listOf("0", ".", "=", "+")
)
```

- Кнопки расположены в виде списка списков, чтобы формировать строки кнопок на экране.
- Кнопки включают арифметические действия, скобки, очистку и “=”.

4. Обработка нажатий handleButtonClick()
Листинг кода:
```
fun handleButtonClick(button: String, current: String): String {
    var data = current
    when (button) {
        "AC" -> return "0"
        "C" -> data = if (data.length > 1) data.dropLast(1) else "0"
        "=" -> return data
    }
```

- "AC" – полная очистка.
- "C" – удаляет последний символ.
- "=" обрабатывается отдельно через evaluateExpression.

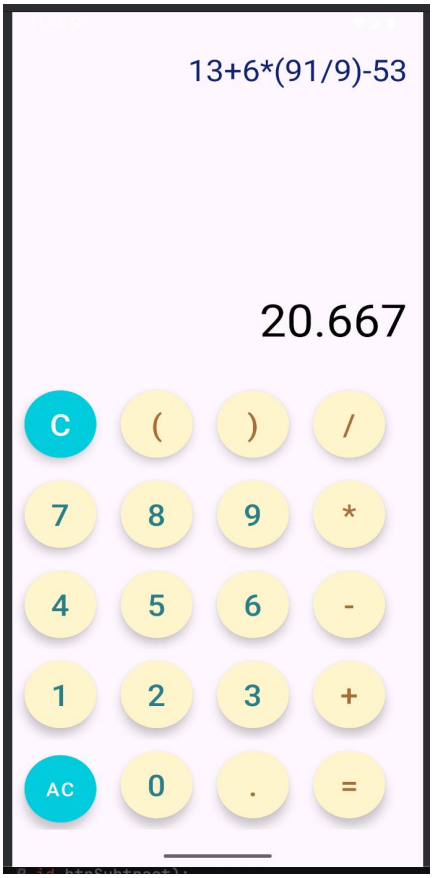
Особенности:
- Автоматическое добавление * перед открывающей скобкой после цифры.
- Предотвращение повторных операторов.
- Контроль правильности десятичной точки и скобок.
- Замена начального нуля при вводе цифры.

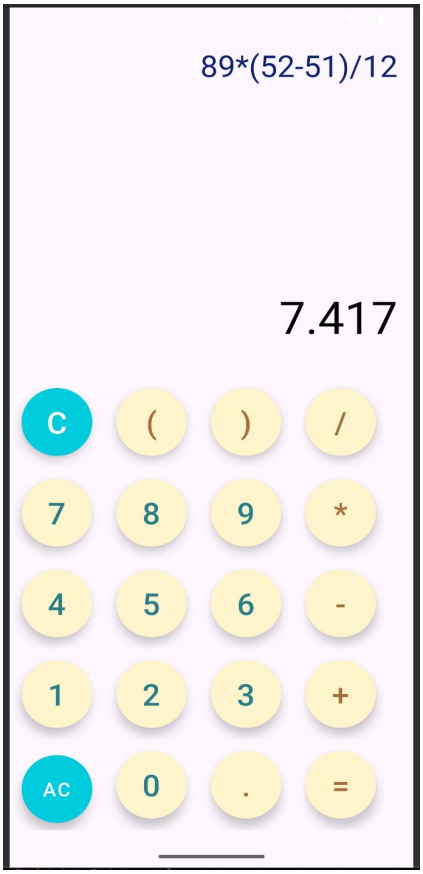
5. Вычисление выражения evaluateExpression()
Листинг кода:

```
fun evaluateExpression(expression: String): String {
    val rhino = Context.enter()
    rhino.optimizationLevel = -1
    return try {
        val scope: Scriptable = rhino.initStandardObjects()
        val result = rhino.evaluateString(scope, expression, "JavaScript", 1,
null).toString()
        DecimalFormat("#.###").format(result.toDouble())
    } catch (e: Exception) {
        "Error"
    } finally {
        Context.exit()
    }
}
```

- Используется движок Rhino (JavaScript engine) для вычисления математических выражений.
- Результат форматируется до трёх знаков после запятой.
- Обработка ошибок: при некорректном выражении возвращается "Error".

Скрины проекта:





2. Туристическое приложение «Travel»

- Основной итоговый проект практики.
- Интерфейс разработан с использованием Jetpack Compose.
- Реализованы основные экраны: приветственный экран, список мест, карточки достопримечательностей.
- Использованы архитектурные паттерны (MVVM, Repository).
- Настроена работа с GitHub, проект загружен в репозиторий.

Структура проекта:

- components – вспомогательные элементы интерфейса
- BtnBooked.kt – кнопка для бронирования
- FilterTabs.kt – табы для фильтрации
- InfoDt.kt – блок информации о деталях места
- PlaceCard.kt – карточка для отображения места в списке
- PlaceCardDt.kt – карточка подробной информации
- SearchBar.kt – строка поиска
- TimTempRat.kt – компонент для отображения времени, температуры и рейтинга
- models
- Place.kt – модель данных для описания туристического места
- navigations
- BottomNavigationBar.kt – нижняя панель навигации
- BottomNavItem.kt – элементы нижней навигации
- NavGraph.kt – граф навигации между экранами
- screens
- Detail.kt – экран с подробной информацией
- Favorite.kt – экран «Избранное»
- MainScreen.kt – главный экран
- Profile.kt – экран профиля
- Recent.kt – экран последних просмотров
- SplashScreen.kt – приветственный экран
- ui.theme – оформление, стили приложения
- MainActivity.kt – главный файл запуска приложения

Разбор кода главного экрана MainScreen.kt

Главный экран отображает приветствие пользователю, строку поиска, фильтры и список доступных туристических мест.

Листинг кода:

```
@Composable
fun MainScreen(navController: NavController) {
   Scaffold(bottomBar = { BottomNavigationBar(navController) })
{ innerPadding ->
      Column(
         modifier = Modifier
            .fillMaxSize()
            .background(Color.White)
            .padding(innerPadding)
            .padding(Dimens.PaddingLarge)
      ) {
         Row(
            modifier = Modifier.fillMaxWidth(),
            verticalAlignment = Alignment.CenterVertically,
            horizontalArrangement = Arrangement.SpaceBetween
         ) {
            Text(
               text = "Привет, Нурбек □",
               fontSize = 28.sp,
               fontWeight = FontWeight.Bold,
               color = Color.Black
            )
            Image(
               painter = painterResource(R.drawable.img_1),
               contentDescription = "Profile Photo",
               modifier = Modifier
                  .size(Dimens.IconSize)
                  .clip(CircleShape)
            )
         }
         Text(text = "Исследуйте мир", fontSize = 22.sp, color =
Color.Gray)
         SearchBar()
         FilterTabs()
         PlacesListScreen(navController)
      }
   }
}
```

Разбор логики:

- Scaffold используется для создания базовой структуры экрана с нижней панелью навигации.

- Column располагает элементы вертикально, с отступами и фоном.
- Row в верхней части экрана показывает приветствие и аватар пользователя.
- SearchBar() и FilterTabs() – компоненты для поиска и фильтрации мест.
- PlacesListScreen(navController) – горизонтальный список карточек мест с навигацией на экран деталей.

Экран деталей Detail.kt
Листинг кода:

```
@Composable
fun Detail(place: Place, navController: NavController) {
   Column(
      modifier = Modifier
         .fillMaxSize()
         .background(Color.White)
         .padding(bottom = Dimens.PaddingMedium)
   ) {
      PlaceCardDt(place, navController)
      Spacer(modifier = Modifier.height(Dimens.SpacerHeight))
      TimTempRat(place)
      Spacer(modifier = Modifier.height(Dimens.SpacerHeight))
      InfoDt(place)
      Spacer(modifier = Modifier.weight(1f))
      BtnBooked()
   }
}
```

Разбор:
- Используется Column, чтобы расположить компоненты вертикально.
- PlaceCardDt отображает основное изображение и ключевую информацию о месте.
- TimTempRat показывает время, температуру и рейтинг.
- InfoDt – блок с детальной информацией.
- BtnBooked – кнопка для бронирования тура.
- Spacer с weight(1f) позволяет кнопке находиться внизу экрана.

Метод getPlaceById(id: Int) возвращает объект Place по идентификатору, что облегчает навигацию между списком мест и экраном деталей.

Карточки мест PlaceCard.kt и PlaceCardDt.kt
PlaceCard.kt – для отображения места в списке:

```
@Composable
fun PlaceCard(place: Place, navController: NavController) {
   var isFavorite by remember { mutableStateOf(false) }

   Card(
      modifier = Modifier
         .width(Dimens.CardWidth)
         .height(Dimens.CardHeight)
         .padding(Dimens.PaddingSmall)
         .clickable { navController.navigate("detail/${place.id}") },
      shape = RoundedCornerShape(Dimens.PaddingMedium),
   ) {
      // Содержимое карточки: изображение, название, локация, кнопка избранного
   }
}
```

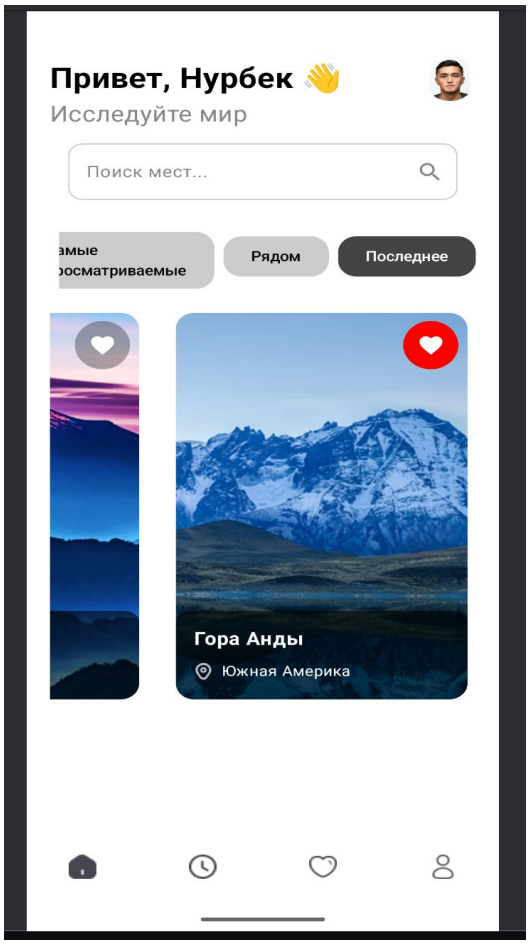
PlaceCardDt.kt – для экрана деталей:

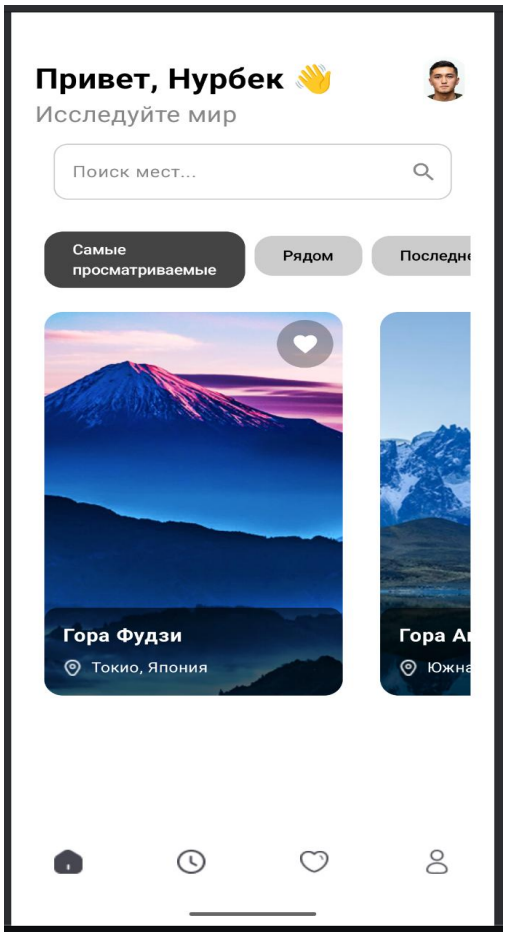
```
@Composable
fun PlaceCardDt(place: Place, navController: NavController) {
   // Показывает фоновое изображение места
   // Кнопка назад с navController.popBackStack()
   // Кнопка сохранить/скачать место
   // Затемнённая панель с названием, локацией и ценой
}
```

Разбор:

- PlaceCard и PlaceCardDt используют Card для оформления.
- Навигация на экран деталей происходит через navController.navigate.
- Состояние избранного или скачанного элемента хранится через remember и mutableStateOf.
- Затемнённый фон и текст реализованы с использованием Box и Column внутри карточки.

Скрины проекта:





Вывод

Производственная практика в ОАО «Optima Bank» позволила закрепить полученные теоретические знания и приобрести практические навыки в области Android-разработки. В процессе практики были изучены и применены современные технологии разработки мобильных приложений: язык Kotlin, фреймворк Jetpack Compose, архитектурные паттерны MVVM, Repository и Singleton.

Было реализовано два проекта:

- **учебное приложение «Калькулятор»**, направленное на закрепление базовых знаний Kotlin и работы с интерфейсами;
- **проект «Travel»**, представляющий собой многоэкранное туристическое приложение с навигацией, поиском и системой избранного.

Практика способствовала развитию профессиональных навыков:

- умению работать с системой контроля версий GitHub;
- проектированию интерфейсов и архитектуры приложений;
- написанию структурированного и оптимизированного кода;
- командной работе и соблюдению корпоративных стандартов разработки.

Итогом прохождения практики стало формирование полноценного опыта участия в реальном проекте и значительное расширение профессиональных компетенций в сфере мобильной разработки.\

Отчёт о прохождении производственной практики в ОАО «Айыл Банк»

Сроки практики: 26 мая 2025 г. – 28 июня 2025 г.
Место практики: ОАО «Айыл Банк», IT-департамент

Введение

Производственная практика проходила в ОАО «Айыл Банк». Основная цель – углубление знаний в области баз данных и алгоритмов, освоение работы в среде Oracle и закрепление навыков программирования на языке SQL и Java. Практика позволила применить теоретические знания в решении прикладных задач, связанных с управлением данными и алгоритмическими процессами.

Цели и задачи практики

Основные задачи практики:

- Ознакомление с деятельностью банка и правилами информационной безопасности.
- Изучение основ работы с Oracle Database.
- Разработка и выполнение SQL-запросов.
- Создание и тестирование таблиц, связей, триггеров, процедур и функций.

- Решение алгоритмических задач на языке Java.
- Приобретение навыков отладки и обработки ошибок в СУБД.

Ход практики

1-я неделя (26.05.25 – 01.06.25):
- Ознакомительный день.
- Оформление документов на практику.
- Инструктаж по технике безопасности.

2-я неделя (02.06.25 – 08.06.25):
- Ознакомление с СУБД Oracle.
- Изучение основ SQL-запросов.
- Создание таблиц, настройка связей и ограничений.
- Работа с триггерами и последовательностями.

3-я неделя (09.06.25 – 15.06.25):
- Получение заданий по алгоритмам.
- Изучение материалов по оптимизации запросов.
- Решение первых алгоритмических задач на Java.

4-я неделя (16.06.25 – 22.06.25):
- Решение второго блока алгоритмических задач.
- Работа над структурой кода, тестирование решений.

5-я неделя (23.06.25 – 28.06.25):
- Проверка выполненных заданий.
- Исправление ошибок.
- Итоговая защита и завершение практики.

Освоенные технологии и инструменты
- Oracle Database: создание таблиц, настройка связей, триггеры, последовательности.
- SQL: SELECT, JOIN, агрегатные функции, создание процедур и функций.
- PL/SQL: обработка ошибок, написание процедур и функций с параметрами.

- Java: решение алгоритмических задач, работа со строками, массивами и структурами данных.
- DBMS_OUTPUT: вывод информации при тестировании PL/SQL-блоков.

Выполненные задания

1. Работа с базами данных в Oracle

Постановка задачи

В рамках практики необходимо было разработать и протестировать базу данных с использованием Oracle. Задачи включали:

- Создание таблиц:
- employees (сотрудники) с полями:
- employee_id (идентификатор сотрудника, уникальный номер),
- first_name (имя),
- last_name (фамилия),
- hire_date (дата найма),
- salary (зарплата),
- department_id (идентификатор отдела, внешний ключ).
- departments (отделы) с полями:
- department_id (идентификатор отдела),
- department_name (название отдела),
- location (местоположение).
- Триггеры:
- Реализовать триггер, автоматически генерирующий ID сотрудника или департамента при вставке в таблицу.
- Заполнение таблиц:
- Добавить несколько тестовых строк для сотрудников и отделов.
- Процедуры:

1. Процедура для увеличения зарплаты сотрудника на заданный процент.

- Параметры:
- p_employee_id – идентификатор сотрудника,
- p_percentage – процент увеличения зарплаты.

2. Процедура для вывода информации о сотруднике по его ID (имя, фамилия, дата найма, зарплата).

- Функции:

- Функция для вычисления средней зарплаты по отделу.
- Параметр: department_id.
- Возврат: средняя зарплата.
- Курсоры:
- Пакет (или анонимный блок), который выводит список сотрудников в заданном отделе, используя неявный курсор.
- Обработка ошибок:
- Добавить обработку ошибок в процедуры и функции (например, если сотрудник с таким ID отсутствует).
- Тестирование:
- Проверить работу всех функций и процедур на тестовых данных.

Решение

Были реализованы SQL-скрипты:

- создание таблиц с ключами и связями,
- настройка последовательностей и триггеров,
- написание процедур, функций и курсоров,
- тестирование всех модулей.

Листинг кода:

```
-- Включаем вывод сообщений из DBMS_OUTPUT
SET SERVEROUTPUT ON;


--  Создание таблиц


CREATE TABLE departments (
    department_id NUMBER PRIMARY KEY,
    department_name VARCHAR2(100),
    location VARCHAR2(100)
);

CREATE TABLE employees (
    employee_id NUMBER PRIMARY KEY,
    first_name VARCHAR2(50),
    last_name VARCHAR2(50),
    hire_date DATE,
    salary NUMBER(10,2),
    department_id NUMBER,
```

```
   FOREIGN KEY (department_id) REFERENCES departments(department_id)
);


--  Создание последовательностей для генерации ID


CREATE SEQUENCE seq_employee_id START WITH 1 INCREMENT BY 1 NOCACHE NOCYCLE;
CREATE SEQUENCE seq_department_id START WITH 1 INCREMENT BY 1 NOCACHE NOCYCLE;


--  Триггеры для автоматической генерации ID


CREATE OR REPLACE TRIGGER trg_employee_id
BEFORE INSERT ON employees
FOR EACH ROW
BEGIN
   IF :NEW.employee_id IS NULL THEN
      :NEW.employee_id := seq_employee_id.NEXTVAL;
   END IF;
END;
/

CREATE OR REPLACE TRIGGER trg_department_id
BEFORE INSERT ON departments
FOR EACH ROW
BEGIN
   IF :NEW.department_id IS NULL THEN
      :NEW.department_id := seq_department_id.NEXTVAL;
   END IF;
END;
/


--  Заполнение таблиц начальными данными


INSERT INTO departments (department_name, location) VALUES ('IT', 'bishkek');
```

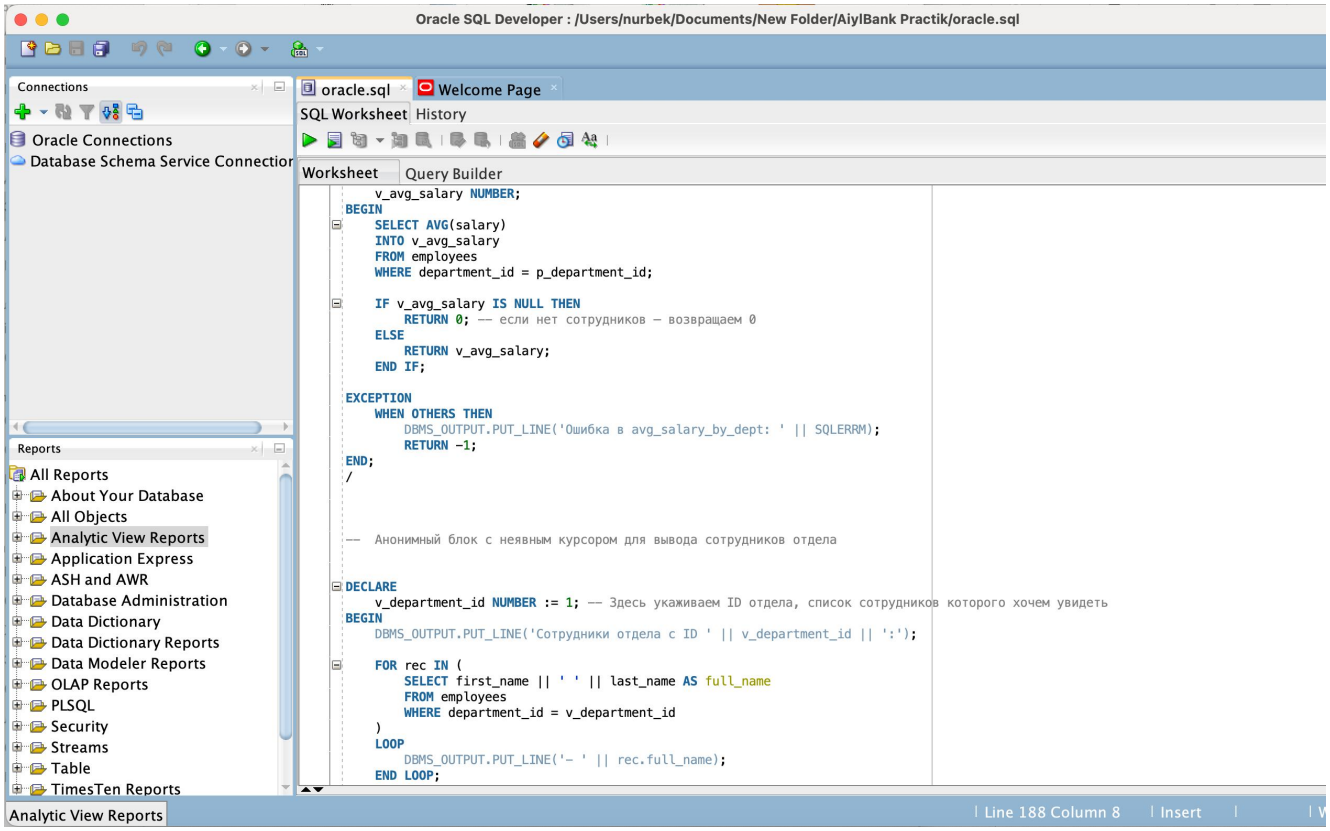
```
INSERT INTO departments (department_name, location) VALUES ('HR', 'bishkek');

INSERT INTO employees (first_name, last_name, hire_date, salary, department_id)
VALUES ('akylbek', 'nurlanov', SYSDATE, 50000, 1);
INSERT INTO employees (first_name, last_name, hire_date, salary, department_id)
VALUES ('azamat', 'aitaliev', SYSDATE, 60000, 1);
INSERT INTO employees (first_name, last_name, hire_date, salary, department_id)
VALUES ('jyldyz', 'kanybekova', SYSDATE, 55000, 2);

COMMIT;


-- Процедура повышения зарплаты


CREATE OR REPLACE PROCEDURE raise_salary(
   p_employee_id IN NUMBER,
   p_percentage IN NUMBER
) IS
BEGIN
   UPDATE employees
   SET salary = salary + (salary * p_percentage / 100)
   WHERE employee_id = p_employee_id;

   IF SQL%ROWCOUNT = 0 THEN
      RAISE_APPLICATION_ERROR(-20001, 'Сотрудник с таким ID не найден.');
   END IF;

EXCEPTION
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('Ошибка в raise_salary: ' || SQLERRM);
END;
/


-- Процедура получения информации о сотруднике
```

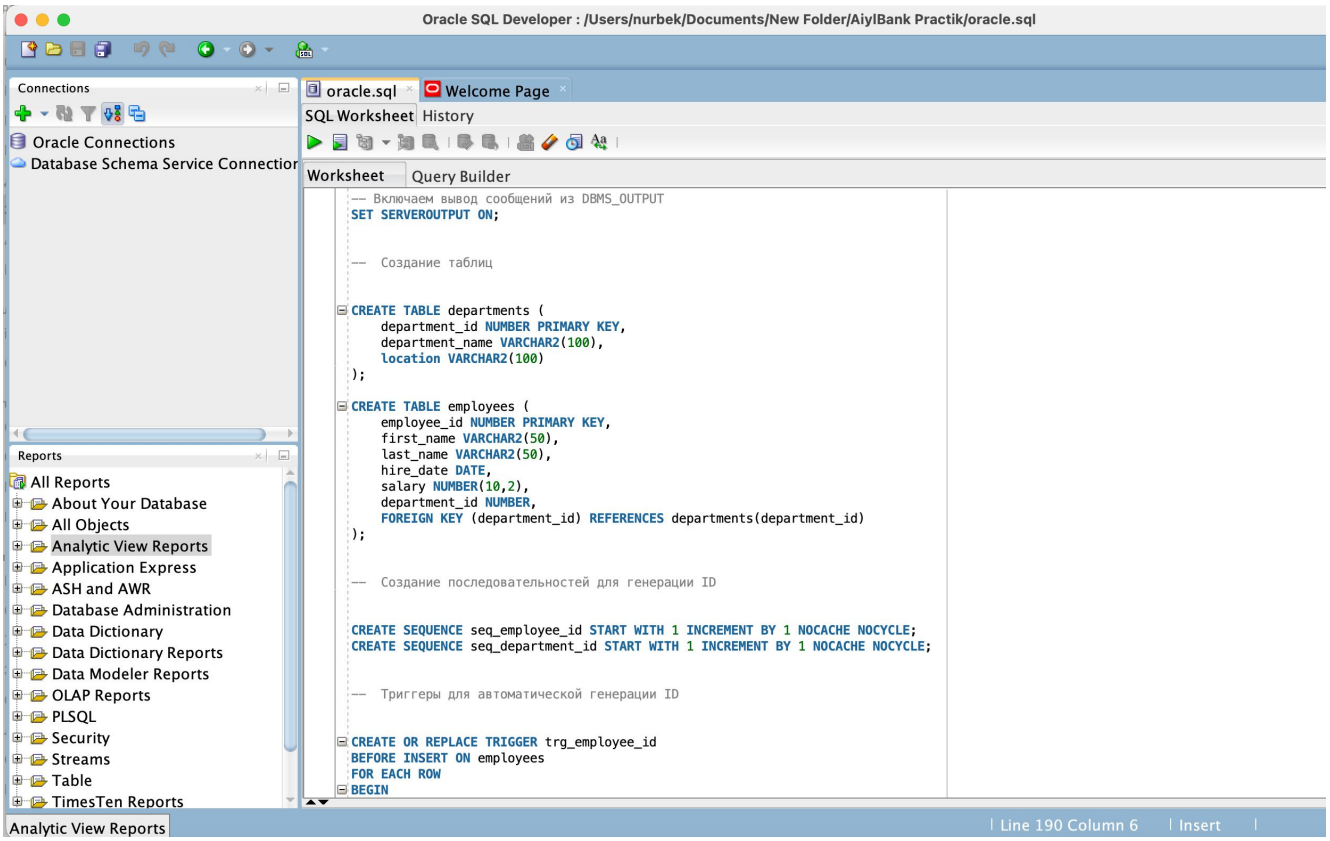
```
CREATE OR REPLACE PROCEDURE get_employee_info(
   p_employee_id IN NUMBER
) IS
   v_first_name employees.first_name%TYPE;
   v_last_name employees.last_name%TYPE;
   v_hire_date employees.hire_date%TYPE;
   v_salary employees.salary%TYPE;
BEGIN
   SELECT first_name, last_name, hire_date, salary
   INTO v_first_name, v_last_name, v_hire_date, v_salary
   FROM employees
   WHERE employee_id = p_employee_id;

   DBMS_OUTPUT.PUT_LINE('Имя: ' || v_first_name);
   DBMS_OUTPUT.PUT_LINE('Фамилия: ' || v_last_name);
   DBMS_OUTPUT.PUT_LINE('Дата найма: ' ||
TO_CHAR(v_hire_date, 'DD.MM.YYYY'));
   DBMS_OUTPUT.PUT_LINE('Зарплата: ' || TO_CHAR(v_salary,
'999999.99'));

EXCEPTION
   WHEN NO_DATA_FOUND THEN
      DBMS_OUTPUT.PUT_LINE('Сотрудник с ID ' || p_employee_id ||
' не найден.');
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('Ошибка в get_employee_info: ' ||
SQLERRM);
END;
/

-- Функция вычисления средней зарплаты по отделу


CREATE OR REPLACE FUNCTION avg_salary_by_dept (
   p_department_id IN NUMBER
) RETURN NUMBER IS
   v_avg_salary NUMBER;
BEGIN
   SELECT AVG(salary)
   INTO v_avg_salary
   FROM employees
   WHERE department_id = p_department_id;
```

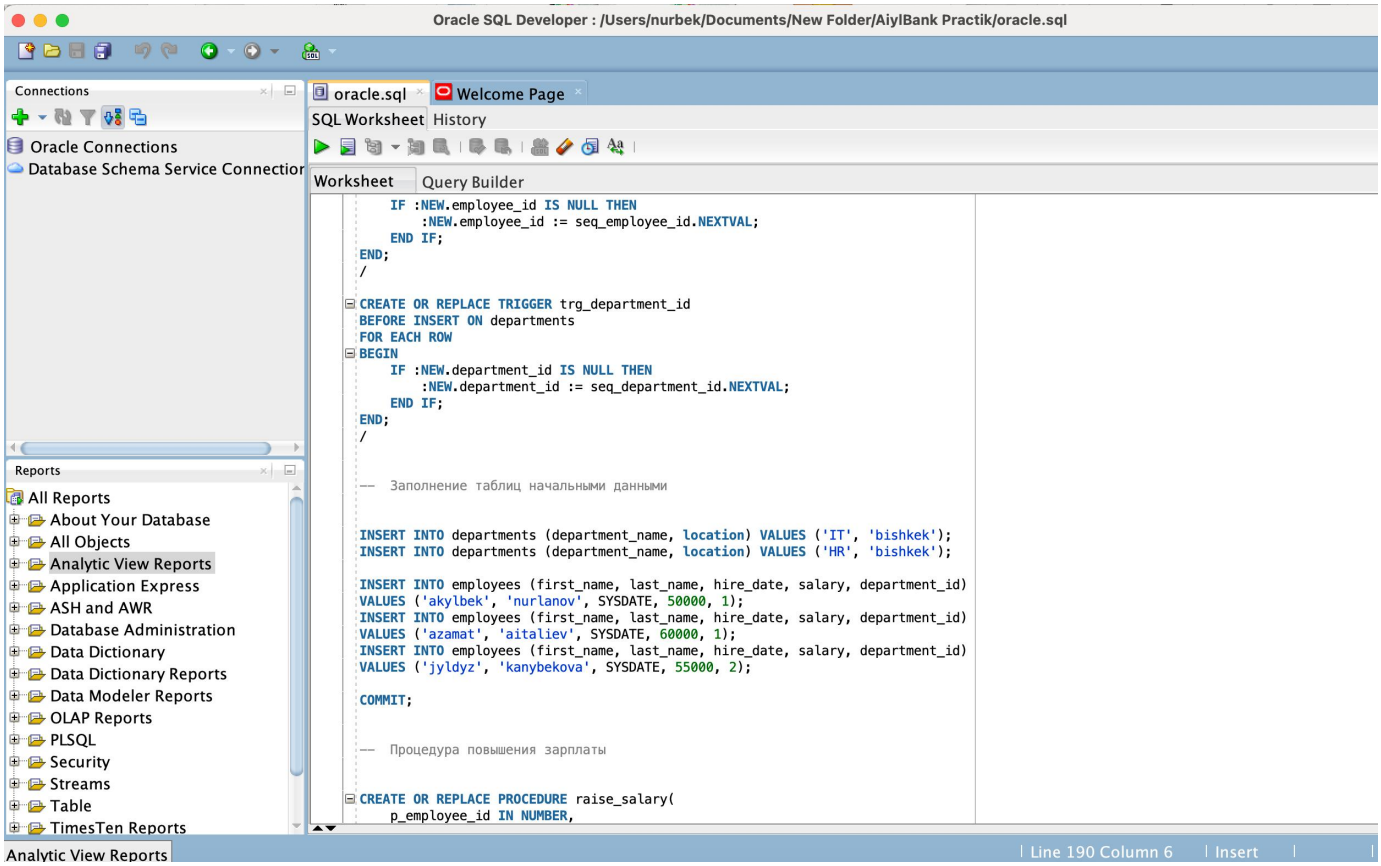
```
   IF v_avg_salary IS NULL THEN
      RETURN 0; -- если нет сотрудников — возвращаем 0
   ELSE
      RETURN v_avg_salary;
   END IF;

EXCEPTION
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('Ошибка в avg_salary_by_dept: ' ||
SQLERRM);
      RETURN -1;
END;
/



--  Анонимный блок с неявным курсором для вывода сотрудников
отдела



DECLARE
   v_department_id NUMBER := 1; -- Здесь укаживаем ID отдела,
список сотрудников которого хочем увидеть
BEGIN
   DBMS_OUTPUT.PUT_LINE('Сотрудники отдела с ID ' ||
v_department_id || ':');

   FOR rec IN (
      SELECT first_name || ' ' || last_name AS full_name
      FROM employees
      WHERE department_id = v_department_id
   )
   LOOP
      DBMS_OUTPUT.PUT_LINE('- ' || rec.full_name);
   END LOOP;
END;
/



-- функция повышение зарплаты
BEGIN
```

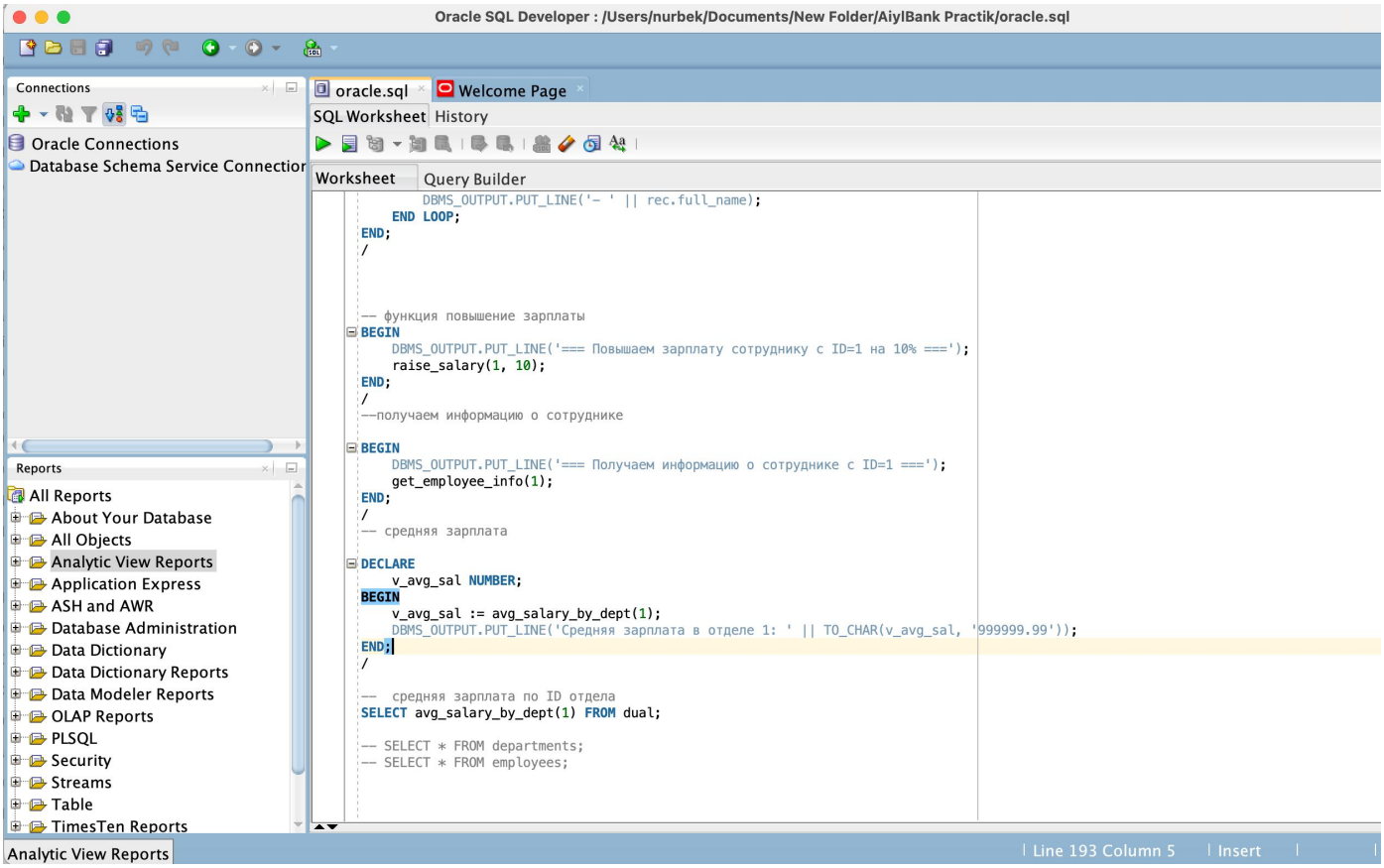
```
   DBMS_OUTPUT.PUT_LINE('=== Повышаем зарплату сотруднику
с ID=1 на 10% ===');
   raise_salary(1, 10);
END;
/
--получаем информацию о сотруднике

BEGIN
   DBMS_OUTPUT.PUT_LINE('===  Получаем  информацию  о
сотруднике с ID=1 ===');
   get_employee_info(1);
END;
/
-- средняя зарплата

DECLARE
   v_avg_sal NUMBER;
BEGIN
   v_avg_sal := avg_salary_by_dept(1);
   DBMS_OUTPUT.PUT_LINE('Средняя  зарплата  в  отделе  1:  ' ||
TO_CHAR(v_avg_sal, '999999.99'));
END;
/

--  средняя зарплата по ID отдела
SELECT avg_salary_by_dept(1) FROM dual;

-- SELECT * FROM departments;
-- SELECT * FROM employees;
```

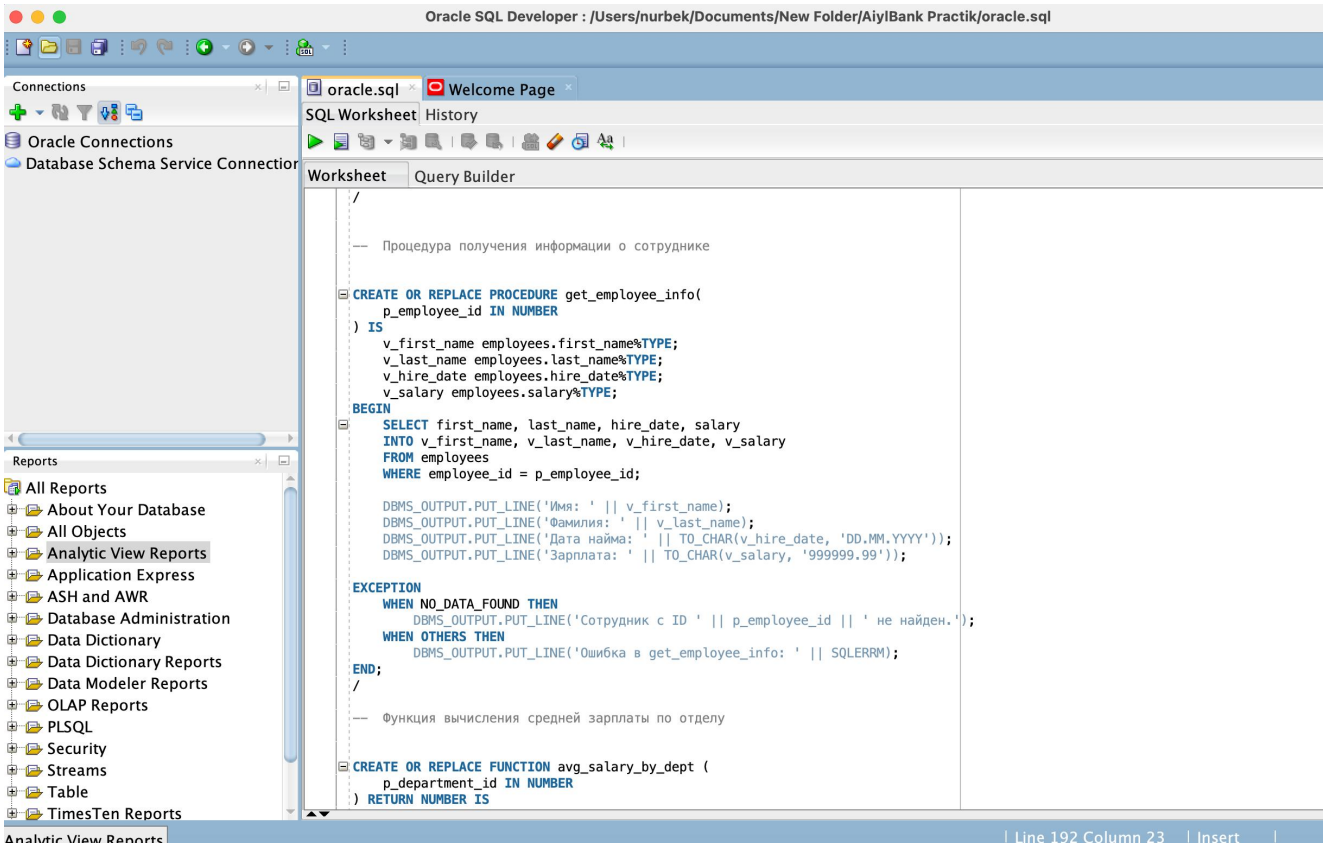
Скрины:


Oracle SQL Developer : /Users/nurbek/Documents/New Folder/AiylBank Practik/oracle.sql
Connections
Oracle Connections
Database Schema Service Connection
oracle.sql
Welcome Page
SQL Worksheet
History
Worksheet
Query Builder
v_avg_salary NUMBER;
BEGIN
SELECT AVG(salary)
INTO v_avg_salary
FROM employees
WHERE department_id = p_department_id;
IF v_avg_salary IS NULL THEN
RETURN 0; -- если нет сотрудников — возвращаем 0
ELSE
RETURN v_avg_salary;
END IF;
EXCEPTION
WHEN OTHERS THEN
DBMS_OUTPUT.PUT_LINE('Ошибка в avg_salary_by_dept: ' || SQLERRM);
RETURN -1;
END;
/
-- Анонимный блок с неявным курсором для вывода сотрудников отдела
DECLARE
v_department_id NUMBER := 1; -- Здесь укаживаем ID отдела, список сотрудников которого хочем увидеть
BEGIN
DBMS_OUTPUT.PUT_LINE('Сотрудники отдела с ID ' || v_department_id || ':');
FOR rec IN (
SELECT first_name || ' ' || last_name AS full_name
FROM employees
WHERE department_id = v_department_id
)
LOOP
DBMS_OUTPUT.PUT_LINE('- ' || rec.full_name);
END LOOP;
Reports
All Reports
About Your Database
All Objects
Analytic View Reports
Application Express
ASH and AWR
Database Administration
Data Dictionary
Data Dictionary Reports
Data Modeler Reports
OLAP Reports
PLSQL
Security
Streams
Table
TimesTen Reports
Analytic View Reports
Line 188 Column 8
Insert


Oracle SQL Developer : /Users/nurbek/Documents/New Folder/AiylBank Practik/oracle.sql
Connections
Oracle Connections
Database Schema Service Connection
oracle.sql
Welcome Page
SQL Worksheet
History
Worksheet
Query Builder
-- Включаем вывод сообщений из DBMS_OUTPUT
SET SERVEROUTPUT ON;
-- Создание таблиц
CREATE TABLE departments (
department_id NUMBER PRIMARY KEY,
department_name VARCHAR2(100),
location VARCHAR2(100)
);
CREATE TABLE employees (
employee_id NUMBER PRIMARY KEY,
first_name VARCHAR2(50),
last_name VARCHAR2(50),
hire_date DATE,
salary NUMBER(10,2),
department_id NUMBER,
FOREIGN KEY (department_id) REFERENCES departments(department_id)
);
-- Создание последовательностей для генерации ID
CREATE SEQUENCE seq_employee_id START WITH 1 INCREMENT BY 1 NOCACHE NOCYCLE;
CREATE SEQUENCE seq_department_id START WITH 1 INCREMENT BY 1 NOCACHE NOCYCLE;
-- Триггеры для автоматической генерации ID
CREATE OR REPLACE TRIGGER trg_employee_id
BEFORE INSERT ON employees
FOR EACH ROW
BEGIN
Reports
All Reports
About Your Database
All Objects
Analytic View Reports
Application Express
ASH and AWR
Database Administration
Data Dictionary
Data Dictionary Reports
Data Modeler Reports
OLAP Reports
PLSQL
Security
Streams
Table
TimesTen Reports
Analytic View Reports
Line 190 Column 6
Insert


Oracle SQL Developer : /Users/nurbek/Documents/New Folder/AiylBank Practik/oracle.sql
Connections
Oracle Connections
Database Schema Service Connection
oracle.sql
Welcome Page
SQL Worksheet
History
Worksheet
Query Builder
IF :NEW.employee_id IS NULL THEN
:NEW.employee_id := seq_employee_id.NEXTVAL;
END IF;
END;
/
CREATE OR REPLACE TRIGGER trg_department_id
BEFORE INSERT ON departments
FOR EACH ROW
BEGIN
IF :NEW.department_id IS NULL THEN
:NEW.department_id := seq_department_id.NEXTVAL;
END IF;
END;
/
-- Заполнение таблиц начальными данными
INSERT INTO departments (department_name, location) VALUES ('IT', 'bishkek');
INSERT INTO departments (department_name, location) VALUES ('HR', 'bishkek');
INSERT INTO employees (first_name, last_name, hire_date, salary, department_id)
VALUES ('akylbek', 'nurlanov', SYSDATE, 50000, 1);
INSERT INTO employees (first_name, last_name, hire_date, salary, department_id)
VALUES ('azamat', 'aitaliev', SYSDATE, 60000, 1);
INSERT INTO employees (first_name, last_name, hire_date, salary, department_id)
VALUES ('jyldyz', 'kanybekova', SYSDATE, 55000, 2);
COMMIT;
-- Процедура повышения зарплаты
CREATE OR REPLACE PROCEDURE raise_salary(
p_employee_id IN NUMBER,
Reports
All Reports
About Your Database
All Objects
Analytic View Reports
Application Express
ASH and AWR
Database Administration
Data Dictionary
Data Dictionary Reports
Data Modeler Reports
OLAP Reports
PLSQL
Security
Streams
Table
TimesTen Reports
Analytic View Reports
Line 190 Column 6
Insert


Oracle SQL Developer : /Users/nurbek/Documents/New Folder/AiylBank Practik/oracle.sql
Connections
Oracle Connections
Database Schema Service Connection
oracle.sql
Welcome Page
SQL Worksheet
History
Worksheet
Query Builder
DBMS_OUTPUT.PUT_LINE('- ' || rec.full_name);
END LOOP;
END;
/
-- функция повышение зарплаты
BEGIN
DBMS_OUTPUT.PUT_LINE('=== Повышаем зарплату сотруднику с ID=1 на 10% ===');
raise_salary(1, 10);
END;
/
--получаем информацию о сотруднике
BEGIN
DBMS_OUTPUT.PUT_LINE('=== Получаем информацию о сотруднике с ID=1 ===');
get_employee_info(1);
END;
/
-- средняя зарплата
DECLARE
v_avg_sal NUMBER;
BEGIN
v_avg_sal := avg_salary_by_dept(1);
DBMS_OUTPUT.PUT_LINE('Средняя зарплата в отделе 1: ' || TO_CHAR(v_avg_sal, '999999.99'));
END;
/
-- средняя зарплата по ID отдела
SELECT avg_salary_by_dept(1) FROM dual;
-- SELECT * FROM departments;
-- SELECT * FROM employees;
Reports
All Reports
About Your Database
All Objects
Analytic View Reports
Application Express
ASH and AWR
Database Administration
Data Dictionary
Data Dictionary Reports
Data Modeler Reports
OLAP Reports
PLSQL
Security
Streams
Table
TimesTen Reports
Analytic View Reports
Line 193 Column 5
Insert



Краткое описание кода и результатов

В работе были реализованы:

1.Создание таблиц

- departments — содержит данные об отделах (ID, название, локация).
- employees — содержит данные о сотрудниках (ID, имя, фамилия, дата найма, зарплата, отдел).

Между ними установлена связь по department_id.

2.Последовательности и триггеры

- seq_employee_id и seq_department_id автоматически генерируют новые ID.
- trg_employee_id и trg_department_id — триггеры, которые подставляют ID при вставке записей.

3.Наполнение данными

Добавлены отделы: IT и HR, а также три сотрудника:

- Akylbek Nurlanov (IT, зарплата 50 000)
- Azamat Aitaliev (IT, зарплата 60 000)
- Jyldyz Kanybekova (HR, зарплата 55 000)

4.Процедуры

- raise_salary — увеличивает зарплату сотрудника на заданный процент.
- get_employee_info — выводит информацию о сотруднике (имя, фамилия, дата найма, зарплата).

5.Функция

- avg_salary_by_dept — вычисляет среднюю зарплату сотрудников в отделе.

6.Анонимные блоки

- Вывод списка сотрудников отдела по ID.
- Тест вызова процедур и функции.

Результаты выполнения

- После вызова raise_salary(1, 10) зарплата Akylbek Nurlanov увеличилась с 50 000 → 55 000.
- Процедура get_employee_info(1) вывела информацию о сотруднике:

Имя: akylbek
Фамилия: nurlanov
Дата найма: <текущая дата>
Зарплата: 55000.00

- Функция avg_salary_by_dept(1) вернула среднюю зарплату в отделе IT:

(55 000 + 60 000) / 2 = 57 500

- Анонимный блок вывел список сотрудников отдела IT:

- akylbek nurlanov
- azamat aitaliev

2. Первая задача по алгоритму

Тема: Числа без одинаковых цифр

Антон записал ряд натуральных чисел в порядке возрастания: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23 и т.д. Затем вычеркнул из него все числа, в которых имеется хотя бы две

одинаковых цифры, и получил последовательность: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 23 и т.д.

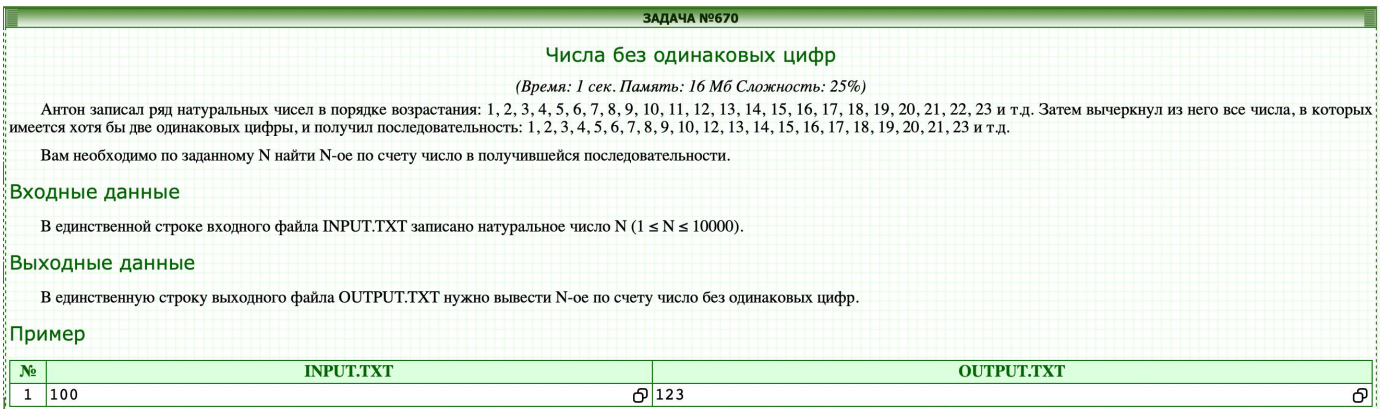
Вам необходимо по заданному N найти N-ое по счету число в получившейся последовательности.

Входные данные

В единственной строке входного файла INPUT.TXT записано натуральное число N ($1 \le N \le 10000$).

Выходные данные

В единственную строку выходного файла OUTPUT.TXT нужно вывести N-ое по счету число без одинаковых цифр.



ЗАДАЧА №670

Числа без одинаковых цифр

(Время: 1 сек. Память: 16 Мб Сложность: 25%)

Антон записал ряд натуральных чисел в порядке возрастания: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23 и т.д. Затем вычеркнул из него все числа, в которых имеется хотя бы две одинаковых цифры, и получил последовательность: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 23 и т.д.

Вам необходимо по заданному N найти N-ое по счету число в получившейся последовательности.

Входные данные

В единственной строке входного файла INPUT.TXT записано натуральное число N ($1 \le N \le 10000$).

Выходные данные

В единственную строку выходного файла OUTPUT.TXT нужно вывести N-ое по счету число без одинаковых цифр.

Пример

№	INPUT.TXT	OUTPUT.TXT
1	100	123

Условие задачи:

Необходимо найти N-ое число в последовательности натуральных чисел, в которых нет одинаковых цифр. Например, последовательность начинается так: 1, 2, 3, …, 10, 12, 13, 14, …

Листинг кода:

```
import java.util.HashSet;
import java.util.Scanner;
import java.util.Set;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int N = scanner.nextInt();
```

```
        int count = 0;
        int number = 1;

        while (true) {
            if (hasAllUniqueDigits(number)) {
                count++;
                if (count == N) {
                    System.out.println(number);
                    break;
                }
            }
            number++;
        }
    }

    // Проверяем, есть ли одинаковые цифры
    public static boolean hasAllUniqueDigits(int num) {
        Set<Character> digits = new HashSet<>();
        char[] chars = String.valueOf(num).toCharArray();
        for (char c : chars) {
            if (digits.contains(c)) {
                return false;
            }
            digits.add(c);
        }
        return true;
    }
}
```

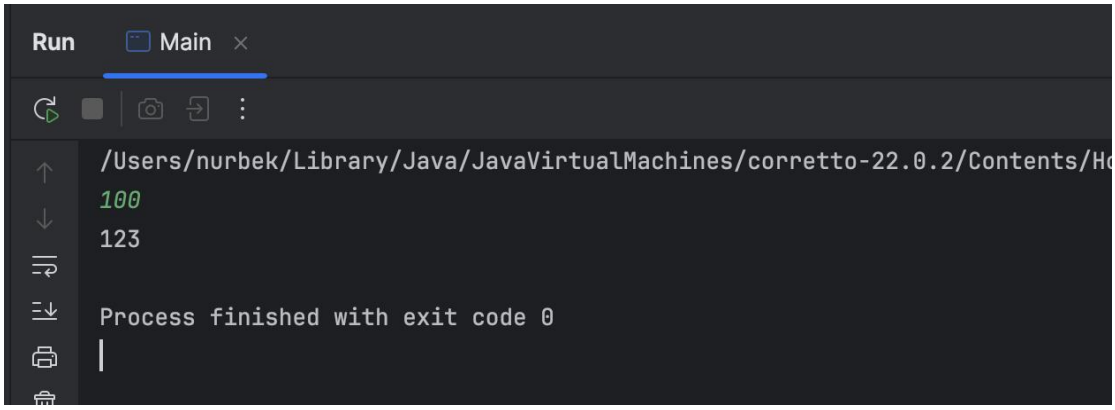
Описание решения:

- Метод hasUniqueDigits(int num) проверяет, что цифры числа не повторяются.
- Используется массив boolean[10] для учёта уже встреченных цифр.
- В цикле перебираются числа, пока не найдено N-ое число без повторяющихся цифр.

Вывод:

- Алгоритм корректно находит N-ое число.
- Пример: при N = 100 результат — 123.

Скрин работы программы:



3. Вторая задача по алгоритму

Тема: Игра со спичками

Двое играют в следующую игру. Из кучки спичек за один ход игрок вытягивает либо 1, либо 2, либо 1000 спичек. Выигрывает тот, кто забирает последнюю спичку. Кто выигрывает при правильной игре?

Входные данные

В единственной строке входного файла INPUT.TXT записано одно натуральное число — N ($1 \leq N \leq 10000$) начальное количество спичек в кучке.

Выходные данные

В единственную строку выходного файла OUTPUT.TXT нужно вывести 1, если выигрывает первый игрок (тот, кто ходит первым), или 2, если выигрывает второй игрок.

ЗАДАЧА №676

Игра со спичками

(Время: 1 сек. Память: 16 Мб Сложность: 28%)

Двое играют в следующую игру. Из кучки спичек за один ход игрок вытягивает либо 1, либо 2, либо 1000 спичек. Выигрывает тот, кто забирает последнюю спичку. Кто выигрывает при правильной игре?

Входные данные

В единственной строке входного файла INPUT.TXT записано одно натуральное число — N ($1 \leq N \leq 10000$) начальное количество спичек в кучке.

Выходные данные

В единственную строку выходного файла OUTPUT.TXT нужно вывести 1, если выигрывает первый игрок (тот, кто ходит первым), или 2, если выигрывает второй игрок.

Примеры

№	INPUT.TXT	OUTPUT.TXT
1	2	1
2	3	2

Условие задачи:

Двое играют с кучкой спичек. За ход можно взять 1, 2 или 1000 спичек. Побеждает тот, кто забирает последнюю спичку. Необходимо определить победителя при правильной игре обоих игроков.

Листинг кода:

```
import java.util.Scanner;


public class MatchstickGame {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int N = sc.nextInt();


        // Игрок 1 выигрывает, если количество спичек % 3 != 0 и N != 1000
        if (N == 1000) {
            System.out.println(2); // проигрыш для первого
        } else if (N % 3 == 0) {
            System.out.println(2); // второй игрок выигрывает
        } else {
            System.out.println(1); // первый игрок выигрывает
        }
    }
}
```

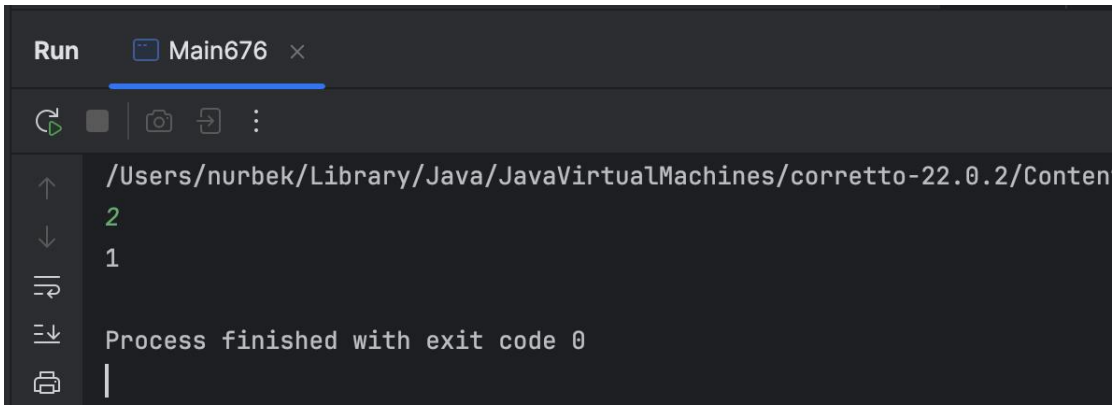
Описание решения:

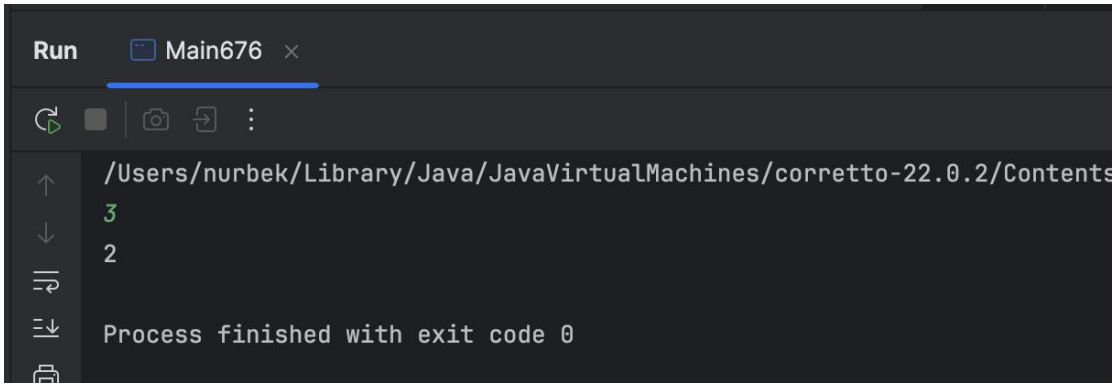
- Определяется стратегия “выигрышного остатка”.
- Если количество спичек кратно 3 (и не равно 1000), первый игрок проигрывает.
- Если N не кратно 3, первый игрок может сделать ход так, чтобы оставить второму кратное 3, обеспечивая победу.
- Особый случай — N = 1000, первый игрок не может выиграть, если второй играет правильно.

Вывод:

- Программа определяет победителя за O(1) операций.
- Примеры:
- N = 2 → выигрывает первый игрок → вывод 1
- N = 3 → выигрывает второй игрок → вывод 2

Скрин работы программы:





Вывод

Производственная практика в ОАО «Айыл Банк» позволила закрепить теоретические знания и приобрести практические навыки в области работы с базами данных, алгоритмами и языком программирования Java. В процессе практики были изучены и применены современные технологии и инструменты работы с данными: Oracle Database, SQL и PL/SQL, а также алгоритмическое решение задач на Java.

Было выполнено несколько ключевых заданий:

- разработка базы данных сотрудников и отделов с использованием таблиц, связей, триггеров, последовательностей, процедур и функций;
- реализация алгоритмических задач на языке Java, включая работу с числами без повторяющихся цифр и решение логических задач;
- тестирование и отладка всех модулей базы данных и алгоритмов.

Практика способствовала развитию профессиональных навыков:

- умению создавать и управлять базами данных в Oracle;
- проектированию и оптимизации SQL-запросов, процедур и функций;
- написанию структурированного, читаемого и корректного кода на Java;
- навыкам обработки ошибок и отладки сложных систем;
- работе с алгоритмами и логическим мышлением при решении практических задач.

Итогом прохождения практики стало формирование опыта работы с корпоративными информационными системами банка, закрепление знаний по базам данных и алгоритмам, а также значительное расширение профессиональных компетенций в сфере IT.

Итоговое заключение

Производственная практика в ОАО «Optima Bank» и ОАО «Айыл Банк» позволила комплексно закрепить теоретические знания, полученные в университете, и приобрести практический опыт работы в разных направлениях информационных технологий.

В ходе практики в «Optima Bank» были освоены современные подходы к разработке мобильных приложений на платформе Android: язык Kotlin, фреймворк Jetpack Compose, архитектурные паттерны MVVM, Repository и Singleton. Реализация проектов «Калькулятор» и «Travel» способствовала развитию навыков проектирования интерфейсов, работы с системой контроля версий GitHub и командной разработки.

Практика в «Айыл Банк» позволила углубить знания в области работы с базами данных, алгоритмами и программированием на Java. Разработка базы данных сотрудников и отделов с использованием таблиц, связей, триггеров, процедур и функций, а также решение алгоритмических задач способствовали формированию умений работы с Oracle Database, SQL, PL/SQL и логическим мышлением при решении практических задач.

В результате обеих практик были достигнуты следующие результаты:

- приобретен опыт работы в реальных корпоративных проектах;
- закреплены навыки программирования, проектирования и оптимизации кода;
- освоены современные технологии и инструменты разработки мобильных приложений и баз данных;
- развиты аналитические и алгоритмические способности, а также навыки командной работы.

Итогом прохождения практики стало значительное расширение профессиональных компетенций в сфере IT и мобильной разработки, а также готовность к эффективной работе в профессиональной среде.